



Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Programmer's Reference for the DG/UX™ System (Volume 3)

093-701102-01

A V I I O N®
P R O D U C T L I N E

Programmer's Reference for the DG/UX™ System (Volume 3)

093-701102-01

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-701102
Copyright © Data General Corporation, 1990, 1991, 1992
Unpublished—all rights reserved under the copyright laws of the United States
Printed in the United States of America
Revision 01, February 1992
Licensed material—property of copyright holder(s)

NOTICE

DATA GENERAL CORPORATION (DGC) HAS PREPARED AND/OR HAS DISTRIBUTED THIS DOCUMENT FOR USE BY DGC PERSONNEL, LICENSEES, AND CUSTOMERS. THE INFORMATION CONTAINED HEREIN IS THE PROPERTY OF THE COPYRIGHT HOLDER(S); AND THE CONTENTS OF THIS MANUAL SHALL NOT BE REPRODUCED IN WHOLE OR IN PART NOR USED OTHER THAN AS ALLOWED IN THE APPLICABLE LICENSE AGREEMENT.

The copyright holder(s) reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS, AND THE TERMS AND CONDITIONS GOVERNING THE LICENSING OF THIRD PARTY SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE APPLICABLE LICENSE AGREEMENT. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW, OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

All software is made available solely pursuant to the terms and conditions of the applicable license agreement which governs its use.

Restricted Rights Legend: Use, duplications, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at [FAR] 52.227-7013 (May 1987).

DATA GENERAL CORPORATION
4400 Computer Drive
Westboro, MA 01580

AVHON, CEO, DASHER, DATAPREP, ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, PRESENT, and TRENDVIEW are U.S. registered trademarks of Data General Corporation. CEO Connection, CEO Connection/LAN, DASHER/One, DASHER/286, DASHER/286-12c, DASHER/286-12j, DASHER/386, DASHER/386-16c, DASHER/386-25, DASHER/386-25k, DASHER/386sx, DASHER/386SX-16, DASHER/386SX-20, DASHER/486-25, DASHER/LN, DATA GENERAL/One, DG/UX, ECLIPSE MV/1000, ECLIPSE MV/1400, ECLIPSE MV/2000, ECLIPSE MV/2500, ECLIPSE MV/3500, ECLIPSE MV/5000, ECLIPSE MV/5500, ECLIPSE MV/5600, ECLIPSE MV/7800, ECLIPSE MV/9300, ECLIPSE MV/9500, ECLIPSE MV/9600, ECLIPSE MV/10000, ECLIPSE MV/15000, ECLIPSE MV/18000, ECLIPSE MV/20000, ECLIPSE MV/30000, ECLIPSE MV/40000, Intellibook, microECLIPSE, microMV, MV/UX, PC Liaison, RASS, SPARE MAIL, TEO, TEO/3D, TEO/Electronics, TURBO/4, UNITE, and XODIAC are trademarks of Data General Corporation.

IBM is a U.S. registered trademark of International Business Machines Corporation.

UNIX is a U.S. registered trademark of American Telephone & Telegraph Company.

NFS is a trademark of Sun Microsystems, Inc.

Portions of this text are reprinted from IEEE Std 1003.1-1988, *Portable Operating System Interface for Computer Environments*, copyright © 1988 by the Institute of Electrical and Electronics Engineers, Inc., with the permission of the IEEE Standards Department. To purchase IEEE Standards, call 800/678-IEEE.

Portions of this material have been previously copyrighted by: American Telephone & Telegraph Company, 1989, 1990; Regents of the University of California, 1980, 1983, 1986.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages. The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc and may not be used without permission.

LEGAL NOTICE TO USERS: Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may also be a trademark of various telephone companies around the world. Sun will be revising future versions of software and documentation to remove references to Yellow Pages.

Programmer's Reference for the DG/UX System (Volume 3)

093-701102-01

Revision History:

Original Release - June 1991
Revision 1 - February 1992

Effective with:

DG/UX 5.4
DG/UX 5.4.1

NAME

dirent – file system independent directory entry

SYNOPSIS

```
#include <sys/dirent.h>
#include <sys/types.h>
```

DESCRIPTION

Different file system types may have different directory entries. The `dirent` structure defines a file system independent directory entry, which contains information common to directory entries in different file system types. A set of these structures is returned by the `getdents(2)` system call.

The `dirent` structure is defined below.

```
struct dirent {
    long          d_ino;
    off_t         d_off;
    unsigned short d_reclen;
    char          d_name[1];
};
```

The `d_ino` is a number which is unique for each file in the file system. The field `d_off` is the offset of that entry in the file system directory. The field `d_name` is the beginning of the character array giving the name of the directory entry. This name is null terminated and may have at most `MAXNAMLEN` characters. This results in file system independent directory entries being variable length entities. The value of `d_reclen` is the record length of this entry. This length is defined to be the number of bytes between the current entry and the next one, so that it will always result in the next entry being on a long boundary.

FILES

`/usr/include/sys/dirent.h`

SEE ALSO

`getdents(2)`.

NAME

dumptab – tape table file for dump2

DESCRIPTION

/etc/dumptab is an ASCII file containing an entry describing media characteristics for each medium made available to dump2.

This table file contains lines in one of three formats:

- a. comment lines (must start with a "#")
- b. lines specifying the capacity of the medium:

medium-name buffer-size <capacity>

- c. lines giving the density, tape length, and IRG for the medium:

medium-name buffer-size density tape-length <IRG>

Fields are separated by white space. The fields are described below:

medium-name

descriptive label for the medium.

buffer-size

size (in 1024-byte blocks) of the buffers written to the medium.

capacity

formatted capacity of the medium (in bytes). The capacity can also be specified as a number followed by a upper or lowercase b, k, m, or g to indicate bytes, kilobytes, megabytes, or gigabytes, respectively.

density density at which data is written to the device (in bpi).

tape-length

length of the tape (in feet).

IRG

inter-record gap size used by the device (in tenths per inch).

SEE ALSO

dump2(1M).

NAME

ethers – Ethernet address to hostname database or YP domain

DESCRIPTION

The `ethers` file contains information regarding the known (48 bit) Ethernet addresses of hosts on the Internet. For each host on an Ethernet, a single line should be present with the following information:

ethernet_address official_hostname

Separate items by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment extending to the end of line.

The standard form for Ethernet addresses is "x:x:x:x:x:x" where *x* is a hexadecimal number between 0 and ff, representing one byte. The address bytes are always in network order. Host names may contain any printable character other than a space, tab, newline, or comment character. It is intended that hostnames in the `ethers` file correspond to the hostnames in the `hosts(4)` file.

The `ether_line()` routine from the Ethernet address manipulation library, `ethers(3N)` may be used to scan lines of the `ethers` file.

EXAMPLE

The following is a sample `/etc/ethers` file:

```
8:0:1b:0:a0:17    dg1
0:0:77:1a:0:6a    sales
8:0:20:0:a7:5d    sun1
```

If you use the domain name system, you should specify fully-qualified names in addition to official hostnames. Here is the same sample `/etc/ethers` file including fully-qualified names:

```
8:0:1b:0:a0:17    dg1
0:0:77:1a:0:6a    sales
8:0:20:0:a7:5d    sun1
8:0:1b:0:a0:17    dg1.tnt.acme.com
0:0:77:1a:0:6a    sales.tnt.acme.com
8:0:20:0:a7:5d    sun1.tnt.acme.com
```

For more information about the domain name system, see *Managing TCP/IP on the DG/UX™ System*.

FILES

`/etc/ethers`

SEE ALSO

`ethers(3N)`, `hosts(4)`

NAME

exports, xtab - directories to export to NFS clients

SYNOPSIS

/etc/exports

/etc/xtab

DESCRIPTION

The /etc/exports file contains entries for directories that can be exported to NFS clients. This file is read automatically by the exportfs(1M) command. If you change this file, you must run exportfs(1M) for the changes to affect the mountd server's operation.

Only when this file is present at boot time does the rc.nfslockd script execute exportfs(1M). The rc.nfsserv script starts the NFS file-system server (daemon), nfsd(1M).

The /etc/xtab file contains entries for directories that are *currently* exported. This file should only be accessed by programs using getexportent (see exportent(3C)). (Use the -u option of exportfs to remove entries from this file).

An entry for a directory consists of a line of the following form:

directory -*option*[, *option*]...

directory is the pathname of a directory (or file).

option is one of

ro Export the directory read-only. If not specified, the directory is exported read-write.

rw=*hostnames*[:*hostname*]...

Export the directory read-mostly. Read-mostly means read-only to most machines, but read-write to those specified. If not specified, the directory is exported read-write to all.

anon=*uid*

If a request comes from an unknown user, use *uid* as the effective user ID. Note: root users (uid 0) are always considered unknown by the NFS server, unless they are included in the root option below. The default value for this option is -2. Setting anon to -1 disables anonymous access. Note: by default secure NFS will accept insecure requests as anonymous, and those wishing for extra security can disable this feature by setting anon to -1.

root=*hostnames*[:*hostname*]...

Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access.

access=*client*[:*client*]...

Give mount access to each *client* listed. A *client* can be either a hostname, or a netgroup (see netgroup(5)). Each *client* in the list is first checked for in the netgroup database, and then the hosts database. The default

value allows any machine to mount the given directory.

`secure`

Require clients to use a more secure protocol when accessing the directory.

A '#' (pound-sign) anywhere in the file indicates a comment that extends to the end of the line.

EXAMPLE

```
/usr      -access=clients      # export to my clients
/usr/local      # export to the world
/usr2      -access=hermes:zip:tutorial # export to only these machines
/usr/dgux  -root=hermes:zip    # give root access only to these
/usr/new    -anon=0           # give all machines root access
/usr/bin    -ro               # export read-only to everyone
/usr/stuff  -access=zip,anon=-3,ro # several options on one line
```

FILES

```
/etc/exports
/etc/xtab
/etc/hosts
/etc/netgroup
```

SEE ALSO

exportfs(1M), nfsd(1M), exportent(3C), hosts(5), netgroup(5).

WARNINGS

You cannot export either a parent directory or a subdirectory of an exported directory that is *within the same filesystem*. It would be illegal, for instance, to export both `/usr` and `/usr/local` if both directories resided on the same disk partition.

NAME

filehdr – file header for common object files

SYNOPSIS

```
#include <filehdr.h>
```

DESCRIPTION

Every common object file begins with a 20-byte header. The following C struct declaration is used:

```
struct filehdr {
    unsigned short f_magic ; /* magic number */
    unsigned short f_nscns ; /* number of sections */
    long          f_timdat ; /* time & date stamp */
    long          f_symptr ; /* file ptr to symtab */
    long          f_nsyms ; /* # symtab entries */
    unsigned short f_opthdr ; /* sizeof(opt hdr) */
    unsigned short f_flags ; /* flags */
} ;
```

f_symptr is the byte offset into the file at which the symbol table can be found. Its value can be used as the offset in `fseek(3S)` to position an I/O stream to the symbol table. The UNIX system optional header is 28-bytes. The magic number for the M88000 is:

```
#define MC88MAGIC 0540
```

The value in *f_timdat* is obtained from the `time(2)` system call. Flag bits currently defined are:

```
#define F_RELFLG 0000001 /* relocation entries stripped */
#define F_EXEC 0000002 /* file is executable */
#define F_LNNO 0000004 /* line numbers stripped */
#define F_LSYMS 0000010 /* local symbols stripped */
#define F_AR32W 0001000 /* non-DEC host */
#define F_BM32B 0020000 /* file contains WE 32100 code */
#define F_BM32MAU 0040000 /* file reqs MAU to execute */
```

SEE ALSO

`time(2)`, `fseek(3S)`, `a.out(4)`.

NAME

fs – file system format

SYNOPSIS

```
#include <ufs/disk_format.h>
```

DESCRIPTION

There is at most one filesystem for each logical disk. The basic components of a the file system are the File Manager Information Areas (FMIA's), Disk Allocation Regions (DAR's), and a table of entries containing information about each DAR called the DAR Information Area.

The FMIA

Two copies of the FMIA are maintained to reduce its vulnerability to corruption. The copies are placed in the first and last blocks of the file system. The FMIA in the first block (the Primary FMIA) is contained in the first DAR, but the FMIA contained in the last block of the logical disk (the Secondary FMIA) is not contained in the last DAR.

The following is the definition of a FMIA. This contains the per-filesystem information. When a filesystem is mounted, this structure is used to generate memory data-bases for the newly mounted entry.

```
typedef struct
{
    df_self_id_type      self_id;
    df_fsid_type         fsid;
    uint32e_type         minor_device_number;
    uint32e_type         dar_size;
    uint32e_type         file_nodes_per_dar;
    boolean16e_type      fsck_required;
    uint16e_type         revision;
    byte8e_type          fname[DF_FS_LABEL_SIZE];
    byte8e_type          fpack[DF_FS_LABEL_SIZE];
    uint8e_type          default_des_exponent;
    uint8e_type          default_ies_exponent;
    uint8e_type          default_dir_des_exponent;
    uint8e_type          default_dir_ies_exponent;
    uint32e_type         first_anniversary;
    uint32e_type         second_anniversary;
    uint32e_type         fs_size;
    uint32e_type         space_used;
    uint32e_type         number_of_used_file_nodes;
    uint32e_type         first_log_lda;
    uint32e_type         second_log_lda;
    uint32e_type         log_size;
    boolean_field_type   shrink_operation_in_progress;
    boolean_field_type   grow_operation_in_progress;
    skip_type            reserved:14;
    byte8e_type          pad_to_block[DF_PADDING_PER_FMIA_BLOCK];
} df_fmia_block_type ;
```

self_id is the self-identification information. The block kind is DF_FMIA_BLOCK. The block number is:

```
#define DF_PRIMARY_FMIA_ADDRESS 0
```

The file node number is:

```
#define DF_NODE_NUMBER_FOR_NON_FILES    012345670123
```

The following fields are assumed to be correct by `fsck(1M)`.

fsid is the filesystem identifier unique among mounted file systems on a single host. It is kept on disk so that it will stay the same if possible from mount to mount. If it doesn't, NFS accesses using filehandles based on a previous mount will fail.

minor_device_number is the assigned extended minor device number. It is kept on disk so that it will stay the same if possible from mount to mount. If the value in this field on disk is not in the valid range for extended minor device numbers, it is file manager's responsibility to correct the problem at mount time.

dar_size is the size of a DAR in blocks. The minimum value for this field is:

```
#define DF_MIN_DAR_SIZE    4032
```

and the maximum value is:

```
#define DF_MAX_DAR_SIZE(fs_size)
```

`mkfs(1M)` defines the default for this field; for efficiency, it should be a multiple of:

```
#define DF_BITS_PER_BITMAP_BLOCK 4032
```

whenever possible; 4 to 12 MB (two to six bitmap blocks' worth) per DAR seems a reasonable default DAR size given current disk sizes. As disks grow by orders of magnitude in size, DAR sizes should likely grow linearly with the square root of the disk sizes.

file_nodes_per_dar is the number of file nodes for each DAR. This value must be a multiple of:

```
#define DF_FILE_NODE_MULTIPLE_REQUIREMENT    64
```

The minimum value for this field is

```
#define DF_MIN_FILE_NODES_PER_DAR 64
```

and the maximum value is:

```
#define DF_MAX_FILE_NODES_PER_DAR(dar_size)
```

`mkfs(1M)` defines this field's default, which is to have about one file node for each four user data blocks, similar to 4.2 BSD.

fsck_required indicates that `fsck(1M)` needs to be run. If this field is not zero (FALSE), the filesystem needs to be checked before it can be mounted.

revision is the revision number of the FMIA. Used to determine the type of filesystem that the FMIA resides on.

Preface

This is Volume 3 of the *Programmer's Reference for the DG/UX™ System*. The *Programmer's Reference* describes the programming features of the DG/UX system. It contains individual manual pages that describe commands, system calls, subroutines, file formats, and other useful topics, such as the ASCII table shown on `ascii(5)`.

This manual is part of a five-volume reference set. The other manuals are the *System Manager's Reference for the DG/UX System* and the *User's Reference for the DG/UX System*. These manuals contain in printed (typeset) form the online entries released with the DG/UX System in `/usr/catman` for access by the `man` command.

The *Programmer's Reference* provides neither a general overview of the DG/UX system nor details of the implementation of the system. For more details about some of the most often used programming tools, see *Programmer's Guide: ANSI C and Programming Support Tools*, *Programmer's Guide: System Services and Application Packaging Tools*, and the Data General supplements to these two manuals. Other related manuals are listed under “Related Documents” at the end of this manual.

Man Pages

For historical reasons, each entry is called a “manual page” or “man page,” though an entry may occupy more than one physical page and may contain more than one entry. If the man page contains more than one entry, it is alphabetized under its “primary” name; for example, the `utmp` manual page describes the `utmp` and `wtmp` files.

Manual pages are assigned to classes ranging from 0 through 8 for easy cross-reference. The class number appears in parentheses following the name; for example, in `accept(1M)` the “1” indicates that `accept` is a command, and the “M” indicates that the man page is in the *System Manager's Reference*.

A command followed by a (1) or (1G) usually means that it is described in the *User's Reference*. (Class 1 commands appropriate for use by programmers are located in the *Programmer's Reference*.) A man page name with a (1M), (4M), (7), or (8) following it means that the entry is in the *System Manager's Reference*. Names with (2) or (3x), (4), (5) [except `editread(5)`], or (6F) are in the *Programmer's Reference*. Occasionally, DG/UX man pages refer to other products' man pages, which are not part of the DG/UX documentation; these are so noted.

Manual Organization

Volume 1 contains two chapters:

Chapter 1: Commands (1)

This chapter describes commands that support C and other programming languages.

Chapter 2: System Calls (2) This chapter describes the access to services provided by the DG/UX kernel, including the C language interface and a description of returned error codes.

Volume 2 contains one chapter:

Chapter 3: Subroutines and Libraries (3) This chapter describes the available subroutines and subroutine libraries. Their binary versions reside in various system libraries in the directories `/lib` and `/usr/lib`. See `intro(3)` for descriptions of these libraries and the files in which they are stored. Although these man pages are alphabetized together, each has a letter associated with the number 3 indicating the pertinent library:

- 3C C Programming Language Libraries
- 3E ELF Library Routines
- 3G General Library Routines
- 3M Mathematical Library Routines
- 3N Networking Support Utilities
- 3R Remote Procedure Call Routines
- 3S Standard I/O Library Routines
- 3W Multinational Language Set (MNLS) Routines
- 3X Specialized Libraries

Volume 3 contains three chapters and one appendix:

Chapter 4: File Formats (4) This chapter documents the structure of particular kinds of files; for example, the format of the output of the link editor is given in `a.out(4)`. Excluded are files used by only one command (for example, the assembler's intermediate files). In general, the C language structures corresponding to these formats can be found in the directories `/usr/include` and `/usr/include/sys`.

Chapter 5: Miscellaneous Features (5) This chapter contains a variety of facilities. Included are descriptions of character sets, macro packages, and other things.

Chapter 6: Communications Protocols (6) This chapter contains a description of the `unix_ipc` communications facility.

Appendix A: Contents and Permuted Index Man Pages

These manual pages contain information extracted from the DG/UX man pages in all five reference volumes.

Man Page Format

Each man page has at least some of the following sections:

NAME	gives the primary name (and secondary names, as the case may be) and briefly states its purpose.
SYNOPSIS	summarizes the usage of the program being described.
DESCRIPTION	discusses how to use these commands.
EXAMPLES	gives examples of usage, where appropriate.
FILES	contains the file names that are referenced by the program.
EXIT CODES	discusses values set when the command terminates. The value set is available in the shell environment variable “?” (see <code>sh(1)</code>).
DIAGNOSTICS	discusses the error messages that may be produced. Messages that are intended to be self-explanatory are not listed.
SEE ALSO	offers pointers to related information.
NOTES	gives information that may be helpful under the particular circumstances described.

Some man pages may contain other heads such as **ENVIRONMENT** and **CAVEATS**.

Man Page Notation Conventions

This manual uses certain symbols and styles of type to indicate different meanings in man pages. Those symbol and typeface conventions are defined in the following list. You should familiarize yourself with these conventions before reading the manual.

The description of convention meanings uses the terms “command line,” “format line,” and “syntax line.” A command line is an example of a command string that you should type verbatim; it is preceded by a system prompt. A format line shows how to structure a command; it shows the variables that must be supplied and the available options. A syntax line is a fragment of program code that shows how to use a particular routine; some syntax lines contain variables.

Convention	Meaning
boldface	This font is used for section heads and subsection heads. It is also used to distinguish input from output in examples where the two are intermixed.
constant width/ monospace	In command formats and code syntax: This typeface indicates text (including punctuation) that you type verbatim from your keyboard. In text: This typeface is used for examples, code samples, pathnames, and the names of commands, files, directories, and manual pages. In all contexts: The following characters, which have special meanings explained below, do not have special meaning but simply represent themselves when they appear in constant-width font: < > [] { } . In constant-width font they are I/O redirection operators, brackets, braces, and the pipe symbol.
<i>italic</i>	In format lines: This font represents variables for which you supply values; for example, the names of your directories and files, your username and password, and possible arguments to commands.
[<i>optional</i>]	In format lines: Regular-font brackets surround an optional argument. Don't type the brackets; they only set off what is optional. These brackets should not be confused with constant-width brackets.
<i>choice1</i> <i>choice2</i>	In format lines: The vertical bar indicates a choice between <i>choice1</i> and <i>choice2</i> .
...	In format lines and syntax lines: You can repeat the preceding argument as many times as desired.
{ }	In format lines: These regular-font braces surround either two or more choices or syntax elements that are repeatable as a group.
< >	In command lines and other examples: Angle brackets distinguish a command sequence or a keystroke (such as <Ctrl-D>, <Esc>, and <3dw>) from surrounding text. Note that these angle brackets are in regular type and that you do not type them; there are, however, constant-width versions of these symbols that you do type.
\$, %, #	In command lines and other examples: These symbols represent the system command prompt symbols used for the Bourne and Korn shells, the C shell, and the superuser, respectively. Note that your system might use different symbols for the command prompts.

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative. A list of related documents appears at the end of this manual with the TIPS order form.

For a complete list of AViiON® and DG/UX™ manuals, see the *Guide to AViiON® and DG/UX™ System Documentation* (069-701085). The on-line version of this manual found in `/usr/release/doc_guide` contains the most current list.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the Data General Customer Support Center (CSC) by calling 1-800-DG-HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive FOCUS monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1-800-932-6663 or 1-508-443-3330.

End of Preface

Contents

Chapter 4 — File Formats

intro(4)	4-2
a.out(4)	4-3
acct(4)	4-9
aliases(4)	4-11
ar(4)	4-14
bootparams(4)	4-17
checklist(4)	4-18
compver(4)	4-19
copyright(4)	4-20
core(4)	4-21
cpio(4)	4-22
d_passwd(4)	4-23
depend(4)	4-24
dialups(4)	4-26
dirent(4)	4-27
dumptab(4)	4-28
ethers(4)	4-29
exports(4)	4-30
filehdr(4)	4-32
fs(4)	4-33
fspec(4)	4-39
fstab(4)	4-40
group(4)	4-43
hfm(4)	4-45
holidays(4)	4-47
hosts(4)	4-48
idl(4)	4-49
inittab(4)	4-68
inode(4)	4-71
issue(4)	4-76
ldfcn(4)	4-77
limits(4)	4-79
linenum(4)	4-81
master(4)	4-82
mfs(4)	4-85
mnttab(4)	4-87
netconfig(4)	4-89
netgroup(4)	4-92
networks(4)	4-93
passwd(4)	4-94
pkginfo(4)	4-97
pkgmap(4)	4-100
profile(4)	4-103
protocols(4)	4-104

prototype(4)	4-105
publickey(4)	4-108
rscfile(4)	4-109
reloc(4)	4-112
rpc(4)	4-113
sccsfile(4)	4-114
scr_dump(4)	4-117
sde-chooser(4)	4-118
sdetab(4)	4-119
services(4)	4-120
space(4)	4-121
statd(4)	4-122
strftime(4)	4-123
svcorder(4)	4-124
syms(4)	4-125
system(4)	4-128
terminfo(4)	4-129
timezone(4)	4-176
updaters(4)	4-179
utmp(4)	4-180
ypfiles(4)	4-182

Chapter 5 — Miscellaneous Features

intro(5)	5-2
ascii(5)	5-3
dg_mknod(5)	5-4
dg_stat(5)	5-6
elink(5)	5-9
environ(5)	5-11
eucioctl(5)	5-17
fcntl(5)	5-19
hier(5)	5-20
hostname(5)	5-25
langinfo(5)	5-26
legend(5)	5-28
math(5)	5-29
misalign(5)	5-30
nL_types(5)	5-33
printcap(5)	5-34
prof(5)	5-36
regex(5)	5-37
sde(5)	5-41
siginfo(5)	5-43
signal(5)	5-46
stat(5)	5-47
statfs(5)	5-49
stdarg(5)	5-51
syslog.conf(5)	5-53
tar(5)	5-55
termcap(5)	5-58
types(5)	5-72

ucontext(5)	5-73
ustat(5)	5-74
values(5)	5-75
varargs(5)	5-76
wstat(5)	5-78

Chapter 6 — Communications Protocols

intro(6)	6-2
dot3(6P)	6-5
inet(6F)	6-6
ip(6P)	6-7
nfs(6P)	6-8
snap(6P)	6-9
tcp(6P)	6-10
udp(6P)	6-12
unix_ipc(6F)	6-13

Appendix A — Contents and Permuted Index Man Pages

contents(0)	A-2
index(0)	A-23

Index

Related Documents

Tables

Table

4-1	Summary of TCP/IP File Format Manual Pages	4-1
4-2	Summary of ONC/NFS File Format Manual Pages	4-1
6-1	Summary of Communications Protocol Manual Pages	6-1

Chapter 4

File Formats

This chapter contains in printed form the online manual entries for DG/UX, TCP/IP, and NFS file formats. The entries are in alphabetical order except for `intro(4)`, which is first.

For other file format manual pages (4M), see the *System Manager's Reference for the DG/UX System*.

Table 4-1 lists the TCP/IP man pages included in this chapter.

Table 4-1 Summary of TCP/IP File Format Manual Pages

Name	Description
<code>aliases(4)</code>	Addresses and aliases for <code>sendmail</code>
<code>ethers(4)</code>	Ethernet address to hostname database or NIS domain
<code>hosts(4)</code>	Host name database
<code>networks(4)</code>	Network name database
<code>protocols(4)</code>	Table of protocols
<code>services(4)</code>	Service name database for DG/UX system
<code>svcborder(4)</code>	File specifying name/address resolution order

Table 4-2 lists the ONC/NFS man pages included in this chapter.

Table 4-2 Summary of ONC/NFS File Format Manual Pages

Name	Description
<code>bootparams(4)</code>	Boot parameter database
<code>exports(4)</code>	Directories to export to NFS clients
<code>netgroup(4)</code>	List of network groups
<code>publickey(4)</code>	Public key database
<code>rpc(4)</code>	RPC program number database
<code>statd(4)</code>	<code>statd</code> directories and file structures
<code>updaters(4)</code>	Configuration file for updating
<code>ypfiles(4)</code>	The NIS database and directory structure

NAME

intro - introduction to file formats

DESCRIPTION

This section outlines the formats of various files. The C structure declarations for the file formats are given where applicable. Usually, the header files containing these structure declarations can be found in the directories `/usr/include` or `/usr/include/sys`. For inclusion in C language programs, however, the syntax `#include <filename.h>` or `#include <sys/filename.h>` should be used.

SEE ALSO

intro(4M).

NAME

a.out – assembler and link editor output

SYNOPSIS

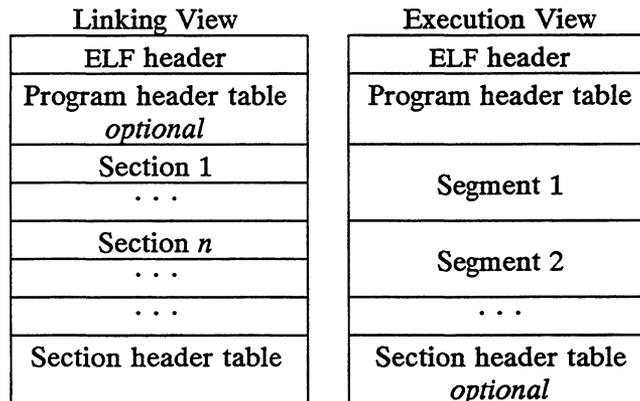
```
#include <elf.h>          /* for ELF executables*/
#include <a.out.h> /* for COFF executables */
```

DESCRIPTION

The filename a.out is the default output filename from the link editor ld(1). The link editor will make a.out executable if there were no errors in linking. The output file of the assembler, as(1), also follows the common object file format of the a.out file although the default filename is different.

ELF (Executable and Linking Format) Files

Programs that manipulate ELF files may use the library that elf(3E) describes. An overview of the file format follows. For more complete information, see the references given below.



An ELF header resides at the beginning and holds a “road map” describing the file’s organization. Sections hold the bulk of object file information for the linking view: instructions, data, symbol table, relocation information, and so on. Segments hold the object file information for the program execution view. As shown, a segment may contain one or more sections.

A program header table, if present, tells the system how to create a process image. Files used to build a process image (execute a program) must have a program header table; relocatable files do not need one. A section header table contains information describing the file’s sections. Every section has an entry in the table; each entry gives information such as the section name, the section size, etc. Files used during linking must have a section header table; other object files may or may not have one.

Although the figure shows the program header table immediately after the ELF header, and the section header table following the sections, actual files may differ. Moreover, sections and segments have no specified order. Only the ELF header has a fixed position in the file.

When an a.out file is loaded into memory for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized, the latter actually being initialized to all 0’s), and a stack. The text segment is not writable by the program; if other processes are executing the same a.out file, the processes will share a single text segment.

The data segment starts at the next maximal page boundary past the last text address. (If the system supports more than one page size, the “maximal page” is the largest

supported size.) When the process image is created, the part of the file holding the end of text and the beginning of data may appear twice. The duplicated chunk of text that appears at the beginning of data is never executed; it is duplicated so that the operating system may bring in pieces of the file in multiples of the actual page size without having to realign the beginning of the data section to a page boundary. Therefore, the first data address is the sum of the next maximal page boundary past the end of text plus the remainder of the last text address divided by the maximal page size. If the last text address is a multiple of the maximal page size, no duplication is necessary. The stack is automatically extended as required. The data segment is extended as requested by the `brk(2)` system call.

COFF (Common Object File Format) Files

A common object file consists of a file header, a UNIX system header (if the file is link editor output), a table of section headers, relocation information, (optional) line numbers, a symbol table, and a string table. The order is given below:

```

File header.
UNIX system header.
Section 1 header.
...
Section n header.
Section 1 data.
...
Section n data.
Section 1 relocation.
...
Section n relocation.
Section 1 line numbers.
...
Section n line numbers.
Symbol table.
String table.
```

The last three parts of an object file (line numbers, symbol table and string table) may be missing if the program was linked with the `-s` option of `ld(1)` or if they were removed by `strip(1)`. Also note that the relocation information will be absent after linking unless the `-r` option of `ld(1)` was used. The string table exists only if the symbol table contains symbols with names longer than eight characters.

The sizes of each section (contained in the header, discussed below) are in bytes.

When an `a.out` file is loaded into memory for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized, the latter actually being initialized to all 0's), and a stack. On the M88K computer the text segment typically starts at location `0x00010000` plus the byte offset in the `a.out` file of the text section data.

The first 16 bits of `a.out` files is the magic number. For non-executable `a.out` files and executables linked in the `m88kbc` SDE, the magic number is 0555. For executables linked in the `dgux` SDE, the magic number is 0541. See `sde(1)`. The optional header of an `a.out` file produced by `ld(1)` also has a magic number whose value is 0413. The headers (file header, optional header, and section headers) appear at the beginning of `a.out` files and determine the address of the text segment when it is loaded into memory. The first text address will equal `0x00010000` plus the size of the headers, and will vary depending upon the number of section headers in the `a.out`

file. In an `a.out` file with three sections (`.text`, `.data`, and `.bss`), the first text address is at `0x000100B8` on the M88K computer. The text segment is not writable by the program; if other processes are executing the same `a.out` file, the processes will share a single text segment.

On the M88K computer the stack begins at location `0xF000000` and grows toward lower addresses. The stack is automatically extended as required. The data segment is extended only as requested by the `brk(2)` system call.

For relocatable files the value of a word in the text or data portions that is not a reference to an undefined external symbol is exactly the value that will appear in memory when the file is executed. If a word in text or data involves a reference to an undefined external symbol, there will be a relocation entry for the word, the storage class of the symbol-table entry for the symbol will be marked as an “external symbol”, and the value and section number of the symbol-table entry will be undefined. When the file is processed by the link editor and the external symbol becomes defined, the value of the symbol will be added to the word in the file.

The format of the `filehdr` header is

```
struct filehdr
{
    unsigned short  f_magic;      /* magic number */
    unsigned short  f_nscns;     /* number of sections */
    long            f_timdat;    /* time and date stamp */
    long            f_symptr;    /* file ptr to symtab */
    long            f_nsyms;     /* # symtab entries */
    unsigned short  f_opthdr;    /* sizeof(opt hdr) */
    unsigned short  f_flags;     /* flags */
};
```

The format of the optional header is

```
typedef struct aouthdr
{
    short          magic;        /* magic number */
    short          vstamp;      /* version stamp */
    long           tsize;        /* text size in bytes, padded */
    long           dsize;        /* initialized data (.data) */
    long           bsize;        /* uninitialized data (.bss) */
    long           entry;        /* entry point */
    long           text_start;   /* base of text used for this file */
    long           data_start;   /* base of data used for this file */
} AOUTHDR;
```

The format of the section header is

```
struct scnhdr
{
    char          s_name[8]; /* section name */
    long         s_paddr;   /* physical address */
    long         s_vaddr;   /* virtual address */
    long         s_size;    /* section size */
    long         s_scnptr;  /* file ptr to raw data */
    long         s_relptr;  /* file ptr to relocation */
    long         s_lnnoptr; /* file ptr to line numbers */
    unsigned long s_nreloc; /* # reloc entries */
    unsigned long s_nlnno; /* # line number entries */
    long         s_flags;   /* flags */
};
```

Object files have one relocation entry for each relocatable reference in the text or data. If relocation information is present, it will be in the following format:

```
struct reloc
{
    long         r_vaddr;   /* (virtual) address of reference */
    long         r_symndx;  /* index into symbol table */
    unsigned short r_type;  /* relocation type */
    unsigned short r_offset; /* high 16 bits of expression */
};
```

The start of the relocation information is *s_relptr* from the section header. If there is no relocation information, *s_relptr* is 0.

The format of each symbol in the symbol table is

```
#define SYMNMLEN 8
#define FILNMLEN 14
#define DIMNUM 4

struct syment
{
    union /* all ways to get a symbol name */
    {
        char          _n_name[SYMNMLEN]; /* name of symbol */
        struct
        {
            long      _n_zeroes; /* == 0L if in string table */
            long      _n_offset; /* location in string table */
        } _n_n;
        char          *_n_nptr[2]; /* allows overlaying */
    } _n;
    long             n_value; /* value of symbol */
    short           n_scnnum; /* section number */
    unsigned short  n_type;   /* type and derived type */
    char            n_sclass; /* storage class */
    char            n_numaux; /* number of aux entries */
    char            n_pad1;   /* pad to 4 byte multiple */
    char            n_pad2;   /* pad to 4 byte multiple */
};
```

```

#define n_name      _n._n_name
#define n_zeroes   _n._n.n._n_zeroes
#define n_offset   _n._n.n._n_offset
#define n_nptr     _n._n_nptr[1]

```

Some symbols require more information than a single entry; they are followed by *auxiliary entries* that are the same size as a symbol entry. The format follows:

```

union auxent {
    struct {
        long x_tagndx;
        union {
            struct {
                unsigned longx_lnno;
                unsigned longx_size;
            } x_lnsz;
            long x_fsize;
        } x_misc;
        union {
            struct {
                long x_lnnoptr;
                long x_endndx;
            } x_fcn;
            struct {
                unsigned shortx_dimen[4];
            } x_ary;
            struct {
                unsigned long x_dimen1[2];
            } x_ary1;
        } x_fcary;
        unsigned short x_tvndx;
        char x_pad1;
        char x_pad2;
    } x_sym;

    struct {
        unsigned long x_dimen2[5];
    } x_ary2;

    union {
        char x_fname[FILNMLEN];
        struct {
            long _x_zeroes; /* 0 if name is in string table*/
            long _x_offset; /* offset into string table */
        } _x_x;
        char *_x_xptr[2]; /* allows for overlaying */
    } x_file;
} x_file;

struct {
    long x_scrlen;

```

```
        unsigned short  x_nreloc;
        unsigned short  x_nlinno;
    } x_scn;

    struct {
        long             x_tvfill;
        unsigned short   x_tvlen;
        unsigned short   x_tvran[2];
    } x_tv;
};
```

Indexes of symbol table entries begin at *zero*. The start of the symbol table is *f_symptr* (from the file header) bytes from the beginning of the file. If the symbol table is stripped, *f_symptr* is 0. The string table (if one exists) begins at *f_symptr* + (*f_nsyms* * SYMESZ) bytes from the beginning of the file.

SEE ALSO

as(1), att_dump(1), cc(1), ld(1), ld-coff(1), brk(2), elf(3E), filehdr(4), ldfcn(4), linenum(4), reloc(4), syms(4).

The “Object Files” chapter in the *Programmer’s Guide: ANSI C and Programming Support Tools*.

NAME

acct - per-process accounting file format

SYNOPSIS

```
#include <sys/acct.h>
```

DESCRIPTION

Files produced as a result of calling acct(2) have records in the form defined by <sys/acct.h>, whose contents are:

```
typedef  ushort comp_t;  /* "floating point" */
                          /* 13-bit fraction, 3-bit exponent */

struct  acct
{
    char    ac_flag;      /* Accounting flag */
    char    ac_stat;      /* Exit status */
    ushort  ac_uid;       /* Accounting user ID */
    ushort  ac_gid;       /* Accounting group ID */
    dev_t   ac_tty;       /* control typewriter */
    time_t  ac_btime;     /* Beginning time */
    comp_t  ac_untime;    /* acctng user time in clock ticks */
    comp_t  ac_stime;     /* acctng system time in clock ticks */
    comp_t  ac_etime;     /* acctng elapsed time in clock ticks */
    comp_t  ac_mem;       /* memory usage in kbytes */
    comp_t  ac_io;        /* chars trnsfrd by read/write */
    comp_t  ac_rw;        /* number of block reads/writes */
    char    ac_comm[8];   /* command name */
};
```

Also defined are the following symbolic names:

```
AFORK /* has executed fork, but no exec */ ASU /* used super-
user privileges */ ACCTF /* record type: 00 = acct */
```

In *ac_flag*, the AFORK flag is turned on by each fork(2) and turned off by an exec(2). The *ac_comm* field is inherited from the parent process and is reset by any exec. Each time the system charges the process with a clock tick, it also adds to *ac_mem* the current process size, computed as follows:

$$(\text{data size}) + (\text{text size}) / (\text{number of in-core processes using text})$$

The value of $ac_mem / (ac_stime + ac_untime)$ can be viewed as an approximation to the mean process size, as modified by text-sharing.

The structure `tacct.h`, which resides with the source files of the accounting commands, represents the total accounting format used by the various accounting commands:

```

/*
 * total accounting (for acct period), also for day
 */

struct tacct {
    uid_t      ta_uid;      /* userid */
    char       ta_name[8]; /* login name */
    float      ta_cpu[2];  /* cum. cpu time, p/np (mins) */
    float      ta_kcore[2]; /* cum kcore-minutes, p/np */
    float      ta_con[2];  /* cum. connect time, p/np, mins */
    float      ta_du;      /* cum. disk usage */
    long       ta_pc;      /* count of processes */
    unsigned short ta_sc;  /* count of login sessions */
    unsigned short ta_dc;  /* count of disk samples */
    unsigned short ta_fee; /* fee for special services */
};

```

SEE ALSO

`acct(2)`, `exec(2)`, `fork(2)`.

`acct(1M)` in the *System Manager's Reference for the DG/UX System*.

`acctcom(1)` in the *User's Reference for the DG/UX System*.

NOTES

The `ac_mem` value for a short-lived command gives little information about the actual size of the command because `ac_mem` may be incremented while a different command (like the shell) is being executed by the process.

NAME

aliases – addresses and aliases for sendmail

DESCRIPTION

These files contain mail addresses or aliases, recognized by `sendmail(1M)`, for the local host:

<code>/etc/passwd</code>	Mail addresses (usernames) of local users.
<code>/etc/aliases</code>	Aliases for the local host, in ASCII format. This file can be edited to add, update, or delete local mail aliases.
<code>/etc/aliases.{dir,pag}</code>	The aliasing information from <code>/etc/aliases</code> , in binary, <code>dbm(3X)</code> format for use by <code>sendmail(1M)</code> . The program <code>newaliases</code> maintains these files.
<code>~/forward</code>	Addresses to which a user's mail is forwarded.
<code>mail.aliases</code>	If you are running <code>ONC/NFS</code> , this Network Information Service (NIS) aliases map contains addresses and aliases available for use across the network.

As distributed, `sendmail(1M)` supports the following types of mail addresses:

- Local usernames. These are listed in the local host's `/etc/passwd` file.
- Local filenames. When mailed to an absolute pathname, a message can be appended to a file.
- Commands. If the first character of the address is a vertical bar, (`|`), `sendmail(1M)` pipes the message to the standard input of the command the bar precedes.
- Internet mail addresses of the form:

name@domain

If *domain* does not contain any '.' (dots), then it is interpreted as the name of a host in the current domain. Otherwise, the message is passed to a *mailhost* that determines how to get to the specified domain. Domains are divided into subdomains that are separated by dots, with the top-level domain on the right. Top-level domains include the following:

<code>.com</code>	Commercial organizations.
<code>.edu</code>	Educational organizations.
<code>.gov</code>	Government organizations.
<code>.mil</code>	Military groups.
<code>.org</code>	Other organizations.

For example, the full address of K. Owen could be:

`owen@cs.unc.edu`

if he can be reached through the subdomain named "cs" at the University of North Carolina.

- `uucp(1)` addresses of the form:

... [*host!*] *host!username*

Addresses such as these are sometimes referred to as "Usenet" addresses. `uucp(1)` provides links to numerous sites throughout the world for the remote copying of files.

Other site-specific forms of addressing can be added by customizing the `sendmail` configuration file. See the `sendmail(1M)` man page and "Configuring and Using `sendmail`" in *Managing TCP/IP on the DG/UX System* for details. Standard addresses are recommended.

The `/etc/aliases` file is formatted as a series of lines of the form

```
aliasname: address[, address]
```

aliasname is the name of the alias or alias group, and *address* is the address of a recipient in the group. Aliases can be nested. That is, an *address* can be the name of another alias group. Because of the way `sendmail` performs mapping from uppercase to lowercase, an *address* that is the name of another alias group must not contain any uppercase letters.

Lines beginning with white space are treated as continuation lines for the preceding alias. Lines beginning with `#` are comments.

Given an alias of the following form:

```
aliasname: address, address, address
```

an alias such as the following:

```
owner-aliasname: erraddress
```

directs error-messages resulting from mail to *aliasname* to *erraddress*, instead of back to the person who sent the message.

An alias of the form:

```
aliasname: :include:pathname
```

with colons as shown, adds the recipients listed in the file *pathname* to the *aliasname* alias. This allows a private list to be maintained separately from the aliases file.

When an alias (or address) is resolved to the name of a user on the local host, `sendmail` checks for a `.forward` file, owned by the intended recipient, in that user's home directory, and with universal read access. This file can contain one or more addresses or aliases as described above, each of which is sent a copy of the user's mail.

Care must be taken to avoid creating addressing loops in the `.forward` file. (See "ONC/NFS-specific Information" below for additional information specific to ONC/NFS.)

A backslash before a username in the `.forward` file inhibits further aliasing. Suppose user `owen` had the following `.forward` file:

```
Postmaster
\owen
```

Mail for `owen` will be redirected to `Postmaster`, but a copy also is sent to `owen`. The `sendmail` program will not alias a username following the backslash.

ONC/NFS-specific Information

If you are running ONC/NFS, the following information applies in addition to the `mail.aliases` file cited above:

Normally, the aliases file on the master NIS server is used for the `mail.aliases` NIS map, which can be made available to every NIS client. Thus, the `/etc/aliases*` files on the various hosts in a network will be largely used to provide host specific aliases. Domain-wide aliases should ultimately be resolved into usernames on specific hosts. For example, if the following were in the domain-wide alias file:

```
mlee:ml@mlmachine
```

then any NIS client could just mail to `mlee` and not have to remember the machine and username for Mike Lee.

When forwarding mail between machines, be sure that the destination machine does not return the mail to the sender through the operation of any NIS aliases. Otherwise, copies of the message may “bounce.” Usually, the solution is to change the NIS alias to direct mail to the proper destination.

FILES

```
/etc/passwd  
/etc/aliases  
/etc/aliases.dir  
/etc/aliases.pag  
~/.forward
```

SEE ALSO

`uucp(1)`, `dbm(3X)`, `sendmail(1M)`.

BUGS

Because of restrictions in `dbm(3X)` a single alias cannot contain more than about 1000 characters. Nested aliases can be used to circumvent this limit.

NAME

ar – DG/UX common archive file format

DESCRIPTION

The archive command `ar` is used to combine several files into one. Archives are used mainly as libraries to be searched by the link editor `ld`.

Each archive begins with the archive magic string.

```
#define ARMAG "!<arch>\n" /* magic string */
#define SARMAG 8 /* length of magic string */
```

Following the archive magic string are the archive file members. Each file member is preceded by a file member header which is of the following format:

```
#define ARFMAG "`\n" /* header trailer string */

struct ar_hdr /* file member header */
{
    char ar_name[16]; /* '/' terminated file member name */
    char ar_date[12]; /* file member date */
    char ar_uid[6]; /* file member user identification */
    char ar_gid[6]; /* file member group identification */
    char ar_mode[8]; /* file member mode (octal) */
    char ar_size[10]; /* file member size */
    char ar_fmag[2]; /* header trailer string */
};
```

All information in the file member headers is in printable ASCII. The numeric information contained in the headers is stored as decimal numbers (except for `ar_mode` which is in octal). Thus, if the archive contains printable files, the archive itself is printable.

If the file member name fits, the `ar_name` field contains the name directly, and is terminated by a slash (/) and padded with blanks on the right. If the member's name does not fit, `ar_name` contains a slash (/) followed by a decimal representation of the name's offset in the archive string table described below.

The `ar_date` field is the modification date of the file at the time of its insertion into the archive. Common format archives can be moved from system to system as long as the portable archive command `ar` is used.

Each archive file member begins on an even byte boundary; a newline is inserted between files if necessary. Nevertheless, the size given reflects the actual size of the file exclusive of padding.

Notice there is no provision for empty areas in an archive file.

Each archive that contains object files [see `a.out(4)`] includes an archive symbol table. This symbol table is used by the link editor `ld` to determine which archive members must be loaded during the link edit process. The archive symbol table (if it exists) is always the first file in the archive (but is never listed) and is automatically created and/or updated by `ar`.

The archive symbol table has a zero length name (i.e., `ar_name[0]` is '/'), `ar_name[1]`==' ', etc.). All "words" in this symbol table have four bytes, using the machine-independent encoding shown below. (All machines use the encoding

described here for the symbol table, even if the machine's "natural" byte order is different.)

0x01020304	0	1	2	3
	01	02	03	04

The contents of this "file" are as follows:

1. The number of symbols. Length: 4 bytes.
2. The array of offsets into the archive file. Length: 4 bytes * "the number of symbols".
3. The name string table. Length: *ar_size* - 4 bytes * ("the number of symbols" + 1).

As an example, the following symbol table defines 4 symbols. The archive member at file offset 114 defines `name` and `object`. The archive member at file offset 426 defines `function` and a second version of `name`.

Offset	+0	+1	+2	+3	
0	4				4 offset entries
4	114				name
8	114				object
12	426				function
16	426				name
20	n	a	m	e	
24	\0	o	b	j	
28	e	c	t	\0	
32	f	u	n	c	
36	t	i	o	n	
40	\0	n	a	m	
44	e	\0			

The number of symbols and the array of offsets are managed with `sget1` and `sput1`. The string table contains exactly as many null terminated strings as there are elements in the offsets array. Each offset from the array is associated with the corresponding name from the string table (in order). The names in the string table are all the defined global symbols found in the common object files in the archive. Each offset is the location of the archive header for the associated symbol.

If some archive member's name is more than 15 bytes long, a special archive member contains a table of file names, each followed by a slash and a new-line. This string table member, if present, will precede all "normal" archive members. The special archive symbol table is not a "normal" member, and must be first if it exists. The *ar_name* entry of the string table's member header holds a zero length name `ar_name[0]=='/'`, followed by one trailing slash (`ar_name[1]=='/'`), followed by blanks (`ar_name[2]==' '`, etc.). Offsets into the string table begin at zero. Example *ar_name* values for short and long file names appear below.

Offset	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
0	f	i	l	e	_	n	a	m	e	_
10	s	a	m	p	l	e	/	\n	l	o
20	n	g	e	r	f	i	l	e	n	a
30	m	e	x	a	m	p	l	e	/	\n

Member Name	<i>ar_name</i>	Note
short-name	short-name/	Not in string table
file_name_sample	/0	Offset 0 in string table
longerfilenameexample	/18	Offset 18 in string table

SEE ALSO

ar(1), ld(1), strip(1), sputl(3X), a.out(4).

NOTES

strip will remove all archive symbol entries from the header. The archive symbol entries must be restored via the `-ts` options of the `ar` command before the archive can be used with the link editor `ld`.

NAME

bootparams – boot parameter data base

SYNOPSIS

/etc/bootparams

DESCRIPTION

The bootparams file contains the list of client entries that diskless clients use for booting. For each diskless client the entry should contain the following information:

- name of client
- a list of keys, names of servers, and pathnames.

The first item of each entry is the name of the diskless client. The subsequent item is a list of keys, names of servers, and pathnames.

Items are separated by TAB characters. A line-continuation character () can be used, but it must be preceded by TAB or SPACE characters (see EXAMPLE).

EXAMPLE

Here is an example of the /etc/bootparams file:

```
myclient    root=myserver:/srv/release/PRIMARY/root/myhost \  
            swap=myserver:/srv/release/PRIMARY/swap/myhost \  
            dump=myserver:/srv/release/PRIMARY/dump/myhost
```

Root specifies the pathname of the executable file to boot. This file must exist to boot the client. Swap gives the pathname of the swap area file. The swap file is a fixed-sized file that must be pre-allocated to an appropriate size. Dump specifies the pathname of the system dump file, where system information is written following a system crash. This file must exist to dump the system crash information. During a system crash, this entry is optional for DG/UX clients; however, non-DG/UX clients may require it. If there is no dump entry, an attempted dump will fail.

FILES

/etc/bootparams

SEE ALSO

bootparamd(1M).

NAME

checklist – list of file systems processed by fsck and ncheck

DESCRIPTION

Checklist may reside in directory `/etc` and contain a list of special file names. Each special file name is contained on a separate line and corresponds to a file system. Each file system will then be automatically processed by the `fsck(1M)` and `ncheck(1M)` commands. You have to create the `checklist` file yourself; the system does not create it for you.

If you have your special files in `fstab`, you do not need to create a `checklist` file to get `fsck` to process them.

SEE ALSO

`fsck(1M)` and `ncheck(1M)` in the *System Manager's Reference for the DG/UX System*.
`fstab(4)`.

NAME

compver – compatible versions file

DESCRIPTION

compver is an ASCII file used to specify previous versions of the associated package which are upward compatible. It is created by a package developer.

Each line of the file specifies a previous version of the associated package with which the current version is backward compatible.

Since some packages may require installation of a specific version of another software package, compatibility information is extremely crucial. Consider, for example, a package called "A" which requires version "1.0" of application "B" as a prerequisite for installation. If the customer installing "A" has a newer version of "B" (version 1.3), the compver file for "B" must indicate that "1.3" is compatible with version "1.0" in order for the customer to install package "A".

NOTES

The comparison of the version string disregards white space and tabs. It is performed on a word-by-word basis. Thus "Version 1.3" and "Version 1.3" would be considered the same.

EXAMPLE

A sample compver file is shown below.

```
Version 1.3  
Version 1.0
```

SEE ALSO

pkginfo(4).

NAME

copyright - copyright information file

DESCRIPTION

copyright is an ASCII file used to provide a copyright notice for a package. The text may be in any format. The full file contents (including comment lines) is displayed on the terminal at the time of package installation.

SEE ALSO

pkginfo(4).

NAME

core – format of core image file

DESCRIPTION

The system writes out a core image of a terminated process when any of several errors occur. See `signal(2)` for the list of reasons; the most common are memory violations, illegal instructions, and user-generated quit signals. The core image is called `core` and is written in the process's working directory (if possible; normal access controls apply). A process with an effective user id different from the real user id will not produce a core image.

The first section of the core image is a copy of the system's per-user data for the process, including the registers as they were at the time of the fault. The remainder represents the actual contents of the user's core area when the core image was written. The text segment is not dumped.

The format of the information in the first section is described by the `user` structure of the system, defined in `/usr/include/sys/user.h`.

SEE ALSO

`sdb(1)`, `dbx(1)`, `setuid(2)`, `signal(2)`.
`crash(1M)` in the *System Manager's Reference for the DG/UX System*.

NAME

cpio - format of cpio archive

DESCRIPTION

The header structure, when the `-c` option of `cpio(1)` is not used, is:

```
struct {
    short    h_magic,
            h_dev;
    ushort  h_ino,
            h_mode,
            h_uid,
            h_gid;
    short    h_nlink,
            h_rdev,
            h_mtime[2],
            h_namesize,
            h_filesize[2];
    char     h_name[h_namesize rounded to word];
} Hdr;
```

When the `-c` option is used, the header information is described by:

```
sscanf(Chdr, "%6o%6o%6o%6o%6o%6o%6o%6o%6o%11lo%6o%11lo%s",
        &Hdr.h_magic, &Hdr.h_dev, &Hdr.h_ino, &Hdr.h_mode,
        &Hdr.h_uid, &Hdr.h_gid, &Hdr.h_nlink, &Hdr.h_rdev,
        &Longtime, &Hdr.h_namesize, &Longfile, Hdr.h_name);
```

Longtime and *Longfile* are equivalent to *Hdr.h_mtime* and *Hdr.h_filesize*, respectively. The contents of each file are recorded in an element of the array of varying length structures, *archive*, with other items describing the file. Every instance of *h_magic* contains the constant 070707 (octal). The items *h_dev* through *h_mtime* have meanings explained in `stat(2)`. The length of the null-terminated path name *h_name*, including the null byte, is given by *h_namesize*.

The last record of the *archive* always contains the name TRAILER!!!. Special files, directories, and the trailer are recorded with *h_filesize* equal to zero.

SEE ALSO

`stat(2)`.
`cpio(1)`, `find(1)` in the *User's Reference for the DG/UX System*.

NAME

d_passwd – log-in programs and passwords for dial-up devices

SYNOPSIS

/etc/d_passwd

DESCRIPTION

This file contains an entry for programs (such as shells) that `login(1)` can invoke for users logging into the system via dial-up devices. Each entry includes the pathname of the shell program for which a dialup password is required and the encrypted password that the user must provide in order to invoke the program. You have to create a `d_passwd` file yourself; the system does not create one for you.

A dial-up device is any device that has an entry in the `/etc/dialups` file. See `dialups(4)`. You have to create a `dialups` file yourself; the system does not create one for you.

When a user logs into a dial-up device, `login` searches the `d_passwd` file to see if it contains an entry for the shell program specified in the user's `passwd` entry. If such an entry is found, `login` requires that the user provide a second ("dial-up") password in addition to their personal password. The program name in the user's `passwd` entry and the program name in the `d_passwd` file must match exactly. E.g., `/bin/csh` and `/usr/bin/csh` will not be matched even though they reference the same file.

The program `/usr/bin/sh` is treated as a special case. If `d_passwd` contains an entry for `/usr/bin/sh`, the password for that entry will be used as the default dial-up password for all users whose `passwd` shell program doesn't match any of the other `d_passwd` entries. In the case where no matching entry is found for a user and no `/usr/bin/sh` entry exists, the user is not prompted for a dial-up password.

Here is a sample `d_passwd` entry:

```
/bin/csh:xxxxxx:
```

where `xxxxxx` is the encrypted password.

SEE ALSO

`login(1)`, `dialups(4)`.

NAME

depend – software dependencies files

DESCRIPTION

depend is an ASCII file used to specify information concerning software dependencies for a particular package. The file is created by a software developer.

Each entry in the depend file describes a single software package. The instance of the package is described after the entry line by giving the package architecture and/or version. The format of each entry and subsequent instance definition is:

```

type pkg name
    (arch)version
    (arch)version
    ...

```

The fields are:

<i>type</i>	Defines the dependency type. Must be one of the following characters:
	<ul style="list-style-type: none"> P Indicates a prerequisite for installation, for example, the referenced package or versions must be installed. I Implies that the existence of the indicated package or version is incompatible. R Indicates a reverse dependency. Instead of defining the package's own dependencies, this designates that another package depends on this one. This type should be used only when an old package does not have a depend file but it relies on the newer package nonetheless. Therefore, the present package should not be removed if the designated old package is still on the system since, if it is removed, the old package will no longer work.
<i>pkg</i>	Indicates the package abbreviation.
<i>name</i>	Specifies the full package name.
<i>(arch)version</i>	Specifies a particular instance of the software. A version name cannot begin with a left parenthesis. The instance specifications, both <i>arch</i> and <i>version</i> , are completely optional but must each begin on a new line that begins with white space. A null version set equates to any version of the indicated package.

EXAMPLE

Here is a sample depend file:

```

I msvr 3B2 Messaging Server
P ctc Cartridge Tape Utilities
P dfm Directory and File Management Utilities
P ed Editing Utilities
P ipc Inter-Process Communication Utilities
P lp Line Printer Spooling Utilities
P shell Shell Programming Utilities
P sys System Header Files
    Release 3.0
P sysadm System Administration Utilities
P term Terminal Filters Utilities

```

P terminfo Terminal Information Utilities
P usrenv User Environment Utilities
P uucp Basic Networking Utilities
P x25 X.25 Network Interface
 Issue 1 Version 1
 Issue 1 Version 2
P windowing AT&T Windowing Utilities
 (3B2)Version 1
R cms 3B2 Call Management System

SEE ALSO

pkginfo(4).

NAME

dialups - devices requiring a dial-up password.

SYNOPSIS

/etc/dialups

DESCRIPTION

This file contains the pathnames of devices that require an additional password, called a dial-up password, from users who attempt to log into it. An example entry might be /dev/tty16. For such devices, the `login(1)` command prompts the user for the dial-up password after the user has provided a valid log-in name and personal password.

Dial-up passwords must appear in the `/etc/d_passwd` file along with the programs (such as a shell) that `login` will execute after a successful log-in at the given device.

You have to create the `dialups` and `d_passwd` files yourself; the system does not create them for you.

SEE ALSO

`login(1)`, `d_passwd(4)`.

`fsck(1M)` will attempt to correct the following fields if they are invalid:

`fname` is used by `statfs(2)`, `fstatfs(2)`, `labelit(1M)`, `volcopy(1M)`, `frec(1M)`, Initialized to zeros, when used it is considered an ASCII string not necessarily terminated by a NULL byte.

`fpack` is used by `statfs(2)`, `fstatfs(2)`, `labelit(1M)`, `volcopy(1M)`, `frec(1M)`, Initialized to zeros, when used it is considered an ASCII string not necessarily terminated by a NULL byte.

The following exponent fields pertain to the size of elements used to access user data blocks. Data elements are equal sized sets of contiguous blocks of a file. These data elements are either pointed to directly from the file node or indirectly through an index structure. Index elements are arrays of block numbers. The index structure is hierarchical; an index block number may point to another index element or, if the bottom is reached, point to a data element. The direct or indexed access of data elements depends on the size of the file and the block being accessed; blocks at the beginning of the file can be accessed through the direct access to provide faster access for smaller files since they are generally more common. The following fields control the sizes of these elements, allowing the user to choose values more suitable for the types of files that will typically fill the file system. For more information about data access from the inode, see `inode(4)`.

`default_des_exponent` specifies the default data element size for non-directory files. The default data element size in blocks is 2 raised to the `default_des_exponent` power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_DES_EXPONENT 4
```

The maximum value is:

```
#define DF_MAX_DES_EXPONENT 31
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

`default_ies_exponent` specifies the default index element size for non-directory files. The default index element size in blocks is 2 raised to the `default_ies_exponent` power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_IES_EXPONENT 0
```

The maximum value is:

```
#define DF_MAX_IES_EXPONENT 15
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

default_dir_des_exponent specifies the default data element size for directories and CPDs. The default data element size in blocks is 2 raised to the *default_dir_des_exponent* power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_DES_EXPONENT 4
```

The maximum value is:

```
#define DF_MAX_DES_EXPONENT 31
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

default_dir_ies_exponent specifies the default index element size for directories and CPDs. The default index element size in blocks is 2 raised to the *default_dir_ies_exponent* power. The default value for this field is:

```
#define DF_DEFAULT_DEFAULT_IES_EXPONENT 0
```

The maximum value is:

```
#define DF_MAX_IES_EXPONENT 15
```

although it is also limited to the base 2 logarithm of the largest power of two that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar).
```

fs_size is the number of blocks in the filesystem. *fsck(1M)* will check this against the disk size as reported by the device driver.

space_used is the total (user and system) space used on this filesystem, including any space wasted at the end due to an incomplete DAR.

number_of_used_file_nodes is the number of file nodes used in the file system, not including the wasted file nodes with node numbers 0 and 1.

first_anniversary is the first anniversary of each file in blocks. When a file first consumes this much space, the filesystem should change the DAR from which it gets space for the file. The minimum value of this field is 2 raised to the *default_des_exponent* power; the default value is:

```
#define DF_DEFAULT_FIRST_ANNIVERSARY(dar_size)
```

second_anniversary the second anniversary of each file in blocks. A file should change the DAR from which the filesystem gets space each time its space utilization crosses a multiple of the second anniversary. The second anniversary must be greater than or equal to the first anniversary. The default value of this field is:

```
#define DF_DEFAULT_SECOND_ANNIVERSARY(dar_size)
```

first_log_lda and *second_log_lda* give the logical disk address of the two halves of the fast recovery log. They will be zero if the file system was not mounted for fast recovery when the filesystem was last mounted or if /f4fsck/fP has been run over the file system.

log_size is the size in 512-byte blocks of each half of the fast recovery log.

shrink_operation_in_progress is set if the filesystem is in the process of being shrunk.

grow_operation_in_progress is set if the filesystem is in the process of being grown.

The Disk Allocation Region (DAR)

The DAR is similar to the BSD cylinder group; however, the DAR is not necessarily associated with a physical disk cylinder as it is in BSD. The purpose of the DAR is to spread files throughout the filesystem while maintaining a locality between inodes and the data blocks associated with them.

The DAR consists of three parts: a bitmap, a file node table, and the data blocks allocated to files as they are needed.

The bitmap records the space allocation in the DAR. A bit in the bitmap represents a block in the DAR (this includes the blocks allocated for the bitmap and the file node table). If the bitmap value is 1, it is used; otherwise, it is free. The size of the bitmap is a function of the size of the DAR and is provided (in blocks) by:

```
#define DF_DAR_BITMAP_SIZE(dar_size)
```

The file node table contains entries for each file in the DAR. A file node entry (called an inode) contains information about the file. The first block of the table is after the bitmap. The number of file nodes in the DAR is a field in the FMIA. The number of blocks allocated to the table (in blocks) is:

```
#define DF_DAR_FILE_NODE_TABLE_SIZE(file_nodes_per_dar)
```

The file node table element (the inode) is discussed in `inode(4)`.

The data blocks take up the remaining blocks of the DAR.

With the exception of the blocks of the DAR Information Area and the Secondary FMIA, all blocks in the file system are contained in DAR's. The number of DAR's in a file system is a function of the size of the file system, the size of each DAR, and the file nodes contained in each DAR. This is provided by:

```
#define DF_NUMBER_OF_DARS(fs_size, dar_size, nodes_per_dar)
```

The last DAR of the file system may be the smaller than the other DAR's. If the space before the DAR Information Area and the Secondary FMIA is large enough to contain the DAR's bitmap and file node table, then the DAR will be created; otherwise, the space between the end of the last DAR and the beginning of the DAR Information Area is wasted. Since the bitmap in the last DAR is the same size as the other DAR's, if the last DAR is smaller the bitmap will have bits indicating the allocation of data blocks that do not exist (in fact it is legal for no data blocks to exist in the last DAR). In this case, the non-existent blocks are marked as allocated. The following macros provide values associated with the space before the DAR Information Area:

```
#define DF_LAST_DAR_SIZE(fs_size, dar_size, nodes_per_dar)
```

```
#define DF_FS_WASTED_SPACE(fs_size, dar_size, nodes_per_dar)
```

The DAR Information Area

At the end of the file system, a table of entries exist for each DAR in the file system. It is located such that its last block of entries is before the last block of the file system containing the Secondary FMIA. This location is provided by:

```
#define DF_DARE_TABLE_ADDRESS(fs_size, dar_size, file_nodes_per_dar)
```

A definition for a DAR entry is:

```
typedef struct
{
    uint32e_type          file_nodes_used;
    uint32e_type          space_used;
    uint32e_type          directories_used;
    df_file_node_number_type free_file_node_number;
    byte8e_type           reserved[DF_RESERVED_BYTES_PER_DAR];
} df_dar_entry_type;
```

file_nodes_used Number of file_nodes in use from the DAR the entry represents.

space_used is the number of data blocks in use from the DAR. This explicitly excludes DAR Information Area blocks, the block containing the Secondary FMIA, and blocks marked as allocated in the last DAR but do not exist. This field includes the following system blocks: the Primary FMIA for the first DAR only, the DAR's bitmap blocks and the DAR's file node blocks.

directories_used is the number of directories in the DAR.

free_file_node_number is the file node number of next free file node in the DAR. This functions as the head of the DAR's free file node list.

SEE ALSO

fstatfs(2), mount(2), statfs(2), inode(4). frec(1M), fsck(1M), labelit(1M), mkfs(1M), volcopy(1M) in the *System Manager's Reference for the DG/UX System*.

NAME

`fspec` – format specification in text files

DESCRIPTION

You may want to maintain text files on the DG/UX system with tabs that are not set at every eighth column. You must usually convert such files to a standard format, frequently by replacing all tabs with the appropriate number of spaces, before they can be processed by DG/UX system commands. A format specification in the first line of a text file specifies how tabs are to be expanded in the rest of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by the brackets `<:` and `:>`. Each parameter consists of a keyletter, possibly followed immediately by a value. The following parameters are recognized:

`t`*tabs* The `t` parameter specifies the tab settings for the file. The value of *tabs* must be one of the following:

1. A list of column numbers separated by commas, indicating tabs set at the specified columns;
2. A `-` followed immediately by an integer *n*, indicating tabs at intervals of *n* columns;
3. A `-` followed by the name of a canned tab specification.

Standard tabs are specified by `t-8`, or equivalently, `t1,9,17,25`, etc. The canned tabs are defined by the `tabs(1)` command.

`s`*size* The `s` parameter specifies a maximum line size. The value of *size* must be an integer. Size is checked after tabs have been expanded, but before the margin is prepended.

`m`*margin* The `m` parameter specifies a number of spaces to be prepended to each line. The value of *margin* must be an integer.

`d` The `d` parameter takes no value. It indicates that the line containing the format specification is to be deleted from the converted file.

`e` The `e` parameter takes no value. It indicates that the current format is to prevail only until another format specification is encountered in the file.

Default values, which are assumed for parameters not supplied, are `t-8` and `m0`. If the `s` parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

```
* <:t5,10,15 s72:> *
```

For programming language source files, if you can disguise a format specification as a comment, you don't need to code the `d` parameter.

SEE ALSO

`ed(1)`, `newform(1)`, `tabs(1)` in the *User's Reference for the DG/UX System*.

NAME

`fstab` – static information about file systems

SYNOPSIS

```
#include <mntent.h>
```

DESCRIPTION

The file `/etc/fstab` describes the file systems and swapping areas used by the local machine. The system administrator can modify it with a text editor or by invoking the `sysadm(1M)` system administration utility. It is read by commands that `mount`, `dump`, `restore`, and `check` the consistency of file systems, as well as by the system in providing swap space. The file consists of a number of lines like this:

```
fsname dir type opts freq passno
```

for example:

```
/dev/dsk/usr /usr dg/ux rw 1 1
```

would indicate a mount for a local file system, and

```
titan:/usr/titan /usr/titan nfs rw,hard 0 0
```

would indicate an NFS file system mount.

A High Sierra CDROM would be indicated using the following line:

```
/dev/pdsk/4 /cdrom cdrom ro 0 0
```

A DOS floppy would be indicated using the following line:

```
/dev/pdsk/3 /pdd/floppy dos rw 0 0
```

A swap area could be indicated using the following line:

```
/dev/dsk/swap1 swap1_area swap sw 0 0
```

The `fstab` format was changed in order to support NFS file systems as well as local file systems. The old-style `fstab` entries are supported, but not recommended.

The entries from this file are accessed using the routines in `getmntent(3C)`, which returns a structure of the following form:

```
struct mntent {
    char *mnt_fsname; /* file system name */
    char *mnt_dir; /* file system path prefix */
    char *mnt_type; /* dg/ux, nfs, swap, cdrom, or ignore */
    char *mnt_opts; /* rw, ro, hard, soft, bg, fg */
    int mnt_freq; /* highest dump level */
    int mnt_passno; /* pass number on parallel fsck */
};
```

Fields are separated by white space; a `#`, as the first non-white character, indicates a comment. The `mnt_type` field determines how the `mnt_fsname` and `mnt_opts` fields will be interpreted. The following is a list of the file system types currently supported, and the way each of them interprets these fields:

<i>Type</i>	<i>Field</i>	<i>Interpretation</i>
dg/ux	mnt_fsname	Must be a block special device unless this is a ramdisk, in which case, it is a symbolic link to the mounted memory file system.
	mnt_opts	Valid options are ro, rw, bg, and fg. If this has the ramdisk option, other options include use_wired_memory, max_file_space and max_file_count.
cdrom	mnt_fsname	Must be a block special device.
	mnt_opts	Valid options are ro, bg, fg.
dos	mnt_fsname	Must be a block special device.
	mnt_opts	Common options are ro, rw, bg, fg.
nfs	mnt_fsname	The hostname of the server and the pathname on the server of the directory to be served. A colon separates the pathname and hostname.
	mnt_opts	Valid options are ro, rw, hard, soft, bg, fg.
swap	mnt_fsname	Must be a block special device swap section.
	mnt_opts	Ignored.

If the *mnt_type* is specified as *ignore*, the entry is ignored. This is useful to show disks not currently used.

Entries identified as *swap* are made available as swap space by the *swapon(1M)* command at the end of the system reboot procedure.

When the *mnt_fsname* field is interpreted as a block special device, programs that require the corresponding character special device must construct the name by changing *dsk* to *rdsk* in the pathname.

If the *mnt_opts* field is a comma-separated list of options that includes *rw* or *ro*, the file system is mounted read-write or read-only. If this includes *hard* or *soft*, the NFS file system is mounted *hard* or *soft*. If the list includes *bg* or *fg*, and failed attempt to mount will cause *mount* to retry in the background or in the foreground. For more details on these options, see *mount(1M)*.

The field *mnt_freq* indicates how often each file system should be dumped by the *dump2(1M)* command (and triggers that command's *w* option, which determines what file systems should be dumped). Most systems set the *mnt_freq* field to 1, indicating that file systems are dumped each day. Some programs, like *sysadm*, may use a different set of entries here.

The final field *mnt_passno* is used by the consistency checking program *fsck(1M)* to allow overlapped checking of file systems during a reboot. All file systems with a *mnt_passno* of 1 are checked first simultaneously, then all file systems with *mnt_passno* of 2 are checked, and so on. A value of 0 indicates that the file system will not be checked. The *<mnt_passno>* of the root file system should be 0, as the

root cannot be checked since it is already mounted.

Programs read the `/etc/fstab` file but never write to it. It is the duty of the system administrator to maintain this file. The order of records in `/etc/fstab` is important because `fsck` and `mount` process the file sequentially; file systems must appear after file systems they are mounted within. For example, if you have an entry for `/usr/spool`, it must appear after the entry for `/usr`.

FILES

`/etc/fstab`

SEE ALSO

`dump2(1M)`, `fsck(1M)`, `mount(1M)`, `swapon(1M)`, `sysadm(1M)`, `getfsent(3C)`, `getmntent(3C)`.

NAME

group – group file

SYNOPSIS

/etc/group

DESCRIPTION

Group is an ASCII file containing a one-line entry for each group recognized by the system. The file format is as follows:

groupname : *password* : *gid* : *user-list*

where:

groupname The name of the group.
password An encrypted password.
gid The group's numerical ID within the system; it must be unique.
user-list A comma-separated list of users allowed in the group.

If the password field is empty, no password is demanded. Because of the encrypted passwords, the group file can and does have general read permission and can be used, for example, to map numerical group IDs to names.

Malformed entries cause routines that read this file to halt, in which case group assignments specified further along are never made. `grpck` can be used to verify entries in the group file. See `pwck(1M)` in the *System Manager's Reference for the DG/UX System*.

ONC/NFS Features

If you are using the DG/UX Open Network Computing/Network File System (ONC/NFS), a group file can have a line beginning with a plus sign (+), which means to incorporate an entry or entries from the Network Information Service (NIS). There are two styles of + entries. By itself, + means to insert the entire contents of the NIS group file at that point; *+groupname* means to insert the entry (if any) for *groupname*. If a + entry has a non-empty *password* or *user-list* field, the contents of that field override the corresponding field from the NIS. The *gid* field cannot be overridden in this way.

An entry can also begin with a minus (-); *-groupname* means to disallow *groupname*. All subsequent entries for the indicated *groupname*, whether originating from the NIS or the local group file, are ignored.

EXAMPLE

```
primary:q.mJzTnu8icF.:10:fred,mary
+myproject:::bill,steve
+:
```

If these entries appear at the end of a group file, then the group `primary` will have members `fred` and `mary`, and a group ID of 10. The group `myproject` will have members `bill` and `steve`, and the password and group ID of the NIS entry for the group `myproject`. All groups listed in the NIS are pulled in and placed after the entry for `myproject`.

FILES

/etc/group

SEE ALSO

`setgroups(2)`, `crypt(3C)`, `crypt(3X)`, `passwd(4)`, `groups(1)`, `newgrp(1)`, `passwd(1)`, `su(1)`, `pwck(1M)`.

NOTES

The `passwd(1)` command won't change group passwords.

Normally, group-ids less than 100 are reserved for system-level use (DG/UX software).

NAME

hfm - high sierra file manager

DESCRIPTION

The DG/UX kernel provides configurable support for High Sierra and ISO 9660 formatted Compact Discs (CDs). The high sierra file manager lets the system administrator mount a CD into the UNIX file system hierarchy. A mounted CD will appear as a readonly UNIX file system. The mode of all files from the CD will be readonly and executable for user, group and other.

Filenames in High Sierra or ISO 9660 format are uppercase, but for convenience, they are translated to lowercase by the high sierra file manager. All input filenames are similarly translated to uppercase. High Sierra and ISO 9660 mounted file systems can be NFS exported in the same way as any normal DG/UX file system. The mount point must be added to `/etc/exports` and the `exportfs(1M)` command must be executed after the file system is mounted. This will be automatic if the mount of the CD is in your `/etc/fstab` file. Since most current CDs available in high sierra or ISO 9660 format are for PC's, the high sierra file manager will be most useful when used with a DOS emulator.

In order to use the high sierra file manager, you must configure the `hfm()` pseudo device into your kernel.

```
sd(inc(),*)
st(inc(),*)
inen()
loop()
pmt()
prf()
meter()
hfm()          # this is the line that must be added.
```

Once the kernel is built and running, you may use the `mount(1M)` command to add the high sierra or ISO 9660 file system to the UNIX file system hierarchy.

```
mount -t cdrom /dev/pdsk/4 /pdd/cdrom
```

The special device mentioned in the mount command is the block special representation of the CD device in `/dev/pdsk`. The type "cdrom" must be used with mount to route the mount request to the correct file manager.

You may add a line to the `/etc/fstab` file to have the mount occur when the system is brought up to init level 3.

```
/dev/pdsk/4 /pdd/cdrom  cdrom ro x 0
```

The `umount(1M)` command may be used to unmount the CD from the file system hierarchy

```
umount /pdd/cdrom
```

To export the file system on the CD, in lieu of adding it to `/etc/exports`:

```
exportfs -iv /pdd/cdrom
```

When the `mount(1M)` command is issued, the CD device will lock the CD platter into the unit until a successful `umount(1M)` is issued.

The high sierra file manager does not support the path table or the extended attribute record from files on the CD, as these are unnecessary to the UNIX file system implementation.

SEE ALSO

`config(1M)`, `exportfs(1M)`, `mount(1M)`, `umount(1M)`, `fstab(4)`.

NAME

holidays - accounting information used to distinguish prime and non-prime days

SYNOPSIS

/usr/lib/acct/holidays

DESCRIPTION

The holidays file distinguishes between *prime* and *non-prime* time for the accounting system. It divides weekdays into two pieces, and it divides the year into prime and non-prime days. Weekends are always non-prime. Additional company holidays can be specified as non-prime.

Comment lines are denoted by an asterisk in column one.

The first non-comment line contains three fields, separated by white space. The first field is the four-digit current year. The second field is the start of prime time, specified as four digits in the form *hhmm* (for hour and minute). The third field is the start of non-prime time, specified in the same way. The hours must be between 0 and 23, inclusive, and the minutes must be between 0 and 59, inclusive.

Subsequent lines define up to 20 non-prime days. The first field is the day of year, where January 1 has the value 1. The second field is the calendar date. The third field is the holiday name.

EXAMPLE

```
* Prime/Nonprime Table for UNIX Accounting System
*
* Curr Prime Non-Prime
* Year Start Start
*
  1989 0830 1700
*
* Day of Calendar Company
* Year Date Holiday
*
   2 Jan 2 New Year's Day Observed
 149 May 29 Memorial Day
 184 Jul 3 Day Before Independence Day
 185 Jul 4 Independence Day
 247 Sep 4 Labor Day
 327 Nov 23 Thanksgiving
 328 Nov 24 Day After Thanksgiving
 359 Dec 25 Christmas Day
```

SEE ALSO

acctcon(1M), acctprc(1M).

NAME

hosts - hostname database

DESCRIPTION

The `hosts` file contains information about the known hosts on the network. For each host, a single line should be present with the following information:

```
Internet_address hostname [ aliases ] [ # comment ]
```

Items are delimited by any number of blanks and/or tab characters. A # character indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file.

For ARPANET systems only: The `hosts` file is often created from the official host database maintained at the DARPA Network Information Center (NIC). However, local changes may be required to update the file for unofficial aliases and/or unknown hosts.

Network addresses are specified in conventional dot notation for use by the `inet_addr` routine from the Internet address manipulation library, `inet(3N)`.

A *hostname* can be a domain name or a single component of a domain name; see `hostname(5)` for details. A component consists of up to 24 characters drawn from the lowercase alphabet (a-z), uppercase alphabet (A-Z), digits (0-9), and minus sign (-). Periods are only allowed when they serve to delimit components of domain names. No blank or space characters are permitted as part of a component. No distinction is made between upper and lower case. The first character of a component must be an alphabetic character. The last character can not be a minus sign or period. Single character names or nicknames are not allowed.

If your system is using either the Network Information Service (NIS) or the Domain Name System (DNS), host names and address mappings are handled differently. For details on NIS, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. For details on DNS, see *Managing TCP/IP on the DG/UX™ System*. The `svcorder` file determines how name/address resolution is done on your system. See the `svcorder(4)` manual page for details.

EXAMPLES

```
85.0.0.31 hostB HOSTB #Comment
85.0.0.32 hostC HOSTC #Greg's Office
```

FILES

`/etc/hosts`

SEE ALSO

`gethostent(3N)`, `svcorder(4)`.

NAME

idl – interface description language

DESCRIPTION

Idl is a language for describing interfaces without respect to the display mechanism which is used to present the information. The interface is described in terms of the menus, screens, and queries that are presented to the user. The language contains statements which can be used to define the entire menu hierarchy.

Idl files are read by the interface description interpreter and compiler, `idi(1)` and `idc(1)`.

Syntax

The language is free format, with whitespace separating tokens. Whitespace is one or more spaces, tabs, or newlines. The character # indicates that the remainder of a line is a comment.

The language is built up from several primitive token types: *names*, *values*, *numbers*, and *keywords*.

A *name* is a sequence of letters, digits, and underscores. A *value* is a sequence of characters other than double quote ("). If the *value* contains whitespace, it must be surrounded by double quotes. A *number* is a sequence of one or more digits (with an optional leading sign character) which is interpreted as a base 10 number.

The following *keywords* are reserved, and may not be used otherwise:

add	boolquery	end	export
menu	operation	querygroup	rangequery
screen	selectquery	set	text
textquery	to		

The syntax of the language is described below:

statement:

```

menu name [ menu-attributes ] ... end
operation name [ operation-attributes ] ... end
text name [ text-attributes ] ... end
screen name [ screen-attributes ] ... end
querygroup name [ querygroup-attributes ] ... end
textquery name [ textquery-attributes ] ... end
boolquery name [ boolquery-attributes ] ... end
selectquery name [ selectquery-attributes ] ... end
rangequery name [ rangequery-attributes ] ... end
set name = value
add name1 to name2
export name

```

These statements, except the `set`, `add`, and `export` statements, create an instance of a database class. For example, a `menu` statement creates an instance of the *menu* class; the instance is named *name* and has the attributes specified by *menu-attributes*.

Each of the attribute lists (*menu-attributes*, *operation-attributes*, and so on) are of the form:

attribute-list:

```
[ attribute-list ] attribute-item
```

attribute-item:

```
name = value
```

Below are the other "types" which the value of an attribute may take.

name-list:

name
"[*name-list*] *name*"

value-list:

value
"[*value-list*] *value*"

command:

value

A *command* is different from a *value* in that a *command* is a string which has meaning if passed to the shell (`sh(1)`) for execution.

boolean:

`#{YES}`
`#{NO}`

direction:

`#{HORIZONTAL}`
`#{VERTICAL}`

menu Class

Instances of the *menu* class are simple containers for *operations*. *Menus* are used to specify the hierarchy of *operations*. *Menus* are added to other *menus* with the `add` statement.

The following attributes are allowed for the *menu* class:

menu Attribute Set		
Name	Type	Default
access-groups	<i>name-list</i>	""
access-names	<i>name-list</i>	"*"
description	<i>value</i>	"No description"
help	<i>value</i>	"No help for this menu."
mnemonic	<i>value</i>	""
name	<i>value</i>	"Unnamed"
title	<i>value</i>	"Untitled"
visible	<i>boolean</i>	"#{YES}"

The attributes have the following meanings:

access-groups

A whitespace-separated list of group names which are allowed access to this menu. A star ("*") means that all groups are allowed access.

access-names

A whitespace-separated list of user names which are allowed access to this menu. A star ("*") means that all users are allowed access.

description

A one-line description of the menu and what is contained under it. The `description` is displayed by the parent menu before this menu is selected, and may be displayed at the same time as the menu's name.

- help** A message to display if the user requests help on this menu.
- mnemonic** A one-character abbreviation for the menu's *name*.
- name** A one or two word name for the menu. The *name* is displayed in the parent menu to identify this menu.
- title** A string, such as "Main Menu" which is used as the title for the menu. This string may be displayed above the items in the menu when this menu is selected.
- visible** A boolean indication of whether this menu will be displayed. If the value is `${NO}`, the menu will not be shown by `idi(1)`.

operation Class

Instances of the *operation* class are the basic actions which can be performed by the user. *Operations* may contain queries which must be answered before performing the action. *Operations* are added to *menus* with the `add` statement.

The following attributes are allowed for the *operation* class:

operation Attribute Set		
Name	Type	Default
access-groups	<i>name-list</i>	""
access-names	<i>name-list</i>	"*"
action	<i>command</i>	""
action-message	<i>command</i>	"Operation failed."
confirm	<i>value</i>	""
description	<i>value</i>	"No description"
entry-action	<i>command</i>	""
entry-action-message	<i>command</i>	"Entry action failed."
exit-action	<i>command</i>	""
exit-action-message	<i>command</i>	"Exit action failed."
help	<i>value</i>	"No help for this operation."
mnemonic	<i>value</i>	""
name	<i>value</i>	"Unnamed"
repeat	<i>value</i>	""
visible	<i>boolean</i>	"\${YES}"

The attributes have the following meanings:

access-groups

A whitespace-separated list of group names which are allowed access to this operation. A star ("`*`") means that all groups are allowed access.

access-names

A whitespace-separated list of user names which are allowed access to this operation. A star ("`*`") means that all users are allowed access.

action A shell command line to execute when this operation is selected (after any queries for the operation are answered and confirmed). This command is not executed if the operation is canceled.

action-message

The error message to display if the `action` for the operation fails.

- confirm** A string to use as a confirmation prompt which must be answered before the operation is executed. If the value of this attribute is the empty string, no confirmation is performed.
- description**
A one-line description of the operation. The `description` is displayed by the parent menu before this operation is selected, and may be displayed at the same time as the operation's name.
- entry-action**
A shell command line to execute as soon as the operation is selected, before any screens or queries are presented. If the value of the `repeat` attribute is not empty, the `entry-action` is performed once for each iteration of the operation.
- entry-action-message**
The error message to display if the `entry-action` for the operation fails.
- exit-action**
A shell command line to execute after all processing of the operation has completed. This command is executed after the `action` command, and is executed even if the operation is canceled. If the value of the `repeat` attribute is not empty, the `exit-action` is performed after all iterations of the operation.
- exit-action-message**
The error message to display if the `exit-action` for the operation fails.
- help** A message to display if the user requests help on the operation.
- mnemonic**
A one-character abbreviation for the operation's *name*.
- name** A one or two word name for the operation. The `name` is displayed in the parent menu to identify this operation.
- repeat** A string to present before repeating the operation. If the value of this attribute is the empty string, the operation is performed only once. Otherwise, the string is presented, and the user is given the opportunity to repeat or cancel the operation.
- visible** A boolean indication of whether the operation will be made available. If the value is `$(NO)`, the operation will appear in the parent menu but will not be available.

text Class

Instances of the *text* class are simple text holders. *Text* objects may be added to *querygroups* with the `add` statement.

The following attributes are allowed for the *text* class:

text Attribute Set		
Name	Type	Default
value	<i>value</i>	""
visible	<i>boolean</i>	"\${YES}"

The attributes have the following meanings:

value A text string to display.

visible A boolean indication of whether the text will be displayed.

screen Class

Instances of the *screen* class are holders for *querygroups*. All of the *querygroups* of a certain *screen* are guaranteed to be evaluated at the same time and before the *querygroups* of any later *screens*. The interface driver may also display *screens* as separate windows. *Screens* may be added to *operations* with the `add` statement.

The following attributes are allowed for the *screen* class:

screen Attribute Set		
Name	Type	Default
entry-action	<i>command</i>	""
entry-action-message	<i>command</i>	"Entry action failed."
exit-action	<i>command</i>	""
exit-action-message	<i>command</i>	"Exit action failed."
title	<i>value</i>	"Untitled"
visible	<i>boolean</i>	"\${YES}"

The attributes have the following meanings:

entry-action

A shell command line to execute when entering the screen.

entry-action-message

The error message to display if the `entry-action` fails.

exit-action

A shell command line to execute when leaving the screen. This is executed after all queries for the screen are validated, and is executed even if the user terminates the screen.

exit-action-message

The error message to display if the `exit-action` fails.

title

A string such as "Add a User" which is used as a title for the screen. This string may be displayed above the queries and *querygroups* which compose the screen.

visible

A boolean indication of whether the screen (and any *querygroups* below it) will be displayed. This attribute is evaluated after an operation is chosen, at the same time as all other screens for the operation, and before the `visible` attributes of the *querygroups* are evaluated.

querygroup Class

Instances of the *querygroup* class are used to group similar queries. The interface driver may use *querygroup* information to display related queries in a more attractive manner. *Querygroups* may be added to *screens* with the `add` statement.

The following attributes are allowed for the *querygroup* class:

querygroup Attribute Set		
Name	Type	Default
orientation	<i>direction</i>	"\${VERTICAL}"
title	<i>value</i>	""
visible	<i>boolean</i>	"\${YES}"

The attributes have the following meanings:

orientation

The preferred layout of queries within the *querygroup*. The value may be either `$VERTICAL` or `$HORIZONTAL`. The default is `$VERTICAL`. This attribute may be ignored by the display driver.

title A string describing the queries within the *querygroup*. This string may be displayed above the queries which compose the *querygroup*.

visible A boolean indication of whether the *querygroup* (and any queries below it) will be displayed. This attribute is evaluated after a screen is entered, and is evaluated at the same time as the *visible* attributes of all other *querygroups* for the screen.

Queries

The following attributes are allowed for all query types: *textquery*, *boolquery*, *selectquery*, and *rangequery*:

Query Attribute Set		
Name	Type	Default
confirm	<i>value</i>	""
confirm-value	<i>value</i>	""
default	<i>value</i>	""
help	<i>value</i>	"No help available."
preserve	<i>boolean</i>	"\${NO}"
prompt	<i>value</i>	""
variable	<i>value</i>	""

The attributes have the following meanings:

confirm The string to use as a confirmation prompt which must be answered by the user before execution continues. Confirmation is performed if the value entered for the query matches the *confirm-value*.

confirm-value

An `ed(1)`-style regular expression. If the value entered for a query matches *confirm-value*, confirmation of the value is sought (using the *confirm* string as the prompt).

default The default value of the *variable*.

help The text string to display if the user requests help on the query.

preserve

An indication of whether the value of *variable* should be saved in a global variable. If the value of this attribute is `${YES}`, the *variable*'s value (after being validated and confirmed) is saved in a global `idl` variable

named *variable*. If the value of this attribute is `#{NO}`, the *variable* is destroyed when the operation is complete.

prompt The text string to be displayed when the query is presented.

variable

The name of an `idl` variable that is set by the query. *variables* may be referenced in other attribute strings by using the `#{variable}` notation.

textquery Class

Instances of the *textquery* class describe how to retrieve an arbitrary text entry from the user. *Textqueries* may be added to *querygroups* or to *selectqueries* with the `add` statement.

The following attributes are allowed for the *textquery* class:

textquery Attribute Set		
Name	Type	Default
confirm	value	""
confirm-value	value	""
default	value	""
help	value	"No help available."
max-columns	number	"40"
max-lines	number	"1"
preserve	boolean	"#{NO}"
prompt	value	"Enter text"
semantics	command	""
semantics-message	value	""
show-columns	number	"40"
show-lines	number	"1"
syntax	command	""
syntax-message	value	""
variable	value	"Text"

The `confirm`, `confirm-value`, `default`, `help`, `preserve`, `prompt`, and `variable` attributes are generic Query Attributes. The other attributes have the following meanings:

max-columns

The maximum number of columns of text accepted for the query.

max-lines

The maximum number of lines of text accepted for the query.

semantics

A command string to execute on the administered host to determine if the value entered for the query is semantically correct. The command must return zero if the value is correct, and return non-zero if the string is not correct. The command may be a builtin command.

semantics-message

The custom error message to display if the semantics check fails. If the value of this attribute is empty, the error message is generated by `idi` from the prompt and the entered value.

show-columns

The maximum number of columns to display at one time. The default

value for this attribute is the value of *max-columns*. This attribute may be ignored by the display driver.

show-lines

The maximum number of lines to display at one time. The default value for this attribute is the value of *max-lines*. This attribute may be ignored by the display driver.

syntax A command string to execute on the administering host to determine if the value entered for the query is syntactically correct. The command must return zero if the value is correct, and return non-zero if the value is not correct. The command may be a builtin command.

syntax-message

The custom error message to display if the syntax check fails. If the value of this attribute is empty, the error message is generated by *idi* from the prompt and the entered value.

boolquery Class

Instances of the *boolquery* class describe how to retrieve a positive or negative response from the user. *Boolqueries* may be added to *querygroups* with the *add* statement.

The following attributes are allowed for the *boolquery* class:

boolquery Attribute Set		
Name	Type	Default
confirm	<i>value</i>	""
confirm-value	<i>value</i>	""
default	<i>boolean</i>	"\${YES}"
help	<i>value</i>	"No help available."
preserve	<i>boolean</i>	"\${NO}"
prompt	<i>value</i>	"Enter yes or no"
variable	<i>value</i>	"Bool"

The *confirm*, *confirm-value*, *default*, *help*, *preserve*, *prompt*, and *variable* attributes are generic Query Attributes.

selectquery Class

Instances of the *selectquery* class describe how to retrieve one or more choices from a list of choices. *Selectqueries* may be added to *querygroups* with the *add* statement.

The following attributes are allowed for the *selectquery* class:

selectquery Attribute Set		
Name	Type	Default
abort-message	value	"No possible values."
assign-values	value-list	""
confirm	value	""
confirm-value	value	""
default	value	""
exclusive	boolean	"\${YES}"
help	value	"No help available."
input-separator	value	","
number	boolean	"\${YES}"
output-separator	value	","
packed	boolean	"\${YES}"
possible-values	value-list	""
preserve	boolean	"\${NO}"
prompt	value	"Enter selection"
variable	value	"Selection"

The confirm, confirm-value, default, help, preserve, prompt, and variable attributes are generic Query Attributes. The other attributes have the following meanings:

abort-message

The message to display if an operation must be aborted because the value of *possible-values* for this query is empty.

assign-values

A newline-separated list of values which may be assigned to the *variable* when the user selects one of the *possible-values*. The value of this attribute may be a backquoted string which is executed to dynamically produce the list described.

exclusive

If the value of this attribute is `${YES}`, only one of the *possible-values* for the query may be selected. If the value of this attribute is `${NO}`, more than one of the values may be selected.

input-separator

The set of characters which may be used by the user to separate multiple selections when selecting more than one *possible-value*. This attribute is used only if the value of the *exclusive* attribute is `${NO}`.

number

If the value of this attribute is `${YES}`, the *possible-values* of the query will be automatically numbered by the interface driver. If the value of this attribute is `${NO}`, the *possible-values* will not be numbered. This attribute should be set to `${NO}` when the *possible-values* are numbers so that there is no confusion between the *possible-values* and the automatically-generated numbers.

output-separator

The character string which is used to separate multiple selections when assigning a value to the query's *variable*. This attribute is used only if the value of the *exclusive* attribute is `${NO}`.

packed

If the value of this attribute is `${YES}`, the interface driver may conserve screen space when presenting the query. If the value is `${NO}`, screen

space may not be conserved.

possible-values

A newline-separated list of choices for the query. The value of this attribute may be a backquoted string which is executed to produce the list of values.

rangequery Class

Instances of the *rangequery* class describe how to retrieve a number within a given range from the user. *Rangequeries* may be added to *querygroups* with the `add` statement.

The following attributes are allowed for the *rangequery* class:

rangequery Attribute Set		
Name	Type	Default
confirm	value	""
confirm-value	value	""
default	value	"0"
help	value	"No help available"
preserve	boolean	"\${NO}"
prompt	value	"Enter value"
range	number-list	"0 1"
semantics	command	""
semantics-message	value	""
syntax	command	""
syntax-message	value	""
variable	value	"Range"

The `confirm`, `confirm-value`, `default`, `help`, `preserve`, `prompt`, and `variable` attributes are generic Query Attributes. The other attributes have the following meanings:

range A whitespace-separated list of two numbers which are the minimum and maximum values for the query. The value of this attribute may be a backquoted string which is executed to produce the list of numbers.

semantics

A command string to execute on the administered host to determine if the value entered for the query is semantically correct. The command must return zero if the value is correct, and return non-zero if the value is not correct. The command may be a builtin command.

semantics-message

The custom error message to display if the semantics check fails. If the value of this attribute is empty, the error message is generated by `idi` from the prompt and the entered value.

syntax A command string to execute on the administering host to determine if the value entered for the query is syntactically correct. The command must return zero if the value is correct, and return non-zero if the value is not correct. The command may be a builtin command.

syntax-message

The custom error message to display if the syntax check fails. If the value of this attribute is empty, the error message is generated by `idi` from the prompt and the entered value.

set Statement

The `set` statement causes the `idl` variable named *name* to take on the value *value*. The *value* is available globally for the duration of the program.

add Statement

The `add` statement causes the database object named *name1* to be added as a sub-object of the database object named *name2*.

The following rules apply:

- a. Both *names* must be defined previously.
- b. Any number of *menus* or *operations* may be added to a *menu*.
- c. Any number of *screens* may be added to an *operation*.
- d. Any number of *querygroups* may be added to a *screen*.
- e. Any number of queries (*textquery*, *boolquery*, *selectquery*, or *rangequery*) may be added to a *querygroup*.
- f. An number of *texts* may be added to a *querygroup*.
- g. At most one *textquery* may be added to a *selectquery*.

export Statement

The `export` statement exports the `idl` variable named *name* (along with the variable's value) into the environment of all sub-shells. This is a function similar to the `export` command of the shell (`sh(1)`).

Compiler Directives

The following compiler directives can be used to alter the behavior of the compiler or interpreter.

`%dir name`

Interpret subsequent `%include` lines relative to *name*. Such a line overrides any previous `%dir` directive.

`%include name`

Read the contents of the file *name* as if the contents were present in the current file.

`%print [object]`

If *object* is given, print debugging information about *object*. Otherwise, print information about all objects.

Variable Substitution

The `action`, `assign-values`, `confirm`, `default`, `help`, `possible-values`, `preserve`, `prompt`, `range`, `semantics`, and `syntax` attributes are processed so that `idl` variables may be used inside of the values for these attributes.

Variable expansion may be indicated by any of these forms:

`$var` or `${var}`

If *var* is set, substitute the value of *var*. Otherwise, substitute an empty string.

`$#var` or `${#var}`

Substitute the number of words found in the value of *var*. Words are separated by whitespace.

`$(var:-val)`
 If *var* is set and non-null, substitute the value of *var*. Otherwise, substitute *val*.

`$(var:+val)`
 If *var* is set and non-null, substitute *val*. Otherwise, substitute an empty string.

`$(var: ?val1:val2)`
 If *var* is set and non-null, substitute *val1*. Otherwise, substitute *val2*.

`$(var: <prefix)`
 If *var* is set and non-null, substitute its value prefixed by *prefix*. Otherwise, substitute an empty string.

`$(var:=text1:value1;text2:value2;textn:valuen)`
 Compare the value of *var* with each of the *texts*, and substitute the *value* associated with the matching *text*. As many text and value pairs as are required may be included. An empty *text* may be specified to indicate a default case. If *var* matches none of the *texts*, substitute an empty string.

If the colon (:) is omitted from the above expressions, *idi* only checks whether *var* is set or not.

In all cases, *var* must be a sequence of alphanumeric characters and underscores, optionally followed by an index specification of the form

name[*index*]

where the *index* is used to select only some of the words or lines from the value of *name*. If the *index* begins with =, the *index*-th line is substituted; otherwise, the *index*-th word is substituted. Words are separated by one or more whitespace characters. The *index* is subjected to variable substitution and may consist of a single number or two numbers separated by a -. The first word or line of a variable's value is numbered 1. If the first number of a range is omitted, it defaults to 1. If the last member of a range is omitted, it defaults to \$#name. The index * selects all words or lines.

If a *val* or *prefix* contains any of colon (:), semi-colon (;), or right brace (}), the character must be preceded by a backslash (\) to escape its special meaning.

Any variables found within double quotes (") are expanded. All characters between back quotes (`) are expanded and passed to the shell (sh(1)) for execution, and the result of the shell execution is inserted in place of the back-quoted string. A backslash (\) preceding either \$ or ` causes the character to lose its special meaning.

The *value* or *text* part of any of the above expressions may contain other variable references.

Pre-defined Variables

The following variables are used internally by *idi*(1) and should not be changed. These variables should be used in place of the strings they represent (for example, always use "\${YES}" instead of "yes").

YES This is defined to be the affirmative string, yes.

NO This is defined to be the negative string, no.

HORIZONTAL

This is defined to be horizontal. This may be used as the value for the

orientation attribute of querygroups.

VERTICAL

This is defined to be `vertical`. This may be used as the value for the orientation attribute of querygroups.

NO_DEFAULT

This is defined to be [`No default`]. This may be used as the value for the `default` attribute of selectqueries. When this is used, the interface driver will leave the `default` for the selectquery empty if possible.

SKILL_LEVELS

This is defined to be the list of possible skill levels: `Novice Intermediate Expert`. Note that this variable's value varies based on the current locale.

The following global variables are set by `idi` at run-time:

`Argc` The number of arguments passed to the `idi` process.

`Argv` The argument vector passed to the `idi` process. The first item of the vector is referenced as `$Argv[1]`.

InterfaceName

The name of the chosen interface. This will be either `ascii` or `motif`. This is the only means for changing the behavior of the program based on the chosen interface.

`Locale` The locale string returned from `setlocale(3C)`.

OperationName

The value of the `name` attribute of the current operation. This may be used to generalize query prompts:

```
prompt = "Host Name to ${OperationName}"
```

The following global variables may be set by the `idl` programmer:

SkillLevel

The chosen level of expertise. This must be one of the values from the `${SKILL_LEVELS}` variable. The default is `Intermediate`.

TitlePrefix

The string which precedes the actual title of windows and screens. The default is the empty string.

TitleSuffix

The string which follows the actual title of windows and screens. The default is the empty string.

Builtin Commands

Several builtin commands are provided for use in values for the `action`, `semantics`, and `syntax` attributes. The builtin commands are the following:

`:Confirm` *confirmation-string*

Present the *confirmation-string* to the user using the appropriate interface driver. Return zero if the string is confirmed; return non-zero if it is not confirmed.

`:DoOp` *operation-name* [*confirmation-string*]

Perform the *operation-name* operation. If the *confirmation-string* is used,

ask for confirmation before the operation is performed. If the confirmation fails, exit with status 0; otherwise, exit with the exit status of the operation.

- :Echo *message*
Echo the *message* to the display.
- :Error *message*
Display the error *message* in a way appropriate for the interface driver.
- :Help *help-text*
Present a help message to the user. The *help-text* is a text object containing the text of the help message.
- :Log *message*
Append the *message* to the log file. The *message* is written regardless of the verbosity level chosen by the user.
- :Match *regexp string*
Return zero if the *string* matches the given egrep(1)-style regular expression, *regexp*; otherwise, return non-zero. This command is useful in the syntax attribute of queries.
- :Numeric *lower-bound upper-bound value*
Return zero if the integer *value* given is within the range specified by *lower-bound* and *upper-bound*. This command is useful in the syntax attribute of queries.
- :Quit *exit-code*
Terminate the program with *exit-code* as the status code.
- :Restart [*command-line*]
Restart the interface driver, optionally using the supplied *command-line*. If the *command-line* is not given, the current command line is used. This operation may be used to take into account new or changed description files.
- :Run *command*
Execute an interactive *command* on the host system. The standard input, output, and error file descriptors are set appropriately.
- :Set *variable value*
Set the global *variable* to *value*. The *variable* is then available for use by other queries. The *variable* is created if it does not exist, or modified if it does exist.
- :Show
Dump the values of all variables to stdout. This is useful for debugging.
- :Unimp *message*
Display a *message* indicating that some feature is unimplemented. *message* should describe the feature not implemented.
- :Unset *variable*
Remove the global *variable* and its value. This command should only be used for *variables* which are set using the :Set builtin command.
- :Warning *message*
Display the warning *message* in a way appropriate for the interface driver.

EXAMPLES

Below is a sample idl file which creates a single menu with several operations which could be used to manage the /etc/ethers database file.

```

#####
#
#   Some patterns used here
#
#####

set STD_HOST_NAME_PATTERN = "[a-zA-Z] [-.a-zA-Z0-9]*\$"

set STD_HOST_NAME_HELP =
"Enter an Internet host name.  A host name may contain the characters:
      a-z  A-Z  0-9  .  -
It should begin with a letter (a-z or A-Z) and be no more
than 32 characters in length.  It should not contain a . or -
as the last character."

set STD_ETHER_ADDRESS_PATTERN =
"^[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+:[0-9a-fA-F]+\$"

set STD_ETHER_ADDRESS_HELP =
"Enter an Ethernet address.  An Ethernet address has the form:
      aa:bb:cc:dd:ee:ff
where a, b, c, d, e, f are two-digit hexadecimal numbers 00 and ff.
The numbers are separated by colons.  You must enter all 17 characters."

set dg_EthersFile = "/etc/ethers"

#####
#
#   Main menu
#
#####

menu main
    name = "Main"
    title = "Main Menu"
    description = "Top level menu"
    help =
    "This is the first level menu.  It contains a sub-menu for
    manipulating the ethers database."
end

#####
#
#   Ether menu
#
#####

menu dg_Ether
    name = "Ether"
    mnemonic = E
    title = "Ethers Menu"
    description = "Manipulate the ethers databases"
    help =

```

```
"This menu provides access to the ethers databases.  There are
operations for adding, deleting, modifying, and listing entries
from the database."
end
```

```
#####
#
# Operations
#
#####
```

```
operation dg_EtherAdd
    name = Add
    mnemonic = A
    action = "admether -o add -a ${NetAddress}"
    description = "Add an entry to the ethers database"
    help =
"The Add operation takes a host name and an Ethernet address and adds
an entry to the ethers database."
    exit-action = ":Unset DefaultString"
end
```

```
operation dg_EtherDelete
    name = Delete
    mnemonic = D
    action = "admether -odelete"
    description = "Delete entry from the ethers database"
    confirm = "Delete ${HostName} from the ethers database?"
    help =
"The Delete operation takes one or more host names and
deletes the corresponding entry or entries from the
ethers database."
end
```

```
operation dg_EtherModify
    name = Modify
    mnemonic = M
    action =
"admether -o modify -n ${NewHostName} -a ${NetAddress}"
    description = "Modify an entry in the ethers database"
    help =
"The Modify operation takes a host name and allows the user to modify
the corresponding entry in the ethers file.
The user may modify the host name and the Ethernet address."
    exit-action = ":Unset DefaultString"
end
```

```
operation dg_EtherList
    name = List
    mnemonic = L
    action = "admether -o list"
    description = "List entries from the ethers database"
    help =
```

```

"The List operation displays the contents of the ethers database
for one or more hosts."
end

#####
#
# Screens, querygroups, and queries
#
#####

screen dg_AddEtherScreen
    title = "Add an Ethers Entry"
    entry-action = ":Set DefaultString 00:00:00:00:00:00 NewName"
end

#
# This querygroup and its queries are used for entering a
# new ether entry. The defaults are stored in the DefaultString
# variable, and should be set by the screen.
#

querygroup dg_NewEtherEntryQG
end

textquery dg_HostNameText
    prompt = "Host Name"
    variable = HostName
    syntax = ":Match ${STD_HOST_NAME_PATTERN} ${HostName}"
    help = "${STD_HOST_NAME_HELP}

This is the name of the host as it should appear in the
ethers database."
    #
    # Do different checks based on whether we're adding or
    # listing.
    #
semantics = "${OperationName=Add:test -z `grep ${HostName} ${dg_EthersFile}`;\
:test -n `grep ${HostName} ${dg_EthersFile}`}"
    default = "${DefaultString[2]}"
end

textquery dg_EthernetText
    prompt = "Ethernet address"
    variable = NetAddress
    syntax = ":Match ${STD_ETHER_ADDRESS_PATTERN} ${NetAddress}"
    help = "${STD_ETHER_ADDRESS_HELP}

This is the Ethernet address of the host as it should appear
in the ethers database."
    default = "${DefaultString[1]}"
end

#

```

```

# This screen, querygroup, and query are shared between Delete
# and List, because both operations need to choose one or more
# existing host names.
#

screen dg_HostNameListScreen
    title = "${OperationName} Ethers Entry(ies)"
end

querygroup dg_HostNameListQG
end

selectquery dg_HostName
    prompt = "Host Name(s)"
    possible-values = "all
`admether -o list -q | cut -f2 -d' '"
    exclusive = "$NO"
    variable = HostName
    default = "${NO_DEFAULT}"
    help = "
This is the name of the host(s) to ${OperationName}."
end

#
# This screen and its queries are used for getting a single
# existing entry which will be modified.
#

screen dg_ModifyEtherScreen1
    title = "Modify an Ethers Entry"
end

querygroup dg_ModifyEtherQG1
end

screen dg_ModifyEtherScreen2
    title = "Modify an Ethers Entry"
    entry-action = ":Set DefaultString `admether -o list -q ${HostName}`"
end

selectquery dg_OldHostName
    prompt = "Old Host Name"
    possible-values = "`admether -o list -q | cut -f2 -d' '"
    exclusive = "$YES"
    variable = HostName
    help = "
This is the name of the host whose database entry is to
be modified."
end

add dg_Ether to main
add dg_EtherAdd to dg_Ether
    add dg_AddEtherScreen to dg_EtherAdd

```

```
    add dg_NewEtherEntryQG to dg_AddEtherScreen
        add dg_HostNameText to dg_NewEtherEntryQG
        add dg_EthernetText to dg_NewEtherEntryQG

add dg_EtherDelete to dg_Ether
    add dg_HostNameListScreen to dg_EtherDelete
        add dg_HostNameListQG to dg_HostNameListScreen
        add dg_HostName to dg_HostNameListQG

add dg_EtherModify to dg_Ether
    add dg_ModifyEtherScreen1 to dg_EtherModify
        add dg_ModifyEtherQG1 to dg_ModifyEtherScreen1
        add dg_OldHostName to dg_ModifyEtherQG1

    add dg_ModifyEtherScreen2 to dg_EtherModify
        add dg_NewEtherEntryQG to dg_ModifyEtherScreen2

add dg_EtherList to dg_Ether
    add dg_HostNameListScreen to dg_EtherList
```

SEE ALSO

ed(1), egrep(1), idi(1), idc(1), sh(1).

NAME

inittab - script for init

DESCRIPTION

The file `/etc/inittab` controls process dispatching by `init`. The processes most typically dispatched by `init` are servers.

The `inittab` file is composed of entries that are position dependent and have the following format:

id : *rstate* : *action* : *process*

Each entry is delimited by a newline, however, a backslash (\) preceding a newline indicates a continuation of the entry. Up to 512 characters per entry are permitted. Comments may be inserted in the *process* field using the convention for comments described in `sh(1)`. There are no limits (other than maximum entry size) imposed on the number of entries in the `inittab` file. The entry fields are:

- id* This is one or two characters used to uniquely identify an entry.
- rstate* This defines the run level in which this entry is to be processed. Run-levels effectively correspond to a configuration of processes in the system. That is, each process spawned by `init` is assigned a run level or run levels in which it is allowed to exist. The run levels are represented by a number ranging from 0 through 6. As an example, if the system is in run level 1, only those entries having a 1 in the *rstate* field are processed. When `init` is requested to change run levels, all processes that do not have an entry in the *rstate* field for the target run level are sent the warning signal `SIGTERM` and allowed a 5-second grace period before being forcibly terminated by the kill signal `SIGKILL`. The *rstate* field can define multiple run levels for a process by selecting more than one run level in any combination from 0 through 6. If no run level is specified, then the process is assumed to be valid at all run levels 0 through 6. There are three other values, a, b and c, which can appear in the *rstate* field, even though they are not true run levels. Entries which have these characters in the *rstate* field are processed only when an `init` or `telinit` process requests them to be run (regardless of the current run level of the system). See `init(1M)`. They differ from run levels in that `init` can never enter run level a, b or c. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an a, b or c command is not killed when `init` changes levels. They are killed only if their line in `inittab` is marked off in the *action* field, their line is deleted entirely from `inittab`, or `init` goes into single-user state.
- action* Key words in this field tell `init` how to treat the process specified in the *process* field. The actions recognized by `init` are as follows:
- `respawn` If the process does not exist, then start the process; do not wait for its termination (continue scanning the `inittab` file), and when the process dies, restart the process. If the process currently exists, do nothing and continue scanning the `inittab` file.
- `wait` When `init` enters the run level that matches the entry's *rstate*, start the process and wait for its termination. All subsequent reads of the `inittab` file while `init` is in the same run level cause `init` to ignore this entry.

once	When <code>init</code> enters a run level that matches the entry's <i>rstate</i> , start the process, do not wait for its termination. When it dies, do not restart the process. If <code>init</code> enters a new run level and the process is still running from a previous run level change, the program is not restarted.
boot	The entry is to be processed only at <code>init</code> 's boot-time read of the <code>inittab</code> file. <code>init</code> is to start the process, not wait for its termination; and when it dies, not restart the process. In order for this instruction to be meaningful, the <i>rstate</i> should be the default or it must match <code>init</code> 's run level at boot time. This action is useful for an initialization function following a hardware reboot of the system.
bootwait	The entry is to be processed the first time <code>init</code> goes from single-user to multi-user state after the system is booted. (If <code>initdefault</code> is set to 2, the process runs right after the boot.) <code>init</code> starts the process, waits for its termination and, when it dies, does not restart the process.
powerfail	Execute the process associated with this entry only when <code>init</code> receives a power fail signal, <code>SIGPWR</code> [see <code>signal(2)</code>].
powerwait	Execute the process associated with this entry only when <code>init</code> receives a power fail signal, <code>SIGPWR</code> , and wait until it terminates before continuing any processing of <code>inittab</code> .
off	If the process associated with this entry is currently running, send the warning signal <code>SIGTERM</code> and wait 5 seconds before forcibly terminating the process with the kill signal <code>SIGKILL</code> . If the process is nonexistent, ignore the entry.
ondemand	This instruction is really a synonym for the <code>respawn</code> action. It is functionally identical to <code>respawn</code> but is given a different keyword in order to divorce its association with run levels. This instruction is used only with the <code>a</code> , <code>b</code> or <code>c</code> values described in the <i>rstate</i> field.
initdefault	An entry with this action is scanned only when <code>init</code> is initially invoked. <code>init</code> uses this entry, if it exists, to determine which run level to enter initially. It does this by taking the highest run level specified in the <i>rstate</i> field and using that as its initial state. If the <i>rstate</i> field is empty, this is interpreted as <code>0123456</code> and <code>init</code> therefore enters run level 6. This will cause the system to loop, that is, it will go to firmware and reboot continuously. Additionally, if <code>init</code> does not find an <code>initdefault</code> entry in <code>inittab</code> , it requests an initial run level from the user at reboot time.
sysinit	Entries of this type are executed before <code>init</code> tries to access the console (i.e., before the <code>Console Login:</code> prompt). It is expected that this entry will be only used to initialize devices on which <code>init</code> might try to ask the run level question. These entries are executed and waited for before continuing.

process This is a command to be executed. The entire *process* field is prefixed with `exec` and passed to a forked `sh` as `sh -c 'exec command'`. For this reason, any legal `sh` syntax can appear in the *process* field.

SEE ALSO

`init(1M)`, `ttymon(1M)`, `exec(2)`, `open(2)`, `signal(2)`

`sh(1)`, `who(1)` in the *User's Reference Manual*

NAME

inode - file node structure

SYNOPSIS

```
#include <ufs/disk_format.h>
```

DESCRIPTION

The inode table for a file system is distributed across the disk: a table exists in each disk allocation region (DAR). For more information about the file system layout, refer to `fs(4)`.

The file node's purpose is to provide access to data blocks associated with the file. The data blocks are allocated in chunks of contiguous physical blocks called data elements. In the case that the file is less than the data element size, the file is fragmented. In this case, the file has only one data element and its size is determined by the fragment exponent field. If the file grows, the fragmented data element is copied to a full sized element, and the allocation to the file will always be in data element sized chunks, causing the actual size of the file to be less than or equal to the blocks allocated to it.

Data elements are accessed directly or indirectly depending on the size of the file. The file node has an array of direct data elements, pointing to the first block of the data element. If the size of the file is greater than the number of direct data element pointers, then indirect access is used.

Indirect data element access is provided through indexing. An index structure consists of index blocks containing pointers to data elements. Depending on the depth of the index structure, index entries point to data elements or other index blocks. There are three index structures rooted in the file node; each of the three differs in the levels of indexing. If the file node represents a directory, only the first index level is used.

In the case of the first index structure, the pointer in the file node points to the first block containing the index entries (an index may span blocks); the entries at this level point to data elements. The second index structure points to the first block containing index entries. Each index entry at this level points to the first block of an index containing the same number of entries as the previous level. These index entries contain pointers to data elements. The third index structure is similar to the previous two but has another level of indexing before the index containing the data element pointers.

This expansion of index levels produces a tree, where the leaves of the tree are data elements. The number at each level multiplies itself by the number of index entries.

To access a data block, it must be determined if it is accessible directly or through indexing. If direct access is possible, the data element needs to be determined along with the particular block within the data element. If the block is deep enough in the file to require indexing, the level of indexing must be determined by finding what range of blocks each index covers. After the index structure is determined, the path of entries through the index structure is required.

The inode table in the DAR is made up of entries of the following structure:

```

typedef struct
{
    boolean_field_type    is_allocated           : 1;
    boolean_field_type    is_fragmented        : 1;
    field_type            fragment_size_exponent : 3;
    field_type            des_exponent          : 5;
    field_type            ies_exponent          : 4;
    field_type            pad_to_double_word    : 9;
    field_type            partial_block_byte_count : 9;
    uint32e_type          whole_block_count;
    uint32e_type          generation_number;
    uint32e_type          dar_index;
    df_file_node_number_type space_parent;
    uint32e_type          maximum_space_usage;
    uint32e_type          current_space_usage;
    uint32e_type          maximum_file_node_usage;
    uint32e_type          current_file_node_usage;
    df_file_mode_type    mode;
    uint16e_type          user_id;
    uint16e_type          group_id;
    int16e_type          link_count;
    df_time_type         time_last_accessed;
    df_time_type         time_last_modified;
    df_time_type         time_attributes_last_changed;
    union
    {
        struct
        {
            uint32e_type    data[DF_DIRECT_ELEMENT_COUNT];
            union
            {
                struct
                {
                    {
                        uint32e_type index_array[DF_MAX_DIR_INDEX_LEVEL];
                        df_din_type  din;
                    } directory;
                } regular;
            } index;
        } element_addresses;
    } contents;
    byte8e_type reserved[DF_RESERVED_BYTES_PER_FILE_NODE];
} df_file_node_type;

```

is_allocated indicates whether this is a free file node or not. If FALSE it is a free file

node; if TRUE, then this is a valid file node.

is_fragmented is TRUE when the first (and only) element of the file is reduced in size from the data element size to the fragment size specified by *fragment_size_exponent*; otherwise, all data elements (if any) are the full data element size and *fragment_size_exponent* is invalid.

fragment_size_exponent specifies, when valid, the size of the fragmented data element which contains the file's data. The size in blocks of the fragment is 2 raised to the *fragment_size_exponent* power. It must be large enough to fit the total size of the file in the fragment. Because all fragments must fit into a single file system buffer, the maximum fragment size is:

```
#define DF_MAX_FRAGMENT_SIZE      16
```

blocks, although the *fragment_size_exponent* field is large enough to support fragment sizes up to 128 (2^7) blocks.

des_exponent specifies the data element size. The data element size in blocks is 2 raised to the *des_exponent* power. The maximum data element size is therefore 2^{31} blocks. The maximum value for this field is:

```
#define DF_MAX_DES_EXPONENT 31
```

although it is also limited to the base 2 logarithm of the largest power of 2 that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

ies_exponent specifies the index element size. The index element size in blocks is 2 raised to the *ies_exponent* power. The maximum index element size is therefore 2^{15} blocks. The maximum value for this field is:

```
#define DF_MAX_IES_EXPONENT 15
```

although it is also limited to the base 2 logarithm of the largest power of 2 that is less than or equal to:

```
#define DF_USER_BLOCKS_PER_DAR(dar_size, file_nodes_per_dar)
```

partial_block_byte_count is the count of the number of bytes to the end of file following the last whole block. All possible values, i.e., 0 to 511, are legal.

whole_block_count is the number of 512 byte blocks logically in the file before EOF. The file size as reported by `stat(2)` is:

```
((whole_block_count * 512) + partial_block_byte_count).
```

generation_number is incremented each time an inode is freed and is kept valid on free nodes so that subsequent uses of the same file node number are guaranteed to have different UFID values.

dar_index is the current allocation hint (index of a DAR to use for data and file node

allocation). DAR indexes are zero based.

space_parent is the parent file node number. In the file node for the root of the filesystem, the value of *space_parent* is:

```
#define DF_ROOT_FILE_NODE_NUMBER 2
```

therefore, the filesystem root is its own space parent.

maximum_space_usage is the maximum usage limit in blocks for the file plus all its space descendants. It must be set to `UIN32_MAX` for non-CPD directories and other non-directory files, as well as for CPD's which have no allocation limit. On the root of each filesystem, this limit is not applied to the superuser.

current_space_usage is the current usage in blocks for the file plus all its space descendants, if any. If not a CPD, then it is the number of blocks actually used to store the file's contents on disk, including both index and data elements. For a CPD, it is that plus the *current_space_allocation* fields of all files which name this CPD as their space parent.

maximum_file_node_usage is the maximum file node usage limit for the file plus all its space descendants. Must be `UIN32_MAX` for non-CPD directories and other non-directory files, as well as for CPDs with no file node allocation limit. On the root of each filesystem, this limit is not applied to the superuser. On all other CPD's it is applied equally to all users.

current_file_node_usage is the current file node usage count for the file plus all its space descendants. It must be 1 for non-CPD directories and other non-directory files. For a CPD, it is 1 plus the *current_file_node_usage* fields of all files which name this CPD as their space parent.

mode is the file's mode. See `stat(2)`.

user_id is user id of the file.

group_id is the group id of the file.

link_count is the number of links (directory entries) to the file. Must be greater than zero.

time_last_accessed is the time the file's contents were last accessed (i.e., read or executed).

time_last_modified is the time the file's contents were last modified (i.e., written or truncated).

time_attributes_last_changed is the time one of the file's attributes (mode, *user_id*, *group_id*, *link_count*, *child_count*, etc.) was last changed.

contents is a union containing *represented_device* for block-special or character-special files, and containing *element_addresses* for all other file types.

represented_device is the device numbers of the device represented by a character or

block special file. The padding bytes (`pad_to_union_size`) must be set to zero.

element_addresses are the disk addresses of the data elements and index elements of the file. The "data" field contains the addresses of the first:

```
#define DF_DIRECT_ELEMENT_COUNT    10
```

data elements in the file. The "index" field contains the addresses of the first index element of each level for regular files. For directory files, we only have 1 level of indexing, with the other two index fields being used to store the directory manager information.

Since all the file nodes in a DAR are not necessarily allocated, a list of free file nodes must be maintained. The head of the list is contained in each DAR entry. The DAR entry contains the file node number of a file node in the DAR, that file node should be unallocated and the following structure contains the fields for a free file node:

```
typedef struct
{
    boolean_field_type    is_allocated : 1;
    df_file_node_number_type next_free_file_node_number;
    uint32e_type          generation_number;
    byte8e_type          pad_to_file_node_size[DF_FREE_FILE_NODE_PADDING];
} df_free_file_node_type;
```

is_allocated is TRUE when this is a valid `file_node`. If FALSE, then this is a free `file_node`.

generation_number is kept valid on free nodes so that subsequent uses of the same file node number are guaranteed to have different UFID values.

next_free_file_node_number is the file node number of next free `file_node` on the DAR free `file_node` list.

SEE ALSO

`stat(2)`, `dg_stat(2)`, `fs(4)`; `fsck(1M)`, `mkfs(1M)` in the *System Manager's Reference for the DG/UX System*.

NAME

issue – issue identification file

DESCRIPTION

The file `/etc/issue` contains the *issue* or project identification to be printed as part of the login prompt. This is an ASCII file containing any text you choose and is read by program `getty` and then written to any terminal spawned or respawned from the `inittab(4)` file.

FILES

`/etc/issue`

SEE ALSO

`ttydefs(4M)`

`login(1)` in the *User's Reference for the DG/UX System*.

NAME

ldfcn – COFF executable file access routines

SYNOPSIS

```
#include <stdio.h>
#include <sys/types.h>
#include <filehdr.h>
#include <ldfcn.h>
```

DESCRIPTION

The executable file access routines are a collection of functions for reading a COFF executable file that is in DG/UX executable file format. Although the calling program must know the detailed structure of the parts of the executable file that it processes, the routines effectively insulate the calling program from knowledge of the overall structure of the executable file.

The interface between the calling program and the executable file access routines is based on `LDFILE` defined as `struct ldfile`, declared in the header file `ldfcn.h`. This structure provides uniform access to simple executable files and to executable files that are members of an archive file.

The function `ldopen(3X)` allocates and initializes the `LDFILE` structure and returns a pointer to the structure to the calling program. The fields of the `LDFILE` structure may be accessed individually through macros defined in `ldfcn.h` and contain the following information:

```
LDFILE          *ldptr;

TYPE(ldptr)     The file magic number, used to distinguish between archive
                members and simple executable files.

IOPTR(ldptr)    The file pointer returned by fopen(3S) and used by the standard
                input/output functions.

OFFSET(ldptr)   The file address of the beginning of the executable file; the offset is
                non-zero if the executable file is a member of an archive file.

HEADER(ldptr)   The file header structure of the executable file.
```

The executable file access functions may be divided into four categories:

- (1) Functions that open or close an executable file
 - `ldopen(3X)` and `ldaopen(3X)` open an executable file
 - `ldclose(3X)` and `ldaclose(3X)` close an executable file
- (2) Functions that read header or symbol table information.
 - `ldahread(3X)` reads the archive header of a member of an archive file
 - `ldfhread(3X)` reads the file header of an executable file
 - `ldshread(3X)` reads a section header of an executable file
 - `ldsyshread(3X)` reads the system header of an executable file
 - `ldtbread(3X)` reads a symbol table entry of an executable file
 - `ldgetname(3X)` retrieves a symbol name from a symbol table entry.
- (3) Functions that position an executable file at (seek to) the start of a particular section.
 - `ldohseek(3X)` seeks to the system header of an executable file
 - `ldsseek(3X)` seeks to a section of an executable file
 - `ldtbseek(3X)` seeks to the symbol table of an executable file

- (4) The function `ldtbindex(3X)` returns the index of a particular executable file symbol table entry.

These functions are described in detail on their respective manual pages.

All the functions except `ldaopen(3X)`, `ldgetname(3X)`, `ldopen(3X)`, and `ldtbindex(3X)` return either `SUCCESS` or `FAILURE`, both constants defined in `ldfcn.h`. `ldaopen(3X)` and `ldopen(3X)` both return pointers to an `LDFILE` structure.

Additional access to an executable file is provided through a set of macros defined in `ldfcn.h`. These macros parallel the standard input/output file reading and manipulating functions, translating a reference of the `LDFILE` structure into a reference to its file descriptor field.

The following macros are provided:

```

GETC(ldptr)
FGETC(ldptr)
GETW(ldptr)
UNGETC(c, ldptr)
FGETS(s, n, ldptr)
FREAD(ptr, sizeof(*ptr), nitems, ldptr)
FSEEK(ldptr, offset, ptrname)
FTELL(ldptr)
REWIND(ldptr)
FEOF(ldptr)
FERROR(ldptr)
FILENO(ldptr)
SETBUF(ldptr, buf)

```

See the manual entries for the corresponding standard input/output library functions for details on these macros.

The program must be loaded with the executable file access routine library `libld.a`.

SEE ALSO

`fseek(3S)`, `ldahread(3X)`, `ldclose(3X)`, `ldfhread(3X)`, `ldgetname(3X)`, `ldohseek(3X)`, `ldopen(3X)`, `ldshread(3X)`, `ldsseek(3X)`, `ldtbindex(3X)`, `ldtbread(3X)`, `ldtbseek(3X)`, `regexp(5)`, `intro(5)`.

NOTES

The executable file format is used only for executable files (load modules), not for object files.

WARNINGS

The macro `FSEEK` defined in the header file `ldfcn.h` translates into a call to the standard input/output function `fseek(3S)`. `FSEEK` should not be used to seek from the end of an archive file because the end of an archive file may not be the same as the end of one of its executable file members!

Note that `<ldfcn.h>` must not be included in a file where `<regexp.h>` is also included, as the macros defined in `<ldfcn.h>` conflict with the macros expected in `<regexp.h>`.

NAME

limits - header file for implementation-specific constants

SYNOPSIS

```
#include <limits.h>
```

DESCRIPTION

The header file `limits.h` is a list of minimal magnitude limitations imposed by a specific implementation of the operating system.

```
ARG_MAX      5120                /* max length of arguments to exec */
CHAR_BIT     8                   /* max # of bits in a "char" */
CHAR_MAX     255                 /* max value of a "char" */
CHAR_MIN     0                   /* min value of a "char" */
CHILD_MAX    25                  /* max # of processes per user id */
EDMC??
CLK_TCK      _sysconf(3)         /* clock ticks per second */
DBL_DIG      15                  /* digits of precision of a "double" */
DBL_MAX      1.79769313486223179E+308 /* max decimal value of a "double" */
DBL_MIN      2.2250738585071991E-308 /* min decimal value of a "double" */
FCHR_MAX     2147483647          /* max size of a file in bytes */
FLT_DIG      6                   /* digits of precision of a "float" */
FLT_MAX      3.40282347E+38F     /* max decimal value of a "float" */
FLT_MIN      1.17549435E-38F     /* min decimal value of a "float" */
HUGE_VAL     7.237005145973118E-75 /* error value returned by Math lib */
INT_MAX      2147483647          /* max value of an "int" */
INT_MIN      (-2147483647-1)     /* min value of an "int" */
LINK_MAX     1000                /* max # of links to a single file */
LOGNAME_MAX  8                   /* max # of characters in a login name */
LONG_BIT     32                  /* # of bits in a "long" */
LONG_MAX     2147483647          /* max value of a "long int" */
LONG_MIN     (-2147483647-1)     /* min value of a "long int" */
MAX_CANON    255                 /* max bytes in a line for canonical
processing */
MAX_INPUT    512                 /* max size of a char input buffer */
MB_LEN_MAX   5                   /* max # of bytes in a multibyte
character */
NAME_MAX     14                  /* max # of characters in a file name */
NGROUPS_MAX  16                  /* max # of groups for a user */
NL_ARGMAX    9                   /* max value of "digit" in calls to the
NLS printf() and scanf() */
NL_LANGMAX   14                  /* max # of bytes in a LANG name */
NL_MSGMAX    32767               /* max message number */
NL_NMAX      1                   /* max # of bytes in N-to-1 mapping
characters */
NL_SETMAX    255                 /* max set number */
NL_TEXTMAX   255                 /* max # of bytes in a message string */
NZERO        20                  /* default process priority */
OPEN_MAX     64                  /* max # of files a process can have
open */
PASS_MAX     8                   /* max # of characters in a password */
```

```

PATH_MAX      1023          /* max # of characters in a path name */
PID_MAX       30000        /* max value for a process ID */
PIPE_BUF      8192        /* max # bytes atomic in write to a pipe */
PIPE_MAX      8192        /* max # bytes written to a pipe
                           in a write */
SCHAR_MAX     127         /* max value of a "signed char" */
SCHAR_MIN     (-128)      /* min value of a "signed char" */
SHRT_MAX      32767       /* max value of a "short int" */
SHRT_MIN      (-32768)    /* min value of a "short int" */
STD_BLK       512        /* # bytes in a physical I/O block */
SYS_NMLN      256        /* 4.0 size of utsname elements */
               /* also defined in sys/utsname.h */
SYSPID_MAX    1          /* max pid of system processes */
TMP_MAX       17576      /* max # of unique names generated
                           by tmpnam */
UCHAR_MAX     255        /* max value of an "unsigned char" */
UID_MAX       60000      /* max value for a user or group ID */
UINT_MAX      4294967295 /* max value of an "unsigned int" */
ULONG_MAX     4294967295 /* max value of an "unsigned long int" */
USHRT_MAX     65535      /* max value of an "unsigned short int" */
USI_MAX       4294967295 /* max decimal value of an "unsigned" */
WORD_BIT      32        /* # of bits in a "word" or "int" */

```

The following POSIX definitions are the most restrictive values to be used by a POSIX conformant application. Conforming implementations shall provide values at least this large.

```

_POSIX_ARG_MAX      4096    /* max length of arguments to exec */
_POSIX_CHILD_MAX    6      /* max # of processes per user ID */
_POSIX_LINK_MAX     8      /* max # of links to a single file */
_POSIX_MAX_CANON    255    /* max # of bytes in a line of input */
_POSIX_MAX_INPUT    255    /* max # of bytes in terminal
                           input queue */
_POSIX_NAME_MAX     14     /* # of bytes in a filename */
_POSIX_NGROUPS_MAX  0      /* max # of groups in a process */
_POSIX_OPEN_MAX     16     /* max # of files a process can have open */
_POSIX_PATH_MAX     255    /* max # of characters in a pathname */
_POSIX_PIPE_BUF     512    /* max # of bytes atomic in write
                           to a pipe */

```

SEE ALSO

passwd(4).

NAME

linenum - line number entries in a common object file

SYNOPSIS

```
#include <linenum.h>
```

DESCRIPTION

When invoked with the `-g` option, the `cc` command generates an entry in the object file for each C source line on which a breakpoint is possible. debuggers such as `sdb(1)` and `dbx(1)` can then reference line numbers in the source. The structure of the line number entries appears below.

```
struct lineno
{
    union
    {
        long    Lsymndx ;
        long    Lpaddr ;
    }
    union
    {
        struct
        {
            unsigned short Llnno;
            unsigned short Lpad;
        } LL;
        long    Llnno;
    }
};
```

Numbering starts with 1 for each function. The initial line number entry for a function has `Llnno` equal to zero, and the symbol table index of the function's entry is in `Lsymndx`. Otherwise, `Llnno` is non-zero, and `Lpaddr` is the physical address of the code for the referenced line. Thus the overall structure is the following:

<i>Laddr</i>	<i>Llnno</i>
function symtab index	0
physical address	line
physical address	line
...	
function symtab index	0
physical address	line
physical address	line
...	

SEE ALSO

`cc(1)`, `sdb(1)`, `dbx(1)`, `a.out(4)`.

NAME

master – format of a master file

DESCRIPTION

Information about configurable kernel components is contained in a set of *master files* that are kept in the *master file directory* (by default, `/usr/etc/master.d`). This information is used by the `config(1M)` program to configure a kernel image. There are four types of configurable kernel components: device drivers, socket protocols, STREAMS modules, and tunable parameters.

Each layered kernel product available on the system has its own master file in the master file directory. For example, the TCP/IP product includes the master file `/usr/etc/master.d/tcpip`. The base DG/UX System itself uses `/usr/etc/master.d/dgux` as its master file. If you create your own device drivers or other configurable kernel components, you will need to create a new master file to supply information about the new components. Remember that every file found in the master file directory is examined when `config(1M)` is run, so backup or duplicate copies of master files should not be stored there, since they will cause errors when components are defined in more than one place. If you are not adding a new configurable component, you will probably only use the master files as reference when setting up your *system file* (see `system(4)`).

A *master file* can contain entries describing device drivers, socket protocols, STREAMS modules, tunable parameters, and aliases. Different types of information are grouped into their own sections with their own entry format. Each section is prefaced by a line containing a section name, whose first character is the dollar sign (\$). A master file may have any number (including zero) of each type of section, and they may appear in any order. Six different types of sections are supported:

<code>\$device</code>	Describes drivers for hardware devices and pseudo-devices.
<code>\$protocol</code>	Describes protocols that can be supported by the <code>socket(2)</code> system call.
<code>\$stream</code>	Describes STREAMS modules.
<code>\$keyword</code>	Describes user-tunable system parameters.
<code>\$alias</code>	Defines aliases for the keywords defined in any of the above types of sections. These aliases can then be used in a system file in place of the master file keywords.
<code>\$local_alias</code>	Defines constants for use only within the master file.

Each entry in a section consists of a single line broken into a number of fields separated by blanks and/or tabs. Comments are preceded by a pound sign (#) and can begin at any position on a line. Blank lines and comments are ignored.

Device Entries

Entries in a `$device` section have three fields:

- Field 1: Device name as specified in the system file. The kernel uses this name as a prefix to names for device driver routines in `conf.c`.
- Field 2: Restriction flags on this device. Flags are:
- o Only one device of this type is allowed.
 - r This device is required and will be automatically be configured into any kernels configured against this master file.

- s This device is a DG/UX-style STREAMS device.
- S This device is a System V-style STREAMS device.
- N This STREAMS device uses the new (System V.4) style open/close interface.
- z This device may be configured either explicitly or implicitly as part of a nested declaration of another device. For example, "st(insc(),4)" declares the device "insc()" implicitly.
- n No restrictions.

Field 3: STREAMS Concurrency Set. The concurrency set name specifies the STREAMS set to which a given STREAMS module or STREAMS device driver belongs. STREAMS concurrency only occurs within each set: modules or drivers belonging to the same set are guaranteed never to run concurrently. A set may contain drivers, modules, or both. Two exceptional cases allow for more concurrency: the pseudo-set named `module` means that each instance of such a STREAMS device or module will have its own private set; and the pseudo-set named `stream` means that locking is granular to the individual STREAMS themselves. All other set name values specify a named set. The concurrency set name has no meaning for non-STREAMS devices, which by convention are assigned to the set named `default`.

Protocol Entries

Entries in a `$protocol` section have six fields:

- Field 1: Name to be used in the system file to reference this protocol.
- Field 2: The protocol's protocol number as defined in the `/etc/protocols` file.
- Field 3: The protocol's domain number as defined in the `<sys/socket.h>` header file.
- Field 4: The protocol's type as defined in the `<sys/socket.h>` header file.
- Field 5: The *infix name*. The kernel will use this name to generate names for the protocol's control routines. You may use any name you want and then match this name with the names of your protocol control routines.
- Field 6: Restriction flags on this protocol. Flags are:
 - r This protocol is required and will be automatically be configured into any kernels configured against this master file.
 - d This protocol will be the default protocol used for `socket(2)` calls of the listed Domain and Type.
 - u This protocol is a UNIX domain protocol.
 - n No restrictions.

STREAMS Module Entries

Entries in a `$stream` section have four fields:

- Field 1: Name of the stream control module as given in the system file.
- Field 2: The *infix name*. The kernel will use this name to generate names for the stream's control module routines. You may use any name you want and then match this name with the names of your stream control module routines.

- Field 3: Restriction flags on this module. Flags are:
- N This STREAMS module uses the new (System V.4) style open/close interface.
 - n No restrictions.
- Field 4: STREAMS Concurrency Set. The concurrency set name specifies the STREAMS set to which a given STREAMS module or STREAMS device driver belongs. STREAMS concurrency only occurs within each set: modules or drivers belonging to the same set are guaranteed never to run concurrently. A set may contain drivers, modules, or both. Two exceptional cases allow for more concurrency: the pseudo-set named `module` means that each instance of such a STREAMS device or module will have its own private set; and the pseudo-set named `stream` means that locking is granular to the individual STREAMS themselves. All other set name values specify a named set.

Tunable Parameter Entries

Entries in a `$keyword` section have four fields, the last of which is optional:

- Field 1: Name of kernel variable to be set.
- Field 2: The default value that the variable will have, unless it is overridden in the system file.
- Field 3: The kernel variable's data type. This must not be a type that requires use of any header file besides `/usr/src/uts/aviion/ext/c_generics.h`.
- Field 4: The implied value for a variable that is listed in the system file without a value. This is useful for things like function pointers, whose value is represented by a string that would otherwise be inconvenient to type.

Alias Entries

Entries in an `$alias` section have two fields:

- Field 1: Alias name.
- Field 2: Name of master file entry being referenced.

Local Alias Entries

Entries in a `$local_alias` section have two fields:

- Field 1: Alias name.
- Field 2: The value which this alias name will have. This can be either a numeric or character string value.

SEE ALSO

`system(4)`.
`config(1M)`, `sysdef(1M)` in the *System Manager's Reference for the DG/UX System Installing the DG/UX System. Customizing the DG/UX System. Managing the DG/UX System.*

NAME

mfs – memory file system

DESCRIPTION

The DG/UX kernel provides support for memory file systems. These are file systems that live entirely in memory without any backing store on disk. Files in memory file systems do not persist between system instantiations. Memory file systems are faster than normal file systems and are ideal for temporary files and for putting common executables in them to avoid any disk I/O on execution. A memory file system has the same semantics as a normal DG/UX file system. Memory file systems can be NFS-exported just like regular DG/UX file systems.

A memory file system can be instantiated via the `mount(1M)` command:

```
mount -o ramdisk /dev/m1 /pdd/memory
```

The "ramdisk" option instructs the DG/UX file system to create a memory file system instead of trying to mount the device `/dev/m1` on the directory. The `/dev/m1` pseudo device must not exist at the time of the mount command. The pseudo device node will be created during the mount to reference the mounted on directory. Any naming convention can be used for this memory device with the exception that the name must reference a path in `/dev`. The example name `/pdd/memory` is the directory in the DG/UX file system hierarchy where the memory file system will be created. This may be any directory.

There are several options:

```
mount -o ramdisk,use_wired_memory /dev/m1 /pdd/memory
```

"use_wired_memory" is a boolean option that will instruct the file manager to use wired memory to hold data for the memory file system instead of unwired memory (the default is to use unwired memory). This is useful if you have lots of expansion memory for the file system, since data in the file system will always reside in memory and never be swapped out. (But see the CAUTIONS section below.)

```
mount -o ramdisk,max_file_space=20000 /dev/m1 /pdd/memory
```

"max_file_space=*n*" gives the number of blocks that can be allocated to the memory file system to hold data. No space is ever allocated up front, so using a high value will not lead to trouble. The amount of actual space that can be given to a memory file system is the minimum of the value assigned by this attribute and the total amount of the resource (wired or unwired memory) available on the system. If space is not available to allocate blocks to a memory file system, then the operation that requests space will return an ENOSPC result. The default amount of space allocated to a memory file system is 2048 blocks.

```
mount -o ramdisk,max_file_count=50000 /dev/m1 /pdd/memory
```

"max_file_count=*n*" gives the number file nodes that can be allocated in the memory file system. This is counted separately from the "max_file_space" attribute. The default number is 16384.

Memory file systems can be unmounted via the `umount(1M)` command:

```
umount /pdd/memory
```

The `umount` will not work until all the files have been removed from the file system. This is to protect against unintended data loss.

There is no limit to the number of memory file systems that may be created on a given system. Memory limitations, both wired and unwired, will ultimately govern how large they may grow.

SEE ALSO

exportfs(1M), mount(1M), umount(1M), fstab(4).

CAUTIONS

Do not over-commit the swap space available to the system. Because of the way DG/UX allocates memory, if you establish a large memory file system, start some very large application, then fill the memory file system, you might exhaust the swap space on the system. This will cause the system to thrash and to kill random processes in order to recover the swap space.

Do not mount a memory file system on `/tmp`, since the recovery mechanism of `ex(1)` and `vi(1)` depends on the persistence of temporary files in the `/tmp` directory.

Do not use the `use_wired_memory` option unless your system has enough expansion (physical) memory.

Use of the `use_wired_memory` option is also strongly discouraged on diskless workstations.

NAME

`mnttab` - mounted file system table

SYNOPSIS

```
#include <mntent.h>
```

DESCRIPTION

`mnttab` resides in the directory `/etc` and consists of a list of currently mounted file systems. The file contains a number of lines like this:

```
fsname dir type opts freq passno
```

for example:

```
/dev/dsk/usr /usr dg/ux rw 1 1
```

would indicate a mount for a local filesystem, and

```
titan:/usr/titan /usr/titan nfs rw,hard 0 0
```

would indicate an NFS filesystem mount. The entries from this file are accessed using the routines in `getmntent(3C)`, which returns a structure of the following form:

```
struct mntent {
    char *mnt_fsname; /* filesystem name */
    char *mnt_dir; /* filesystem path prefix */
    char *mnt_type; /* dg/ux, nfs, swap, cdrom, or ignore */
    char *mnt_opts; /* rw, ro, hard, soft, fg, bg, memory */
    int mnt_freq; /* highest dump level */
    int mnt_passno; /* pass number on parallel fsck */
};
```

Fields are separated by white space; a `#`, as the first non-white character, indicates a comment. The `mnt_type` field determines how the `mnt_fsname` and `mnt_opts` fields will be interpreted. The following is a list of the filesystem types currently supported, and the way each of them interprets these fields:

<i>Type</i>	<i>Field</i>	<i>Interpretation</i>
dg/ux	mnt_fsname	Must be a block special device.
	mnt_opts	Valid options are ro, rw, bg, and fg. If this has the ramdisk option, other options include use_wired_memory, max_file_space and max_file_count.
cdrom	mnt_fsname	Must be a block special device.
nfs	mnt_fsname	The hostname of the server and the pathname on the server of the directory to be served. A colon separates the pathname and hostname.
	mnt_opts	Valid options are ro, rw, hard, soft.
swap	mnt_fsname	Must be a block special device swap section.
	mnt_opts	Ignored.

If the *mnt_type* is specified as `ignore` then the entry is ignored. This is useful to show disks not currently used.

Entries identified as `swap` are made available as swap space by the `swapon(1M)` command at the end of the system reboot procedure.

When the *mnt_fsname* field is interpreted as a block special device, programs that require the corresponding character special device must construct the name by changing `dsk` to `rdsk` in the pathname.

If the *mnt_opts* field is a comma-separated list of options that includes `ro` or `rw`, then the filesystem is mounted read-write or read-only. If this includes `hard` or `soft`, then the NFS filesystem is mounted `hard` or `soft`.

The field *mnt_freq* indicates how often each filesystem should be dumped by the `dump(1M)` command (and triggers that command's `w` option, which determines what filesystems should be dumped). Most systems set the *mnt_freq* field to 1, indicating that filesystems are dumped each day.

The final field *mnt_passno* is used by the consistency checking program `fsck(1M)` to allow overlapped checking of filesystems during a reboot. All filesystems with a *mnt_passno* of 1 are checked first simultaneously, then all filesystems with *mnt_passno* of 2 are checked, and so on. The `<mnt_passno>` of the root filesystem should be 0, as the root cannot be checked since it is already mounted.

The maximum number of entries in `mnttab` is based on the system parameter `NMOUNT` located in `/usr/src/uts/mv/cf/config.h`, which defines the number of allowable mounted special files.

SEE ALSO

`mount(1M)`, `setmnt(1M)` in the *System Manager's Reference for the DG/UX System*.

NAME

netconfig – network configuration database

SYNOPSIS

```
#include <netconfig.h>
```

DESCRIPTION

The network configuration database, `/etc/netconfig`, is a system file used to store information about networks connected to the system and available for use. The `netconfig` database and the routines that access it [see `getnetconfig(3N)`] are part of the UNIX System V Network Selection component. The Network Selection component also includes the environment variable `NETPATH` and a group of routines that access the `netconfig` database using `NETPATH` components as links to the `netconfig` entries. `NETPATH` is described in `sh(1)`; the `NETPATH` access routines are discussed in `getnetpath(3N)`.

`netconfig` contains an entry for each network available on the system. Entries are separated by newlines. Fields are separated by whitespace and occur in the order in which they are described below. Whitespace can be embedded as “`\blank`” or “`\tab`”. Backslashes may be embedded as “`\\`”. Each field corresponds to an element in the `struct netconfig` structure. `struct netconfig` and the identifiers described on this manual page are defined in `/usr/include/netconfig.h`.

network ID

A string used to uniquely identify a network. *network ID* consists of non-null characters, and has a length of at least 1. No maximum length is specified. This namespace is locally significant and the local system administrator is the naming authority. All *network IDs* on a system must be unique.

semantics

The *semantics* field is a string identifying the “semantics” of the network, i.e., the set of services it supports, by identifying the service interface it provides. The *semantics* field is mandatory. The following semantics are recognized.

<code>tpi_clts</code>	Transport Provider Interface, connectionless
<code>tpi_cots</code>	Transport Provider Interface, connection oriented
<code>tpi_cots_ord</code>	Transport Provider Interface, connection oriented, supports orderly release.

flag The *flag* field records certain two-valued (“true” and “false”) attributes of networks. *flag* is a string composed of a combination of characters, each of which indicates the value of the corresponding attribute. If the character is present, the attribute is “true.” If the character is absent, the attribute is “false.” “_” indicates that none of the attributes is present. Only one character is currently recognized:

<code>v</code>	Visible (“default”) network. Used when the environment variable <code>NETPATH</code> is unset.
----------------	--

protocol family

The *protocol family* and *protocol name* fields are provided for protocol-specific applications.

The *protocol family* field contains a string that identifies a protocol family. The *protocol family* identifier follows the same rules as those for *network IDs*, that is, the string consists of non-null characters; it has a length of at least 1; and there is no maximum length specified. A “_” in the *protocol family* field

indicates that no protocol family identifier applies, that is, the network is experimental. The following are examples:

loopback	Loopback (local to host).
inet	Internetwork: UDP, TCP, etc.
implink	ARPANET imp addresses
pup	PUP protocols: e.g. BSP
chaos	MIT CHAOS protocols
ns	XEROX NS protocols
nbs	NBS protocols
ecma	European Computer Manufacturers Association
datakit	DATAKIT protocols
ccitt	CCITT protocols, X.25, etc.
sna	IBM SNA
decnet	DECNET
dli	Direct data link interface
lat	LAT
hylink	NSC Hyperchannel
appletalk	Apple Talk
nit	Network Interface Tap
ieee802	IEEE 802.2; also ISO 8802
osi	Umbrella for all families used by OSI (e.g., protosw lookup)
x25	CCITT X.25 in particular
osinet	AFI = 47, IDI = 4
gossip	U.S. Government OSI

protocol name

The *protocol name* field contains a string that identifies a protocol. The *protocol name* identifier follows the same rules as those for *network IDs*, that is, the string consists of non-NULL characters; it has a length of at least 1; and there is no maximum length specified. The following protocol names are recognized. A “-” indicates that none of the names listed applies.

tcp	Transmission Control Protocol
udp	User Datagram Protocol
icmp	Internet Control Message Protocol

network device

The *network device* is the full pathname of the device used to connect to the transport provider. Typically, this device will be in the /dev directory. The *network device* must be specified.

directory lookup libraries

The *directory lookup libraries* support a “directory service” (a name-to-address mapping service) for the network. This service is implemented by the UNIX System V Name-to-Address Mapping feature. If a network is not provided with such a library, the *netdir* feature will not work. A “-” in this field indicates the absence of any lookup libraries, in which case name-to-address mapping for the network is non-functional. The directory lookup library field consists of a comma-separated list of full pathnames to dynamically linked libraries. Commas may be embedded as “\,”; backslashes as “\\”.

Lines in /etc/netconfig that begin with a sharp sign (#) in column 1 are treated as comments.

The `struct netconfig` structure includes the following members corresponding to the fields in the `netconfig` database entries:

<code>char * nc_netid</code>	Network ID, including NULL terminator
<code>unsigned long nc_semantics</code>	Semantics
<code>unsigned long nc_flag</code>	Flags
<code>char * nc_protofmly</code>	Protocol family
<code>char * nc_proto</code>	Protocol name
<code>char * nc_device</code>	Full pathname of the network device
<code>unsigned long nc_nlookups</code>	Number of directory lookup libraries
<code>char ** nc_lookups</code>	Full pathnames of the directory lookup libraries themselves
<code>unsigned long nc_unused[9]</code>	Reserved for future expansion (not advertised to user level)

The `nc_semantics` field takes the following values, corresponding to the semantics identified above:

```
NC_TPI_CLTS
NC_TPI_COTS
NC_TPI_COTS_ORD
```

The `nc_flag` field is a bitfield. The following bit, corresponding to the attribute identified above, is currently recognized. `NC_NOFLAG` indicates the absence of any attributes.

```
NC_VISIBLE
```

FILES

```
/etc/netconfig
/usr/include/netconfig.h
```

SEE ALSO

```
netdir_getbyname(3N), getnetconfig(3N), getnetpath(3N), netconfig(4)
Network Programmer's Guide
System Administrator's Guide
```

NAME

netgroup – list of network groups

DESCRIPTION

netgroup defines network wide groups, used for permission checking when doing remote mounts, remote logins, and remote shells. For remote mounts, the information in netgroup is used to classify machines; for remote logins and remote shells, it is used to classify users. Each line of the netgroup file defines a group and has the format

```
groupname member1 member2 ....
```

where member_{*i*} is either another group name, or a triple:

```
(hostname, username, domainname)
```

Any of these three fields can be empty, in which case it signifies a wild card. Thus

```
universal ( , , )
```

defines a group to which everyone belongs.

A gateway machine should be listed under all possible hostnames by which it may be recognized:

```
wan (gateway, , ) (gateway-ebb, , )
```

Field names that begin with something other than a letter, digit or underscore (such as '-') work in precisely the opposite fashion. For example, consider the following entries:

```
justmachines(analytica,-,dgux)
justpeople (-,babbage,dgux)
```

The machine *analytica* belongs to the group *justmachines* in the domain *dgux*, but no users belong to it. Similarly, the user *babbage* belongs to the group *justpeople* in the domain *dgux*, but no machines belong to it.

The *domainname* field refers to the domain in which the triple is valid, not the name containing the trusted host.

FILES

/etc/netgroup

SEE ALSO

makedbm(1M), ypserv(1M), getnetgrent(3N), exports(4).

NAME

networks - network name database

DESCRIPTION

The `networks` file contains information on the networks known to your system. Each `networks` file should contain a single line for each network with the following information:

```
net_name net_number [ aliases ] [# comment ]
```

Items are separated by any number of blanks and/or tab characters. A # indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file. This file is normally created from the official network database maintained at the Network Information Center (NIC), though local changes may be required to update the file for unofficial aliases and/or unknown networks.

Network names may contain any printable character other than a field delimiter (blanks, tabs, CR, ESC), New Line, or comment character. Network numbers must be specified in four-part dot notation with trailing zeros omitted. For example, you would not specify 128.223 as 128.223.0.0.

If your system is using NIS, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System* for details.

EXAMPLE

```
loop-net 127 loop # s/w test net  
mfg-net 85 #mfg network
```

FILES

/etc/networks

SEE ALSO

getnetent(3N).

NAME

passwd – password file

SYNOPSIS

/etc/passwd

DESCRIPTION

The `passwd` file is an ASCII file containing basic information about each user's account. The file contains a one-line entry for each user allowed to log in to the system. Each entry has the following format:

```
username : password : uid : gid : gcos-field : home-dir : login-shell
```

where

<i>username</i>	User's login name. This field contains no uppercase characters, and must not be more than <code>USR_NAME</code> [see <code>limits(4)</code>] characters long.
<i>password</i>	The user's encrypted password. If this field is empty, <code>login(1)</code> does not request a password before logging the user in.
<i>uid</i>	The user's user identification number (UID) for the system. The UID must be unique; otherwise, users with the same UID will be able to access each other's files. <i>uid</i> is generally a value between 0 and 32767.
<i>gid</i>	The user's group identification number (GID) for the system. <i>gid</i> is generally a value between 0 and 32767.
<i>gcos-field</i>	The user's real name, along with information to pass along in a mail-message heading. Some system administrators use this field to contain the user's office, extension, home phone, and so on. It is called the GCOS field for historical reasons. An ampersand (&) in this field stands for the login name (in cases where the login name appears in a user's real name).
<i>home-dir</i>	The pathname of the directory to which the user is initially positioned when logging in.
<i>login-shell</i>	The user's initial shell program. If this field is empty, the default shell is <code>/bin/sh</code> unless you are running the Network File System (NFS); in that case the default is <code>/usr/bin/sh</code> .

Because the encrypted passwords on a secure system are kept in the `passwd.adjunct` file, `/etc/passwd` has general read permission on all systems, and can be used by routines that map UIDs to names.

The encrypted password consists of 13 characters chosen from a 64-character alphabet (`. , / , 0-9 , A-Z , a-z`), except when the password is null. In that case, the encrypted password is also null. Password aging is affected for a particular user if the user's encrypted password in the password file is followed by a comma and a non-null string of characters from the above alphabet (such a string must first be introduced by the superuser).

The first character of the age denotes the maximum number of weeks for which a password is valid. If you try to login after your password has expired, you must supply a new one. The next character denotes the minimum period in weeks that must elapse before the password may be changed. The remaining characters define the week (counted from the beginning of 1970) when the password was last changed (a

null string is equivalent to zero). The first and second characters have numerical values in the range 0-63 that correspond to the 64-character alphabet shown above (i.e., / = 1 week; z = 63 weeks). If both characters are equal to zero (derived from the string "." or ".."), you must change your password the next time you login. The age will disappear from your entry in the password file. If the second character is greater than the first (signified, e.g., by the string "./"), then only the superuser will be able to change the password.

ONC/NFS Features

If you are using DG/UX Open Network Computing/Network File System (ONC/NFS), additional features are available. The `passwd` file can also have lines beginning with a plus (+), which means to incorporate entries from the Network Information Service (NIS).

There are three styles of + entries: by itself, + means to insert the entire contents of the NIS password file at that point; `+name` means to insert the entry (if any) for `name` from NIS at that point; `+@netgroup` means to insert the entries for all members of the network group `netgroup` at that point. If a `+name` entry has a non-null `password`, `gcos-field`, `home-dir`, or `login-shell` field, they will override what is contained in NIS. The `uid` and `gid` fields cannot be overridden.

Entries beginning with a minus sign (-) are also allowed. They have two formats: `-name` and `-@name`. The meaning of these formats is the same as for `+name` and `+@name`, respectively, except that the action is reversed; all members matched are considered to be excluded from the password file, regardless of subsequent entries. Minus entries can be used to exclude specific entries from NIS.

Appropriate precautions must be taken to lock the `/etc/passwd` file against simultaneous changes if it is to be edited with a text editor; `vipw(1M)` does the necessary locking.

EXAMPLE

Here is a sample `/etc/passwd` file:

```
root:q.mJzTnu8icF.:0:10:God:/:/bin/csh
tut:6k/7KCFRPNVXg:508:10:Bill Tuthill:/usr/tut:/bin/csh
+john:
-@documentation:no-login:
+:::Guest
john::605:20:John Smith:/usr/john:
```

In this example, there are specific entries for users `root` and `tut`, in case NIS is not running. (See *Managing ONC/NFS and Its Facilities on the DG/UX System.*) The user `john` will have his password entry in NIS incorporated without change; anyone in the `netgroup documentation` will have their password field disabled, and anyone else will be able to login with their usual password, shell, and home directory, but with a GCOS field of `Guest`.

The second entry for `john` in this example will not be used if NIS is running; the first entry for a given user name will be used if multiple entries exist.

Appropriate precautions must be taken to lock the `/etc/passwd` file against simultaneous changes if it is to be edited with a text editor; `vipw(1M)` does the necessary locking. The password file can be scanned for inconsistencies using `pwck(1M)`.

ONC/NFS Example

The following example relates to ONC/NFS and NIS:

```
root:q.mJzTnu8icF.:0:10:Super User:/:/bin/csh
fred:6k/7KCFRPNVXg:508:10:% Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+:::Guest
```

In this example, there are specific entries for users `root` and `fred`, to assure that they can log in even when the system is running standalone. The user `john` will have his password entry in the Network Information Service incorporated without change; anyone in the netgroup `documentation` will have their password field disabled, and anyone else will be able to log in with their usual password, shell, and home directory, but with a GCOS field of `Guest`.

FILES

/etc/passwd

SEE ALSO

`login(1)`, `mail(1)`, `passwd(1)`, `pwck(1M)`, `sendmail(1M)`, `useradd(1M)`, `vipw(1M)`, `crypt(3C)`, `crypt(3X)`, `getpwent(3C)`, `group(4)`, `limits(4)`.

BUGS

The `mail(1)` and `sendmail(1M)` programs use the GCOS field to compose the `From:` line for addressing mail messages, but these programs get confused by nested parentheses when composing replies. This problem can be avoided by using different types of brackets within the GCOS field; for example:

```
(& Fredricks [Podunk U <EE/CIS>] {818}-555-5555)
```

NAME

`pkginfo` – package characteristics file

DESCRIPTION

`pkginfo` is an ASCII file that describes the characteristics of the package along with information that helps control the flow of installation. It is created by the software package developer.

Each entry in the `pkginfo` file is a line that establishes the value of a parameter in the following form:

PARAM="value"

There is no required order in which the parameters must be specified within the file. Each parameter is described below. Only fields marked with an asterisk are mandatory.

- PKG**** Abbreviation for the package being installed, generally three characters in length (for example, `dir` or `pkg`). All characters in the abbreviation must be alphanumeric and the first may not be numeric. The abbreviation is limited to a maximum length of nine characters. `install`, `new`, and `all` are reserved abbreviations.
- NAME**** Text that specifies the package name (maximum length of 256 ASCII characters).
- ARCH**** A comma-separated list of alphanumeric tokens that indicate the architecture (for example, `3B2`) associated with the package. The `pkgmk` tool may be used to create or modify this value when actually building the package. The maximum length of a token is 16 characters and it cannot include a comma.
- VERSION**** Text that specifies the current version associated with the software package. The maximum length is 256 ASCII characters and the first character cannot be a left parenthesis. The `pkgmk` tool may be used to create or modify this value when actually building the package.
- CATEGORY**** A comma-separated list of categories under which a package may be displayed. A package must at least belong to the system or application category. Categories are case-insensitive and may contain only alphanumerics. Each category is limited in length to 16 characters.
- DESC*** Text that describes the package (maximum length of 256 ASCII characters).
- VENDOR*** Used to identify the vendor that holds the software copyright (maximum length of 256 ASCII characters).
- HOTLINE*** Phone number and/or mailing address where further information may be received or bugs may be reported (maximum length of 256 ASCII characters).
- EMAIL*** An electronic address where further information is available or bugs may be reported (maximum length of 256 ASCII characters).
- VSTOCK*** The vendor stock number, if any, that identifies this product (maximum length of 256 ASCII characters).
- CLASSES*** A space-separated list of classes defined for a package. The order of the list determines the order in which the classes are installed. Classes listed first will be installed first (on a media by media basis).

	This parameter may be modified by the request script.
<i>ISTATES</i>	A list of allowable run states for package installation (for example, "S s 1").
<i>RSTATES</i>	A list of allowable run states for package removal (for example, "S s 1").
<i>BASEDIR</i>	The pathname to a default directory where "relocatable" files may be installed. If blank, the package is not relocatable and any files that have relative pathnames will not be installed. An administrator can override the default directory.
<i>ULIMIT</i>	If set, this parameter is passed as an argument to the <code>ulimit</code> command, which establishes the maximum size of a file during installation.
<i>ORDER</i>	A list of classes defining the order in which they should be put on the medium. Used by <code>pkgmk</code> in creating the package. Classes not defined in this field are placed on the medium using the standard ordering procedures.
<i>MAXINST</i>	The maximum number of package instances that should be allowed on a machine at the same time. By default, only one instance of a package is allowed. This parameter must be set in order to have multiple instances of a package.
<i>PSTAMP</i>	Production stamp used to mark the <code>pkgmap</code> file on the output volumes. Provides a means for distinguishing between production copies of a version if more than one is in use at a time. If <code>PSTAMP</code> is not defined, the default is used. The default consists of the UNIX system machine name followed by the string "YYMMDDHHMM" (year, month, date, hour, minutes).
<i>INTONLY</i>	Indicates that the package should only be installed interactively when set to any non-NULL value.
<i>PREDEPEND</i>	Used to maintain compatibility with pre-SVR4 package dependency checking. Pre-SVR4 dependency checks were based on whether or not the name file for the required package existed in the <code>/var/options</code> directory. This directory is not maintained for SVR4 packages since the <code>depend</code> file is used for checking dependencies. However, entries can be created in this directory to maintain compatibility. Setting the <code>PREDEPEND</code> parameter to <code>y</code> or <code>yes</code> creates a <code>/usr/option</code> entry for the package. (Packages that are new for SVR4 do not need to use this parameter.)

EXAMPLES

Here is a sample `pkginfo`:

```

PKG="oam"
NAME="OAM Installation Utilities"
VERSION="3"
VENDOR="AT&T"
HOTLINE="1-800-ATT-BUGS"
EMAIL="attunix!olsen"
VSTOCK="0122c3f5566"
CATEGORY="system.essential"
ISTATES="S 2"
RSTATES="S 2"

```

SEE ALSO

compver(4), copyright(4), depend(4), pkgmap(4).

NOTES

Developers may define their own installation parameters by adding a definition to this file. A developer-defined parameter must begin with a capital letter,

NAME

pkgmap – package contents description file

DESCRIPTION

pkgmap is an ASCII file that provides a complete listing of the package contents. It is automatically generated by **pkgmk(1)** using the information in the **prototype** file.

Each entry in **pkgmap** describes a single “deliverable object file.” A deliverable object file includes shell scripts, executable objects, data files, directories, etc. The entry consists of several fields of information, each field separated by a space. The fields are described below and must appear in the order shown.

part An optional field designating the part number in which the object resides. A part is a collection of files, and is the atomic unit by which a package is processed. A developer can choose the criteria for grouping files into a part (e.g., based on class). If no value is defined in this field, part 1 is assumed.

f_{type} A one-character field that indicates the file type. Valid values are:

- f** a standard executable or data file
- e** a file to be edited upon installation or removal
- v** volatile file (one whose contents are expected to change)
- d** directory
- x** an exclusive directory
- l** linked file
- p** named pipe
- c** character special device
- b** block special device
- i** installation script or information file
- s** symbolic link

class The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. It is not specified if the **f_{type}** is **i** (information file).

pathname The pathname where the object will reside on the target machine, such as **/usr/bin/mail**. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable.

For linked files (**f_{type}** is either **l** or **s**), **pathname** must be in the form of **path1=path2**, with **path1** specifying the destination of the link and **path2** specifying the source of the link.

pathname may contain variables which support relocation of the file. A **\$parameter** may be embedded in the pathname structure. **\$BASEDIR** can be used to identify the parent directories of the path hierarchy, making the entire package easily relocatable. Default values for **parameter** and **BASEDIR** must be supplied in the **pkginfo** file and may be overridden at installation.

major The major device number. The field is only specified for block or character special devices.

minor The minor device number. The field is only specified for block or character special devices.

mode The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files,

packaging information files or non-installable files.

owner The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what owner an installation script will be executed.

Can be a variable specification in the form of `#[A-Z]`. Will be resolved at installation time.

group The group to which the file belongs (for example, `"bin"` or `"sys"`). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what group an installation script will be executed.

Can be a variable assignment in the form of `#[A-Z]`. Will be resolved at installation time.

size The actual size of the file in bytes. This field is not specified for named pipes, special devices, directories or linked files.

cksum The checksum of the file contents. This field is not specified for named pipes, special devices, directories or linked files.

modtime The time of last modification, as reported by the `stat(2)` function call. This field is not specified for named pipes, special devices, directories or linked files.

Each `pkgmap` must have one line that provides information about the number and maximum size (in 512-byte blocks) of parts that make up the package. This line is in the following format:

```
:number_of_parts maximum_part_size
```

Lines that begin with `"#"` are comment lines and are ignored.

When files are saved during installation before they are overwritten, they are normally just copied to a temporary pathname. However, for files whose mode includes execute permission (but which are not editable), the existing version is linked to a temporary pathname and the original file is removed. This allows processes which are executing during installation to be overwritten.

EXAMPLES

The following is an example of a `pkgmap` file.

```
:2 500
1 i pkginfo 237 1179 541296672
1 b class1 /dev/diskette 17 134 0644 root other
1 c class1 /dev/rdiskette 17 134 0644 root other
1 d none bin 0755 root bin
1 f none bin/INSTALL 0755 root bin 11103 17954 541295535
1 f none bin/REMOVE 0755 root bin 3214 50237 541295541
1 l none bin/UNINSTALL=bin/REMOVE
1 f none bin/cmda 0755 root bin 3580 60325 541295567
1 f none bin/cmdb 0755 root bin 49107 51255 541438368
1 f class1 bin/cmdc 0755 root bin 45599 26048 541295599
```

```
1 f class1 bin/cmdd 0755 root bin 4648 8473 541461238
1 f none bin/cmde 0755 root bin 40501 1264 541295622
1 f class2 bin/cmdf 0755 root bin 2345 35889 541295574
1 f none bin/cmdg 0755 root bin 41185 47653 541461242
2 d class2 data 0755 root bin
2 p class1 data/apipe 0755 root other
2 d none log 0755 root bin
2 v none log/logfile 0755 root bin 41815 47563 541461333
2 d none save 0755 root bin
2 d none spool 0755 root bin
2 d none tmp 0755 root bin
```

SEE ALSO

pkginfo(4).

NOTES

The pkgmap file may contain only one entry per unique pathname.

NAME

profile - setting up an environment at login time

DESCRIPTION

If you are using the Bourne shell and your login directory contains a file named `.profile`, that file will be executed (via `exec .profile`) before your session begins; `.profiles` are handy for setting exported environment variables and terminal modes. If the file `/etc/profile` exists, it will be executed for every user before the `.profile`. The following example is typical (except for the comments):

```
# Make some environment variables global
export MAIL PATH
# Set file creation mask
umask 22
# Tell me when new mail comes in
MAIL=/usr/mail/myname
# Add my /bin directory to the shell search sequence
PATH=$PATH:$HOME/bin
```

FILES

```
$HOME/.profile
/etc/profile
```

SEE ALSO

`environ(5)`, `term(5)`.
`env(1)`, `login(1)`, `mail(1)`, `sh(1)`, `stty(1)`, `su(1)` in the *User's Reference for the DG/UX System*.

NAME

protocols - protocol name database

DESCRIPTION

The protocols file contains information about the known protocols used in the networks. Each protocol should have a one-line entry in the protocols file with the following information:

```
official protocol name
protocol number
aliases (optional)
# comment (optional)
```

Items are separated by any number of blanks and/or tab characters. A # indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file. Use only decimal numerals to specify protocol numbers.

Names in the protocols file may contain any printable character other than a field delimiter (blanks, tabs, CR, ESC), New Line, or comment character.

If your system is using the Network Information System (NIS), see Chapters 3 and 4 in *Managing ONCTM/NFS[®] and Its Facilities on the DG/UXTM System* for information on how to update the protocols database.

EXAMPLES

```
tcp 6 TCP # transmission control protocol
```

FILES

```
/etc/protocols
```

SEE ALSO

```
getprotoent(3N).
```

NAME

prototype – package information file

DESCRIPTION

prototype is an ASCII file used to specify package information. Each entry in the file describes a single deliverable object. An object may be a data file, directory, source file, executable object, etc. This file is generated by the package developer.

Entries in a prototype file consist of several fields of information separated by white space. Comment lines begin with a “#” and are ignored. The fields are described below and must appear in the order shown.

part An optional field designating the part number in which the object resides. A part is a collection of files, and is the atomic unit by which a package is processed. A developer can choose criteria for grouping files into a part (e.g., based on class). If this field is not used, part 1 is assumed.

f_{type} A one-character field which indicates the file type. Valid values are:

- f a standard executable or data file
- e a file to be edited upon installation or removal
- v volatile file (one whose contents are expected to change)
- d directory
- x an exclusive directory
- l linked file
- p named pipe
- c character special device
- b block special device
- i installation script or information file
- s symbolic link

class The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. The field is not specified for installation scripts. (admin and all classes beginning with capital letters are reserved class names.)

pathname The pathname where the file will reside on the target machine, e.g., /usr/bin/mail or bin/ras_proc. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable. The form

path1=path2

may be used for two purposes: to define a link and to define local pathnames.

For linked files, *path1* indicates the destination of the link and *path2* indicates the source file. (This format is mandatory for linked files.)

For local pathnames, *path1* indicates the pathname an object should have on the machine where the entry is to be installed and *path2* indicates either a relative or fixed pathname to a file on the host machine which contains the actual contents.

A pathname may contain a variable specification, which will be resolved at the time of installation. This specification should have the form $\$[A-Z]$.

major The major device number. The field is only specified for block or character special devices.

minor The minor device number. The field is only specified for block or character special devices.

- mode* The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.
- owner* The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.
- Can be a variable specification in the form of `$(A-Z)`. Will be resolved at installation time.
- group* The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.
- Can be a variable specification in the form of `$(A-Z)`. Will be resolved at installation time.

An exclamation point (!) at the beginning of a line indicates that the line contains a command. These commands are used to incorporate files in other directories, to locate objects on a host machine, and to set permanent defaults. The following commands are available:

- search* Specifies a list of directories (separated by white space) to search for when looking for file contents on the host machine. The basename of the *path* field is appended to each directory in the ordered list until the file is located.
- include* Specifies a pathname which points to another prototype file to include. Note that *search* requests do not span *include* files.
- default* Specifies a list of attributes (mode, owner, and group) to be used by default if attribute information is not provided for prototype entries which require the information. The defaults do not apply to entries in *include* prototype files.
- param=value* Places the indicated parameter in the current environment.

The above commands may have variable substitutions embedded within them, as demonstrated in the two example prototype files below.

Before files are overwritten during installation, they are copied to a temporary pathname. The exception to this rule is files whose mode includes execute permission, unless the file is editable (i.e., *ftype* is `e`). For files which meet this exception, the existing version is linked to a temporary pathname, and the original file is removed. This allows processes which are executing during installation to be overwritten.

EXAMPLES

Example 1:

```
!PROJDIR=/usr/proj
!BIN=$PROJDIR/bin
!CFG=$PROJDIR/cfg
!LIB=$PROJDIR/lib
!HDRS=$PROJDIR/hdrs
```

```

!search /usr/myname/usr/bin /usr/myname/src /usr/myname/hdrs
i pkginfo=/usr/myname/wrap/pkginfo
i depend=/usr/myname/wrap/depend
i version=/usr/myname/wrap/version
d none /usr/wrap 0755 root bin
d none /usr/wrap/usr/bin 0755 root bin
! search $BIN
f none /usr/wrap/bin/INSTALL 0755 root bin
f none /usr/wrap/bin/REMOVE 0755 root bin
f none /usr/wrap/bin/addpkg 0755 root bin
!default 755 root bin
f none /usr/wrap/bin/audit
f none /usr/wrap/bin/listpkg
f none /usr/wrap/bin/pkgmk
# the following file starts out zero length but grows
v none /usr/wrap/logfile=/dev/null 0644 root bin
# the following specifies a link (dest=src)
l none /usr/wrap/src/addpkg=/usr/wrap/bin/rmpkg
! search $SRC
!default 644 root other
f src /usr/wrap/src/INSTALL.sh
f src /usr/wrap/src/REMOVE.sh
f src /usr/wrap/src/addpkg.c
f src /usr/wrap/src/audit.c
f src /usr/wrap/src/listpkg.c
f src /usr/wrap/src/pkgmk.c
d none /usr/wrap/data 0755 root bin
d none /usr/wrap/save 0755 root bin
d none /usr/wrap/spool 0755 root bin
d none /usr/wrap/tmp 0755 root bin
d src /usr/wrap/src 0755 root bin

```

Example 2:

```

# this prototype is generated by 'pkgproto' to refer
# to all prototypes in my src directory
!PROJDIR=/usr/dew/projx
!include $PROJDIR/src/cmd/prototype
!include $PROJDIR/src/cmd/audmerg/protofile
!include $PROJDIR/src/lib/proto

```

SEE ALSO

pkginfo(4), pkgmk(1).

NOTES

Normally, if a file is defined in the prototype file but does not exist, that file is created at the time of package installation. However, if the file pathname includes a directory that does not exist, the file will not be created. For example, if the prototype file has the following entry:

```
f none /usr/dev/bin/command
```

and that file does not exist, it will be created if the directory /usr/dev/bin already exists or if the prototype also has an entry defining the directory:

```
d none /usr/dev/bin
```

NAME

publickey – public key database

SYNOPSIS

/etc/publickey

DESCRIPTION

NOTE: Secure RPC using DES Authentication is an additional feature that must be purchased separately from the DG/UX™ ONC™/NFS® package. You must have this feature to use the database described in this manual page.

/etc/publickey is the public key database used for secure networking. Each entry in the database consists of a network user name (which may either refer to a user or a hostname), followed by the user's public key (in hex notation), a colon, and then the user's secret key encrypted with its login password (also in hex notation).

This file is altered either by the user through the `chkey(1)` command or by the system administrator through the `newkey(1M)` command. The file `/etc/publickey` should only contain data on the Network Information Service master machine, where it is converted into the NIS database `publickey.byname`.

SEE ALSO

`chkey(1)`, `newkey(1M)`, `ypupdated(1M)`, `des_crypt(3R)`, `publickey(3R)`, `rpc(3N)`.

NAME

rcsfile - format of RCS file

DESCRIPTION

An RCS file's contents are described by the grammar below.

The text is free format: space, backspace, tab, newline, vertical tab, form feed, and carriage return (collectively, *white space*) have no significance except in strings. However, an RCS file must end in a newline character.

Strings are enclosed by @. If a string contains a @, it must be doubled; otherwise, strings may contain arbitrary binary data.

The meta syntax uses the following conventions: '|' (bar) separates alternatives; '{' and '}' enclose optional phrases; '{' and '*}' enclose phrases that may be repeated zero or more times; '{' and '+}' enclose phrases that must appear at least once and may be repeated; Terminal symbols are in **boldface**; nonterminal symbols are in *italics*.

```

rcstext ::= admin {delta}* desc {deltatext}*
admin   ::= head      {num};
          { branch   {num}; }
          access     {id}*;
          symbols    {id : num}*;
          locks      {id : num}*; {strict ;}
          { comment  {string}; }
          { expand   {string}; }
          { newphrase }*

delta   ::= num
          date       num;
          author     id;
          state      {id};
          branches   {num}*;
          next       {num};
          { newphrase }*

desc    ::= desc     string

deltatext ::= num
           log       string
           { newphrase }*
           text      string

num     ::= {digit{.} }+
digit   ::= 0 | 1 | ... | 9
id      ::= letter{idchar}*
letter  ::= any letter
idchar  ::= any visible graphic character except special
special ::= $ | , | . | : | ; | @
string  ::= @{any character, with @ doubled}*@
newphrase ::= id word* ;
word     ::= id | num | string | :

```

Identifiers are case sensitive. Keywords are in lower case only. The sets of keywords and identifiers may overlap. In most environments RCS uses the ISO 8859/1 encoding: letters are octal codes 101–132, 141–172, 300–326, 330–366 and 370–377, visible graphic characters are codes 041–176 and 240–377, and white space characters are codes 010–015 and 040.

The *newphrase* productions in the grammar are reserved for future extensions to the format of RCS files. No *newphrase* will begin with any keyword already in use.

The *delta* nodes form a tree. All nodes whose numbers consist of a single pair (e.g., 2.3, 2.1, 1.3, etc.) are on the trunk, and are linked through the `next` field in order of decreasing numbers. The `head` field in the *admin* node points to the head of that sequence (i.e., contains the highest pair). The `branch` node in the *admin* node indicates the default branch (or revision) for most RCS operations. If empty, the default branch is the highest branch on the trunk.

All *delta* nodes whose numbers consist of $2n$ fields (n) (e.g., 3.1.1.1, 2.1.2.2, etc.) are linked as follows. All nodes whose first $2n-1$ number fields are identical are linked through the `next` field in order of increasing numbers. For each such sequence, the *delta* node whose number is identical to the first $2n-2$ number fields of the deltas on that sequence is called the branchpoint. The `branches` field of a node contains a list of the numbers of the first nodes of all sequences for which it is a branchpoint. This list is ordered in increasing numbers.

Example:

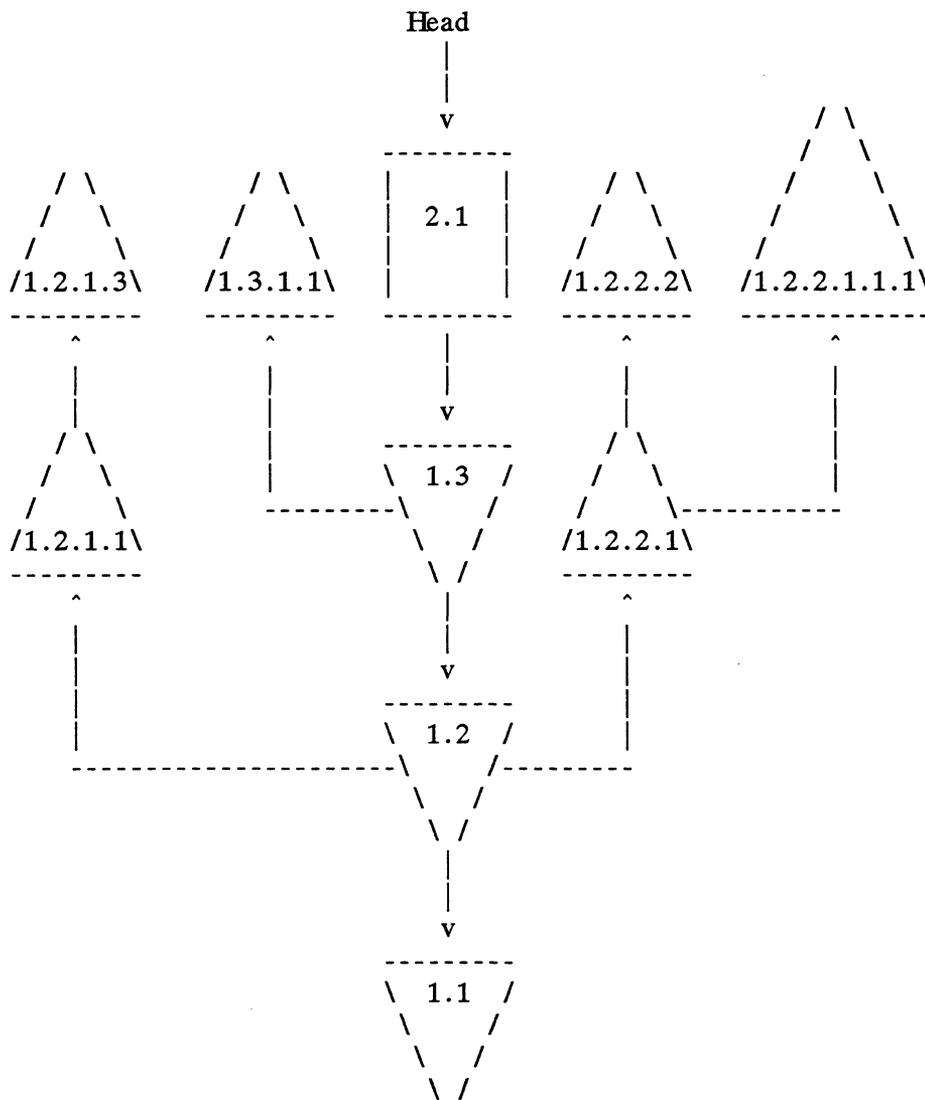


Fig. 1: A revision tree

IDENTIFICATION

Author: Walter F. Tichy, Purdue University, West Lafayette, IN, 47907.

Revision Number: 4.1.1.8; Release Date: 1992/01/07.

Copyright © 1982, 1988, 1989 by Walter F. Tichy.

Copyright © 1990, 1991 by Paul Eggert.

SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsmerge(1), rlog(1),
 Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* 15, 7 (July 1985), 637-654.

NAME

reloc – relocation information for a common object file

SYNOPSIS

```
#include <reloc.h>
```

DESCRIPTION

Common object (COFF) files have one relocation entry for each relocatable reference in the text or data. If relocation information is present, it will be in the following format:

```
struct reloc
{
    long    r_vaddr ; /* (virtual) address of reference */
    long    r_symndx ; /* index into symbol table */
    ushort  r_type ; /* relocation type */
    unsigned short r_offset; /* high 16 bits of expression*/
} ;

#define    R_ABS      0
#define    R_PCR16L   128
#define    R_PCR26L   129
#define    R_VRT16    130
#define    R_HVRT16   131
#define    R_LVRT16   132
#define    R_VRT32    133
```

As the link editor reads each input section and performs relocation, the relocation entries are read. They direct how references found within the input section are treated.

R_ABS The reference is absolute and no relocation is necessary. The entry will be ignored.

R_PCR16L A "PC-relative" 16-bit reference to the symbol's virtual address.

R_PCR26L A "PC-relative" 26-bit reference to the symbol's virtual address.

R_VRT16 Direct 16-bit reference to the symbol's virtual address.

R_HVRT16 Same as **R_VRT16**, except, only the high 16 bits are used in the relocation.

R_LVRT16 Same as **R_VRT16**, except, only the low 16 bits are used in the relocation.

R_VRT32 Direct 32-bit reference to the symbol's virtual address.

Relocation entries are generated automatically by the assembler and automatically used by the link editor. Link editor options exist for both preserving and removing the relocation entries from object files.

SEE ALSO

as(1), ld-coff(1), a.out(4), syms(4).

NAME

rpc - rpc program number data base

SYNOPSIS

/etc/rpc

DESCRIPTION

The rpc file contains user readable names that can be used in place of rpc program numbers. Each line has the following information:

name of server for the rpc program
 rpc program number
 aliases

Items are separated by any number of blanks and/or tab characters. A “#” indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Here is an example of a partial /etc/rpc file from the DG/UX System.

```
# $What: <@(#) rpc.4,v 4.1.1.7> $
portmapper      100000    portmap sunrpc
rstatd          100001    rstat rup perfmeter
rusersd        100002    rusers
nfs             100003    nfsprog
ypserv         100004    ypprog
mountd         100005    mount showmount
ypbind         100007
walld          100008    rwall shutdown
yppasswdd      100009    yppasswd
etherstatd     100010    etherstat
rquotad        100011    rquotaprog quota rquota
sprayd         100012    spray
3270_mapper    100013
rje_mapper     100014
selection_svc  100015    selnsvc
database_svc   100016
rex            100017    rex
alis           100018
sched          100019
llockmgr       100020
nlockmgr       100021
x25.inr        100022
statmon        100023
status         100024
bootparam     100026
ypupdated      100028    yppupdate
keyserver     100029    keyserver
tfsd           100037
nsed           100038
```

FILES

/etc/rpc

SEE ALSO

getrpcent(3N), rpc(3N).

NAME

sccsfile - format of SCCS file

DESCRIPTION

An SCCS file is an ASCII file. It consists of six logical parts:

checksum

delta table information about each delta

user names

login names and/or numerical group IDs of users who may add deltas

flags definitions of internal keywords

comments arbitrary descriptive information about the file

body the actual text lines intermixed with control lines

Throughout an SCCS file there are lines that begin with the ASCII SOH (start of heading) character (octal 001). We call this character *the control character*, and represent it graphically as @. Any line described below that does not begin with the control character is prevented from doing so.

Entries of the form DDDDD represent a five-digit string (a number between 00000 and 99999).

Each logical part of an SCCS file is described in detail below.

Checksum

The checksum is the first line of an SCCS file. The form of the line is:

@hDDDDD

The value of the checksum is the sum of all characters, except those of the first line. The @h provides a *magic number* of (octal) 064001.

Delta table

The delta table consists of a variable number of entries of the form:

@s DDDDD/DDDDD/DDDDD

@d *type* <SCCS ID> yr/mo/da hr:mi:se *pgmr* DDDDD DDDDD

@i DDDDD ...

@x DDDDD ...

@g DDDDD ...

@m <MR number>

.

.

.

@c *comments* ...

.

.

.

@e

The first line (@s) contains the number of lines inserted/deleted/unchanged. The second line (@d) contains the type of the delta (currently, normal: D, and removed: R); the SCCS ID of the delta; the date and time of creation of the delta; the login name corresponding to the real user ID at the time the delta was created; and the serial numbers of the delta and its predecessor

The @i, @x, and @g lines are optional; they contain the serial numbers of deltas included, excluded, and ignored, respectively.

The @m lines (optional) each contain one MR number associated with the delta; the @c lines contain comments associated with the delta.

The @e line ends the delta table entry.

User names

The list of login names and/or numerical group IDs of users who may add deltas to the file, separated by new-lines. The lines containing these login names and/or numerical group IDs are surrounded by the bracketing lines @u and @U. An empty list lets anyone to make a delta. Any line starting with a ! prohibits the succeeding group or user from making deltas.

Flags

Keywords used internally (see admin(1) for more information on their use). Each flag line takes the form:

```
@f flag<optional text>
```

The following flags are defined:

```
@f t <type of program>
@f v <program name>
@f i <keyword string>
@f b
@f m <module name>
@f f <floor>
@f c <ceiling>
@f d <default-sid>
@f n
@f j
@f l <lock-releases>
@f q <user defined>
@f z <reserved for use in interfaces>
```

The t flag defines the replacement for the %Y% identification keyword. The v flag controls prompting for MR numbers as well as comments; if the optional text is present it defines an MR number validity checking program.

The i flag controls the warning/error aspect of the No id keywords message. When the i flag is not present, this message is only a warning; when the i flag is present, this message will cause a fatal error; the file will not be gotten, or the delta will not be made.

When the b flag is present the -b keyletter may be used on the get command to cause a branch in the delta tree.

The m flag defines the first choice for the replacement text of the %M% identification keyword. The f flag defines the the release below which no deltas may be added (also known as the floor release).

The c flag defines the the release above which no deltas may be added (also known as the ceiling release).

The *d* flag defines the default SID to be used when none is specified on a *get* command.

The *n* flag causes *delta* to insert a null delta (a delta that applies *no* changes) in those releases that are skipped when a delta is made in a *new* release (e.g., when delta 5.1 is made after delta 2.7, releases 3 and 4 are skipped). The absence of the *n* flag causes skipped releases to be completely empty.

The *j* flag causes *get* to allow concurrent edits of the same base SID.

The *l* flag defines a *list* of releases that are *locked* against editing (*get*(1) with the *-e* keyletter).

The *q* flag defines the replacement for the *%Q%* identification keyword.

The *z* flag is used in certain specialized interface programs.

Comments

Arbitrary text is surrounded by the bracketing lines *@t* and *@T*. The comments section typically will contain a description of the file's purpose.

Body

The body consists of text lines and control lines. Text lines do not begin with the control character, control lines do. There are three kinds of control lines:

<i>@I</i> DDDDD	Insert
<i>@D</i> DDDDD	Delete
<i>@E</i> DDDDD	End

The digit string is the serial number corresponding to the delta for the control line.

SEE ALSO

admin(1), *delta*(1), *get*(1), *prs*(1) in the *User's Reference for the DG/UX System*.

NAME

scr_dump - format of curses screen image file

SYNOPSIS

scr_dump(*file*)

DESCRIPTION

The `curses(3X)` function `scr_dump()` copies the contents of the screen into a file. The format of the screen image is as described below.

The name of the tty is 20 characters long and the modification time (the *mtime* of the tty that this is an image of) is of the type *time_t*. All other numbers and characters are stored as `ctype` (see `<curses.h>`). No newlines are stored between fields.

```

<magic number: octal 0433>
<name of tty>
<mod time of tty>
columns <lines>
<line length> <chars in line>   for each line on the screen
<line length> <chars in line>
.
.
.
<labels?>                       1, if soft screen labels are present
<cursor row> <cursor column>

```

Only as many characters as are in a line will be listed. For example, if the *<line length>* is 0, there will be no characters following *<line length>*. If *<labels?>* is TRUE, following it will be

```

<number of labels>
<label width>
<chars in label 1>
<chars in label 2>
.
.
.

```

SEE ALSO

`curses(3X)`.

NAME

sde-chooser - execute environment-sensitive tool

SYNOPSIS

sde-chooser [-e *sde-target*] *path* [*tool-args*]

DESCRIPTION

The action of a number of software development tools depends on the current software development environment [see *sde(5)*]. Such tools have different versions in each environment. *Sde-chooser* finds and executes the correct version of such a tool.

For example, when a command line such as “as foo.s” is executed, a small program named as in */usr/bin* executes *sde-chooser* with the appropriate arguments. *Sde-chooser* in turn executes the correct version of *as*.

Sde-chooser is not normally invoked from a shell command line, but it can be with the following arguments:

-e sde-target Specifies a software development environment explicitly. If this option is not given, *sde-chooser* uses the current software development environment [see *sde-target(1)*].

path The path to the desired tool within an environment. Path is given as an absolute path but it is interpreted as being relative to */usr/sde/<sde-target>*. For example, */usr/bin/as* invokes */usr/sde/<sde-target>/usr/bin/as*, where *<sde-target>* is a software development environment.

tool-args All remaining arguments to *sde-chooser* are passed to the selected tool as the *argv* array. The first of these arguments, *argv[0]*, should be the command name.

For example, the command line

```
sde-chooser -e m88kdguxcoff /usr/bin/cc cc -v
```

will invoke the COFF version of *cc* with the *-v* option. The effect, in this example, is the same as issuing the command *cc -v* in the *m88kdguxcoff* software development environment.

SEE ALSO

sde-target(1), *sde(5)*, *elink(5)*.

NAME

sdetab - software development environment data base

DESCRIPTION

The sdetab file contains information used by certain software development tools to customize SDE targets. The actual file used is /usr/etc/sdetab, which is an e`link` to the appropriate file (see s`de`(5) and e`link`(5)).

Each entry in the sdetab file consists of a key followed by one or more attributes separated by a colon, :. Blank lines and comments (from the pound sign, #, to the end of the line) are ignored. The backslash, \, may be used to quote characters.

Currently, ld(1) uses the key f`magic` to determine the magic number of the executable it produces.

FILES

/usr/etc/sdetab

SEE ALSO

s`de-target`(1), s`de`(5), e`link`(5).

NAME

services - service name database

DESCRIPTION

The `services` file contains information about the known services available in the DARPA Internet. For each service, a single line with the following information should be present:

name port/protocol [aliases] [# comment]

Items are separated by any number of blanks and/or tab characters. The *port* number and *protocol* name are considered a single item ; a slash (/) separates the *port* number and *protocol* name (e.g., 512/tcp).

Use only decimal numbers to specify port numbers in `/etc/services`. Ports 1-1023 are reserved by DG/UX for system servers to listen for incoming connections from other machines. Other ports in the range 1024 to 2**16-1 are available for user-implemented services.

If you specify an *alias*, you may refer to the service by that name rather than the official service *name*. A # indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file.

Names in the `services` file may contain any printable character other than a field delimiter (blanks, tabs, CR, ESC), New Line, or comment character.

If your system uses Network Information Service (NIS), see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System* for details.

EXAMPLES

```
ftp 21/tcp
telnet 23/tcp
```

FILES

`/etc/services`

SEE ALSO

`getservent(3N)`, `protocols(4)`.

NAME

space – disk space requirement file

DESCRIPTION

space is an ASCII file that gives information about disk space requirements for the target environment. It defines space needed beyond that which is used by objects defined in the `prototype` file—for example, files which will be installed with the `installf` command. It should define the maximum amount of additional space which a package will require.

The generic format of a line in this file is:

pathname blocks inodes

Definitions for the fields are as follows:

pathname Specifies a directory name which may or may not be the mount point for a filesystem. Names that do not begin with a slash (/) indicate relocatable directories.

blocks Defines the number of disk blocks required for installation of the files and directory entries contained in the *pathname* (using a 512-byte block size).

inodes Defines the number of inodes required for installation of the files and directory entries contained in the *pathname*.

EXAMPLE

```
# extra space required by config data which is
# dynamically loaded onto the system
data 500 1
```

SEE ALSO

`installf(1M)`, `prototype(4)`

NAME

sm, sm.bak, state - statd directories and file structures

SYNOPSIS

/etc/sm /etc/sm.bak /etc/state

DESCRIPTION

/etc/sm and /etc/sm.bak are directories generated by statd. Each entry in /etc/sm represents the name of the machine to be monitored by the statd daemon. Each entry in /etc/sm.bak represents the name of the machine to be notified by the statd daemon upon the remote machine's recovery.

/etc/state is a file generated by statd to record its version number. This version number is incremented each time a crash or recovery takes place.

FILES

/etc/sm
/etc/sm.bak
/etc/state

SEE ALSO

lockd(1M), statd(1M).

NAME

strptime - language specific strings

DESCRIPTION

There can exist one printable file per locale to specify its date and time formatting information. These files must be kept in the directory `/usr/lib/locale/<locale>/LC_TIME`. The contents of these files are:

1. abbreviated month names (in order)
2. month names (in order)
3. abbreviated weekday names (in order)
4. weekday names (in order)
5. default strings that specify formats for locale time (%X) and locale date (%x).
6. default format for ctime, if the argument for ctime is zero or null.
7. AM (ante meridian) string
8. PM (post meridian) string

Each string is on a line by itself. All white space is significant. The order of the strings in the above list is the same order in which they must appear in the file.

EXAMPLE

```
/usr/lib/locale/C/LC_TIME
```

```
Jan
Feb
...
January
February
...
Sun
Mon
...
Sunday
Monday
...
%H:%M:%S
%m/%d/%y
%a %b %d %T %Z %Y
AM
PM
```

FILES

```
/usr/lib/locale/<locale>/LC_TIME
```

SEE ALSO

ctime(3C), setlocale(3C), strptime(3C).

NAME

svccorder – file that specifies name/address resolution order

DESCRIPTION

Use the file `/etc/svccorder` to configure the order and the means to perform name/address resolution. If the file `/etc/svccorder` exists, it contains one record made up of one or two fields.

The first field determines the order in which different address resolution methods are tried. This first field consists of any combination of a subset of the following values separated by a colon (:):

YP or yp for Network Information Services (NIS)

RES or res for the resolver

EHOSTS or ehosts for the `/etc/hosts` file

The second field is an optional comment. You separate the two fields with white space, and begin a comment with a # character.

EXAMPLE

The following line specifies that address resolution should be attempted by NIS first.

```
yp:ehosts:res
```

If NIS is running and cannot resolve the name/address pair, name/address resolution goes no farther. However, you can configure NIS so that it uses the domain name system (DNS) when it cannot resolve a name/address pair after consulting its own database. Note that DNS must be set up on your system before you can use it. See *Managing TCP/IP on the DG/UX System* for more information on DNS. You can configure NIS to use DNS in two ways. You can enter the following command line:

```
make INTERDOMAIN=-b hosts
```

You can also change the "INTERDOMAIN=" line in the `/etc/yp/Makefile` script to read "INTERDOMAIN=-b". In either case, make sure there is no space before or after the equals (=) sign. The latter method enables the domain name system automatically whenever a change is made in the hosts map. For more information, see Chapter 3 in *Managing ONC/NFS and Its Facilities on the DG/UX System*.

If NIS is not running, `/etc/hosts` is consulted for name address resolution. If `/etc/hosts` does not produce an answer, the domain name system is consulted.

If the `/etc/svccorder` file does not exist, the order in which different resolution methods are tried is as above.

FILES

`/etc/svccorder`

SEE ALSO

gethostbyname(3N), gethostbyaddr(3N), resolver(3C).

NAME

syms - common object file symbol table format

SYNOPSIS

```
#include <syms.h>
```

DESCRIPTION

Common object files contain information to support symbolic software testing [see `sdb(1)`]. Line number entries [see `linenum(4)`] and extensive symbolic information permit testing at the C *source* level. Every object file's symbol table is organized as shown below.

File name 1.

Function 1.

Local symbols for function 1.

Function 2.

Local symbols for function 2.

...

Static externs for file 1.

File name 2.

Function 1.

Local symbols for function 1.

Function 2.

Local symbols for function 2.

...

Static externs for file 2.

...

Defined global symbols.

Undefined global symbols.

The entry for a symbol is a fixed-length structure. The members of the structure hold the name (null padded), its value, and other information. The C structure is given below.

```
#define SYMNMLEN 8
#define FILNMLEN 14
#define DIMNUM 4

struct syment
{
    union /* all ways to get symbol name */
    {
        char _n_name[SYMNMLEN]; /* symbol name */
        struct
        {
            long _n_zeroes; /* == 0L when in string table */
            long _n_offset; /* location of name in table */
        } _n_n;
        char *_n_nptr[2]; /* allows overlaying */
    } _n;
    long n_value; /* value of symbol */
    short n_scnun; /* section number */
    unsigned short n_type; /* type and derived type */
    char n_sclass; /* storage class */
}
```

```

        char          n_numaux;      /* number of aux entries */
        char          n_pad1;       /* pad to 4 byte multiple */
        char          n_pad2;       /* pad to 4 byte multiple */
};
};
};

#define n_name      _n._n_name
#define n_zeroes   _n._n.n._n_zeroes
#define n_offset   _n._n.n._n_offset
#define n_nptr     _n._n_nptr[1]

```

Meaningful values and their explanations can be found in `syms.h`; anyone who needs to interpret the entries should seek more information there. Some symbols require more information than a single entry; they are followed by *auxiliary entries* that are the same size as a symbol entry. The format follows:

```

union auxent
{
    struct
    {
        long          x_tagndx;
        union
        {
            struct
            {
                unsigned shortx_lnno;
                unsigned shortx_size;
            } x_lnsz;
            long      x_fsize;
        } x_misc;
        union
        {
            struct
            {
                long x_lnnoptr;
                long x_endndx;
            } x_fcn;
            struct
            {
                unsigned shortx_dimen[DIMNUM];
            } x_ary;
        } x_fcary;
        unsigned short x_tvndx;
        char pad1;
        char pad2;
    } x_sym;
    struct
    {
        char x_fname[FILNMLEN];
    } x_file;
    struct
    {
        long x_scnlen;
    }

```

```
        unsigned short  x_nreloc;
        unsigned short  x_nlinno;
    }
        x_scn;

    struct
    {
        long            x_tvfill;
        unsigned short  x_tvlen;
        unsigned short  x_tvran[2];
    } x_tv;
};
```

Indexes of symbol table entries begin at *zero*.

SEE ALSO

sdb(1), a.out(4), linenum(4).

CAUTION

Symbols declared as type `long` are recorded in the symbol table as type `int`.

NAME

`system` – format of a kernel description file

DESCRIPTION

The *system file* contains information about the hardware and system-dependent parameters found on your system. This information is used in conjunction with one or more *master files* as input into the `config(1M)` program. The `config(1M)` program is used to generate a `conf.c` file, which is then compiled and linked with kernel libraries to form a kernel image. A more complete description of the system file is found in *Managing the DG/UX System*.

Each line in a the system file is a separate entry. An entry contains one or more fields, separated by one or more space and/or tab characters. Any line with a number sign (#) in column 1 is treated as a comment and is ignored. Blank lines are also ignored. Each non-comment entry represents a device, STREAMS module, protocol, or tunable system parameter. Entries of any type may appear in any order.

Device Entries

An entry of the form:

`devname(parameters)`

or

`devname@devcode(parameters)`

specifies a device or pseudo-device to be configured into the kernel.

The device name *devname* must be listed in a `$device` section of one of the master files.

The *devcode* notation, if present, specifies that a non-default hardware device code will be used for that device. The device code must appear as a two-digit hexadecimal number.

The *parameters* string represents a specific unit or instantiation of the device; its interpretation is left to the specific device driver. If *parameters* is the null string, the driver's default parameter values will be used. Note that the *parameters* string may itself be a device specification, such as:

`sd(inc(),*)`

Protocol Entries

Each single-word entry that matches an entry in a master file's `$protocol` section specifies a socket protocol to be configured into the kernel.

STREAMS Module Entries

Each single-word entry that matches an entry in a master file's `$stream` section specifies a STREAMS module to be configured into the kernel.

Tunable Parameter Entries

Each one or two-word entry whose first word matches an entry in a master file's `$keyword` section specifies a tunable system parameter for which a non-default value should be configured into the kernel. The first word of the entry names the parameter that is to be tuned; the second word specifies its value. The value field may be omitted if an implied value is specified in the master file. Note that the implied value may be different from the default value.

SEE ALSO

`config(1M)`, `sysdef(1M)`, `master(4)`.

Installing the DG/UX System, *Customizing the DG/UX System*, *Managing the DG/UX System*.

NAME

`terminfo` – terminal and printer capability database

DESCRIPTION

`Terminfo` is a compiled database of terminal and printer device capabilities. The capabilities of each type of device are described in a data file that has a name of the following form: `/usr/lib/terminfo/?/*`, where `*` stands for the device name and `?` stands for the first character of the name. For example,

`/usr/lib/terminfo/d/d215`

is the `terminfo` entry for Data General's DASHER D215 terminal and terminals that behave like it.

`Terminfo` data files are obtained by compiling source descriptions with the `tic(1M)` command. `Terminfo` source descriptions describe, in special code, how basic operations are performed on a terminal or printer. They also describe padding requirements, initialization sequences, and so on. The section entitled "Preparing a `Terminfo` Description" explains how to build a `terminfo` source description. Applications such as `vi(1)` and `curses(3X)` refer to the compiled `terminfo` database so that they can work with a variety of terminals without changes to the program code.

Entries in a `terminfo` source file consist of a number of comma-separated fields. The white space after each comma is ignored. The first line names the device, and the remaining lines describe its capabilities.

Device Names

The first line of each device description in the `terminfo` source file gives the names by which `terminfo` knows the device. Each name is separated by bar (|) characters. The first name specifies the most common abbreviation for the device (this is the one to use for the environment variable `TERM`; see `profile(4)`). The last name should be a long name that fully identifies the device. All other names are synonyms for the device name. All names but the last should contain no blanks; the last, verbose name may contain blanks for readability.

Device names (except for the verbose entry) should be chosen using the following conventions. First, the particular vendor and model of the device should be specified in the root name, for example, `att4425` for the AT&T 4425 terminal. Second, device modes or user preferences should be indicated by appending a hyphen and an indicator of the mode, for example, `d410-w` for the Data General DASHER D410 series in wide mode (more than 80 columns). See `term(5)` for examples and more information on choosing names and synonyms.

Device Capabilities

Lines after the first line of a device description describe the device's capabilities. `Terminfo` device capabilities are of three general types: boolean capabilities indicate that the device has some particular feature, numeric capabilities specify a numeric value associated with a particular feature, for example, the size of a terminal screen, and string capabilities give a sequence which can be used to perform particular device operations.

In the table below, the `variable` is the name by which a C programmer (at the `terminfo` level) accesses the capability. The `capname` is the short name for this variable used in the text of the database. It is used by a person updating the database and by the `tput(1)` command when asking what the value of the capability is for a particular device. See Also refers to the numbered subsection in "Terminfo Terminal Capabilities" or the lettered subsection in "Terminfo Printer Capabilities" where the capability is described in detail.

Capability names have no fixed length limit, but an informal limit of 5 characters has been adopted to keep them short. Most of the time, names are chosen to be the same as or similar to the ANSI X3.64-1979 standard. Semantics are also intended to match those of the description.

All string capabilities listed below may have padding described, with the exception of those used for input. Input capabilities, listed under the strings section in the table below, have names beginning with `key_`. The following indicators may appear at the end of the description for a variable.

- (G) indicates that the string needs to be instantiated by `tparam()` with arguments (parms) as given (`#i` as described below). `tparam()` will substitute the arguments into the string to create a customized version. (See `curses(3X)` for more information on `tparam()` and the strings it creates.)
- (*) indicates that padding may be based on the number of lines affected.
- (#_i) indicates the *i*th parameter.

Variable	Cap-name	See Also	Description
Boolean Capabilities:			
<code>auto_left_margin</code>	<code>bw</code>	1	<code>cub1</code> wraps back from column 0
<code>auto_right_margin</code>	<code>am</code>	1,13	Device has automatic margins
<code>back_color_erase</code>	<code>bce</code>	12	Screen erased with background color
<code>can_change</code>	<code>ccc</code>	12	Device can redefine existing color
<code>ceol_standout_glitch</code>	<code>xhp</code>	14	Standout not erased by overwriting (HP)
<code>col_addr_glitch</code>	<code>xhpa</code>	B	Only positive motion for <code>hpa/mhpa</code>
<code>cpi_changes_res</code>	<code>cpix</code>	A,G	Character pitch affects resolution
<code>cr_cancels_micro_mode</code>	<code>crxm</code>	B	Using <code>cr</code> disables micro mode
<code>eat_newline_glitch</code>	<code>xenl</code>	14	Newline ignored after 80 columns (Concept)
<code>erase_overstrike</code>	<code>eo</code>	6	Overstrikes are erased by blanks
<code>generic_type</code>	<code>gn</code>	13	Generic line type (e.g., dialup, switch)
<code>hard_copy</code>	<code>hc</code>	1	Hardcopy device
<code>hard_cursor</code>	<code>chts</code>	6	Cursor is hard to see
<code>has_meta_key</code>	<code>km</code>	13	Device can send meta-characters (e.g., key sets eighth bit)
<code>has_print_wheel</code>	<code>daisy</code>	E	Printer needs operator to change character sets
<code>has_status_line</code>	<code>hs</code>	10	Terminal has extra "status line"
<code>hue_lightness_saturation</code>	<code>hls</code>	12	Device uses only HLS color notation (Tektronix)
<code>insert_null_glitch</code>	<code>in</code>	5	Insert mode distinguishes nulls
<code>lpi_changes_res</code>	<code>lpix</code>	A,G	Line pitch affects resolution
<code>memory_above</code>	<code>da</code>	4	Display may be retained above screen
<code>memory_below</code>	<code>db</code>	4	Display may be retained below screen
<code>move_insert_mode</code>	<code>mir</code>	5	Safe to move in insert mode
<code>move_standout_mode</code>	<code>msgr</code>	6	Safe to move in standout modes

needs_xon_xoff	nxon	14	Padding won't work, XON/XOFF needed
no_esc_ctlc	xsb	14	Beehive (F1=<ESC>, F2=<Ctrl-C>)
non_rev_rmcup	nrrmc	6	smcup does not reverse rmcup
no_pad_char	npc	13	Pad character doesn't exist
over_strike	os	1,6	Device overstrikes (hardcopy device)
prtr_silent	mc5i	13	Printer won't echo on screen
row_addr_glitch	xvpa	B	Only positive motion for vpa/mvpa
semi_auto_right_margin	sam	B	Printing in last column causes cr
status_line_esc_ok	eslok	10	Escape sequences work on status line
dest_tabs_magic_smsc	xt	13	Destructive tabs, magic smsc character (t1061)
tilde_glitch	hz	14	Hazeltine; can't print tildes (~)
transparent_underline	ul	6	Underline character overstrikes
xon_xoff	xon	1,13	Device uses XON/XOFF handshaking

Numeric Capabilities:

buffer_capacity	bufsz	I	Bytes buffered before printing
columns	cols	1	Number of columns in a line
dot_vert_spacing	spinv	F	Vertical pin spacing (pins/inch)
dot_horz_spacing	spinh	F	Horizontal dot spacing (dots/inch)
init_tabs	it	8	Initial spacing of tab settings
label_height	lh	7	Number of rows in each soft label
label_width	lw	7	Number of columns in each soft label
lines	lines	1	Number of lines on screen or page
lines_of_memory	lm	13	Lines of memory; variable if 0
magic_cookie_glitch	xmc	6	Number of blanks left by smsc/rmsc
max_colors	colors	12	Maximum number of colors on-screen
max_micro_address	maddr	B	Maximum limit on micro..._address
max_micro_jump	mjump	B	Maximum limit on parm..._micro
max_pairs	pairs	12	Maximum number of color-pairs
micro_col_size	mcs	A	Horizontal step size in micro mode
micro_line_size	mls	A	Vertical step size in micro mode
no_color_video	ncv	12	Video attributes unusable with color
number_of_pins	npins	F	Number of pins in print head
num_labels	nlab	7	Number of soft labels available (starting from 1)
output_res_char	orc	A	Horizontal resolution (steps/column)
output_res_line	orl	A	Vertical resolution (steps/line)
output_res_horz_inch	orhi	A	Horizontal resolution (steps/inch)
output_res_vert_inch	orvi	A	Vertical resolution (steps/inch)
padding_baud_rate	pb	9	Lowest baud rate requiring padding
print_rate	cps	I	Average speed (characters/second)
virtual_terminal	vt	13	UNIX system virtual terminal number
wide_char_size	widcs	A	Character size in double wide mode
width_status_line	wsl	10	Number of columns in status line

String Capabilities:

acs_chars	acsc	11	Graphic character set pairs aAbBcC (vt100+)
-----------	-------------	----	---

back_tab	cbt	8	Back tab
bell	bel	1	Audible signal (bell)
carriage_return	cr	1,9	Carriage return (*)
change_char_pitch	cpi	A,G	Set pitch to #1 characters/inch (G)
change_line_pitch	lpi	A,G	Set pitch to #1 lines/inch (G)
change_res_horz	chr	A	Set horizontal resolution to #1 (G)
change_res_vert	cvr	A	Set vertical resolution to #1 (G)
change_scroll_region	csr	4	Scrolling area lines #1 through #2 (vt100) (G)
char_padding	rmp	5	Like ip but when in replace mode
char_set_names	csnm	E	Name of character set #1 (G)
clear_all_tabs	tbc	8	Clear all tab stops
clear_margins	mgc	8	Clear left and right soft margins
clear_screen	clear	1	Clear screen and home cursor (*)
clr_bol	el1	3	Clear to beginning of line
clr_eol	el	3,14	Clear to end of line
clr_eos	ed	3	Clear to end of display (*)
column_address	hpa	2	Horizontal position to column #1 (G)
command_character	cmdch	13	Prototype settable command character
cursor_address	cup	2	Move cursor to row #1, column #2 (G)
cursor_down	cud1	1	Move cursor down one line
cursor_home	home	2	Home cursor (especially if no cup)
cursor_invisible	civis	6	Make cursor invisible
cursor_left	cub1	1	Move cursor left one space
cursor_mem_address	mrcup	2	Like cup but memory relative (G)
cursor_normal	cnorm	6	Make cursor normal (undo civis/cvvis)
cursor_right	cuf1	1	Move cursor right one space (non-destructive)
cursor_to_ll	ll	2	Move cursor to column 0 of last line
cursor_up	cuu1	2	Move cursor up one line
cursor_visible	cvvis	6	Make cursor very visible
define_char	defc	E	Define character #1 with width #2 and descender #3 (G)
delete_character	dch1	5	Delete character (*)
delete_line	dll	4	Delete line (*)
dis_status_line	dsl	10	Disable status line
down_half_line	hd	13	Move cursor down one half-line (forward 1/2 linefeed)
ena_acs	enacs	6	Initialize alternate character set
enter_alt_charset_mode	smacs	6	Enable alternate character set mode
enter_am_mode	smam	13	Enable automatic margins
enter_blink_mode	blink	6	Enable blinking mode
enter_bold_mode	bold	6	Enable bold (extra bright) mode
enter_ca_mode	smcup	6	String to send before using cup
enter_delete_mode	smdc	5	Begin delete mode
enter_dim_mode	dim	6	Enable half-bright mode
enter_doublewide_mode	swidm	D	Enable double wide printing
enter_draft_quality	sdrfq	G	Set draft quality printing
enter_insert_mode	smir	5	Begin insert mode

enter_italics_mode	sitm	D	Enable italics
enter_leftward_mode	slm	B	Enable leftward carriage motion
enter_micro_mode	smicm	B	Enable micro motion capabilities
enter_near_letter_quality	snlq	G	Set near-letter-quality printing
enter_normal_quality	snrmq	G	Set normal quality printing
enter_protected_mode	prot	6	Enable protected mode
enter_reverse_mode	rev	6	Enable reverse video mode
enter_secure_mode	invis	6	Enable blank mode (invisible text)
enter_shadow_mode	sshm	D	Enable shadow printing
enter_standout_mode	smso	6	Enable standout mode
enter_subscript_mode	ssubm	D	Enable subscript printing
enter_superscript_mode	ssupm	D	Enable superscript printing
enter_underline_mode	smul	6	Enable underscore mode
enter_upward_mode	sum	B	Enable upward carriage motion
enter_xon_mode	smxon	13	Enable XON/XOFF handshaking
erase_chars	ech	5	Erase #1 characters (G)
exit_alt_charset_mode	rmacs	6	Disable alternate character set mode
exit_am_mode	rmam	13	Disable automatic margins
exit_attribute_mode	sgr0	6	Disable all video attributes (G)
exit_ca_mode	rmcup	6	String to send when done with cup
exit_delete_mode	rmdc	5	End delete mode
exit_doublewide_mode	rwidm	D	Disable double wide printing
exit_insert_mode	rmir	5	End insert mode
exit_italics_mode	ritm	D	Disable italics
exit_leftward_mode	rlm	B	Enable rightward carriage motion (the normal state)
exit_micro_mode	rmicm	B	Disable micro motion capabilities
exit_shadow_mode	rshm	D	Disable shadow printing
exit_standout_mode	rmso	6	Disable standout mode
exit_subscript_mode	rsubm	D	Disable subscript printing
exit_superscript_mode	rsupm	D	Disable superscript printing
exit_underline_mode	rmul	6	Disable underscore mode
exit_upward_mode	rum	B	Enable downward carriage motion (the normal state)
exit_xon_mode	rmxon	13	Disable XON/XOFF handshaking
flash_screen	flash	6	Visible bell (must not move cursor)
form_feed	ff	13	Hardcopy device page eject (*)
from_status_line	fsl	10	Return from status line
init_1string	is1	8	Device initialization string 1
init_2string	is2	8	Device initialization string 2
init_3string	is3	8	Device initialization string 3
init_file	if	8	Name of initialization data file
init_prog	iprog	8	Path name of initialization program
initialize_color	initc	12	Define color #1 as RGB #2-#4 (G)
initialize_pair	initp	12	Define color-pair #1 as RGB #2-#7 (G)
insert_character	ich1	5	Insert new blank character
insert_line	ill	4	Add new blank line (*)
insert_padding	ip	5	Padding after character inserted (*)
key_a1	ka1	7	KEY_A1, Upper left of keypad

key_a3	ka3	7	KEY_A3, Upper right of keypad
key_b2	kb2	7	KEY_B2, Center of keypad
key_backspace	kbs	7	KEY_BACKSPACE, Sent by backspace key
key_beg	kbeg	7	KEY_BEG, Sent by beginning key (beg key)
key_btab	kcbt	7	KEY_BTAB, Sent by back-tab key
key_c1	kc1	7	KEY_C1, Lower left of keypad
key_c3	kc3	7	KEY_C3, Lower right of keypad
key_cancel	kcan	7	KEY_CANCEL, Sent by cancel key
key_catab	ktbc	7	KEY_CATAB, Sent by clear-all-tabs key
key_clear	kclr	7	KEY_CLEAR, Sent by clear-screen key (erase key)
key_close	kclo	7	KEY_CLOSE, Sent by close key
key_command	kcmd	7	KEY_COMMAND, Sent by command key (cmd key)
key_copy	kcpy	7	KEY_COPY, Sent by copy key
key_create	kcrt	7	KEY_CREATE, Sent by create key
key_ctab	kctab	7	KEY_CTAB, Sent by clear-tab key
key_dc	kdch1	7	KEY_DC, Sent by delete-character key
key_dl	kdll	7	KEY_DL, Sent by delete-line key
key_down	kcud1	7	KEY_DOWN, Sent by cursor-down key (down-arrow key)
key_eic	krmir	7	KEY_EIC, Sent by end-insert-mode key
key_end	kend	7	KEY_END, Sent by end key
key_enter	kent	7	KEY_ENTER, Sent by enter/send key
key_eol	kel	7	KEY_EOL, Sent by clear-to-end-of-line key
key_eos	ked	7	KEY_EOS, Sent by clear-to-end-of-screen key
key_exit	kext	7	KEY_EXIT, Sent by exit key
key_f0	kf0	7	KEY_F(0), Sent by function key F0
key_f1	kf1	7	KEY_F(1), Sent by function key F1
key_f2	kf2	7	KEY_F(2), Sent by function key F2
key_f3	kf3	7	KEY_F(3), Sent by function key F3
key_f4	kf4	7	KEY_F(4), Sent by function key F4
key_f5	kf5	7	KEY_F(5), Sent by function key F5
key_f6	kf6	7	KEY_F(6), Sent by function key F6
key_f7	kf7	7	KEY_F(7), Sent by function key F7
key_f8	kf8	7	KEY_F(8), Sent by function key F8
key_f9	kf9	7	KEY_F(9), Sent by function key F9
key_f10	kf10	7	KEY_F(10), Sent by function key F10
key_f11	kf11	7	KEY_F(11), Sent by function key F11
key_f13	kf13	7	KEY_F(12), Sent by function key F12
key_f14	kf14	7	KEY_F(13), Sent by function key F13
key_f14	kf14	7	KEY_F(14), Sent by function key F14
key_f15	kf15	7	KEY_F(15), Sent by function key F15
key_f16	kf16	7	KEY_F(16), Sent by function key F16
key_f17	kf17	7	KEY_F(17), Sent by function key F17
key_f18	kf18	7	KEY_F(18), Sent by function key F18

key_f19	kf19	7	KEY_F(19), Sent by function key F19
key_f20	kf20	7	KEY_F(20), Sent by function key F20
key_f21	kf21	7	KEY_F(21), Sent by function key F21
key_f22	kf22	7	KEY_F(22), Sent by function key F22
key_f23	kf23	7	KEY_F(23), Sent by function key F23
key_f24	kf24	7	KEY_F(24), Sent by function key F24
key_f25	kf25	7	KEY_F(25), Sent by function key F25
key_f26	kf26	7	KEY_F(26), Sent by function key F26
key_f27	kf27	7	KEY_F(27), Sent by function key F27
key_f28	kf28	7	KEY_F(28), Sent by function key F28
key_f29	kf29	7	KEY_F(29), Sent by function key F29
key_f30	kf30	7	KEY_F(30), Sent by function key F30
key_f31	kf31	7	KEY_F(31), Sent by function key F31
key_f32	kf32	7	KEY_F(32), Sent by function key F32
key_f33	kf33	7	KEY_F(13), Sent by function key F33
key_f34	kf34	7	KEY_F(34), Sent by function key F34
key_f35	kf35	7	KEY_F(35), Sent by function key F35
key_f36	kf36	7	KEY_F(36), Sent by function key F36
key_f37	kf37	7	KEY_F(37), Sent by function key F37
key_f38	kf38	7	KEY_F(38), Sent by function key F38
key_f39	kf39	7	KEY_F(39), Sent by function key F39
key_f40	kf40	7	KEY_F(40), Sent by function key F40
key_f41	kf41	7	KEY_F(41), Sent by function key F41
key_f42	kf42	7	KEY_F(42), Sent by function key F42
key_f43	kf43	7	KEY_F(43), Sent by function key F43
key_f44	kf44	7	KEY_F(44), Sent by function key F44
key_f45	kf45	7	KEY_F(45), Sent by function key F45
key_f46	kf46	7	KEY_F(46), Sent by function key F46
key_f47	kf47	7	KEY_F(47), Sent by function key F47
key_f48	kf48	7	KEY_F(48), Sent by function key F48
key_f49	kf49	7	KEY_F(49), Sent by function key F49
key_f50	kf50	7	KEY_F(50), Sent by function key F50
key_f51	kf51	7	KEY_F(51), Sent by function key F51
key_f52	kf52	7	KEY_F(52), Sent by function key F52
key_f53	kf53	7	KEY_F(53), Sent by function key F53
key_f54	kf54	7	KEY_F(54), Sent by function key F54
key_f55	kf55	7	KEY_F(55), Sent by function key F55
key_f56	kf56	7	KEY_F(56), Sent by function key F56
key_f57	kf57	7	KEY_F(57), Sent by function key F57
key_f58	kf58	7	KEY_F(58), Sent by function key F58
key_f59	kf59	7	KEY_F(59), Sent by function key F59
key_f60	kf60	7	KEY_F(60), Sent by function key F60
key_f61	kf61	7	KEY_F(61), Sent by function key F61
key_f62	kf62	7	KEY_F(62), Sent by function key F62
key_f63	kf63	7	KEY_F(63), Sent by function key F63
key_find	kfnd	7	KEY_FIND, Sent by find key
key_help	khlp	7	KEY_HELP, Sent by help key
key_home	khome	7	KEY_HOME, Sent by home key
key_ic	kich1	7	KEY_IC, Sent by insert-character key

			(enter-insert-mode key)
key_il	kill	7	KEY_IL, Sent by insert-line key
key_left	kcub1	7	KEY_LEFT, Sent by cursor-left key (left-arrow key)
key_ll	kill	7	KEY_LL, Sent by home-down key
key_mark	kmrk	7	KEY_MARK, Sent by mark key
key_message	kmsg	7	KEY_MESSAGE, Sent by message key
key_move	kmov	7	KEY_MOVE, Sent by move key
key_next	knxt	7	KEY_NEXT, Sent by next-object key
key_npage	knp	7	KEY_NPAGE, Sent by next-page key
key_open	kopn	7	KEY_OPEN, Sent by open key
key_options	kopt	7	KEY_OPTIONS, Sent by options key
key_ppage	kpp	7	KEY_PPAGE, Sent by previous-page key
key_previous	kprv	7	KEY_PREVIOUS, Sent by previous-object key
key_print	kpri	7	KEY_PRINT, Sent by print key (copy key)
key_redo	krdo	7	KEY_REDO, Sent by redo key
key_reference	kref	7	KEY_REFERENCE, Sent by reference key (ref key)
key_refresh	krfr	7	KEY_REFRESH, Sent by refresh key
key_replace	krpl	7	KEY_REPLACE, Sent by replace key
key_restart	krst	7	KEY_RESTART, Sent by restart key
key_resume	kres	7	KEY_RESUME, Sent by resume key
key_right	kcuf1	7	KEY_RIGHT, Sent by cursor-right key (right-arrow key)
key_save	ksav	7	KEY_SAVE, Sent by save key
key_sbeg	kBEG	7	KEY_SBEG, Sent by shifted beginning key
key_scancel	kCAN	7	KEY_SCANCEL, Sent by shifted cancel key
key_scommand	kCMD	7	KEY_SCOMMAND, Sent by shifted command key (cmd key)
key_scopy	kCPY	7	KEY_SCOPY, Sent by shifted copy key
key_screate	kCRT	7	KEY_SCREATE, Sent by shifted create key
key_sdc	kDC	7	KEY_SDC, Sent by shifted delete-character key
key_sdl	kDL	7	KEY_SDL, Sent by shifted delete-line key
key_select	kslt	7	KEY_SELECT, Sent by select key
key_send	kEND	7	KEY_SEND, Sent by shifted end key
key_seol	kEOL	7	KEY_SEOL, Sent by shifted clear-to-end-of-line key
key_sexit	kEXT	7	KEY_SEXIT, Sent by shifted exit key
key_sf	kind	7	KEY_SF, Sent by scroll-forward key (scroll-down key)
key_sfind	kFND	7	KEY_SFIND, Sent by shifted find key
key_shelp	kHLP	7	KEY_SHELP, Sent by shifted help key

key_shome	kHOM	7	KEY_SHOME, Sent by shifted home key
key_sic	kIC	7	KEY_SIC, Sent by shifted input key
key_sleft	kLFT	7	KEY_SLEFT, Sent by shifted cursor-left key (left-arrow key)
key_smessage	kMSG	7	KEY_SMESSAGE, Sent by shifted message key
key_smove	kMOV	7	KEY_SMOVE, Sent by shifted move key
key_snext	kNXT	7	KEY_SNEXT, Sent by shifted next key
key_soptions	kOPT	7	KEY_SOPTIONS, Sent by shifted options key
key_sprevious	kPRV	7	KEY_SPREVIOUS, Sent by shifted previous-object key
key_sprint	kPRT	7	KEY_SPRINT, Sent by shifted print key
key_sr	kri	7	KEY_SR, Sent by scroll-backward key (scroll-up key)
key_sredo	kRDO	7	KEY_SREDO, Sent by shifted redo key
key_sreplace	kRPL	7	KEY_SREPLACE, Sent by shifted replace key
key_sright	kRIT	7	KEY_SRIGHT, Sent by shifted cursor-right key (right-arrow key)
key_sresume	kRES	7	KEY_SRSUME, Sent by shifted resume key
key_ssave	kSAV	7	KEY_SSAVE, Sent by shifted save key
key_ssuspend	kSPD	7	KEY_SSUSPEND, Sent by shifted suspend key
key_stab	khts	7	KEY_STAB, Sent by set-tab key
key_sundo	kUND	7	KEY_SUNDO, Sent by shifted undo key
key_suspend	kspd	7	KEY_SUSPEND, Sent by suspend key
key_undo	kund	7	KEY_UNDO, Sent by undo key
key_up	kcuu1	7	KEY_UP, Sent by cursor-up key (up-arrow key)
keypad_local	rmkx	7	Disable “keypad-transmit” mode
keypad_xmit	smkx	7	Enable “keypad-transmit” mode
lab_f0	lf0	7	Label on function key F0 if not F0
lab_f1	lf1	7	Label on function key F1 if not F1
lab_f2	lf2	7	Label on function key F2 if not F2
lab_f3	lf3	7	Label on function key F3 if not F3
lab_f4	lf4	7	Label on function key F4 if not F4
lab_f5	lf5	7	Label on function key F5 if not F5
lab_f6	lf6	7	Label on function key F6 if not F6
lab_f7	lf7	7	Label on function key F7 if not F7
lab_f8	lf8	7	Label on function key F8 if not F8
lab_f9	lf9	7	Label on function key F9 if not F9
lab_f10	lf10	7	Label on function key F10 if not F10
label_off	rmln	7	Disable soft labels
label_on	smln	7	Enable soft labels
meta_off	rmm	13	Disable “meta mode”
meta_on	smm	13	Enable “meta mode” (eight-bit I/O)

micro_column_address	mhpa	B	Like column_address for micro adjustment (G)
micro_down	mcud1	B	Like cursor_down for micro adjustment
micro_left	mcub1	B	Like cursor_left for micro adjustment
micro_right	mcuf1	B	Like cursor_right for micro adjustment
micro_row_address	mvpa	B	Like row_address for micro adjustment (G)
micro_up	mcuu1	B	Like cursor_up for micro adjustment
newline	nel	1	Newline (like CR followed by LF)
order_of_pins	porder	F	Matches data bits to print head pins
orig_colors	oc	12	Set all color(-pair)s to defaults
orig_pair	op	12	Set color-pair to the default (G)
pad_char	pad	13	Pad character (rather than null)
parm_dch	dch	5	Delete #1 characters (G*)
parm_delete_line	dl	4	Delete #1 lines (G*)
parm_down_cursor	cud	1	Move cursor down #1 lines (G*)
parm_down_micro	mcud	B	Like parm_down_cursor for micro adjustment (G)
parm_ich	ich	4	Insert #1 blank characters (G*)
parm_index	indn	1	Scroll forward #1 lines (G)
parm_insert_line	il	4	Add #1 new blank lines (G*)
parm_left_cursor	cub	1	Move cursor left #1 spaces (G)
parm_left_micro	mcub	B	Like parm_left_cursor for micro adjustment (G)
parm_right_cursor	cuf	1	Move cursor right #1 spaces (G*)
parm_right_micro	mcuf	B	Like parm_right_cursor for micro adjustment (G)
parm_rindex	rin	1	Scroll backward #1 lines (G)
parm_up_cursor	cuu	1	Move cursor up #1 lines (G*)
parm_up_micro	mcuu	B	Like parm_up_cursor for micro adjustment (G)
pkey_key	pfkey	7	Program PFkey #1 to type #2 (G)
pkey_local	pfloc	7	Program PFkey #1 to execute #2 (G)
pkey_xmit	px	7	Program PFkey #1 to transmit #2 (G)
plab_norm	pln	7	Program soft label #1 to show #2 (G)
print_screen	mc0	13	Print contents of screen
prtr_non	mc5p	13	Enable printer for #1 bytes
prtr_off	mc4	13	Disable printer
prtr_on	mc5	13	Enable printer
repeat_char	rep	13	Repeat character #1 #2 times (G*)
req_for_input	rfi	13	Send next input character (for ptys)
reset_1string	rs1	8	Device full reset string 1
reset_2string	rs2	8	Device full reset string 2
reset_3string	rs3	8	Device full reset string 3
reset_file	rf	8	Name of file containing reset string
restore_cursor	rc	4,10	Move cursor to position of last sc

row_address	vpa	2	Vertical position to row #1 (G)
save_cursor	sc	4,10	Save cursor position for next rc
scroll_forward	ind	1	Scroll text up one line
scroll_reverse	ri	1	Scroll text down one line
select_char_set	scs	E	Select character set #1 (G)
set_attributes	sgr	6	Define video attributes #1-#9 (G)
set_background	setb	12	Set active background color to #1 (G)
set_bottom_margin	smgb	C	Set bottom margin at current line
set_bottom_margin_parm	smgbp	C	Set bottom margin at line #1 or #2 lines from bottom (G)
set_color_pair	scp	12	Set current color-pair to #1 (G)
set_foreground	setf	12	Set active foreground color to #1 (G)
set_left_margin	smgl	8	Set soft left margin
set_left_margin_parm	smglp	C	Set left margin at column #1 (right margin at #2) (G)
set_right_margin	smgr	8	Set soft right margin
set_right_margin_parm	smgrp	C	Set right margin at column #1 (G)
set_tab	hts	8	Set tab in all rows, current column
set_top_margin	smgt	C	Set top margin at current line
set_top_margin_parm	smgtp	C	Set top margin at line #1 (bottom margin at line #2) (G)
set_window	wind	4	Set current window to lines #1-#2, columns #3-#4 (G)
start_bit_image	sbim	F	Start printing bit image graphics, #1 dots wide (G)
start_char_set_def	scsd	E	Start defining character set #1, containing #2 characters (G)
stop_bit_image	rbim	F	End printing bit image graphics
stop_char_set_def	rcsd	E	End defining character set #1 (G)
subscript_characters	subcs	D	“Subscript-able” characters
superscript_characters	supcs	D	“Superscript-able” characters
tab	ht	8	Tab to next hardware tab stop
these_cause_cr	docr	B	Any of these characters causes cr
to_status_line	tsl	10	Go to status line, column #1 (G)
underline_char	uc	6	Underscore character and move past
up_half_line	hu	13	Move up one half-line (reverse 1/2 linefeed)
xoff_character	xoffc	13	XOFF character
xon_character	xonc	13	XON character
zero_motion	zerom	B	No motion for subsequent character

PREPARING A TERMINFO DESCRIPTION

At a minimum for a terminal, a `terminfo` source file should specify capabilities to do the following:

- Clear the screen
- Specify screen size
- Specify how to scroll the screen
- Specify how to move the cursor to any point on the screen
- Display whatever graphic embellishments are available (e.g., reverse video)
- Specify whether the cursor wraps around when it reaches the end of a line
- Specify a scrolling region, if possible
- Insert and delete lines and characters, if available

- Save and restore the cursor position, if possible
- Describe special keys, if any
- Specify how to handle special cases of terminal behavior, if any

The most effective way to prepare a new device description is by imitating the description of a similar device in `terminfo` and building up the new description gradually, testing whether `vi(1)` works with the compiled description. That is, first create a `terminfo` source file that includes what you have determined to be the minimum set of capabilities needed for the new device. Next, compile the source with the `tic(1M)` command. Use `vi(1)` and determine whether the device displays what it is supposed to display. Make alterations or add more advanced capabilities to the source file as appropriate, recompile the source, and repeat the test. Repeat this cycle until the description is complete and correct.

You can obtain the source description for a given device by using the `-I` option of `infocmp(1M)`. You may copy and edit this description to accurately describe the device that you wish to enter into the `terminfo` database. Most reference manuals for terminals and printers list the codes that make the device perform specific operations. Use these codes to describe capabilities of the new device.

To test a new device description, set the environment variable `TERMINFO` to the pathname of a directory containing the compiled description. Programs will then search that directory for terminal information instead of `/usr/lib/terminfo`. To get the padding for insert-line correct on a terminal (if the manufacturer did not document it) a severe test is to comment out `xon`, edit a large file at 9600 baud with `vi(1)`, delete 16 or so lines from the middle of the screen, then hit the `u` key several times quickly. If the display is corrupted, more padding is usually needed. An analogous test can be used for insert-character.

Be aware that a very unusual device may expose deficiencies in the ability of `terminfo` to describe it or the ability of programs such as `vi(1)` to work with that device.

Similar Devices

If there are two very similar devices, one can be defined as being just like the other with certain exceptions. The string capability `use` can be given with the name of the similar device. The capabilities given before `use` override those in the device type included by `use`.

More than one `use` capability may be specified. Statements that contain `use` exhibit left-to-right precedence. That is, the earliest `use` statement has priority when more than one statement defines the same capability.

A capability can be canceled by placing `@` to the left of the capability definition. For example:

```
att4424-2|Teletype 4424 in display function group ii,
    rev@, sgr@, smul@, use=att4424,
```

defines an AT&T 4424 terminal that does not have the `rev`, `sgr`, and `smul` capabilities, and hence cannot do highlighting. This is useful for different modes of a device, or for different user preferences.

Parameterized Strings

Cursor addressing and other strings requiring parameters for the device are described by a parameterized string capability, with `printf(3S)`-like escapes (`%x`) in it. The parameter mechanism uses a stack and special `%` codes to manipulate it in the manner of a Reverse Polish Notation (postfix) calculator.

Typically a sequence pushes one of the parameters onto the stack and then prints it in some format. When a sequence pushes a value, the value is placed onto the top of the `terminfo` stack, leaving the source unchanged. The complement to a "push" is the "pop", which removes the topmost value from the `terminfo` stack, storing it elsewhere or using it in the current calculation.

Stack and Variable Manipulation

Parameterized strings can access arguments passed to `tparam()`. The arguments are referenced positionally, by number from 1 to 9. `Terminfo` also provides 52 variables that parameterized strings can use. The variables are referenced by letter from a to z and from A to Z. The lowercase variable names represent automatic variables that do not retain their values between parameterized strings. The uppercase variable names represent static variables that do retain their values.

`%p[1-9]` Push the indicated parameter.
`%'c'` Push the character constant 'c'.
`%{n}` Push the one or two digit decimal number constant *n*.
`%P[a-zA-Z]` Pop the stack into the indicated variable.
`%g[a-zA-Z]` Push the current contents of the indicated variable.

Printing Operations

The following escapes print a value in a specified format.

`%%` Print the '%' character.
`%c` Pop the stack and print the value without interpretation, that is, as a single character.
`%[[:]flags][width[.precision]][doxXs]`
 Pop the stack and print the value as a formatted string, converting to decimal (d), octal (o), lowercase hexadecimal (x), uppercase hexadecimal (X), or character (s) data as indicated. For information on the *flags*, *width*, and *precision* fields, and more information on the conversions, consult `printf(3S)`. (The *flags* supported are -, +, #, and the space character.)

NOTE: The - flag must be preceded by a colon (:) to differentiate the flag from the %- escape described below.

Arithmetic Operations

The following escapes pop one or two operands off the stack, perform some arithmetic operation, and then push the result onto the stack. Binary operations are in postfix form and expect the first operand to be on the top of the stack.

NOTE: Whether arithmetic is signed or unsigned is unspecified.

`%+` Push the sum of the two topmost values on the stack.
`%-` Push the difference of the two topmost values on the stack.
`%*` Push the product of the two topmost values on the stack.
`%/` Push the quotient of the two topmost values on the stack.
`%m` Push the modulus of the two topmost values on the stack.
`%&` Push the bitwise AND of the two topmost values on the stack.
`%|` Push the bitwise OR of the two topmost values on the stack.
`%^` Push the bitwise exclusive OR of the two topmost values on the stack.
`%~` Bitwise complement the topmost value on the stack.

Logical Operations

The following escapes are like arithmetic operations except that they return boolean values. They pop one or two operands off the stack, perform some logical operation,

and then push the result onto the stack. Possible results are 0 for FALSE, or 1 for TRUE.

NOTE: For logical operands, any nonzero value is considered TRUE.

- %= Push TRUE if the two topmost operands are numerically equal.
- %> Push TRUE if the second operand is greater than the topmost operand.
- %< Push TRUE if the second operand is less than the topmost operand.
- %A Push TRUE if the two topmost operands are both logically TRUE (AND).
- %O Push TRUE if either of the two topmost operands are logically TRUE (OR).
- %! Logically invert the topmost operand (NOT).

Miscellaneous Operations

- %l Pop the stack, then push the length of the string indicated by that value. This escape is similar to `strlen(3C)`.
- %i Add one to the first two parameters passed to `tparam()`, or to the single parameter if just one was passed. This is useful for ANSI terminals, which number cursor positions starting from one instead of zero.

`expr%then%`;

`expr%then%else%`;

"If-Then" and "If-Then-Else" (conditional) statements. *Expr*, *then*, and *else* are all parameterized substrings. In operation, `terminfo` evaluates *expr* and then pops the stack. If the popped value is logically TRUE, *then* is evaluated. Otherwise, if *else* was provided, *else* is evaluated. (*expr* typically calculates some logical expression, and *then* and *else* typically print corresponding strings.)

"If-Then-ElseIf" conditionals can be written as a string of "If-Then-Else" statements ala Algol 68, that is:

```
⊘? c1 ⊘t b1 ⊘e c2 ⊘t b2 ... ⊘e cN ⊘t bN ⊘e E ⊘;
```

where $c[1-N]$ are conditionals like *expr*, $b[1-N]$ are bodies like *then*, and *E* is a body like *else*.

A Sample Entry

The following entry, which describes the Concept-100 terminal, is among the more complex entries in the `terminfo` file as of this writing. It is provided here to illustrate the form and content of a `terminfo` entry, and to provide a point of reference for the text that follows.

```
concept100|c100|concept|c104|c100-4p|concept 100,
am, db, eo, in, mir, ul, xenl,
cols#80, lines#24, pb#9600, vt#8,
bel='G, blank=\EH, blink=\EC, clear='L$<2*>, cnorm=\Ew, cr='M$9,
cub1='H, cud1='J, cuf1='\E=, cup='\Ea%p1%' '%+%c%p2%' '%+%c,
cuu1='\E;, cvvis='\EW, dch1='\E^A$<16*>, dim='\EE, dl1='\E^B$<3*>,
ed='\E^C$<16*>, el='\E^U$I6, flash='\Ek$<20>\EK, ht='\t$8, il1='\E^R$<3*>,
.ind='J$9, ind='J, ip='$<16*>,
is2='\EU\Ef\E7\E5\E8\EI\ENH\EK\E\0\Eo&\0\Eo\47\E, kbs='h, kcub1='\E>,
kcud1='\E<, kcu1='\E=, kcuu1='\E;, kf1='\E5, kf2='\E6, kf3='\E7, khome='\E?,
prot='\EI, rep='\Er%p1%c%p2%' '%+%c$<.2*>, rev='\ED,
rmcup='\Ev\s\s\s$<6>\Ep\r\n, rmir='\E\0, rmkx='\Ex, rmso='\Ed\Ee,
rmul='\Eg, rmul='\Eg, sgr0='\EN\0, smcup='\EU\Ev\s\s8p\Ep\r, smir='\E^P,
smkx='\EX, smso='\EE\ED, smul='\EG,
```

Entries may continue onto multiple lines by placing white space at the beginning of each line except the first. Lines beginning with “#” are interpreted as comments.

How to Describe Device Capabilities

In the example, the boolean capabilities appear in the second line. The numeric capabilities appear in the line that follows the booleans. The remainder of the entry consists of string capabilities.

The fact that a device has “automatic margins” (that is, an automatic return and linefeed when the end of a line is reached) is indicated by the boolean capability `am`. Thus, the device description simply gives `am`. Numeric capabilities are followed by the character ‘#’ and then the value assigned. Thus `cols`, which indicates the number of columns the device has, specifies the value `80` for the Concept 100 as `cols#80`. The value may be specified in decimal, octal, or hexadecimal using normal C conventions. Finally, string-valued capabilities, such as `bel` (sound an audible alarm) are specified by the two- to five-character capability name, or capname for short, an ‘=’, and then a string ending at the next following comma. The concept 100 responds to `<Ctrl-G>` by sounding its bell, so the description specifies `bel=^G`.

A delay in milliseconds may appear anywhere in a string capability, bracketed by `$<.>`, as in `el=\EK$<3>`. Padding characters are supplied by `tputs()` (see `curses(3X)`) to provide this delay. The delay can be either a number (for example, `20`); or a number followed by an ‘*’ (for example, `3*`), a ‘/’ (for example, `5/`), or both (for example, `10*/`). A ‘*’ indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. (In the case of insert character, the factor is still the number of lines affected. This is always 1 unless the terminal has `in` defined and the software uses it.) When an ‘*’ is specified, it is sometimes useful to give a delay of the form `3.5` to specify a delay per unit to tenths of milliseconds. (Only one decimal place is allowed.) A ‘/’ indicates that the padding is mandatory. Otherwise, if the device has `xon` defined, the padding information is advisory and is only used for cost estimates or when the device is in raw mode. Mandatory padding is transmitted regardless of the setting of `xon`.

A number of escape sequences are provided in the string valued capabilities for easy encoding of characters there. Both `\E` and `\e` map to an ESCAPE character, `^x` maps to a `<Ctrl-x>` for any appropriate `x`, and the sequences `\n`, `\l`, `\r`, `\t`, `\b`, `\f`, and `\s` give a newline, linefeed, return, tab, backspace, formfeed, and space, respectively. Other escapes include: `\^` for caret (^); `\` for backslash (\); `\,` for comma (,); `\:` for colon (:); and `\0` for null. (`\0` actually produces `\200`, which does not terminate a string but behaves as a null character on most devices.) Finally, characters may be given as three octal digits after a backslash (e.g., `\123`).

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the first `ind` in the example above. Note that when capabilities are defined more than once, a prior definition overrides a later definition.

TERMINFO TERMINAL CAPABILITIES

The following subsections describe `terminfo` terminal capabilities in detail. Subsections are numbered for cross-reference to the table that appears earlier in this man page.

1. Basic Capabilities

The number of columns on each line for the terminal is given by the `cols` numeric capability. If the terminal has a screen, then the number of lines on the screen is given by the `lines` capability. If the terminal cursor wraps around to the beginning of the next line when it reaches the right margin, then the `am` capability should be given. If the terminal can clear its screen, leaving the cursor in the home position, then this is given by the `clear` string capability. If the terminal overstrikes (rather than clearing a position when a character is overwritten) then it should have the `os` capability. If the terminal is a printing terminal, with no soft copy unit, give it both `hc` and `os`. (`os` applies to storage scope terminals, such as the Tektronix 4010 series, as well as hardcopy and APL terminals.) If there is a code to move the cursor to the left edge of the current row, give this as `cr`. (Normally this is carriage return, `^M`.) If there is a code to produce an audible signal (bell, beep, etc) give this as `bel`. If the terminal uses the XON-XOFF flow control protocol, like most terminals, specify the boolean capability `xon`.

If there is a code to move the cursor one position to the left (such as backspace) that capability should be given as `cub1`. Similarly, codes to move to the right, up, and down should be given as `cuf1`, `cuu1`, and `cud1`. These local cursor motions should not alter the text they pass over; for example, you would not normally use `cuf1=\s` because the space would erase the character moved over.

It is important to remember that the local cursor motions encoded in `terminfo` are undefined at the left and top edges of a screen terminal. Programs should never attempt to backspace around the left edge, unless `bw` is specified, and should never attempt to move the cursor up locally off the top.

To scroll text up, a program moves the cursor to the bottom left corner of the screen and sends the `ind` (index) string. To scroll text down, a program moves the cursor to the top left corner of the screen and sends the `ri` (reverse index) string. The strings `ind` and `ri` are undefined when the cursor is not on their respective corners of the screen.

Parameterized versions of the scrolling sequences are `indn` and `rin` which have the same semantics as `ind` and `ri` except that they take one parameter, and scroll that many lines. They are also undefined except at the appropriate corners of the screen.

The `am` capability tells whether the cursor sticks at the right edge of the screen when text is output, but this does not necessarily apply to a `cuf1` from the last column. The only local motion which is defined from the left edge is if `bw` is given, then a `cub1` from the left edge moves to the right edge of the previous row. If `bw` is not given, the effect is undefined. `bw` is useful for drawing a box around the edge of the screen, for example. If the terminal has switch selectable automatic margins, the `terminfo` file usually assumes that this is on; i.e., `am`. If the terminal has a command which moves to the first column of the next line, that command can be given as `nel` (newline). It does not matter if the command clears the remainder of the current line, so if the terminal has no `CR` and `LF` it may still be possible to craft a working `nel` out of one or both of them.

These capabilities suffice to describe hardcopy and screen terminals. Thus the model 33 teletype is described as follows:

```
33|tty33|tty|model 33 teletype,
    bel=^G, cols#72, cr=^M, cud1=^J, hc, ind=^J, os,
```

The Lear Siegler ADM-3 is described as follows:

```
adm3|lsl adm3,
am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,
cud1=^J, ind=^J, lines#24,
```

2. Cursor Motions

If the terminal has a fast way to home the cursor (to the very upper left corner of the screen) then this can be given as `home`; similarly a fast way of getting to the lower left-hand corner can be given as `ll`; this may involve going up with `cuu1` from the home position, but a program should never do this itself (unless `ll` does) because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as addressing to (0,0): to the top left corner of the screen, not of memory. (Thus, the `\EH` sequence on Hewlett-Packard terminals cannot be used for `home` without losing some of the other features on the terminal.)

If the terminal has a way to move the cursor to any selected position on the screen, specify this with the `cup` string capability, which takes two parameters: the row and column of the new cursor position. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory.) If the terminal has memory relative cursor addressing, that can be indicated by the string capability `mr cup`.

If the terminal has row or column absolute cursor addressing, these can be given as single parameter capabilities `hpa` (horizontal position absolute) and `vpa` (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to `cup`. If there are parameterized local motions (e.g., move *n* spaces to the right) these can be given as `cud`, `cub`, `cuf`, and `cuu` with a single parameter indicating how many spaces to move. These are primarily useful if the terminal does not have `cup`, as with the Tektronix 4025.

3. Area Clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as `e1`. If the terminal can clear from the beginning of the line to the current position inclusive, leaving the cursor where it is, this should be given as `e11`. If the terminal can clear from the current position to the end of the display, then this should be given as `ed`. `ed` is only defined from the first column of a line. (Thus, it can be simulated by a request to delete a large number of lines, if a true `ed` is not available.)

4. Insert/delete line

If the terminal can open a new blank line before the line containing the cursor, this should be given as `i11`; this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as `d11`; this is done only from the first position on the line to be deleted. Versions of `i11` and `d11` which take a single parameter and insert or delete that many lines can be given as `i1` and `d1`.

If the terminal has a destructive programmable scrolling region (like the VT100), the command to set the region can be described with the `csr` string capability, which takes two parameters: the top and bottom lines of the scrolling region. It is possible to get the effect of insert or delete line using this command – the `sc` and `rc` (save and restore cursor) string capabilities are also useful. The cursor position is, alas, undefined after using this command. It must be reset using other terminfo capabilities such as `cup`, `home`, or `rc`. Inserting lines at the top or bottom of the screen can also be done using `ri` or `ind` on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

To determine whether a terminal has destructive scrolling regions or non-destructive scrolling regions, create a scrolling region in the middle of the screen, place data on the bottom line of the scrolling region, move the cursor to the top line of the scrolling region, and do a reverse index (`ri`) followed by a delete line (`d11`) or index (`ind`). If the data that was originally on the bottom line of the scrolling region was restored into the scrolling region by the `d11` or `ind`, then the terminal has non-destructive scrolling regions. Otherwise, it has destructive scrolling regions. Do not specify `csr` if the terminal has non-destructive scrolling regions, unless `ind`, `ri`, `indn`, `rin`, `d1`, and `d11` all simulate destructive scrolling.

If the terminal has the ability to define a window as part of memory, which all commands affect, it should be given as the parameterized string `wind`. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above, then the `da` boolean capability should be given; if display memory can be retained below, then `db` should be given. These indicate that deleting a line or scrolling a full screen may bring non-blank lines up from below or that scrolling back with `ri` may bring down non-blank lines.

5. Insert/Delete Character

There are two basic kinds of intelligent terminals with respect to insert/delete character operations which can be described using `terminfo`. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly (i.e., all characters to the right of the insertion or deletion shift as a unit). Other terminals, such as the Concept-100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks.

You can determine the kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type “`abc def`” using local cursor motions (not spaces) between the `abc` and the `def`. Then position the cursor before the `abc` and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to “fall off” the end, then your terminal does not distinguish between blanks and untyped positions. If the `abc` shifts over to the `def` which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and thus you should define the boolean capability `in`, which stands for “insert null”. While these are two logically separate attributes (one line versus multiline insert mode, and special treatment of untyped spaces), we have seen no terminals whose insert mode cannot be described with the single attribute.

`Terminfo` can describe both terminals which have an insert mode and terminals which send a simple sequence to open a blank position on the current line. Give as `smir` the sequence to get into insert mode. Give as `rmi.r` the sequence to leave insert mode. Now give as `ich1` any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode do not specify `ich1`; terminals which send a sequence to open a screen position should specify it here. (If your terminal has both, insert mode is usually preferable to `ich1`. Do not give both unless the terminal actually requires both to be used in combination.)

If post-insert padding is needed, give this as a number of milliseconds padding in `ip` (a string capability). Any other sequence that may need to be sent after an insert of a single character may also be given in `ip`. If your terminal needs both to be placed into an ‘insert mode’ and a special code to precede each inserted character, then both

`smir/rmir` and `ich1` can be given, and both are used.

The `ich` capability, with one parameter, n , repeats the effects of `ich1` n times.

If padding is necessary between characters typed while not in insert mode, give this as a number of milliseconds padding in `rmp`.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (e.g., if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability `mir` to speed up inserting in this case. Omitting `mir` affects only speed. Some terminals (notably Datamedia's) must not have `mir` because of the way their insert mode works.

Finally, you can give `dch1` to delete a single character, `dch` with one parameter, n , to delete n characters, and `smdc` and `rmdc` to enter and exit delete mode (any mode the terminal needs to be placed in for `dch1` to work).

A command to erase n characters (equivalent to outputting n blanks without explicitly moving the cursor) can be given as `ech` with one parameter.

6. Highlighting, Underlining, and Visible Bells

If your terminal has one or more kinds of display attributes (graphic embellishments to text), these can be represented in a number of different ways. You should choose one display form as “standout mode” (see `curses(3X)`), representing a good, high contrast, easy-on-the-eyes format for highlighting error messages and other attention getters. (If you have a choice, reverse video plus half-bright is good, or reverse video alone; however, different users have different preferences on different terminals.)

The sequences to enter and exit standout mode are given as `smso` and `rmso`, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, as on the TVI 912 and the Teleray 1061, then `xmc` should be given to tell how many spaces are left.

Codes to begin underlining and end underlining can be given as `smul` and `rmul` respectively. If the terminal has a code to underline the current character and move the cursor one space to the right, such as the Micro-Term MIME, this can be given as `uc`.

Other capabilities to enter various highlighting modes include `blink` (blinking), `bold` (bold or extra-bright), `dim` (dim or half-bright), `invis` (blinking or invisible text), `prot` (protected), `rev` (reverse video), `sgr0` (turn off all attribute modes), `smacs` (enter alternate-character-set mode), and `rmacs` (exit alternate-character-set mode). Turning on any of these modes singly may or may not turn off other modes. If a command is necessary before alternate character set mode is entered, give the sequence in `enacs` (enable alternate-character-set mode).

If there is a sequence to set arbitrary combinations of modes, this should be given as `sgr` (set attributes), taking nine parameters. Each parameter is either zero or non-zero, as the corresponding attribute is on or off. The nine parameters are, in order: standout, underline, reverse, blink, dim, bold, invisible, protected, and alternate character set. Not all modes need be supported by `sgr`, only those for which corresponding separate attribute commands exist. (See the example at the end of this section.)

Terminals with the “magic cookie” glitch (`xmc`) deposit special “cookies” when they receive mode-setting sequences, rather than having extra attribute bits for each character. These “cookies” affect the display algorithm to provide video attributes, but also take up (blank) space on the screen.

Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when the cursor is moved to a new line or is addressed. Programs using standout mode should exit standout mode before moving the cursor or sending a newline, unless the `msgcr` capability, asserting that it is safe to move in standout mode, is present.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), then this can be given as `flash`; it must not move the cursor. A good flash can be done by changing the screen into reverse video, padding for 200 ms, then returning the screen to normal video.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to make, for example, a non-blinking underline into an easier to find block or blinking underline) give this sequence as `cvvis`. The boolean `chts` should also be given. If there is a way to make the cursor completely invisible, give that as `civis`. The capability `cnorm` should be given which undoes the effects of either of these modes.

If the terminal needs to be in a special mode when running a program that uses `terminfo` capabilities, the codes to enter and exit this mode can be given as `smcup` and `rmcup`. This arises, for example, from terminals like the Concept-100 with more than one page of memory. If the terminal has only memory relative cursor addressing and not screen relative cursor addressing, a window the size of the screen must be fixed into the terminal for cursor addressing to work properly. This is also used for the Tektronix 4025, where `smcup` sets the command character to the one used by `terminfo`. If the `smcup` sequence does not restore the screen after an `rmcup` sequence is output (to the state prior to outputting `rmcup`), specify the boolean capability `nrrmc`.

If your terminal generates underlined characters by using the underline character (with no special codes needed) even though it does not otherwise overstrike characters, then you should give the capability `ul`. For terminals where a character overstriking another leaves both characters on the screen, give the capability `os`. If overstrikes are erasable with a blank, then this should be indicated by giving `eo`.

Here is an example of highlighting: assume that a terminal needs the following escape sequences to turn on various modes.

tparam parameter	attribute	escape sequence
	none	\E[0m
p1	standout	\E[0;4;7m
p2	underline	\E[0;3m
p3	reverse	\E[0;4m
p4	blink	\E[0;5m
p5	dim	\E[0;7m
p6	bold	\E[0;3;4m
p7	invis	\E[0;8m
p8	protect	not available
p9	altcharset	^O (off) ^N(on)

Note that each escape sequence requires a 0 to turn off other modes before turning on its own mode. Combinations of attributes are allowed by appending a digit that represents each attribute, separated by a semicolon. For instance, underline + blink needs the sequence `\E[0;3;5m`. Note that, as suggested above, *standout* is set up to be the combination of *reverse* and *dim*. Also, since this terminal has no *bold* mode,

bold is set up as the combination of *reverse* and *underline*. The terminal doesn't have *protect* mode, either, but that cannot be simulated in any way, so *p8* is ignored. The *altcharset* mode is different in that it requires either `<Ctrl-O>` or `<Ctrl-N>` depending on whether it is to be turned off or on. If all modes were to be turned on, the sequence would be `\E[0;3;4;5;7;8m^N`.

Now look at the cases in which different sequences are output. For example, `;3` is output when either *p2* or *p6* is true, that is, if either *underline* or *bold* modes are turned on. Writing out the above sequences, along with their dependencies, gives the following:

sequence	when to output	terminfo translation
<code>\E[0</code>	always	<code>\E[0</code>
<code>;3</code>	if <i>p2</i> or <i>p6</i>	<code>%?%p2%p6% %t;3%;</code>
<code>;4</code>	if <i>p1</i> or <i>p3</i> or <i>p6</i>	<code>%?%p1%p3% %p6% %t;4%;</code>
<code>;5</code>	if <i>p4</i>	<code>%?%p4%t;5%;</code>
<code>;7</code>	if <i>p1</i> or <i>p5</i>	<code>%?%p1%p5% %t;7%;</code>
<code>;8</code>	if <i>p7</i>	<code>%?%p7%t;8%;</code>
<code>m</code>	always	<code>m</code>
<code>^N</code> or <code>^O</code>	if <i>p9</i> <code>^N</code> , else <code>^O</code>	<code>%?%p9%t^N%e^O%;</code>

Putting this all together into the `sgr` sequence gives:

```
sgr=\E[0%?%p2%p6%|%t;3%;%?%p1%p3%|%p6%|%t;4%;%?%p5%t;5%;
%?%p1%p5%|%t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;
```

7. Keypad

If the terminal has a keypad that transmits codes when special keys are pressed, this information can be given. Note that it is not possible to handle terminals where the keypad only works in local mode (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, give these codes as `smkx` and `rmkx`. Otherwise the keypad is assumed to always transmit.

The codes sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as `kcub1`, `kcuf1`, `kcuu1`, `kcud1`, and `khome` respectively. If there are function keys such as `F0`, `F1`, ..., `F63`, the codes they send can be given as `kf0`, `kf1`, ..., `kf63`. If the first 11 keys have labels other than the default `F0` through `F10`, the labels can be given as `lf0`, `lf1`, ..., `lf10`. The codes transmitted by certain other special keys can be given: `kll` (home down), `kbs` (backspace), `ktbc` (clear all tabs), `kctab` (clear the tab stop in this column), `kclr` (clear screen or erase), `kdch1` (delete character), `kdl1` (delete line), `krmir` (exit insert mode), `kel` (clear to end of line), `ked` (clear to end of screen), `kich1` (insert character or enter insert mode), `kil1` (insert line), `knp` (next page), `kpp` (previous page), `kind` (scroll forward/down), `kri` (scroll backward/up), `khts` (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, the other five keys can be given as `ka1`, `ka3`, `kb2`, `kc1`, and `kc3`. These keys are useful when the effects of a 3 by 3 directional pad are needed. Further keys are defined above in the capabilities list.

Strings to program function keys can be given as `pfkey`, `pfloc`, and `pfx`. A string to program their soft screen labels can be given as `pln`. Each of these strings takes two parameters: the function key number to program (from 0 to 10) and the string to program it with. Function key numbers out of this range may program undefined keys in a terminal-dependent manner. The difference between the capabilities is that `pfkey` causes the given key to act as if the user had typed the given string; `pfloc` causes the string to be executed by the terminal in local mode; and `pfx` causes the

string to be transmitted to the computer. The capabilities `nlab`, `lw`, and `lh` define how many soft labels there are and how wide and high they are. If there are commands to turn the labels on and off, give them as `smln` and `rmln`. `smln` is normally output after one or more `pln` sequences to make sure that the change becomes visible.

8. Tabs and Initialization

If the terminal has hardware tabs, the command to advance to the next tab stop can be given as `ht` (usually *Ctrl-I*). A “backtab” command which moves leftward to the previous tab stop can be given as `cbt`. By convention, if the terminal driver modes indicate that tabs are being expanded by the computer rather than being sent to the terminal, programs should not use `ht` or `cbt` even if they are present, since the user may not have the tab stops properly set.

If the terminal has hardware tabs which are initially set every n spaces when the terminal is powered up, the numeric parameter `it` should be given, showing the number of spaces n to which the tabs are set. This is normally used by `tput init` (see `tput(1)`) to determine whether to set the mode for hardware tab expansion and whether to set the tab stops.

If the terminal has tab stops that can be saved in nonvolatile memory, the `terminfo` description can assume that they are properly set. If there are commands to set and clear tab stops, they can be given as `tbc` (clear all tab stops) and `hts` (set a tab stop in the current column of every row).

Other capabilities include: `is1`, `is2`, and `is3`, initialization strings for the terminal; `ipro`, the path name of a program to run to initialize the terminal; and `if`, the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the `terminfo` description. They must be sent to the terminal each time the user logs in and be output in the following order: run the program `ipro`; output `is1`; output `is2`; set the margins using `mgc`, `smgl`, and `smgr`; set the tabs using `tbc` and `hts`; print the file `if`; and finally output `is3`. This is usually done using the `init` option of `tput(1)`; see `profile(4)`.

Most initialization is done with `is2`. Special terminal modes can be set up without duplicating strings by putting the common sequences in `is2` and special cases in `is1` and `is3`. Sequences that do a harder reset from a totally unknown state can be given as `rs1`, `rs2`, `rf`, and `rs3`, analogous to `is1`, `is2`, `if`, and `is3`. (The method using files, `if` and `rf`, is used for a few terminals, from `/usr/lib/tabset/*`; however, the recommended method is to use the initialization and reset strings.) These strings are output by `tput reset`, which is used when the terminal gets into a wedged state. Commands are normally placed in `rs1`, `rs2`, `rs3`, and `rf` only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set a terminal into 80-column mode would normally be part of `is2`, but on some terminals it causes an annoying glitch on the screen and is not normally needed since the terminal is usually already in 80-column mode.

If a more complex sequence is needed to set the tabs than can be described by using `tbc` and `hts`, the sequence can be placed in `is2` or `if`.

If there are commands to set and clear margins, they can be given as `mgc` (clear all margins), `smgl` (set left margin), and `smgr` (set right margin).

9. Delays

Certain capabilities control padding in the terminal driver (see `termio(7)` and `ttcompat(7)`). These are primarily needed by hardcopy terminals, and are used by `tput init` to set terminal driver modes appropriately. Delays embedded in the capabilities `cr`, `ind`, `cubl`, `ff`, and `tab` can be used to set the appropriate delay bits in the terminal driver. If `pb` (padding baud rate) is given, these values can be ignored at baud rates below the value of `pb`.

10. Status Lines

If the terminal has an extra “status line” that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, into which a program can cursor address normally (such as the Heathkit h19’s 25th line, or the 24th line of a VT100 which is set to a 23-line scrolling region), the capability `hs` should be given. Special strings that go to a given column of the status line and return from the status line can be given as `tsl` and `fsl`. (`fsl` must leave the cursor position in the same place it was before `tsl`. If necessary, the `sc` and `rc` strings can be included in `tsl` and `fsl` to get this effect.) The capability `tsl` takes one parameter, which is the column number of the new cursor position in the status line.

If escape sequences and other special commands, such as `tab`, work while in the status line, the flag `eslok` can be given. A string which turns off the status line (or otherwise erases its contents) should be given as `dsl`. If the terminal has commands to save and restore the position of the cursor, give them as `sc` and `rc`. The status line is normally assumed to be the same width as the rest of the screen, e.g., `cols`. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded) the width, in columns, can be indicated with the numeric parameter `ws1`.

11. Line Graphics

If the terminal has a line drawing alternate character set, the mapping of glyph to character would be given in `acsc`. The definition of this string is based on the alternate character set used in the DEC VT100 terminal, extended slightly with some characters from the AT&T 4410v1 terminal.

glyph name	vt100+ character
arrow pointing right	+
arrow pointing left	,
arrow pointing down	.
solid square block	0
lantern symbol	I
arrow pointing up	-
diamond	‘
checker board (stipple)	a
degree symbol	f
plus/minus	g
board of squares	h
lower right corner	j
upper right corner	k

upper left corner	l
lower left corner	m
plus	n
scan line 1	o
horizontal line	q
scan line 9	s
left tee (┌)	t
right tee (┐)	u
bottom tee (└)	v
top tee	w
vertical line	x
bullet	~

The best way to describe a new terminal's line graphics set is to add a third column to the above table with the characters for the new terminal that produce the appropriate glyphs when the terminal is in the alternate character set mode. For example,

glyph name	vt100+ char	new tty char
upper left corner	l	R
lower left corner	m	F
upper right corner	k	T
lower right corner	j	G
horizontal line	q	,
vertical line	x	.

Now write down the characters left to right, as in `acsc=lRmFkTjGq\,x`.

12. Color Manipulation

Let us define two methods of color manipulation: the Tektronix method and the HP method. The Tektronix method uses a set of *N* predefined colors (usually 8) from which a program can select "current" foreground and background colors. Thus a terminal can support up to *N* colors mixed into *N***N* color-pairs to be displayed on the screen at the same time. When using an HP method the program cannot define the foreground independently of the background, or vice-versa. Instead, the program must define an entire color-pair at once. Up to *M* color-pairs, made from 2**M* different colors, can be defined this way. Most existing color terminals belong to one of these two classes.

The numeric capabilities `colors` and `pairs` define the number of colors and color-pairs that can be displayed on the screen at the same time. If a terminal can change the definition of a color (for example, the Tektronix 4100 and 4200 series terminals), this should be specified with the boolean capability `ccc` (can change color). To change the definition of a color (Tektronix method), use the parameterized string capability `initc` (initialize color). It requires four parameters: color number (ranging from 0 to `colors-1`) and three RGB (red, green, and blue) values (ranging from 0 to 1000).

Tektronix 4100 series terminals use a type of color notation called HLS (Hue Lightness Saturation) instead of RGB color notation. For such terminals one must define a boolean capability `hls`. The last three parameters of the `initc` string would then be HLS values: *H*, ranging from 0 to 360; and *L* and *S*, ranging from 0 to 100.

To set the current foreground or background to a given color, use parameterized string capabilities `setf` (set foreground) and `setb` (set background). They each require one parameter: the number of the color. To initialize a color-pair (HP

method), use `initp` (initialize pair). It requires seven parameters: the number of a color-pair (ranging from 0 to `pairs-1`), and six RGB values: three for the foreground followed by three for the background. (When `initc` or `initp` is used, RGB or HLS arguments should be in the order "red, green, blue" or "hue, lightness, saturation", respectively.) To make a color-pair current, use the parameterized string capability `scp` (set color-pair). It takes one parameter, the number of a color-pair.

If a terminal can change the definitions of colors, but uses a color notation different from RGB and HLS, a mapping to either RGB or HLS must be developed and encoded in the `initc` and `initp` capabilities.

Some terminals (for example, most color terminal emulators for PCs) erase areas of the screen using the current background color. In such cases, the boolean capability `bce` (background color erase) should be defined. The string capability `op` (original pair) contains a sequence for setting the foreground and background colors to what they were at the terminal start-up time. Similarly, `oc` (original colors) contains a sequence for setting all colors (for the Tektronix method) or color-pairs (for the HP method) to the values they had at the terminal start-up time.

Some video attributes on some color terminals should not be combined with colors. For instance, some color terminals substitute color for video attributes, so each attribute can be displayed in only one color. Information about these video attributes should be packed into the numeric capability `ncv` (no color video). There is a one-to-one correspondence between the nine least significant bits of this capability and the video attributes. The following table depicts this correspondence.

Attribute	Bit Position	Decimal Value
<code>A_STANDOUT</code>	0	1
<code>A_UNDERLINE</code>	1	2
<code>A_REVERSE</code>	2	4
<code>A_BLINK</code>	3	8
<code>A_DIM</code>	4	16
<code>A_BOLD</code>	5	32
<code>A_INVIS</code>	6	64
<code>A_PROTECT</code>	7	128
<code>A_ALTCHARSET</code>	8	256

When a particular video attribute should not be used with colors, the corresponding `ncv` bit should be set to 1; otherwise it should be set to zero. To determine the information to pack into the `ncv` capability, you must add together the decimal values corresponding to those attributes that cannot coexist with colors. For example, if the terminal uses colors to simulate reverse video (bit number 2 and decimal value 4) and bold (bit number 5 and decimal value 32), the resulting value for `ncv` will be 36 (4 + 32).

13. Miscellaneous

If the terminal requires any character other than a null (zero) as a pad, then this can be given as `pad`. Only the first character of the `pad` string is used. If the terminal does not have a pad character, specify `npc`.

If the terminal can move up or down half a line, this can be indicated with `hu` (half-line up) and `hd` (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as `ff` (usually `^L`).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters) this can be indicated with the parameterized string `rep`. The first parameter is the character to be repeated and the second is the number of times to repeat it. Thus, `tparam(repeat_char, 'x', 10)` produces the same effect as `xxxxxxxxxx`.

If the terminal has a programmable command character, such as the Tektronix 4025, this can be indicated with `cmdch`. A prototype command character is chosen which is used in all capabilities. This character is given in the `cmdch` capability to identify it. The following convention is supported on some UNIX systems: If the environment variable `CC` exists, all occurrences of the prototype character are replaced with the character in `cc`.

Terminal descriptions that do not represent a specific kind of known terminal, such as `switch`, `dialup`, `patch`, and `network`, should include the `gn` (generic) capability so that programs can complain that they do not know how to talk to the terminal. (This capability does not apply to virtual terminal descriptions for which the escape sequences are known.) If the terminal is one of those supported by the UNIX system virtual terminal protocol, the terminal number can be given as `vt`. A line-turn-around sequence to be transmitted before doing reads should be specified in `rfi`.

If the terminal uses XON/XOFF handshaking for flow control, define `xon`. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters are not transmitted. Sequences to turn on and off XON/XOFF handshaking may be given in `smxon` and `rmxon`. If the characters used for handshaking are not `<Ctrl-S>` and `Ctrl-Q`, they may be specified with `xonc` and `xoffc`.

If the terminal has a “meta key” which acts as a shift key, setting the eighth bit of any character transmitted, this can be specified with the boolean capability `km`. Otherwise, software assumes that the eighth bit is parity and it is usually cleared. If strings exist to turn this “meta mode” on and off, they can be specified as `smm` and `rmm`.

If the terminal has more lines of memory than can fit on the screen at once, the number of lines of memory can be indicated with `lm`. A value of zero for `lm` indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

If the terminal cursor can wrap around to the beginning of the next line when it reaches the right margin, this can be specified with the boolean capability `am`. If a string exists to enable this wrapping, specify it as `smam`. A string to make the cursor stick in the last column of a line is specified as `rmam`.

Media copy strings which control an auxiliary printer connected to the terminal can be given as `mc0`: print the contents of the screen, `mc4`: turn off the printer, and `mc5`: turn on the printer. When the printer is on, all text sent to the terminal is sent to the printer. A variation, `mc5p`, takes one parameter, and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. If the text is not displayed on the terminal screen when the printer is on, specify `mc5i` (silent printer). All text, including `mc4`, is transparently passed to the printer while an `mc5p` is in effect.

14. Special Cases

The working model used by `terminfo` fits most terminals reasonably well. However, some terminals do not completely match that model, requiring special support by `terminfo`. These are not meant to be construed as deficiencies in the terminals;

they are just differences between the working model and the actual hardware. They may be unusual devices or, for some reason, do not have all the features of the `terminfo` model implemented.

Terminals which cannot display tilde (~) characters, such as certain Hazeltine terminals, should indicate `hz`.

Terminals which ignore a linefeed immediately after an `am` wrap, such as the Concept-100, should indicate `xenl`. Those terminals whose cursor remains on the rightmost column until another character has been received, rather than wrapping immediately upon receiving the rightmost character, such as the VT100, should also indicate `xenl`.

If `e1` is required to get rid of standout mode (instead of writing normal text on top of it), `xhp` should be given.

Those Teleray terminals whose tabs overwrite blanks should indicate `xt` (destructive tabs). This capability is also taken to mean that it is not possible to position the cursor on top of a "magic cookie"; therefore, to erase standout mode, it is instead necessary to use delete and insert line.

Those Beehive Superbee terminals which do not transmit the `<ESC>` or `<Ctrl-C>` characters should specify `xsb`, indicating that the F1 key is to be used for `<ESC>` and the F2 key for `Ctrl-C`.

Most terminals can use padding as an alternative to XON-XOFF flow control. Some terminals, though, require XON-XOFF flow control. For these, specify the boolean capability `nxon`.

TERMINFO PRINTER CAPABILITIES

The `terminfo` database allows you to define capabilities of printers as well as terminals. To find out what capabilities are available for printers as well as for terminals, see the table in the "Device Capabilities" section. Most subsections below are lettered for cross-reference to that table.

Rounding Values

Because parameterized string capabilities work only with integer values, we recommend that `terminfo` designers create strings that expect numeric values that have been rounded. Application designers should note this and should always round values to the nearest integer before using them with a parameterized string capability.

Printer Resolution

A printer's resolution is defined to be the smallest spacing of characters it can achieve. In general printers have independent resolution horizontally and vertically. Thus the vertical resolution of a printer can be determined by measuring the smallest achievable distance between consecutive printing baselines, while the horizontal resolution can be determined by measuring the smallest achievable distance between the leftmost edges of consecutive printed, identical, characters. (The terms "smallest distance" and "smallest step" will be used later to refer to these smallest achievable distances.)

All printers are assumed to be capable of printing with a uniform horizontal and vertical resolution. The view of printing that `terminfo` currently presents is one of printing inside a uniform matrix: All characters are printed at fixed positions relative to each "cell" in the matrix; furthermore, each cell has the same size given by the smallest horizontal and vertical step sizes dictated by the resolution. (The cell size can be changed as will be seen later.)

Many printers are capable of “proportional printing,” where the horizontal spacing depends on the size of the character last printed. Terminfo does not make use of this capability, although it does provide enough capability definitions to allow an application to simulate proportional printing.

A printer must not only be capable of printing characters as close together as the horizontal and vertical resolutions suggest, but also of “moving” to a position an integral multiple of the resolution from a previous position. Thus printed characters can be spaced apart a distance that is an integral multiple of the smallest distance, up to the length or width of a single page.

Some printers can have different resolutions depending on different “modes.” In “normal mode,” the existing terminfo capabilities are assumed to work on columns and lines, just like a video terminal. Thus the old `lines` capability would give the length of a page in lines, and the `cols` capability would give the width of a page in columns. In “micro mode,” many terminfo capabilities work on increments of lines and columns. With some printers the micro mode may be concomitant with normal mode, so that all the capabilities work at the same time.

A. Specifying Printer Resolution

The printing resolution of a printer is given in several ways. Each specifies the resolution as the number of smallest steps per distance:

<u>Numeric Capabilities for Specifying Characteristic Number of Smallest Steps</u>	
orhi	Steps per inch horizontally
orvi	Steps per inch vertically
orc	Steps per column
orl	Steps per line

When printing in normal mode, each character printed causes movement to the next column, except in special cases described later; the distance moved is the same as the per-column resolution. Some printers cause an automatic movement to the next line when a character is printed in the rightmost position; the distance moved vertically is the same as the per-line resolution. When printing in micro mode, these distances can be different, and may be zero for some printers.

<u>Numeric Capabilities for Specifying Automatic Motion after Printing</u>	
<i>Normal Mode:</i>	
orc	Steps moved horizontally
orl	Steps moved vertically
<i>Micro Mode:</i>	
mcs	Steps moved horizontally
mls	Steps moved vertically

Some printers are capable of printing wide characters. The distance moved when a wide character is printed in normal mode may be different from when a regular width character is printed. The distance moved when a wide character is printed in micro mode may also be different from when a regular character is printed in micro mode, but the differences are assumed to be related: If the distance moved for a regular character is the same whether in normal mode or micro mode (**mcs=orc**), then the distance moved for a wide character is also the same whether in normal mode or micro mode. This doesn't mean the normal character distance is necessarily the same as the wide character distance, just that the distances don't change with a change in normal to micro mode. However, if the distance moved for a regular character is

different in micro mode from the distance moved in normal mode ($mcs < orc$), the micro mode distance is assumed to be the same for a wide character printed in micro mode, as the table below shows.

<u>Numeric Capabilities for Specifying Automatic Motion after Printing Wide Character</u>	
<i>Normal Mode or Micro Mode</i> ($mcs = orc$):	
widcs	Steps moved horizontally
<i>Micro Mode</i> ($mcs < orc$):	
mcs	Steps moved horizontally

There may be control sequences to change the number of columns per inch (the character pitch) and to change the number of lines per inch (the line pitch). If these are used, the resolution of the printer changes, but the type of change depends on the printer:

<u>String and Boolean Capabilities for Changing the Character/Line Pitches</u>	
cp	Change character pitch
cpix	If set, cp changes orhi , otherwise changes orc
lp	Change line pitch
lpix	If set, lp changes orvi , otherwise changes orl
chr	Change steps per column
cvr	Change steps per line

The **cp** and **lp** string capabilities each require a single parameter, the pitch in columns (or characters) and lines per inch, respectively. The **chr** and **cvr** string capabilities each require a single parameter, the number of steps per column and line, respectively.

Using any of the control sequences in these strings will imply a change in some of the values of **orc**, **orhi**, **orl**, and **orvi**. Also, the distance moved when a wide character is printed, **widcs**, changes in relation to **orc**. The distance moved when a character is printed in micro mode, **mcs**, changes similarly, with one exception: if the distance is 0 or 1, then no change is assumed (see items marked with † in the following table).

Programs that use **cp**, **lp**, **chr**, or **cvr** should recalculate the printer resolution (and should recalculate other values — see the topic "Effect of Changing Printing Resolution" in the section "Dot-Matrix Graphics").

<u>Specification of Printer Resolution Effects of Changing the Character/Line Pitches</u>	
<u>Before</u>	<u>After</u>
<i>Using cp with cpix clear:</i>	
orhi †	orhi
orc †	$orc = \frac{orhi}{V_{cp}}$

Using cpi with cpix set:

orhi '	$\mathbf{orhi=orc \cdot V_{cpi}}$
orc '	orc

Using lpi with lpix clear:

orvi '	orvi
orl '	$\mathbf{orl= \frac{orvi}{V_{lpi}}}$

Using lpi with lpix set:

orvi '	$\mathbf{orvi=orl \cdot V_{lpi}}$
orl '	orl

Using chr:

orhi '	orhi
orc '	V_{chr}

Using cvr:

orvi '	orvi
orl '	V_{cvr}

Using cpi or chr:

widcs '	$\mathbf{widcs=widcs \cdot \frac{orc}{orc}}$
---------	--

mcs ' †	$\mathbf{mcs=mcs \cdot \frac{orc}{orc}}$
---------	--

V_{cpi} , V_{lpi} , V_{chr} , and V_{cvr} are the parameters required by `cpi`, `lpi`, `chr`, and `cvr`, respectively. The ' mark indicates the old value.

B. Capabilities that Cause Movement

In the following descriptions, “movement” refers to the motion of the “current position.” With video terminals this would be the cursor; with some printers this is the carriage position. Other printers have different equivalents. In general, the current position is where a character would be displayed if printed.

Terminfo has string capabilities for control sequences that cause movement a number of full columns or lines. It also has equivalent string capabilities for control sequences that cause movement a number of smallest steps.

String Capabilities for Specifying
Single and Multiple Motions

mcub1	Move 1 step left
mcuf1	Move 1 step right
mcuu1	Move 1 step up
mcud1	Move 1 step down
mcub	Move <i>N</i> steps left
mcuf	Move <i>N</i> steps right
mcuu	Move <i>N</i> steps up
mcud	Move <i>N</i> steps down
mhpa	Move <i>N</i> steps from the left
mvpa	Move <i>N</i> steps from the top

The latter six strings each require a single parameter, *N*.

Some printers limit the motion to less than the width or length of a page. Also, some printers don't accept absolute motion to the left of the current position. `Terminfo` has capabilities for specifying these limits.

Numeric and Boolean Capabilities for
Specifying Limits to Motion

mjump	Limit on use of <code>mcub1</code> , <code>mcuf1</code> , <code>mcuu1</code> , and <code>mcud1</code>
maddr	Limit on use of <code>mhpa</code> and <code>mvpa</code>
xhpa	If set, <code>hpa</code> and <code>mhpa</code> cannot move left
xvpa	If set, <code>vpa</code> and <code>mvpa</code> cannot move up

If a printer needs to be in a "micro mode" for the motion capabilities described above to work, there are string capabilities defined to enter and exit this mode. A boolean capability is available for those printers where using a carriage return causes an automatic return to normal mode.

String and Boolean Capabilities for
Entering and Exiting Micro Mode

smicm	Enter micro mode
rmicm	Exit micro mode
crxm	If set, using <code>cr</code> exits micro mode

The movement made when a character is printed in the rightmost position varies among printers. Some make no movement, some move to the beginning of the next line, others move to the beginning of the same line. `Terminfo` has boolean capabilities for describing all three cases.

Boolean Capabilities for Specifying
What Happens After Character
Printed in Rightmost Position

sam	Automatic move to beginning of same line
------------	--

Some printers can be put in a mode where the normal direction of motion is reversed. This mode can be especially useful when there are no capabilities for leftward or upward motion, because those capabilities can be built from the motion reversal capability and the rightward or downward motion capabilities. It is best to leave it up to an application to build the leftward or upward capabilities, though, and not enter them in the `terminfo` database. This allows several reverse motions to be

strung together without intervening wasted steps that leave and reenter reverse mode.

String Capabilities for
Entering and Exiting Reverse Modes

slm	Reverse sense of horizontal motions
rlm	Restore sense of horizontal motions
sum	Reverse sense of vertical motions
rum	Restore sense of vertical motions

While sense of horizontal motions reversed:

mcub1	Move 1 step right
mcuf1	Move 1 step left
mcub	Move <i>N</i> steps right
mcuf	Move <i>N</i> steps left
cub1	Move 1 column right
cuf1	Move 1 column left
cub	Move <i>N</i> columns right
cuf	Move <i>N</i> columns left

While sense of vertical motions reversed:

mcuu1	Move 1 step down
mcud1	Move 1 step up
mcuu	Move <i>N</i> steps down
mcud	Move <i>N</i> steps up
cuu1	Move 1 line down
cud1	Move 1 line up
cuu	Move <i>N</i> lines down
cud	Move <i>N</i> lines up

The reverse motion modes should not affect the `mvp` and `mhp` absolute motion capabilities. The reverse vertical motion mode should, however, also reverse the action of the line “wrapping” that occurs when a character is printed in the rightmost position. Thus printers that have the standard `terminfo` capability `am` defined should experience motion to the beginning of the previous line when a character is printed in the rightmost position under reverse vertical motion mode.

The action when any other motion capabilities are used in reverse motion modes is not defined; thus, programs must exit reverse motion modes before using other motion capabilities.

Two miscellaneous capabilities complete the list of new motion capabilities. One of these is needed for printers that move the current position to the beginning of a line when certain control characters, such as “linefeed” or “formfeed,” are used. The other is used for the capability of suspending the motion that normally occurs after printing a character.

String Capabilities for Specifying
Miscellaneous Motion

docr	List of control characters causing <code>cr</code>
zerom	Prevent auto motion after printing next single character

C. Margins

`Terminfo` provides two strings for setting margins on terminals: one for the left margin and one for the right. Printers, however, have two additional margins, for the top and bottom of each page. Furthermore, instead of using motion strings to move the current position to a margin and then fixing the margin there, some printers require

the specification of where a margin should be regardless of the current position. Therefore `terminfo` offers six additional strings for defining margins with printers.

String Capabilities for
Setting Margins

smgl	Set left margin at current column
smgr	Set right margin at current column
smgb	Set bottom margin at current line
smgt	Set top margin at current line
smgbp	Set bottom margin at line <i>N</i>
smglp	Set left margin at column <i>N</i>
smgrp	Set right margin at column <i>N</i>
smgtp	Set top margin at line <i>N</i>

The last four strings each require one or more parameters that give the position of the margin or margins to set. If both of `smglp` and `smgrp` are defined, each requires a single parameter, *N*, that gives the column number of the left and right margin, respectively. If both of `smgtp` and `smgbp` are defined, they are used to set the top and bottom margin, respectively: `smgtp` requires a single parameter, *N*, the line number of the top margin; however, `smgbp` requires two parameters, *N* and *M*, that each give the line number of the bottom margin, the first counting from the top of the page and the second counting from the bottom. This accommodates the two methods used by different manufacturers to specify the bottom margin. When coding a `terminfo` entry for a printer that has a settable bottom margin, only the first or second parameter should be used, depending on the printer. When writing an application that uses `smgbp` to set the bottom margin, both arguments must be given.

If only one of `smglp` and `smgrp` is defined, then it requires two parameters, the column numbers of the left and right margins, in that order. Likewise, if only one of `smgtp` and `smgbp` is set, then it requires two parameters that give the top and bottom margins, in that order, counting from the top of the page. Thus when coding a `terminfo` entry for a printer that requires setting both left and right or top and bottom margins simultaneously, only one of `smglp` and `smgrp`, or `smgtp` and `smgbp`, should be defined; the other capability of the pair should not be included in the entry. When writing an application that uses these string capabilities, each pair should first be checked to see if both members of the pair are defined or if only one is defined; the defined capabilities should then be instantiated accordingly.

In counting lines or columns, line zero is the top line and column zero is the leftmost column. A zero value for the second argument with `smgbp` means the bottom line of the page.

All margins can be cleared with `mgc`.

D. Shadows, Italics, Wide Characters, Superscripts, Subscripts

Five new sets of string capabilities are used to describe the methods printers have of enhancing printed text.

String Capabilities for Specifying
Enhanced Printing

sshm	Enter shadow-printing mode
rshm	Exit shadow-printing mode

sitm	Enter italicizing mode
ritm	Exit italicizing mode
swidm	Enter wide character mode
rwidm	Exit wide character mode
ssupm	Enter superscript mode
rsupm	Exit superscript mode
supcs	List of characters available as superscripts
ssubm	Enter subscript mode
rsubm	Exit subscript mode
subcs	List of characters available as subscripts

If a printer requires the `sshm` control sequence before every character to be shadow-printed, the `rshm` string should be left undefined. Thus programs that find a control sequence in `sshm` but none in `rshm` should use the `sshm` control sequence before every character to be shadow-printed; otherwise, the `sshm` control sequence should be used once before the set of characters to be shadow-printed, followed by `rshm`. The same is also true of each of the `sitm/ritm`, `swidm/rwidm`, `ssupm/rsupm`, and `ssubm/rsubm` pairs.

Note that `terminfo` also has a capability for printing emboldened text (**bold**). While shadow printing and emboldened printing are similar in that they “darken” the text, many printers produce these two types of print in slightly different ways. Generally, emboldened printing is done by overstriking the same character one or more times. Shadow printing likewise usually involves overstriking, but with a slight movement up and/or to the side so that the character is “fatter.”

`Terminfo` requires that enhanced printing modes be independent, so that it would be possible, for instance, to shadow print italicized subscripts.

As mentioned earlier, the amount of motion automatically made after printing a wide character should be given in the numeric capability `widcs`.

If only a subset of the printable ASCII characters can be printed as superscripts or subscripts, they should be listed in the `supcs` or `subcs` strings, respectively. If the `ssupm` (or `ssubm`) string contains control sequences, but the corresponding `supcs` (or `subcs`) string is undefined, a program can assume that all printable ASCII characters are available as superscripts (or subscripts).

Automatic motion made after printing a superscript or subscript must be the same as for regular characters. Thus, for example, printing any of the following two-character sequences will result in equivalent motion: `Bi B1 B1`

Note that the existing `msgr` boolean capability describes whether motion control sequences can be used while in “standout mode.” This capability has been extended to cover the enhanced printing modes added here. `msgr` should be set for those printers that accept any motion control sequences without affecting shadow, italicized, widened, superscript, or subscript printing. Conversely, if `msgr` is not set, a program should exit these modes before attempting any motion.

E. Alternate Character Sets

In addition to allowing you to define line graphics (described in the “Line Graphics” section), `terminfo` lets you define alternate character sets. The following capabilities cover printers and terminals with multiple selectable or definable character sets.

String and Boolean Capabilities for Specifying
Alternate Character Sets

scs	Select character set <i>N</i>
scsd	Start definition of character set <i>N</i> , <i>M</i> characters
defc	Define character <i>A</i> , <i>B</i> dots wide, descender <i>D</i>
rcsd	End definition of character set <i>N</i>
csnm	List of character set names
daisy	If set, printer has manually changed print wheels

The *scs*, *rcsd*, and *csnm* strings each require a single parameter, *N*, a number from 0 to 63 that identifies the character set. The *scsd* string also requires the parameter *N* and another, *M*, that gives the number of characters in the set. The *defc* string requires three parameters: *A* gives the ASCII code representation for the character, *B* gives the width of the character in dots, and *D* is zero or one depending on whether the character is a “descender” or not. The *defc* string is also followed by a string of “image data” bytes that describe how the character looks (see below).

Character set 0 is the default character set present after the printer has been initialized. Not every printer has 64 character sets, of course; using *scs* with an argument that doesn’t select an available character set should cause a null result from *tparm()*.

If a character set has to be defined before it can be used, the *scsd* control sequence must be used before defining the character set, and *rcsd* must be used after. They should also cause a null result from *tparm()* when used with an argument *N* that doesn’t apply. If a character set still has to be selected after being defined, the *scs* control sequence must follow the *rcsd* control sequence. By examining the results of using each of the *scs*, *scsd*, and *rcsd* strings with a character set number in a call to *tparm()*, a program can determine which of the three are needed.

Between use of the *scsd* and *rcsd* strings, the *defc* string should be used to define each character. To print any character on printers covered by *terminfo*, the ASCII code is sent to the printer. This is true for characters in an alternate set as well as “normal” characters. Thus the definition of a character includes the ASCII code that represents it. In addition, the width of the character in dots is given, along with an indication of whether the character should descend below the print line (such as the lower case letter *g* in most character sets). The width of the character in dots also indicates the number of image data bytes that will follow the *defc* string. These image data bytes indicate where in a dot-matrix pattern ink should be applied to “draw” the character; the number of these bytes and their form are defined below in the “Dot-Matrix Graphics” section.

It’s easiest for the creator of *terminfo* entries to refer to each character set by number; however, these numbers will be meaningless to the application developer. The *csnm* string alleviates this problem by providing names for each number.

When used with a character set number in a call to *tparm()*, the *csnm* string will produce the equivalent name. These names should be used as a reference only. No naming convention is specified, although anyone who creates a *terminfo* entry for a printer should use names consistent with the names found in user documents for the printer. Application developers should allow a user to specify a character set by number (leaving it up to the user to examine the *csnm* string to determine the correct number), or by name, where the application examines the *csnm* string to determine the corresponding character set number.

The boolean `daisy` indicates printers that have manually changed print wheels or font cartridges. However, the capabilities described above are likely to be used only with dot-matrix printers.

F. Dot-Matrix Graphics

Dot-matrix printers typically have the capability of reproducing “raster graphics” images. Three new numeric capabilities and three new string capabilities help a program draw raster graphics images independent of the type of dot-matrix printer or the number of pins or dots the printer can handle at one time.

Numeric and String Capabilities for Specifying Dot-Matrix Graphics

npins	Number of pins, N , in print head
spinv	Spacing of pins vertically in pins per inch
spinh	Spacing of dots horizontally in dots per inch
porder	Matches software bits to print head pins
sbim	Start printing bit image graphics, B bits wide
rbim	End printing bit image graphics

The `sbim` string requires a single parameter, B , the width of the image in dots.

The model of dot-matrix or raster graphics that `terminfo` presents is similar to the technique used for most dot-matrix printers: Each pass of the printer’s print head is assumed to produce a dot-matrix that is N dots high and B dots wide. This is typically a wide, squat, rectangle of dots. The height of this rectangle in dots will vary from one printer to the next; this is given in the `npins` numeric capability. The size of the rectangle in fractions of an inch will also vary; it can be deduced from the `spinv` and `spinh` numeric capabilities. With these three values an application can divide a complete raster graphics image into several horizontal strips, perhaps interpolating to account for different dot spacing vertically and horizontally.

The `sbim` and `rbim` strings start and end a dot-matrix image, respectively. The `sbim` string requires a single parameter that gives the width of the dot-matrix in dots. A sequence of “image data” bytes is sent to the printer after the `sbim` string and before the `rbim` string. The number of bytes is an integral multiple of the width of the dot-matrix; the multiple and the form of each byte are determined by the `porder` string as described below.

The `porder` string is a comma-separated list of pin numbers optionally followed by a numerical offset. The offset, if given, is separated from the list with a semicolon. The position of each pin number in the list corresponds to a bit in an eight-bit data byte. The pins are numbered consecutively from 1 to `npins`, with 1 being the top pin. Note that the term “pin” is used loosely here; “ink-jet” dot-matrix printers don’t have pins, but can be considered to have an equivalent method of applying a single dot of ink to paper. The bit positions in `porder` are in groups of eight; the first position of each group is the most significant bit and the last position is the least significant bit. An application produces eight-bit bytes in the order of the groups in `porder`.

An application computes the “image data” bytes from its internal image, mapping vertical dot positions in each print head pass into eight-bit bytes, using a 1 bit where ink should be applied and 0 where no ink should be applied. This can be reversed (0 bit for ink, 1 bit for no ink) by giving a negative pin number in `porder`. If a position is skipped in `porder`, a 0 bit is used. If a position has a lower case ‘x’ instead of a pin number, a 1 bit is used in the skipped position. For consistency, a lower case ‘o’ can be used to represent a 0 filled (no-ink) bit. There must be a multiple of 8 bit

Applications that use these values should recognize the variability in print rate. Straight text, in short lines, with no embedded control sequences will probably print at close to the advertised print rate and probably faster than the rate in `cps`. Graphics data with a lot of control sequences, or very long lines of text, will print at well below the advertised rate and below the rate in `cps`. If the application is using `cps` to decide how long it should take a printer to print a block of text, the application should pad the estimate. If the application is using `cps` to decide how much text has already been printed, it should shrink the estimate. The application will thus err in favor of the user, who wants, above all, to see all the output in its correct place.

TERMINFO/TERMCAP CORRESPONDENCE

The table below presents the correspondence between `terminfo` and `termcap(5)` codes. The first two columns correspond to the first two columns in the previously presented table of `terminfo` capabilities. The last column shows the `Termcap Code`, which is the two-letter code that corresponds to the `termcap(5)` capability. The table is sorted alphabetically by `Capname`.

Variable	Cap- name	Termcap Code
<code>acs_chars</code>	<code>acsc</code>	<code>ac</code>
<code>auto_right_margin</code>	<code>am</code>	<code>am</code>
<code>back_color_erase</code>	<code>bce</code>	<code>be</code>
<code>bell</code>	<code>bel</code>	<code>bl</code>
<code>enter_blink_mode</code>	<code>blink</code>	<code>mb</code>
<code>enter_bold_mode</code>	<code>bold</code>	<code>md</code>
<code>buffer_capacity</code>	<code>bufsz</code>	<code>Ya</code>
<code>auto_left_margin</code>	<code>bw</code>	<code>bw</code>
<code>back_tab</code>	<code>cbt</code>	<code>bt</code>
<code>can_change</code>	<code>ccc</code>	<code>cc</code>
<code>change_res_horz</code>	<code>chr</code>	<code>ZC</code>
<code>hard_cursor</code>	<code>chts</code>	<code>HC</code>
<code>cursor_invisible</code>	<code>civis</code>	<code>vi</code>
<code>clear_screen</code>	<code>clear</code>	<code>cl</code>
<code>command_character</code>	<code>cmdch</code>	<code>CC</code>
<code>cursor_normal</code>	<code>cnorm</code>	<code>ve</code>
<code>max_colors</code>	<code>colors</code>	<code>Co</code>
<code>columns</code>	<code>cols</code>	<code>co</code>
<code>change_char_pitch</code>	<code>cpi</code>	<code>ZA</code>
<code>cpi_changes_res</code>	<code>cpix</code>	<code>YF</code>
<code>print_rate</code>	<code>cps</code>	<code>Ym</code>
<code>carriage_return</code>	<code>cr</code>	<code>cr</code>
<code>cr_cancels_micro_mode</code>	<code>crxm</code>	<code>YB</code>
<code>char_set_names</code>	<code>csnm</code>	<code>Zy</code>
<code>change_scroll_region</code>	<code>csr</code>	<code>cs</code>
<code>parm_left_cursor</code>	<code>cub</code>	<code>LE</code>
<code>cursor_left</code>	<code>cub1</code>	<code>le</code>
<code>parm_down_cursor</code>	<code>cud</code>	<code>DO</code>
<code>cursor_down</code>	<code>cud1</code>	<code>do</code>
<code>parm_right_cursor</code>	<code>cuf</code>	<code>RI</code>

cursor_right	cuf1	nd
cursor_address	cup	cm
parm_up_cursor	cuu	UP
cursor_up	cuu1	up
change_res_vert	cvr	ZD
cursor_visible	cvvis	vs
memory_above	da	da
has_print_wheel	daisy	YC
memory_below	db	db
parm_dch	dch	DC
delete_character	dch1	dc
define_char	defc	ZE
enter_dim_mode	dim	mh
parm_delete_line	dl	DL
delete_line	dl1	dl
these_cause_cr	docr	Zw
dis_status_line	dsl	ds
erase_chars	ech	ec
clr_eos	ed	cd
clr_eol	el	ce
clr_bol	el1	cb
ena_acs	enacs	eA
erase_overstrike	eo	eo
status_line_esc_ok	eslok	es
form_feed	ff	ff
flash_screen	flash	vb
from_status_line	fsl	fs
generic_type	gn	gn
hard_copy	hc	hc
down_half_line	hd	hd
hue_lightness_saturation	hls	hl
cursor_home	home	ho
column_address	hpa	ch
has_status_line	hs	hs
tab	ht	ta
set_tab	hts	st
up_half_line	hu	hu
tilde_glitch	hz	hz
parm_ich	ich	IC
insert_character	ich1	ic
init_file	if	if
parm_insert_line	il	AL
insert_line	il1	al
insert_null_glitch	in	in
scroll_forward	ind	sf
parm_index	indn	SF
initialize_color	initc	Ic
initialize_pair	initp	Ip
enter_secure_mode	invis	mk

insert_padding	ip	ip
init_prog	ipro	iP
init_1string	is1	i1
init_2string	is2	is
init_3string	is3	i3
init_tabs	it	it
key_sbeg	kBEG	&9
key_scancel	kCAN	&0
key_scommand	kCMD	*1
key_scopy	kCPY	*2
key_screate	kCRT	*3
key_sdc	kDC	*4
key_sdl	kDL	*5
key_send	kEND	*7
key_seol	KEOL	*8
key_sexit	kEXT	*9
key_sfind	kFND	*0
key_shelp	kHLP	#1
key_shome	kHOM	#2
key_sic	kIC	#3
key_sleft	kLFT	#4
key_smove	kMOV	%b
key_smessage	kMSG	%a
key_snext	kNXT	%c
key_soptions	kOPT	%d
key_sprint	kPRT	%f
key_sprevious	kPRV	%e
key_sredo	kRDO	%g
key_sresume	kRES	%j
key_sright	kRIT	%i
key_sreplace	kRPL	%h
key_ssave	kSAV	!1
key_ssuspend	kSPD	!2
key_sundo	kUND	!3
key_a1	ka1	K1
key_a3	ka3	K3
key_b2	kb2	K2
key_beg	kbeg	@1
key_backspace	kbs	kb
key_c1	kc1	K4
key_c3	kc3	K5
key_cancel	kcan	@2
key_btab	kcbt	kB
key_close	kclo	@3
key_clear	kclr	kC
key_command	kcmd	@4
key_copy	kcpy	@5
key_create	kcrt	@6
key_ctab	kctab	kt

key_left	kcub1	kl
key_down	kcud1	kd
key_right	kcuf1	kr
key_up	kcuu1	ku
key_dc	kdch1	kD
key_dl	kdl1	kL
key_eos	ked	kS
key_eol	kel	kE
key_end	kend	@7
key_enter	kent	@8
key_exit	kext	@9
key_f0	kf0	k0
key_f1	kf1	k1
key_f10	kf10	k;
key_f11	kf11	F1
key_f12	kf12	F2
key_f13	kf13	F3
key_f14	kf14	F4
key_f15	kf15	F5
key_f16	kf16	F6
key_f17	kf17	F7
key_f18	kf18	F8
key_f19	kf19	F9
key_f2	kf2	k2
key_f20	kf20	FA
key_f21	kf21	FB
key_f22	kf22	FC
key_f23	kf23	FD
key_f24	kf24	FE
key_f25	kf25	FF
key_f26	kf26	FG
key_f27	kf27	FH
key_f28	kf28	FI
key_f29	kf29	FJ
key_f3	kf3	k3
key_f30	kf30	FK
key_f31	kf31	FL
key_f32	kf32	FM
key_f33	kf33	FN
key_f34	kf34	FO
key_f35	kf35	FP
key_f36	kf36	FQ
key_f37	kf37	FR
key_f38	kf38	FS
key_f39	kf39	FT
key_f4	kf4	k4
key_f40	kf40	FU
key_f41	kf41	FV
key_f42	kf42	FW

key_f43	kf43	FX
key_f44	kf44	FY
key_f45	kf45	FZ
key_f46	kf46	Fa
key_f47	kf47	Fb
key_f48	kf48	Fc
key_f49	kf49	Fd
key_f5	kf5	k5
key_f50	kf50	Fe
key_f51	kf51	Ff
key_f52	kf52	Fg
key_f53	kf53	Fh
key_f54	kf54	Fi
key_f55	kf55	Fj
key_f56	kf56	Fk
key_f57	kf57	Fl
key_f58	kf58	Fm
key_f59	kf59	Fn
key_f6	kf6	k6
key_f60	kf60	Fo
key_f61	kf61	Fp
key_f62	kf62	Fq
key_f63	kf63	Fr
key_f7	kf7	k7
key_f8	kf8	k8
key_f9	kf9	k9
key_find	kfnd	@0
key_help	khlp	%1
key_home	khome	kh
key_stab	khts	kT
key_ic	kich1	kI
key_il	kil1	kA
key_sf	kind	kF
key_ll	kll	kH
has_meta_key	km	km
key_move	kmov	%4
key_mark	kmrk	%2
key_message	kmsg	%3
key_npage	knp	kN
key_next	knxt	%5
key_open	kopn	%6
key_options	kopt	%7
key_ppage	kpp	kP
key_print	kppt	%9
key_previous	kprv	%8
key_redo	krdo	%0
key_reference	kref	&1
key_resume	kres	&5
key_refresh	krfr	&2

key_sr	kri	kR
key_eic	krmir	kM
key_replace	krpl	&3
key_restart	krst	&4
key_save	ksav	&6
key_select	kslt	*6
key_suspend	kspd	&7
key_catab	ktbc	ka
key_undo	kund	&8
lab_f0	lf0	l0
lab_f1	lf1	l1
lab_f10	lf10	la
lab_f2	lf2	l2
lab_f3	lf3	l3
lab_f4	lf4	l4
lab_f5	lf5	l5
lab_f6	lf6	l6
lab_f7	lf7	l7
lab_f8	lf8	l8
lab_f9	lf9	l9
label_height	lh	lh
lines	lines	li
cursor_to_ll	ll	ll
lines_of_memory	lm	lm
change_line_pitch	lpi	ZB
lpi_changes_res	lpix	YG
label_width	lw	lw
max_micro_address	maddr	Yd
print_screen	mc0	ps
prtr_off	mc4	pf
prtr_on	mc5	po
prtr_silent	mc5i	5i
prtr_non	mc5p	pO
micro_col_size	mcs	Yf
parm_left_micro	mcub	Zg
micro_left	mcub1	Za
parm_down_micro	mcud	Zf
micro_down	mcud1	ZZ
parm_right_micro	mcuf	Zh
micro_right	mcuf1	Zb
parm_up_micro	mcuu	Zi
micro_up	mcuu1	Zd
clear_margins	mgc	MC
micro_column_address	mhpa	ZY
move_insert_mode	mir	mi
max_micro_jump	mjump	Ye
micro_line_size	mls	Yg
cursor_mem_address	mrcup	CM
move_standout_mode	msgr	ms

micro_row_address	mvp	Zc
no_color_video	ncv	NC
newline	nel	nw
num_labels	nlab	NI
no_pad_char	npc	NP
number_of_pins	npins	Yh
non_rev_rmcup	nrrmc	NR
needs_xon_xoff	nxon	nx
orig_colors	oc	oc
orig_pair	op	op
output_res_char	orc	Yi
output_res_horz_inch	orhi	Yk
output_res_line	orl	Yj
output_res_vert_inch	orvi	Yl
over_strike	os	os
pad_char	pad	pc
max_pairs	pairs	pa
padding_baud_rate	pb	pb
pkey_key	pfkey	pk
pkey_local	pfloc	pl
pkey_xmit	pfx	px
plab_norm	pln	pn
order_of_pins	porder	Ze
enter_protected_mode	prot	mp
stop_bit_image	rbim	Zs
restore_cursor	rc	rc
stop_char_set_def	rcsd	Zt
repeat_char	rep	rp
enter_reverse_mode	rev	mr
reset_file	rf	rf
req_for_input	rfi	RF
scroll_reverse	ri	sr
parm_rindex	rin	SR
exit_italics_mode	ritm	ZR
exit_leftward_mode	rlm	ZS
exit_alt_charset_mode	rmacs	ae
exit_am_mode	rmam	RA
exit_ca_mode	rmcup	te
exit_delete_mode	rmdc	ed
exit_micro_mode	rmicm	ZT
exit_insert_mode	rmir	ei
keypad_local	rmkx	ke
label_off	rmln	LF
meta_off	rmm	mo
char_padding	rmp	rP
exit_standout_mode	rms	se
exit_underline_mode	rmul	ue
exit_xon_mode	rmxon	RX
reset_lstring	rs1	r1

reset_2string	rs2	r2
reset_3string	rs3	r3
exit_shadow_mode	rshm	ZU
exit_subscript_mode	rsubm	ZV
exit_superscript_mode	rsupm	ZW
exit_upward_mode	rum	ZX
exit_doublewide_mode	rwidm	ZQ
semi_auto_right_margin	sam	YE
start_bit_image	sbim	Zq
save_cursor	sc	sc
set_color_pair	scp	sp
select_char_set	scs	Zj
start_char_set_def	scsd	Zr
enter_draft_quality	sdrfq	ZG
set_background	setb	Sb
set_foreground	setf	Sf
set_attributes	sgr	sa
exit_attribute_mode	sgr0	me
enter_italics_mode	sitm	ZH
enter_leftward_mode	slm	ZI
enter_alt_charset_mode	smacs	as
enter_am_mode	smam	SA
enter_ca_mode	smcup	ti
enter_delete_mode	smdc	dm
set_bottom_margin	smgb	Zk
set_bottom_margin_parm	smgbp	Zl
set_left_margin	smgl	ML
set_left_margin_parm	smglp	Zm
set_right_margin	smgr	MR
set_right_margin_parm	smgrp	Zn
set_top_margin	smgt	Zo
set_top_margin_parm	smgtp	Zp
enter_micro_mode	smicm	ZJ
enter_insert_mode	smir	im
keypad_xmit	smkx	ks
label_on	smln	LO
meta_on	smm	mm
enter_standout_mode	smso	so
enter_underline_mode	smul	us
enter_xon_mode	smxon	SX
enter_near_letter_quality	snlq	ZK
enter_normal_quality	snrmq	ZL
dot_horz_spacing	spinh	Yc
dot_vert_spacing	spinv	Yb
enter_shadow_mode	sshm	ZM
enter_subscript_mode	ssubm	ZN
enter_superscript_mode	ssupm	ZO
subscript_characters	subcs	Zu
enter_upward_mode	sum	ZP

superscript_characters	supcs	Zv
enter_doublewide_mode	swidm	ZF
clear_all_tabs	tbc	ct
to_status_line	tsl	ts
underline_char	uc	uc
transparent_underline	ul	ul
row_address	vpa	cv
virtual_terminal	vt	vt
wide_char_size	widcs	Yn
set_window	wind	wi
width_status_line	wsl	ws
eat_newline_glitch	xenl	xn
ceol_standout_glitch	xhp	xs
col_addr_glitch	xhpa	YA
magic_cookie_glitch	xmc	sg
xoff_character	xoffc	XF
xon_xoff	xon	xo
xon_character	xonc	XN
no_esc_ctlc	xsb	xb
dest_tabs_magic_sms0	xt	xt
row_addr_glitch	xvpa	YD
zero_motion	zerom	Zx

FILES

`/usr/lib/terminfo/?/*`
 compiled device description database

`/usr/src/lib/libcurses/terminfo/*.ti`
 source device descriptions

`/usr/lib/tabset/*`
 tab settings for some devices, in a format appropriate to be output to the device (escape sequences that set margins and tabs)

SEE ALSO

`curses(3X)`, `printf(3S)`, `term(5)`, `profile(4)`, `termcap(5)`,
`captainfo(1M)`, `infocmp(1M)`, `tic(1M)`, `termio(7)`, `ttcompat(7)` in the *System Manager's Reference for the DG/UX System*.
`tput(1)` in the *User's Reference for the DG/UX System*.

CAUTIONS

As described in the "Tabs and Initialization" section above, a device's initialization strings, `is1`, `is2`, and `is3`, if defined, must be output before a `curses(3X)` program is run. An available mechanism for outputting such strings is `tput init` (see `tput(1)` and `profile(4)`).

If a null character (`\0`) is encountered in a string, the null and all characters after it are lost. Therefore it is not possible to code a null character (`\0`) in a string capability and send it to a device (either a terminal or a printer). The suggestion of sending `\0200` where `\0` (null) is needed can succeed only if the device ignores the eighth bit. For example, because all eight bits are used in the standard international ISO character set, devices that adhere to this standard will treat `\0200` differently from `\0`.

Tampering with entries in `/usr/lib/terminfo/?/*` (for example, changing or removing an entry) can affect programs such as `vi(1)` that expect the entry to be present and correct. In particular, removing the description for the dumb terminal causes unexpected problems.

NAME

timezone - set default system time zone and locale

SYNOPSIS

/etc/TIMEZONE, /etc/TIMEZONE.csh

DESCRIPTION

The files /etc/TIMEZONE and /etc/TIMEZONE.csh set and export the following environment variables:

TZ time zone
 NLSPATH search path for message catalogs
 LANG local language

These files are included into other shell scripts (for example, /etc/profile and /etc/cshrc) to establish this localization information. /etc/TIMEZONE is also read by /etc/init to initialize the timezone and locale information for the system startup procedures.

To change the values of these environment variables, you may edit these files directly, or use `admdate(1M)` and `admnls(1M)`, which can be invoked from `sysadm(1M)`.

If /etc/TIMEZONE is missing, it is created at system startup by copying the file /etc/TIMEZONE.proto. If /etc/TIMEZONE.csh is missing, it is created at system startup by copying the file /etc/TIMEZONE.csh.proto.

NLSPATH and LANG are described in `environ(5)` and `setlocale(3C)`. The default value of NLSPATH (in the proto files) is "/usr/lib/nls/msg/%L/%N". The default value of LANG is "C".

TZ can be either the name of a timezone database file found under the directory /usr/lib/locale/TZ, preceded by a colon (e.g. ":US/Eastern"), or else a string that describes the timezone rules. The syntax of such a rule string can be described as follows:

```
TZ                                 →                 zone
                                                      | zone signed_time
                                                      | zone signed_time zone
                                                      | zone signed_time zone dst
zone                                 →                 letter letter letter
signed_time                         →                 sign time
                                                      | time
time                                 →                 hour
                                                      | hour : minute
                                                      | hour : minute : second
dst                                 →                 signed_time
                                                      | signed_time , dst_date , dst_date
                                                      | , dst_date , dst_date
dst_date                             →                 julian
                                                      | julian / time
letter                               →                 a | A | b | B | ... | z | Z
hour                                 →                 00 | 01 | ... | 23
minute                               →                 00 | 01 | ... | 59
second                               →                 00 | 01 | ... | 59
julian                               →                 001 | 002 | ... | 366
sign                                 →                 - | +
```

EXAMPLES

The contents of the file `/etc/TIMEZONE` could be

```
# Time Zone
TZ=:US/Eastern
export TZ
# Message catalog search path
NLSPATH=/usr/lib/nls/msg/%L/%N
export NLSPATH
# Language
LANG=C
export C
```

A simple setting for `TZ` for New Jersey could be

```
TZ=EST5EDT
```

where `EST` is the abbreviation for the main time zone, `5` is the difference, in hours, between GMT (Greenwich Mean Time) and the main time zone, and `EDT` is the abbreviation for the alternate time zone.

The most complex representation of the same setting, for the year 1986, is

```
TZ="EST5:00:00EDT4:00:00,117/2:00:00,299/2:00:00"
```

where `EST` is the abbreviation for the main time zone, `5:00:00` is the difference, in hours, minutes, and seconds between GMT and the main time zone, `EDT` is the abbreviation for the alternate time zone, `4:00:00` is the difference, in hours, minutes, and seconds between GMT and the alternate time zone, `117` is the number of the day of the year (Julian day) when the alternate time zone will take effect, `2:00:00` is the number of hours, minutes, and seconds past midnight when the alternate time zone will take effect, `299` is the number of the day of the year when the alternate time zone will end, and `2:00:00` is the number of hours, minutes, and seconds past midnight when the alternate time zone will end.

A southern hemisphere setting such as the Cook Islands could be

```
TZ="KDT9:30KST10:00,64/5:00,303/20:00"
```

This setting means that `KDT` is the abbreviation for the main time zone, `KST` is the abbreviation for the alternate time zone, `KST` is 9 hours and 30 minutes later than GMT, `KDT` is 10 hours later than GMT, the starting date of `KDT` is the 64th day at 5 AM, and the ending date of `KDT` is the 303rd day at 8 PM.

Starting and ending times are relative to the alternate time zone. If the alternate time zone start and end dates and the time are not provided, the days for the United States that year will be used and the time will be 2 AM. If the start and end dates are provided but the time is not provided, the time will be midnight.

Note that in most installations, `TZ` is set to the correct value by default when the user logs on, via the local `/etc/profile` file (see `profile(4)`).

NOTES

When the longer format is used, the `TZ` variable must be surrounded by double quotes as shown.

The system administrator must change the Julian start and end days annually if the longer form of the `TZ` variable is used.

Setting the time during the interval of change from the main time zone to the alternate time zone or vice versa can produce unpredictable results.

SEE ALSO

zic(1M), ctime(3C), setlocale(3C), profile(4), environ(5).

NAME

updaters - configuration file for NIS updating

SYNOPSIS

/etc/yp/updaters

DESCRIPTION

The file `/etc/yp/updaters` is a makefile (see `make(1)`) which is used for updating NIS databases. Databases can only be updated in a secure network, that is, one that has a `publickey(4)` database. Each entry in the file is a make target for a particular NIS database. For example, if there is an NIS database named `passwd.byname` that can be updated, there should be a make target named `passwd.byname` in the `updaters` file with the command to update the file.

The information necessary to make the update is passed to the update command through standard input. The information passed is described below (all items are followed by a NEWLINE, except for bullets four and six, below):

- Network name of client wishing to make the update (a string)
- Kind of update (an integer)
- Number of bytes in key (an integer)
- Actual bytes of key
- Number of bytes in data (an integer)
- Actual bytes of data

After getting this information through standard input, the command to update the particular database should decide whether the user is allowed to make the change. If not, it should exit with the status `YPERR_ACCESS`. If the user is allowed to make the change, the command should make the change and exit with a status of zero. If there are any errors that may prevent the updater from making the change, it should exit with the status that matches a valid NIS error code described in `<rpcsvc/ypclnt.h>`.

FILES

/etc/yp/updaters

SEE ALSO

`make(1)`, `ypupdated(1M)`, `ypupdate(3R)`.

NAME

utmp, wtmp - utmp and wtmp entry formats

SYNOPSIS

```
#include <sys/types.h>
#include <limits.h>
#include <utmp.h>
```

DESCRIPTION

These files, which hold user and accounting information for such commands as who(1), write(1), and login(1), have the following structure as defined by <utmp.h>:

```
#define UTMP_FILE "/etc/utmp"
#define WTMP_FILE "/etc/wtmp"
#define ut_name   ut_user

struct utmp {
    char    ut_user[USR_NAME]; /* User login name */
    char    ut_id[4];         /* /etc/inittab id (usually line #) */
    char    ut_line[12];     /* device name (console, lnxx) */
    short   ut_pid;          /* process id */
    short   ut_type;         /* type of entry */
    struct  exit_status {
        short e_termination; /* Process termination status */
        short e_exit;        /* Process exit status */
    } ut_exit;              /* The exit status of a process
                             * marked as DEAD_PROCESS. */
    time_t  ut_time;        /* time entry was made */
    char    ut_host[16];    /* hostname, if remote */
};

/* Definitions for ut_type */
#define EMPTY          0
#define RUN_LVL        1
#define BOOT_TIME      2
#define OLD_TIME       3
#define NEW_TIME       4
#define INIT_PROCESS   5 /* Process spawned by "init" */
#define LOGIN_PROCESS  6 /* A "getty" process waiting for login */
#define USER_PROCESS   7 /* A user process */
#define DEAD_PROCESS   8
#define ACCOUNTING     9
#define UTMAXTYPE      ACCOUNTING /* Largest legal value of ut_type */

/* Special strings or formats used in the "ut_line" field when */
/* accounting for something other than a process */
/* No string for the ut_line field can be more than 11 chars + */
/* a NULL in length */

#define RUNLVL_MSG     "run-level %c"
#define BOOT_MSG       "system boot"
#define OTIME_MSG      "old time"
#define NTIME_MSG      "new time"
```

FILES

/usr/include/utmp.h
/etc/utmp
/etc/wtmp

SEE ALSO

login(1), who(1), write(1), getut(3C), limits(4).

NAME

ypfiles – the Network Information Service database and directory structure

DESCRIPTION

The Network Information Service (NIS) network lookup service uses a distributed, replicated database of dbm files contained in the `/etc/yp` directory hierarchy on each NIS server. A dbm database consists of two files, created by calls to the `ndbm(3C)` library package. One has the filename extension `.pag` and the other has the filename extension `.dir`. For instance, the database named `hosts.byname`, is implemented by the pair of files `hosts.byname.pag` and `hosts.byname.dir`.

A dbm database served by the NIS is called an NIS *map*. An NIS *domain* is a subdirectory of `/etc/yp` containing a set of NIS maps. Any number of NIS domains can exist. Each may contain any number of maps.

No maps are required by the NIS lookup service itself, although they may be required for the normal operation of other parts of the system. There is no list of maps which NIS serves — if the map exists in a given domain, and a client asks about it, the NIS will serve it. For a map to be accessible consistently, it must exist on all NIS servers that serve the domain. To provide data consistency between the replicated maps, an entry to run `ypxfr` periodically should be made in the super-user's `crontab` file on each server. More information on this topic is in `ypxfr(1M)`.

NIS maps should contain two distinguished key-value pairs. The first is the key `YP_LAST_MODIFIED`, having as a value a ten-character ASCII order number. The order number should be the system time in seconds when the map was built. The second key is `YP_MASTER_NAME`, with the name of the NIS master server as a value. `makedbm(1M)` generates both key-value pairs automatically. A map that does not contain both key-value pairs can be served by the YP, but the `ypserv` process will not be able to return values for “Get order number” or “Get master name” requests. See `ypserv(1M)`. In addition, values of these two keys are used by `ypxfr` when it transfers a map from a master NIS server to a slave. If `ypxfr` cannot figure out where to get the map, or if it is unable to determine whether the local copy is more recent than the copy at the master, you must set extra command line switches when you run it.

NIS maps must be generated and modified only at the master server. They are copied to the slaves using `ypxfr(1M)` to avoid potential byte-ordering problems among NIS servers running on machines with different architectures, and to minimize the amount of disk space required for the dbm files. The NIS database can be initially set up for both masters and slaves by using `ypinit(1M)`.

After the server databases are set up, it is probable that the contents of some maps will change. In general, some ASCII source version of the database exists on the master, and it is changed with a standard text editor. The update is incorporated into the NIS map and is propagated from the master to the slaves by running `/etc/yp/Makefile`. All maps supplied with this OS have entries in `/etc/yp/Makefile`; if you add an NIS map, edit this file to support the new map. The makefile uses `makedbm(1M)` to generate the NIS map on the master, and `yppush(1M)` to propagate the changed map to the slaves. `yppush` is a client of the map `ypservers`, which lists all the NIS servers. For more information on this topic, see `yppush(1M)`.

FILES

`/etc/yp`
`/etc/yp/Makefile`

SEE ALSO

makedbm(1M), rpcinfo(1M), ypinit(1M), ypmake(1M), yppoll(1M),
yppush(1M), ypserv(1M), ypxfr(1M), dbm(3X).

End of Chapter

Chapter 5

Miscellaneous Features

This chapter contains in printed form the online manual entries for miscellaneous features. The entries are in alphabetical order except for **intro(5)**, which is first.

This chapter contains only DG/UX manual pages except for **hostname(5)**, which is a TCP/IP man page.

NAME

intro – introduction to miscellany

DESCRIPTION

This section describes miscellaneous facilities, such as macro packages and character set tables.

NAME

ascii - map of ASCII character set

DESCRIPTION

ascii is a map of the ASCII character set, giving both octal and hexadecimal equivalents of each character, to be printed as needed. It contains:

000 nul	001 soh	002 stx	003 etx	004 eot	005 enq	006 ack	007 bel
010 bs	011 ht	012 nl	013 vt	014 np	015 cr	016 so	017 si
020 dle	021 dc1	022 dc2	023 dc3	024 dc4	025 nak	026 syn	027 etb
030 can	031 em	032 sub	033 esc	034 fs	035 gs	036 rs	037 us
040 sp	041 !	042 "	043 #	044 \$	045 %	046 &	047 ' /
050 (051)	052 *	053 +	054 ,	055 -	056 .	057 /
060 0	061 1	062 2	063 3	064 4	065 5	066 6	067 7
070 8	071 9	072 :	073 ;	074 <	075 =	076 >	077 ?
100 @	101 A	102 B	103 C	104 D	105 E	106 F	107 G
110 H	111 I	112 J	113 K	114 L	115 M	116 N	117 O
120 P	121 Q	122 R	123 S	124 T	125 U	126 V	127 W
130 X	131 Y	132 Z	133 [134 \	135]	136 ^	137 _
140 ' /	141 a	142 b	143 c	144 d	145 e	146 f	147 g
150 h	151 i	152 j	153 k	154 l	155 m	156 n	157 o
160 p	161 q	162 r	163 s	164 t	165 u	166 v	167 w
170 x	171 y	172 z	173 {	174	175 }	176 ~	177 del

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0a nl	0b vt	0c np	0d cr	0e so	0f si
10 dle	11 dc1	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1a sub	1b esc	1c fs	1d gs	1e rs	1f us
20 sp	21 !	22 "	23 #	24 \$	25 %	26 &	27 ' /
28 (29)	2a *	2b +	2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a :	3b ;	3c <	3d =	3e >	3f ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [5c \	5d]	5e ^	5f _
60 ' /	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c	7d }	7e ~	7f del

SEE ALSO

terminfo(4).

NAME

dg_mknod – data returned by the dg_mknod system call

SYNOPSIS

```
#include <sys/types.h>
```

DESCRIPTION

The system call dg_mknod takes a parameter that is a pointer to the structure defined by this include file. This structure defines the node that is created.

```
struct dg_mknod
{
    mode_t      extended_mode;
    dev_t      device_number;
    char *      symbolic_link_target;
    unsigned long  desired_data_element_blocks;
    unsigned long  data_element_blocks_limit;
    unsigned long  desired_index_element_blocks;
    unsigned long  index_element_blocks_limit;
};
```

The fields of this structure are defined as follows:

extended_mode

The file type and access permissions of the file. The file type is available by AND-ing this field with DG_FILE_TYPE_MASK. The access bits are available by AND-ing this field with (~ DG_FILE_TYPE_MASK). The file type and access are encoded using the constants defined in stat.h and dg_stat.h

device_number

The device specifier to be used if the file to be created is of type 'block-special' or 'character-special'. This field is ignored otherwise.

symbolic_target_link

A null-terminated pathname which will be the target of the file to be created if that file is of type 'symbolic link'. This field is ignored otherwise.

desired_data_element_blocks

The preferred size (in 512-byte blocks) of the data elements of the file to be created. If this size is 0, then the default data element size for the containing file system will be used.

data_element_blocks_limit

The maximum size (in 512-byte blocks) of the data elements of the file to be created. Values in the range starting at the preferred size and working towards the limit are tried until a valid data element size is found.

desired_index_element_blocks

The preferred size (in 512-byte blocks) of the index elements of the file to be created. If this size is 0, then the default data element size for the containing file system will be used.

index_element_blocks_limit

The maximum size (in 512-byte blocks) of the index elements of the file to be created. Values in the range starting at the preferred size and working towards the limit are tried until a valid data element size is found.

FILES

/usr/include/sys/dg_mknod.h
/usr/include/sys/types.h

SEE ALSO

dg_mknod(2), dg_stat(5), types(5).

NAME

dg_stat – data returned by dg_stat and dg_fstat system call

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/dg_stat.h>
```

DESCRIPTION

The system calls dg_stat, and dg_fstat return data whose structure is defined by this include file.

```
struct dg_stat
{
    dev_t      st_dev;
    ino_t      st_ino;
    mode_t     st_mode;
    nlink_t    st_nlink;
    uid_t      st_uid;
    gid_t      st_gid;
    dev_t      st_rdev;
    off_t      st_size;
    time_t     st_atime;
    unsigned long st_ausec;
    time_t     st_mtime;
    unsigned long st_musec;
    time_t     st_ctime;
    unsigned long st_cusec;
    long       st_pad1[114];
    unsigned long st_blocks;
    mode_t     extended_mode;
    unsigned long data_element_blocks;
    unsigned long index_element_blocks;
    unsigned long max_cpd_blocks;
    unsigned long max_cpd_file_nodes;
    unsigned long cur_cpd_blocks;
    unsigned long cur_cpd_file_nodes;
};
```

The fields of this structure are defined as follows:

st_dev

An identifier of the flat file store containing the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_ino

An identifier of the per-file database within the flat file store. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_mode

The mode of the file, encoded using the constants defined in stat.h. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_nlink
The number of links to the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_uid
The user-id of the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_gid
The group-id of the file. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_rdev
The represented device, giving the major and minor device numbers of the device represented by a special file. This field is meaningful only if the file is of type 'block-special' or 'character-special'. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_size
The size of the file in bytes. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_atime
The last time the file was accessed. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_ausec
The extended-precision portion of st_atime, in microseconds. If such precision is not available, this field will be zero.

st_mtime
The last time the file's contents were modified. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_musec
The extended-precision portion of st_mtime, in microseconds. If such precision is not available, this field will be zero.

st_ctime
The last time the file's attributes were changed. The meaning of this field is the same as that of the field of the same name in the stat structure.

st_cusec
The extended-precision portion of st_ctime, in microseconds. If such precision is not available, this field will be zero.

st_pad
Reserved space.

st_blocks
The actual number of blocks allocated for the file.

extended_mode
The extended mode of the file, encoded using the constants defined below and in stat.h.

data_element_blocks
The number of 512-byte blocks used in each of the file's data elements.

index_element_blocks
The number of 512-byte blocks used in each of the file's index elements.

max_cpd_blocks

The maximum number of 512-byte blocks that can be allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero. A node is a space descendant of a CPD if it is found in the directory tree descending from the CPD and if no file system mount point boundaries are crossed.

max_cpd_file_nodes

The maximum number of file nodes that can be allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero.

cur_cpd_blocks

The current number of 512-byte blocks that have been allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero.

cur_cpd_file_nodes

The current number of file nodes that have been allocated by this file and all of its space descendants. This field has meaning only if the file is a control-point directory. Otherwise, it will be zero.

```
#define DG_FILE_TYPE_MASK    ((unsigned_long) 0xFFFFF000)
```

The bitmask used to extract the file's type from the *extended_mode* field. The result of AND-ing the file's *extended_mode* with this mask will be one of the following: DG_IFCPD, S_IFDIR, S_IFCHR, S_IFBLK, S_IFREG, S_IFLNK, S_IFIFO, S_IFSOCK. Logically, this field is equivalent to the S_IFMT mask defined in stat.h, except that DG_FILE_TYPE_MASK allows for detection of DG/UX-only extended file types, such as DG_IFCPD (see below).

```
#define DG_IFCPD            ((unsigned long) 0x00010000)
```

Control-point directory file type.

```
#define DG_IFSTREAMS       ((unsigned long) 0x00020000)
```

Streams special file type.

FILES

```
/usr/include/sys/dg_stat.h
/usr/include/sys/types.h
```

SEE ALSO

dg_stat(2), dg_fstat(2), stat(5), types(5).

NAME

elink - Environment variable sensitive file link

DESCRIPTION

An elink is the mechanism used to encode environment variable-sensitive references into symbolic links. This non-standard use of symbolic links is used by a number of software development tools such as `cc` to find files that pertain to a development environment selected with `sde-target(1)`.

The elink mechanism is incorporated into a number of software development tools to support the generation of programs and libraries that conform to different standards on the same machine. It is implemented by inserting code into the error paths of special versions of some system library routines.

An elink is a symbolic link whose value conforms to the following grammar:

```

<elink>      ::= "elink:" <sp> <pathname> <sp> <comment>
<pathname>  ::= <pathname> <evref> <pathname>
              | <pathchars>
<evref>     ::= "$" <evname>
              | "${" <evname> "}"
              | "${" <evname> ":" <default> "}"
<evname>    ::= <id>
<default>   ::= <id>
<pathchars> ::= <id>
              | <pathchars> "/" <pathchars>
<comment>   ::= "#" <text>
              |

```

<sp> is zero or more tab or space characters.

<id> is a sequence of identifier characters.

<text> is zero or more of any character except null.

This grammar is ambiguous in a number of ways that are not significant. For example, you can't tell how <evref> terminates if it is not the "\${}" form and it is followed by an <id>.

Within one of the specially modified tools, when an operation such as `open(2)` is performed, nothing is done unless an error would be reported. In that case, the pathname argument is checked to see if it or any component is a symbolic link. If one is found, then the contents of the link are checked to see if they conform to the above grammar. If so, the <pathname> component is extracted, environment variable substitution is performed, and the operation is tried again, substituting the newly created pathname for the value of the symbolic link in the original argument. The previous steps are repeated until the operation succeeds or the argument does not resolve to a valid symbolic link (and an error is reported).

Environment variable substitution is defined as the replacement of all <evref> components in the <pathname> with the appropriate environment variable value. If a given environment variable is not defined, then the <default> value is used if it is supplied; otherwise "" is used.

For example, consider the following symbolic link:

```

/usr/lib/libc.a ->
elink:/usr/sde/${TARGET_BINARY_INTERFACE:-m88kdgux}/
usr/lib/libc.a # See sde-target(1)

```

Links begin with "elink:" to give a visual cue that something is different about this symbolic link. The comment allows the insertion of other informational pointers.

This link makes reference to one environment variable although more could have been used. If the environment variable `TARGET_BINARY_INTERFACE` is not defined when a tool such as `ld(1)` attempts to open `/usr/lib/libc.a` then the tool will use the path `/usr/sde/m88kdgux/usr/lib/libc.a`. If `TARGET_BINARY_INTERFACE` is some value such as `m88kbcs`, the the path used to find `libc.a` will include the value of the variable such as `/usr/sde/m88kbcs/usr/lib/libc.a`.

It should be noted that the `elink` mechanism is incorporated only in a small set of tools. Other tools that attempt to use a pathname that contains an `elink` will get an error indicating that the file does not exist.

SEE ALSO

`sde-target(1)`, `sde(5)`.

NAME

environ – user environment

DESCRIPTION

When a process begins execution, `exec` routines make available an array of strings called the environment [see `exec(2)`]. By convention, these strings have the form *variable=value*, for example, `PATH=/sbin:/usr/sbin`. These environmental variables provide a way to make information about a program's environment available to programs. The following environmental variables can be used by applications and are expected to be set in the target run-time environment.

- HOME** The name of the user's login directory, set by `login(1)` from the password file (see `passwd(4)`).
- LANG** The string used to specify localization information that allows users to work with different national conventions. The `setlocale(3C)` function looks for the `LANG` environment variable when it is called with "" as the *locale* argument. `LANG` is used as the default locale if the corresponding environment variable for a particular category is unset.

For example, when `setlocale()` is invoked as

```
setlocale(LC_CTYPE, ""),
```

`setlocale()` will query the `LC_CTYPE` environment variable first to see if it is set and non-null. If `LC_CTYPE` is not set or null, then `setlocale()` will check the `LANG` environment variable to see if it is set and non-null. If both `LANG` and `LC_CTYPE` are unset or null, the default C locale will be used to set the `LC_CTYPE` category.

Most commands will invoke

```
setlocale(LC_ALL, "")
```

prior to any other processing. This allows the command to be used with different national conventions by setting the appropriate environment variables.

The system-wide default value for `LANG` can be changed with the `sysadm(1M)` command.

The following environment variables are supported to correspond with each category of `setlocale(3C)`:

- LC_COLLATE** This category specifies the collation sequence being used. The information corresponding to this category is stored in a database created by the `colltbl(1M)` command. This environment variable affects `strcoll(3C)`, `strxfrm(3C)` and the regular expression code (see `regexpr(3C)`).
- LC_CTYPE** This category specifies character classification, character conversion, and widths of multibyte characters. The information corresponding to this category is stored in a database created by the `chrtbl(1M)` command. The default C locale corresponds to the 7-bit ASCII character set. This environment variable is used by `ctype(3C)`, `mbchar(3C)`, and many commands; for example: `cat(1)`, `ed(1)`, `ls(1)`, and `vi(1)`.

- LC_MESSAGES** This category specifies the language of the AT&T-style message database being used. For example, an application may have one message database with French messages, and another database with German messages. Message databases are created by the `mkmsgs(1M)` command. This environment variable is used by `exstr(1)`, `gettext(1)`, `gettext(3C)`, and `srchtxt(1)`. The X/Open-style message facility does not use this variable.
- LC_MONETARY** This category specifies the monetary symbols and delimiters used for a particular locale. The information corresponding to this category is stored in a database created by the `montbl(1M)` command. This environment variable is used by `localeconv(3C)`.
- LC_NUMERIC** This category specifies the decimal and thousands delimiters. The information corresponding to this category is stored in a database created by the `chrtbl(1M)` command. The default C locale corresponds to "." as the decimal delimiter and no thousands delimiter. This environment variable is used by `localeconv(3C)`, `printf(3C)`, and `strtod(3C)`.
- LC_TIME** This category specifies date and time formats. The information corresponding to this category is stored in a database specified in `strftime(4)`. The default C locale corresponds to U.S. date and time formats. This environment variable is used by many commands and functions; for example: `at(1)`, `calendar(1)`, `date(1)`, `strftime(3C)`, and `getdate(3C)`.
- MSGVERB** Controls which standard format message components `fmtmsg` selects when messages are displayed to `stderr` [see `fmtmsg(1)` and `fmtmsg(3C)`].
- SEV_LEVEL** Define severity levels and associate and print strings with them in standard format error messages [see `addseverity(3C)`, `fmtmsg(1)`, and `fmtmsg(3C)`].
- NETPATH** A colon-separated list of network identifiers. A network identifier is a character string used by the Network Selection component of the system to provide application-specific default network search paths. A network identifier must consist of non-NULL characters and must have a length of at least 1. No maximum length is specified. Network identifiers are normally chosen by the system administrator. A network identifier is also the first field in any `/etc/netconfig` file entry. `NETPATH` thus provides a link into the `/etc/netconfig` file and the information about a network contained in that network's entry. `/etc/netconfig` is maintained by the system administrator. The library routines described in `getnetpath(3N)` access the `NETPATH` environment variable.
- NLSPATH** Contains a sequence of templates which the X/Open-style message facility uses when attempting to locate message catalogs (see `catopen(3C)`). The AT&T-style message facility does not use this variable. Each template consists of an optional prefix, one or more substitution fields, a

filename and an optional suffix.

For example:

```
NLSPATH="/usr/lib/nls/msg/%N.cat"
```

defines that `catopen()` should look for all message catalogs in the directory `/usr/lib/nls/msg`, where the catalog name should be constructed from the *name* parameter passed to `catopen()`, `%N`, with the suffix `.cat`.

Substitution fields consist of a `%` symbol, followed by a single-letter keyword. The following keywords are currently defined:

<code>%N</code>	The value of the <i>name</i> parameter passed to <code>catopen()</code> .
<code>%L</code>	The value of <code>LANG</code> .
<code>%l</code>	The language element from <code>LANG</code> .
<code>%t</code>	The territory element from <code>LANG</code> .
<code>%c</code>	The codeset element from <code>LANG</code> .
<code>%%</code>	A single <code>%</code> character.

An empty string is substituted if the specified value is not currently defined. The separators “`_`” and “`.`” are not included in `%t` and `%c` substitutions.

Templates defined in `NLSPATH` are separated by colons (`:`). A leading colon or two adjacent colons (`::`) is equivalent to specifying `%N`.

For example:

```
NLSPATH=":%N.cat:/usr/lib/nls/msg/%L/%N.cat"
```

indicates to `catopen()` that it should look for the requested message catalog in *name*, *name.cat* and `/usr/lib/nls/msg/$LANG/name.cat`.

The system-wide default value for `NLSPATH` can be changed with the `sysadm(1M)` command.

PATH	The sequence of directory prefixes that <code>sh(1)</code> , <code>time(1)</code> , <code>nice(1)</code> , <code>nohup(1)</code> , etc., apply in searching for a file known by an incomplete path name. The prefixes are separated by colons (<code>:</code>). <code>login(1)</code> sets <code>PATH=/usr/bin</code> . (For more detail, see <code>sh(1)</code> .)
TERM	The kind of terminal for which output is to be prepared. This information is used by commands, such as <code>vi(1)</code> , which may exploit special capabilities of that terminal.
CFTIME	Historically, the default format string to be used by the <code>date(1)</code> command and the <code>ascftime()</code> and <code>cftime()</code> routines (see <code>strftime(3C)</code>). If <code>CFTIME</code> is not set or is null, the default format string specified in the <code>/lib/cftime/LANGUAGE</code> file (if it exists) is used in its place (see <code>cftime(4)</code>). The use of <code>CFTIME</code> has generally been subsumed by <code>LANG</code> and <code>LC_TIME</code> .
CHRCLASS	Historically, a value that corresponds to a file in <code>/lib/chrclass</code> containing character classification and conversion information. This information was used by commands (such as <code>cat(1)</code> , <code>ed(1)</code> , and <code>sort(1)</code>) to classify characters as alphabetic, printable, upper case, and so on,

and to convert characters to upper or lower case. The use of CHRCLASS has generally been subsumed by LANGF1 and LC_CTYPE. For more detail, see ctype(3C).

LANGUAGE Historically, a language for which a printable file by that name exists in /lib/cftime. This information was used by commands (such as date(1), ls(1), and sort(1)) to print date and time information in the language specified. The use of LANGUAGE has generally been subsumed by LANG and LC_TIME.

TZ Time zone information. The contents of the environment variable named TZ are used by the functions ctime(3C), localtime() (see ctime(3C)), strftime(3C) asctime() (see strftime(3C)), cftime() (see strftime(3C)), and mktime(3C) to override the default timezone. The value of TZ has one of the two forms (spaces inserted for clarity):

:characters

or:

std offset dst offset, rule

If TZ is of the first format (i.e., if the first character is a colon), the string following the colon is the name of the timezone that will be loaded in from the /usr/lib/locale/TZ directory. For example, if TZ was set to :US/Eastern, it would load the /usr/lib/local/TZ/US/Eastern timezone definition file. The timezones under this directory are produced with the zic(1) command.

The expanded format (for all TZs whose value does not have a colon as the first character) is as follows:

std offset [dst [offset] , [start [/time] , end [/time]]]

Where:

std and *dst*

Three or more bytes that are the designation for the standard (*std*) and daylight savings time (*dst*) timezones. Only *std* is required, if *dst* is missing, then daylight savings time does not apply in this locale. Upper- and lower-case letters are allowed. Any characters except a leading colon (:), digits, a comma (,), a minus (-), a plus (+), or an ASCII NUL are allowed.

offset Indicates the value one must add to the local time to arrive at Coordinated Universal Time. The offset has the form:

hh [: mm [: ss]]

The minutes (*mm*) and seconds (*ss*) are optional. The hour (*hh*) is required and may be a single digit. The *offset* following *std* is required. If no *offset* follows *dst*, daylight savings time is assumed to be one hour ahead of standard time. One or more digits may be used; the value is always interpreted as a decimal number. The hour must be between 0 and 24, and the minutes (and seconds) if present between 0 and 59. Out of range values may cause unpredictable behavior. If preceded by a “-”, the timezone is east of the Prime Meridian; otherwise it is west (which may be indicated by an optional preceding “+” sign).

rule Indicates when to change to and back from summer time. The *rule* has the form:

start/time, end/time

Which indicates when to change to and back from daylight savings time, where *start/time* describes when the change from standard time to daylight savings time occurs, and *end/time* describes when the change back happens. Each *time* field describes when, in current local time, the change is made.

The formats of *start* and *end* are one of the following:

Jn The Julian day n ($1 \leq n \leq 365$). Leap days are not counted. That is, in all years, February 28 is day 59 and March 1 is day 60. It is impossible to refer to the occasional February 29.

n The zero-based Julian day ($0 \leq n \leq 365$). Leap days are counted, and it is possible to refer to February 29.

Mm.n.d The d^{th} day, ($0 \leq d \leq 6$) of week n of month m of the year ($1 \leq n \leq 5$, $1 \leq m \leq 12$), where week 5 means “the last d -day in month m ” which may occur in either the fourth or the fifth week). Week 1 is the first week in which the d^{th} day occurs. Day zero is Sunday.

The *time* has the same format as *offset* except that no leading sign (“-” or “+”) is allowed. The default, if *time* is not given is 02:00:00.

Further names may be placed in the environment by the `export` command and *name=value* arguments in `sh(1)`, or by `exec(2)`. It is unwise to conflict with certain shell variables that are frequently exported by `.profile` files: `MAIL`, `PS1`, `PS2`, `IFS` (see `profile(4)`).

Whenever `asctime(3)`, `cftime(3)`, `ctime(3)`, `localtime(3)`, `mktime(3)`, or `strptime(3)` is called, the time zone names contained in the external variable `tzname(3)` shall be set as if the `tzset(3)` function had been called.

Applications are explicitly allowed to change `TZ` and have the changed `TZ` apply to themselves.

The system-wide default value for `TZ` can be changed with the `sysadm(1M)` command.

NOTE:

There is an unfortunate potential for confusion with time zones identified by an offset from GMT. The `TZ` value `GMT+5`, according to the rules presented here, is equivalent to `EST5 — 5 hours West of GMT`. There is also a timezone definition file that can be used by setting `TZ` to `:GMT+5`, but this file defines the time zone 5 hours *East* of GMT. Existing practice requires that both these notations be supported.

SEE ALSO

`chrtbl(1M)`, `colltbl(1M)`, `montbl(1M)`, `netconfig(4)`, `strptime(4)`, `passwd(4)`, `profile(4)` in the *System Manager's Reference*.
`exec(2)`, `addseverity(3C)`, `catopen(3C)`, `ctime(3C)`, `ctype(3C)`, `fmtmsg(3C)`, `getdate(3C)`, `getenv(3C)`, `gettxt(3C)`, `localeconv(3C)`, `mbchar(3C)`, `mktime(3C)`, `printf(3C)`, `strcoll(3C)`, `strptime(3C)`, `strtod(3C)`,

strxfrm(3C), strftime(4), time(4), timezone(4).
cat(1), date(1), ed(1), gencat(1), fmtmsg(1), ls(1), login(1), mkmsgs(1),
nice(1), nohup(1), sh(1), sort(1), time(1), vi(1), zic(1) in the *User's Reference*.
getnetpath(3N), in the *Programmer's Guide: Networking Interfaces*.

COPYRIGHTS

Portions of this text are reprinted from IEEE Std 1003.1-1988, *Portable Operating System Interface for Computer Environment*, copyright © 1988 by the Institute of Electrical and Electronics Engineers, Inc., with the permission of the IEEE Standards Department. To purchase IEEE Standards, call 800/678-IEEE.

In the event of a discrepancy between the electronic and the original printed version, the original version takes precedence.

NAME

eucioctl – generic interface to EUC handling TTY drivers and modules

SYNOPSIS

```
#include <sys/eucioctl.h>

ioctl(int fd, I_STR, struct strioctl *sb);
```

DESCRIPTION

This interface is implemented in TTY drivers and pushable *STREAMS* modules that handle EUC codes. It is intended as a generic interface for EUC handling, to eliminate an explosion of “module specific” `ioctl` calls that would otherwise be necessary, and to provide uniformity in dealing with EUC codesets in the TTY subsystem.

Several calls are defined. The first two calls take an argument, which is expected to be a pointer to an `eucioc` structure, defined in the header file `<sys/eucioctl.h>`:

```
struct eucioc {
    unsigned char eucw[4];
    unsigned char scrw[4];
};
typedef struct eucioc eucioc_t;
```

In all cases, these calls return non-zero on failure. Failure should be usually taken as an indication that the current driver, or line discipline module, does not support EUC in which case `errno` will be set to `EINVAL`. For the `EUC_WSET` and `EUC_WGET` calls `errno` will be set to `EPROTO` if the `struct eucioc` argument is invalid.

- `EUC_WSET` This call takes a pointer to an `eucioc` structure, and uses it to set the EUC line discipline’s local definition for the codeset widths to be used for subsequent operations. Within the *STREAM*, the line discipline may optionally notify other modules of this setting via `M_CTL` messages.
- `EUC_WGET` This call takes a pointer to an `eucioc` structure, and returns in it the EUC codeset widths currently in use by the EUC line discipline. It need be recognized *only* by line discipline modules.

The following calls take no arguments. They should only fail if the driver (at the bottom of the TTY *STREAM*) does not recognize EUC codes. Drivers that support EUC, whether the *STREAM* contains modules that respond to the calls or not, will *recognize* the calls and acknowledge them. These calls are normally only *interpreted* by modules that have modes other than ASCII, and/or do some form of I/O conversion that normally prevents a program from receiving non-EUC characters in its byte stream. All of these calls, when received by modules, are passed down the TTY *STREAM*, to be ultimately acknowledged by the TTY driver.

- `EUC_MSAVE` This call has no effect on modules that are currently in ASCII mode. Otherwise (i.e., for modules *not* in ASCII mode), the following actions are taken by all modules that recognize this call: (1) the current “mode” status is saved, (2) the mode is changed to ASCII mode immediately.
- `EUC_MREST` If a mode was saved via a previous `EUC_MSAVE` call, the saved mode is restored, and the “saved state” flag is cleared. If the mode was not previously saved, this call has no effect. (The exact semantics are somewhat dependent on the module, since some

modules may respond to specific user-requests to switch modes, even while a mode is being saved via `EUC_MSAVE`.)

<code>EUC_IXLOFF</code>	If a module is currently in a state where “input conversion” is being performed on the incoming byte stream, then input conversion is turned off, and the module’s “mode” status is saved. If no input conversion is being performed, there is no effect on the module. The purpose of this call is to provide a way of insuring a “pure” byte stream to the program. The byte stream while input conversion is off is, of course, not guaranteed to be a stream of EUC characters. Turning off input conversion is roughly equivalent to the old concept of “raw” mode, if used in conjunction with <code>ICANON</code> off. It should normally not be used by applications.
<code>EUC_IXLON</code>	If a module previously saved its state and turned off input conversion, then input conversion is restored (i.e., turned back on); otherwise, there is no effect.
<code>EUC_OXLOFF</code>	In a manner similar to <code>EUC_IXLOFF</code> , any “output conversion” is turned off, and the current mode status saved.
<code>EUC_OXLON</code>	In a manner similar to <code>EUC_IXLON</code> , any saved “output conversion” status is restored (i.e., output conversion is turned back on if previously turned off via <code>EUC_OXLOFF</code>).

Limitations

Drivers and modules that support EUC should all respond appropriately to these calls, depending on their type. Line disciplines must respond to `EUC_WSET` and `EUC_WGET`, changing their current codeset sizes to match `EUC_WSET` requests. All TTY STREAMS modules that do any input or output conversion should recognize the other calls; modules that do no codeset conversion are not required to recognize the calls, but *must* pass them through. Drivers that support EUC TTY STREAMS must all acknowledge the ON/OFF calls, whether the drivers themselves are affected or not, since these calls are purposely *not* acknowledged by modules which receive them; they are intended to be made available for affecting all modules in *the whole STREAM*.

FILES

`/usr/include/sys/eucioctl.h`

SEE ALSO

`eucset(1)`.

NOTES

Adherence to this protocol for all EUC handling modules is strongly encouraged in order to increase portability and language-independence of applications. These calls are intended as a small set of primitives to help reduce an anticipated plethora of module- and language-dependent operations.

NAME

fcntl – file control options

SYNOPSIS

```
#include <fcntl.h>
```

DESCRIPTION

The fcntl(2) function helps you control open files. This include file describes *commands* and *arguments* to fcntl and open(2).

```
/* Flag values accessible to open(2) and fcntl(2) */
/* (The first three can only be set by open) */
```

```
#define O_RDONLY 0
#define O_WRONLY 1
#define O_RDWR 2
#define O_NDELAY 04 /* Non-blocking I/O */
#define O_APPEND 010 /* append (writes guaranteed at the end) */
```

```
/* Flag values accessible only to open(2) */
#define O_CREAT 00400 /* open with file create (uses 3rd open arg)*/
#define O_TRUNC 01000 /* open with truncation */
#define O_EXCL 02000 /* exclusive open */
```

```
/* fcntl(2) commands */
#define F_DUPFD 0 /* Duplicate fildes */
#define F_GETFD 1 /* Get the `close-on-exec' flag */
#define F_SETFD 2 /* Set the `close-on-exec' flag */
#define F_GETFL 3 /* Get file flags */
#define F_SETFL 4 /* Set file flags */
#define F_GETLK 5 /* Get record lock status */
#define F_SETLK 6 /* Set record lock or fail */
#define F_SETLKW 7 /* Set record lock or pend */
#define F_CHKFL 8 /* Check flags for validity */
#define F_FREESP 11 /* Free up file space */
#define F_GETOWN 65536 /* Get owner of fildes */
#define F_SETOWN 65537 /* Set owner of fildes */
```

SEE ALSO

fcntl(2), open(2).

NAME

hier - DG/UX file system hierarchy

DESCRIPTION

The following outline gives a quick tour through a representative directory hierarchy. The basis of the outline is the DG/UX operating system. It is not exhaustive.

```

/      root
/admin  typical home directory for sysadm
/bin    a symbolic link to /usr/bin
/dev/   devices (7)
        console    system console
        dsk/*      logical disks
        error      the error device, error(7)
        kmem       logical kernel memory
        lp         line printer, lp(7)
        mem        physical memory
        mt/*       magnetic tapes
        null       the null device; i.e., the "bit bucket"
        pdsk/*     physical disks
        rdk/*      raw logical disks
        rmt/*      raw magnetic tapes
        rpdsk/*    raw physical disks
        tty[0-9]* terminals, ttcompat(7)
        ttyp[0-9]* pseudo terminals
        ...
/dgux   the kernel binary (DG/UX System itself)
/etc/   essential data and maintenance utilities; section (1M)
        cron.d     main cron directory also containing scheduler for at(1)
        default    directory containing defaults for various programs
        cron       file specifying actions for cron(1M) to take
        ...
        dumpdates  dump history, dump(1M)
        erm/       directory containing error message files
        ermes      file containing text of system error messages refer-
                    enced by perror(3C)
        ...
        fstab      file system configuration table, fstab(4)
        group      group file, group(4)
        getty      initial part of login sequence, getty(1M)
        hosts      host name to network address mapping file, hosts(4)
        init.d     scripts for rc.d directories init(1M), rc(1M)
        inittab    the init configuration table, inittab(4)
        login.csh  global csh(1) startup script, csh(1)
        mnttab     mounted file table, mnttab(4)
        motd       message of the day, login(1)
        networks   network name to network number mapping file, networks(4)
        passwd     password file, passwd(4)
        profile    global sh(1) startup script, sh(1)
        protocols  protocol name to protocol number mapping file, protocols(4)
        rc.init    shell program to enter init states (0, 1, ...), init(1M),
                    rc(1M)

```

```

rc[S0123456i].d
    links to init.d scripts for actions in init states, init(1M),
    rc(1M)
services    network services definition file, services(4)
ttydefs     terminal modes for getty, ttydefs(4)
wtmp, utmp  login history, utmp(4)
...
/lib        a symbolic link to /usr/lib
/lost+found
    directory for connecting detached files for fsck(1M)
/sbin/
    basic utilities
    halt     stop the system processor, halt(1M)
    init     the parent of all processes, init(1M)
    mount    mount(1M)
    sh       Bourne shell, sh(1)
    ...
/srv/       server directory
release     release directory
            PRIMARY directory for the primary release
            root     directory containing root directories of
                    diskless clients
swap        directory containing swap space for diskless clients
...
/tmp/       temporary files, usually on a fast device, cf. /var/tmp/
e*          used by ed(1)
ctm*        used by cc(1)
...
/udd/       directory containing a local file system of user directories and possibly
            mounted file systems of remote user directories (name udd is optional)
local       local directory containing user directories
            wd       user's initial working directory, typically the user's
                    login name
                    .profile
                                set environment for sh(1), environ(5)
            .cshrc   startup file for csh(1)
            .editreadrc
                                startup file for Editread command-line
                                editor
            .exrc    startup file for ex(1)
            .mailrc  startup file for mail(1)
            .netrc   startup file for various network programs
            calendar
                                user's datebook for calendar(1)
            remote   mounted remote directory containing user directories
/usr/       system software directory, typically read-only
bin/       utility programs
as         Data General macro assembler
cc         C compiler executive, cf. /usr/lib/ccomp,
            /lib/cpp
csh        C shell, csh(1)
sh         Bourne shell, sh(1)

```

```

...
catman/  online manual pages for man(1)
u_man/   User's Reference for the DG/UX System
          man0/   general: contents, permuted index
                  contents.0.z
                  index.0.z
          man1/   user commands and application programs
                  acctcom.1.z
                  alpq.1.z
                  ...
          man5/   miscellaneous features
                  editread.5.z
                  ...
a_man/   System Manager's Reference for the DG/UX System
          man1/   system maintenance commands
                  accept.1m.z
                  acct.1m.z
                  ...
          man4/   file formats for system maintenance com-
                  mands
                  dfm.4m.z
                  ...
          man7/   special files
                  alp.7.z
                  ...
          man8/   system maintenance procedures
                  crash.8.z
                  ...
p_man/   Programmer's Reference for the DG/UX System
          man1/   programmer commands
                  admin.1.z
                  ar.1.z
                  ...
          man2/   system calls
                  accept.2.z
                  access.2.z
                  ...
          man3/   runtime libraries
                  a64l.3c.z
                  ...
          man4/   file formats
                  a.out.4.z
                  ...
          man5/   miscellaneous features
                  ascii.5.z
                  ...
          man6/   networking protocols
                  dot3.6.z
                  ...
include/  standard #include files
          a.out.h  object file layout, a.out(4)
          stdio.h  standard I/O, intro(3)

```

```

math.h (3M)
net/ network header files
sys/ system-defined layouts
...
lib/ object libraries, etc.
acct/* account programs and shell scripts
gcc symbolic link to gcc-1
gcc-1 directory for GNU C preprocessor and compiler
libc.a elink pointing to directory containing system calls,
standard I/O, etc. (2,3,3S)
locale directory containing locale-specific information
...
uucp/ programs and data for uucp(1)
L.sys remote system names and numbers
uucico the real copy program
...
unittab conversion tables for units(1)
sbin/ utility programs
cron the clock server, cron(1M)
dump dump program, dump(1M)
restore restore program, restore(1M)
...
sde/ software development environment directory
m88kdguxelf
Motorola 88000 ELF binary interface, elf(3E)
usr/bin utility programs
usr/lib libraries
lint[12] subprocesses for lint(1)
llib-1c dummy declarations for
/lib/libc.a, used by lint(1)
llib-1m dummy declarations for
/lib/libc.m
...
...
tmp/ symbolic link to /var/tmp
stm* used by sort(1)
/var/ directory to contain various writable directories
adm/ administrative information
acct/* system accounting data files
sulog log of the invocations of the su(1) command
mail/* the directory where mail messages are stored
news/* the directory where news items are stored
preserve/ editor temporaries preserved here after crashes/hangups
spool/ delayed execution files
at/ used by at(1)
uucp/ work files and staging area for uucp(1)
LOGFILE summary log
LOG.* log file for one transaction
tmp/ directory to avoid writing temporary files to /usr

```

SEE ALSO

find(1), grep(1), ls(1) in the *User's Reference for the DG/UX System*.

CAUTION

The position of files is subject to change without notice.

NAME

hostname – hostname resolution description

DESCRIPTION

Hostnames are expressed as domain names, where a domain name is a hierarchical, dot-separated list of labels; for example, the machine `abc`, in the `de` subdomain of the `COM` subdomain would be represented as `abc.de.COM` (with no trailing dot).

A label consists of up to 24 characters drawn from the lowercase alphabet (`a-z`), uppercase alphabet (`A-Z`), digits (`0-9`), and minus sign (`-`). You cannot include blank or space characters in a label. No distinction is made between upper and lower case.

Hostnames are often used with network client and server programs, which must generally translate the name to an address for use. (This translation is generally performed by the library routine `gethostbyname(3N)`.) Hostnames are resolved by the domain name resolver in the following way.

If the hostname consists of a single component, that is, contains no dot, and if the environment variable `HOSTALIASES` is set to the name of a file, that file is searched for a string matching the hostname. The file should consist of lines made up of two strings separated by white space, the first of which is the hostname alias, and the second of which is the complete hostname to be substituted for that alias. If a case-sensitive match is found between the hostname to be resolved and the first field of a line in the file, the substituted name is looked up with no further processing.

If the input hostname ends with a trailing dot, the trailing dot is removed, and the remaining hostname is looked up with no further processing. A hostname that ends with a trailing dot is called a "fully-qualified" hostname.

If the input hostname does not end with a trailing dot, it is looked up in the local domain and its parent domains until either a match is found or fewer than two components of the local domain remain. For example, in the domain `tnt.acme.COM`, the name `spectre.bucky` will be checked first as `spectre.bucky.tnt.acme.COM`, then as `spectre.bucky.acme.COM`, then as `spectre.bucky.COM`, and then as `spectre.bucky`.

If you use the Domain Name System (DNS) you must either 1) set the default domain in `resolv.conf(4)` and use hostnames that consist of a single component or 2) consistently use fully-qualified hostnames.

SEE ALSO

`named(1M)`, `gethostbyname(3N)`, `resolv.conf(4)`, RFC883.

NAME

langinfo - language information constants

SYNOPSIS

```
#include <langinfo.h>
```

DESCRIPTION

This header file contains the constants used to identify items of langinfo data. The mode of *items* is given in *nl_types*.

DAY_1	Locale's equivalent of 'sunday'
DAY_2	Locale's equivalent of 'monday'
DAY_3	Locale's equivalent of 'tuesday'
DAY_4	Locale's equivalent of 'wednesday'
DAY_5	Locale's equivalent of 'thursday'
DAY_6	Locale's equivalent of 'friday'
DAY_7	Locale's equivalent of 'saturday'
ABDAY_1	Locale's equivalent of 'sun'
ABDAY_2	Locale's equivalent of 'mon'
ABDAY_3	Locale's equivalent of 'tue'
ABDAY_4	Locale's equivalent of 'wed'
ABDAY_5	Locale's equivalent of 'thur'
ABDAY_6	Locale's equivalent of 'fri'
ABDAY_7	Locale's equivalent of 'sat'
MON_1	Locale's equivalent of 'january'
MON_2	Locale's equivalent of 'february'
MON_3	Locale's equivalent of 'march'
MON_4	Locale's equivalent of 'april'
MON_5	Locale's equivalent of 'may'
MON_6	Locale's equivalent of 'june'
MON_7	Locale's equivalent of 'july'
MON_8	Locale's equivalent of 'august'
MON_9	Locale's equivalent of 'september'
MON_10	Locale's equivalent of 'october'
MON_11	Locale's equivalent of 'november'
MON_12	Locale's equivalent of 'december'
ABMON_1	Locale's equivalent of 'jan'
ABMON_2	Locale's equivalent of 'feb'
ABMON_3	Locale's equivalent of 'mar'
ABMON_4	Locale's equivalent of 'apr'
ABMON_5	Locale's equivalent of 'may'

ABMON_6	Locale's equivalent of 'jun'
ABMON_7	Locale's equivalent of 'jul'
ABMON_8	Locale's equivalent of 'aug'
ABMON_9	Locale's equivalent of 'sep'
ABMON_10	Locale's equivalent of 'oct'
ABMON_11	Locale's equivalent of 'nov'
ABMON_12	Locale's equivalent of 'dec'
RADIXCHAR	Locale's equivalent of '.'
THOUSEP	Locale's equivalent of ','
YESSTR	Locale's equivalent of 'yes'
NOSTR	Locale's equivalent of 'no'
CRNCYSTR	Locale's currency symbol
D_T_FMT	Locale's default format for date and time
D_FMT	Locale's default format for the date
T_FMT	Locale's default format for the time
AM_STR	Locale's equivalent of 'AM'
PM_STR	Locale's equivalent of 'PM'

This information is retrieved by `nl_langinfo`.

The items `CRNCYSTR`, `RADIXCHAR` and `THOUSEP` are extracted from the fields `currency_symbol`, `decimal_point` and `thousands_sep` in the structure returned by `localeconv`.

The items `T_FMT`, `D_FMT`, `D_T_FMT`, `YESSTR` and `NOSTR` are retrieved from a special message catalog named `Xopen_info` which should be generated for each locale supported and installed in the appropriate directory [see `gettext(3C)` and `mkmsgs(1)`]. This catalog should have the messages in the order `T_FMT`, `D_FMT`, `D_T_FMT`, `YESSTR` and `NOSTR`.

All other items are as returned by `strftime`.

SEE ALSO

`chrtbl(1M)`, `mkmsgs(1)`, `gettext(3C)`, `localeconv(3C)`, `nl_langinfo(3C)`, `strftime(3C)`, `strftime(4)`, `nl_types(5)`.

NAME

legend – Debugging information technology

DESCRIPTION

Legend debugging information (or legends for short) is used by the `sdb(1)` and `dbx(1)` debuggers when debugging an ELF executable and always used by the `mxd(1)` debugger. It is created during compilation typically by `as(1)` which calls the `ctl(1)` translator.

Traditional UNIX compilation systems control debugging information by the use of a `-g` option. If the `-g` option is present on the compiler command line (e.g. "`cc -g`") then debugging information is generated. Legend technology provides a number of options that can't be coded into a single yes or no option but many existing applications have makefiles and shell scripts that users don't want to modify. The legend options, therefore, are controlled by an environment variable called `LEGENDS`.

OPTIONS

The following values can be placed in the `LEGENDS` environment variable, separated by blanks, to control the generation of legends.

-external

Store the legend data in a separate file. If the target file is named "`prog.o`", then the legend will be stored in a file named "`prog.lg`". This reduces the size of object files, libraries and executables, significantly saving link time as well as disk space.

-no-external

Store legend data in the object file. This is the default.

-compress

Legends come in two forms that allow you to make a speed/space trade-off. If present, this option requests that legends be generated in a compressed form. You can mix compressed and uncompressed legends into the same application.

-no-compress

Don't compress the legend. This is the default.

-keep-std

This option only makes sense when creating a COFF object file. If present, it directs the legend translator to preserve the COFF information in addition to generating a legend. This allows the use of COFF debuggers in addition to `mxd(1)` on resulting executables. By default the COFF information is deleted.

-no-keep-std

Don't preserve COFF information. This is the default.

-v Print the version of `ctl` to `stderr`.

-warn Print warning messages. They are suppressed by default.

SEE ALSO

`ctl(1)`, `cc(1)`, `gcc(1)`, `ghcc(1)`, `ghf77(1)`, `ghpc(1)`, `as(1)`, `mxd(1)`, `sdb(1)`, `dbx(1)`

NAME

math – math functions and constants

SYNOPSIS

```
#include <math.h>
```

DESCRIPTION

This file contains declarations of all the functions in the Math Library (described in Section 3M), as well as various functions in the C Library (Section 3C) that return floating-point values.

It defines the structure and constants used by the `matherr(3M)` error-handling mechanisms, including the following constant used as a error-return value:

HUGE The maximum value of a single-precision floating-point number.

The following mathematical constants are defined for user convenience:

M_E The base of natural logarithms (e).

M_LOG2E The base-2 logarithm of e .

M_LOG10E The base-10 logarithm of e .

M_LN2 The natural logarithm of 2.

M_LN10 The natural logarithm of 10.

M_PI π , the ratio of the circumference of a circle to its diameter.

M_PI_2 $\pi/2$.

M_PI_4 $\pi/4$.

M_1_PI $1/\pi$.

M_2_PI $2/\pi$.

M_2_SQRTPI $2/\sqrt{\pi}$.

M_SQRT2 The positive square root of 2.

M_SQRT1_2 The positive square root of $1/2$.

The following mathematical constants are also defined in this header file:

MAXFLOAT The maximum value of a non-infinite single-precision floating point number.

HUGE_VAL positive infinity.

For the definitions of various machine-dependent constants, see `values(5)`.

SEE ALSO

`intro(3)`, `matherr(3M)`, `values(5)`.

NAME

`misalign` - handle misaligned memory access faults

DESCRIPTION

The Motorola M88000 microprocessor family, on which the Data General AViiON computers are based, requires that data be aligned in memory to their lengths. If the address of a datum is not an integral multiple of the datum's length, a reference to the datum will cause a misaligned access fault. For example, if a program attempts to fetch a 16-bit value from an odd address, a misaligned access fault occurs. A misaligned access fault results in the delivery of a SIGBUS signal to the application. If the application has not defined a SIGBUS signal handler, the application terminates with a "Bus error" message.

A program can use the facilities defined herein to repair misaligned access faults that it incurs. These facilities can be useful in porting applications that were written for computers that don't impose alignment restrictions as strict as those of the M88000 family. The facilities are offered in three forms, for generality and convenience:

- functions to repair misaligned access faults with which you can construct your own SIGBUS signal handler
- predefined SIGBUS signal handlers that are built from the repair functions mentioned above
- a link-time mechanism to have one of the predefined SIGBUS signal handlers installed automatically when your program runs

To use these facilities in any of the three forms you must specify the misalignment handling library, `libmisalign.a`, to the linker. To do this you can simply include `-lmisalign` on the `cc` or `ld` command line. If you use the `ld` command, be sure to specify the misalignment handling library before specifying `libc`, as with `-lc`.

If your program does not care to handle SIGBUS signals other than those representing misaligned access faults, you can simply specify `-u misalign.auto_install` to the linker before specifying the misalignment handling library. With such a specification, a SIGBUS handler that catches SIGBUS signals and repairs misaligned access faults will be installed automatically when your program runs. You do not need to modify your original program to use misalignment handling in this way.

If your program does not care to handle SIGBUS signals other than those representing misaligned access faults but does want to establish signal handlers explicitly, you can use the predefined signal handlers `misalignment_sigbus_handler_ocs1` and `misalignment_sigbus_handler_abi1`. These signal handlers catch SIGBUS signals and repair misaligned access faults in the same way; they differ only in the target environments for which they are appropriate. If you establish the signal handler in a COFF environment (such as `m88kbc`, `m88kocs`, or `m88kdguxcoff`), use `misalignment_sigbus_handler_ocs1`. If you establish the signal handler in an ELF environment (such as `m88kdguxelf`), use `misalignment_sigbus_handler_abi1`.

If a predefined signal handler catches a SIGBUS signal that does not represent a misaligned access fault, or if it cannot repair a misaligned access fault for any reason, it aborts the program by sending a SIGBUS signal to its own process using the `kill()` function. This same failure response occurs when `-u misalign.auto_install` is used, because one of the predefined handlers is installed automatically in that case.

If the failure treatment of the predefined handlers is inappropriate for your program, or if you want to handle SIGBUS signals other than those representing misaligned access faults, you can use the functions `repair_misalignment_ocs1` and

`repair_misalignment_abi1`. These functions attempt to repair misaligned access faults and indicate their success or failure. You can call one of these functions from your program's SIGBUS signal handler, then take other appropriate action in the case of failure. The two functions act the same; they differ only in their argument lists and the target environments for which they are appropriate.

`repair_misalignment_ocs1` takes two arguments, the same arguments received by a signal handler that was established in a COFF environment.

`repair_misalignment_abi1` takes three arguments, the same arguments received by a signal handler that was established in an ELF environment by a call to `sigaction(2)` with the `SA_SIGINFO` flag set.

The repair functions return an integer whose value indicates whether the repair was successful. If the return value is negative, the repair failed; otherwise, it succeeded. Furthermore, if the return value is zero, the site of the misaligned access fault was patched so that future faults will not occur; if the return value is positive, patching was not possible.

The remainder of this description applies to repair of misaligned access faults by any of the three forms described above (automatic installation of predefined handler, explicit installation of predefined handler, or direct use of repair function). The common facilities are referred to collectively as "misalignment handling."

Misalignment handling can not only emulate the faulting memory access but also patch the faulting instruction so that future faults will not occur. Patching can greatly speed up an application that suffers misaligned access faults. Note, however, that patching renders your program's text area less sharable. Pages that contain faulting instructions that are patched become private to your process.

If a faulting instruction appears to be in a delay slot (that is, the instruction appears to follow a flow control instruction with delayed branching selected), it is assumed that the instruction is indeed in a delay slot, and instructions are generated to patch the flow control instruction as well as the faulting instruction. Patching an instruction in a delay slot requires more instructions. If the resulting performance of your program is inadequate due to a large number of misaligned access faults, you may wish to instruct the compiler not to perform delay slot optimization. For `gcc`, use the `-fno-delayed-branch` option. For `cc`, use the `-w0,-fno-delayed-branch` option. For Green Hills compilers, use the `-x307` option.

Three M88000 instructions can incur misaligned access faults: `ld`, `st`, and `xmem`. Misalignment handling handles all three instructions, but cannot maintain atomicity in most cases because the access must be done in pieces. The loss of atomicity is generally not important except for `xmem`, which is not typically generated by compilers.

You can control the behavior of misalignment handling by including an options file among the object files presented to the linker. The file `misalign-options.c` is provided as a prototype from which you can create your own version. The following table shows what behaviors the options file controls and what the defaults are when no options file is present. See the commentary in the prototype options file for complete information.

Behavior	Default
Whether to patch	yes
Whether to patch in delay slots	yes
What registers to treat as scratch	r26 through r29
How much bss area to preallocate	none
How to abort on failure	send SIGBUS signal to self

EXAMPLE

The following `gcc` command compiles a program for debugging with `mxd(1)` and links it with misalignment handling.

```
gcc -g -mlegend -o example example.c -u misalign.auto_install -lmisalign
```

`Mxd` can be used to determine where misaligned accesses occur. The following shell script produces a backtrace of the stack on each misaligned access. It then continues the program which allows misalignment handling to fix the access.

```
mxd example <<EOF

,, Do a walkback on each SIGBUS.

signal, catch bus, \
    action { \
        new-line; \
        write MISALIGNED ACCESS; \
        walkback, arg, locals; \
        continue \
    }

continue      ,, Start the program.
bye           ,, Quit when it is done.
```

EOF

The backslashes shown above are necessary.

If you use the above approach with patching enabled (the default), you should note two things. First, warnings of the following form may result but can be ignored:

```
Warning: instruction 00000000 not yet supported, ignored
```

Second, misaligned access faults can occur in the patch code sequences themselves. You need not worry about these faults, because in these cases the original faulting instruction is "repatched."

SEE ALSO

`mxd(1)`, `sigaction(2)`, `kill(2)`, `sde(5)`,
Using the Multi-Extensible Debugger (Mxd for DG/UX and 386/ix Systems),
88open Binary Compatibility Standard,
88open Object Compatibility Standard,
MC88100 RISC Microprocessor User's Manual.

NAME

nl_types – native language data types

SYNOPSIS

```
#include <nl_types.h>
```

DESCRIPTION

This header file contains the following definitions that relate to the X/open-style message facility:

nl_catd	used by the message catalog functions <code>catopen</code> , <code>catgets</code> and <code>catclose</code> to identify a catalogue
nl_item	used by <code>nl_langinfo</code> to identify items of <code>langinfo</code> data. Values for objects of type <code>nl_item</code> are defined in <code>langinfo.h</code> .
NL_SETD	used by <code>gencat</code> when no <code>\$set</code> directive is specified in a message text source file. This constant can be used in subsequent calls to <code>catgets</code> as the value of the set identifier parameter.
NL_MGSMAX	maximum number of messages per set
NL_SETMAX	maximum number of sets per catalogue.
NL_TEXTMAX	maximum size of a message in bytes. " 41" counts as one byte; a multibyte character counts as more than one byte.
DEF_NLSPATH	the default search path for locating catalogues.

SEE ALSO

`gencat(1M)`, `catgets(3C)`, `catopen(3C)`, `nl_langinfo(3C)`, `langinfo(5)`, `mkmsgs(1)`, `gettext(3C)` — AT&T-style message facility.

NAME

printcap - printer capability data base

SYNOPSIS

/etc/printcap

DESCRIPTION

Printcap is a simplified version of the `termcap(5)` data base used to describe line printers. The spooling system accesses the `printcap` file every time it is used, allowing dynamic addition and deletion of printers. Each entry in the data base is used to describe one printer. This data base may not be substituted for, as is possible for `termcap`, because it may allow accounting to be bypassed.

The default printer is normally `lp`, though the environment variable `PRINTER` may be used to override this. Each spooling utility supports an option, `-Pprinter`, to allow explicit naming of a destination printer.

Capabilities

Refer to `termcap(5)` for a description of the file layout.

Name	Type	Default	Description
af	str	NULL	name of accounting file
br	num	none	if <code>lp</code> is a tty, set baud rate (ioctl call)
cf	str	NULL	cifplot data filter
df	str	NULL	tex data filter (DVI format)
fc	num	0	if <code>lp</code> is a tty, clear flag bits (sgtty.h)
ff	str	"\f"	string to send for a form feed
fo	bool	false	print a form feed when device is opened
fs	num	0	like "fc" but set bits
gf	str	NULL	graph data filter (plot (3X) format)
hl	bool	false	print the burst header page last
ic	bool	false	driver supports nonstandard ioctl to indent printout
if	str	NULL	name of text filter which does accounting
lf	str	"/dev/console"	error logging file name
lo	str	"lock"	name of lock file
lp	str	"/dev/lp"	device name to open for output
mx	num	1000	maximum file size (in BUFSIZ blocks), 0 = unlimited
nd	str	NULL	next directory for list of queues (unimplemented)
nf	str	NULL	ditroff data filter (device independent troff)
of	str	NULL	name of output filtering program
pc	num	200	price per foot or page in hundredths of cents
pl	num	66	page length (in lines)
pw	num	132	page width (in characters)
px	num	0	page width in pixels (horizontal)
py	num	0	page length in pixels (vertical)
rf	str	NULL	filter for printing FORTRAN style text files
rg	str	NULL	restricted group; only group members can access
rm	str	NULL	machine name for remote printer
rp	str	"lp"	remote printer name argument
rs	bool	false	restrict remote users to those with local accounts
rw	bool	false	open the printer device for reading and writing
sb	bool	false	short banner (one line only)
sc	bool	false	suppress multiple copies
sd	str	"/usr/spool/lpd"	spool directory
sf	bool	false	suppress form feeds

sh	bool	false	suppress printing of burst page header
st	str	"status"	status file name
tf	str	NULL	troff data filter (cat phototypesetter)
tr	str	NULL	trailer string to print when queue empties
vf	str	NULL	raster image filter
xc	num	0	if lp is a tty, clear local mode bits [tty(4)]
xs	num	0	like "xc" but set bits

If the local line printer driver supports indentation, the server must understand how to invoke it.

Filters

The `lpd(1M)` server creates a pipeline of *filters* to process files for various printer types. The filters selected depend on the flags passed to `lpr(1)`. The pipeline set up is:

```

-p    pr | if      regular text + pr(1)
none  if          regular text

```

The `if` filter is invoked with arguments:

```
if [ -c ] -wwidth -llength -iindent -n login -h host acct-file
```

The `-c` flag is passed only if the `-l` flag (pass control characters literally) is specified to `lpr`. *Width* and *length* specify the page width and length (from `pw` and `pl` respectively) in characters. The `-n` and `-h` parameters specify the login name and host name of the owner of the job respectively. *Acct-file* is passed from the `af` *printcap* entry.

If no `if` is specified, `of` is used instead, with the distinction that `of` is opened only once, while `if` is opened for every individual job. Thus, `if` is better suited to performing accounting. The `of` is only given the *width* and *length* flags.

All other filters are called as:

```
filter -xwidth -ylength -n login -h host acct-file
```

where *width* and *length* are represented in pixels, specified by the `px` and `py` entries respectively.

All filters take *stdin* as the file, *stdout* as the printer, may log either to *stderr* or using `syslog(3)`, and must not ignore SIGINT.

Logging

Error messages generated by the line printer programs themselves (that is, the *lp** programs) are logged by `syslog(3)` using the *LPR* facility. Messages printed on *stderr* of one of the filters are sent to the corresponding `lf` file. The filters may, of course, use *syslog* themselves.

Error messages sent to the console have a carriage return and a line feed appended to them, rather than just a line feed.

SEE ALSO

`lpc(1M)`, `lpd(1M)`, `lpq(1)`, `lpr(1)`, `lprm(1)`, `termcap(5)`.

NAME

prof – profile within a function

SYNOPSIS

```
#define MARK
#include <prof.h>

void MARK (name);
```

DESCRIPTION

MARK introduces a mark called *name* that is treated the same as a function entry point. Execution of the mark adds to a counter for that mark, and program-counter time spent is accounted to the immediately preceding mark or to the function if there are no preceding marks within the active function.

name may be any combination of letters, numbers, or underscores. Each *name* in a single compilation must be unique, but may be the same as any ordinary program symbol.

For marks to be effective, the symbol MARK must be defined before the header file `prof.h` is included, either by a preprocessor directive as in the synopsis, or by a command line argument:

```
cc -p -DMARK foo.c
```

If MARK is not defined, the `MARK(name)` statements may be left in the source files containing them and are ignored. `prof -g` must be used to get information on all labels.

EXAMPLE

In this example, marks can be used to determine how much time is spent in each loop. Unless this example is compiled with MARK defined on the command line, the marks are ignored.

```
#include <prof.h>
foo( )
{
    int i, j;
    . . .
    MARK(loop1);
    for (i = 0; i < 2000; i++) {
        . . .
    }
    MARK(loop2);
    for (j = 0; j < 2000; j++) {
        . . .
    }
}
```

SEE ALSO

prof(1), profil(2), monitor(3C).

NAME

regex: compile, step, advance – regular expression compile and match routines

SYNOPSIS

```
#define INIT declarations
#define GETC(void) getc code
#define PEEKC(void) peekc code
#define UNGETC(void) ungetc code
#define RETURN(ptr) return code
#define ERROR(val) error code

#include <regex.h>

char *compile(char *instring, char *expbuf, char *endbuf, int eof);
int step(char *string, char *expbuf);
int advance(char *string, char *expbuf);
extern char *loc1, *loc2, *locs;
```

DESCRIPTION

These functions are general purpose regular expression matching routines to be used in programs that perform regular expression matching. These functions are defined by the <regex.h> header file.

The functions `step` and `advance` do pattern matching given a character string and a compiled regular expression as input.

The function `compile` takes as input a regular expression as defined below and produces a compiled expression that can be used with `step` or `advance`.

A regular expression specifies a set of character strings. A member of this set of strings is said to be matched by the regular expression. Some characters have special meaning when used in a regular expression; other characters stand for themselves.

The regular expressions available for use with the regex functions are constructed as follows:

<i>Expression</i>	<i>Meaning</i>
<i>c</i>	the character <i>c</i> where <i>c</i> is not a special character.
<i>\c</i>	the character <i>c</i> where <i>c</i> is any character, except a digit in the range 1–9.
<i>^</i>	the beginning of the line being compared.
<i>\$</i>	the end of the line being compared.
<i>.</i>	any character in the input.
<i>[s]</i>	any character in the set <i>s</i> , where <i>s</i> is a sequence of characters and/or a range of characters, e.g., <i>[c-c]</i> .
<i>[^s]</i>	any character not in the set <i>s</i> , where <i>s</i> is defined as above.
<i>r*</i>	zero or more successive occurrences of the regular expression <i>r</i> . The longest leftmost match is chosen.
<i>rx</i>	the occurrence of regular expression <i>r</i> followed by the occurrence of regular expression <i>x</i> . (Concatenation)
<i>r\{m,n\}</i>	any number of <i>m</i> through <i>n</i> successive occurrences of the regular expression <i>r</i> . The regular expression <i>r\{m\}</i> matches exactly <i>m</i> occurrences;

`r\{m,\}` matches at least *m* occurrences.

`\(r\)` the regular expression *r*. When `\n` (where *n* is a number greater than zero) appears in a constructed regular expression, it stands for the regular expression *x* where *x* is the *n*th regular expression enclosed in `\(` and `\)` that appeared earlier in the constructed regular expression. For example, `\(r\)x\(\y\)z\2` is the concatenation of regular expressions *rxzy*.

Characters that have special meaning except when they appear within square brackets (`[]`) or are preceded by `\` are: `.`, `*`, `[`, `\`. Other special characters, such as `$` have special meaning in more restricted contexts.

The character `^` at the beginning of an expression permits a successful match only immediately after a newline, and the character `$` at the end of an expression requires a trailing newline.

Two characters have special meaning only when used within square brackets. The character `-` denotes a range, `[c-c]`, unless it is just after the open bracket or before the closing bracket, `[-c]` or `[c-]` in which case it has no special meaning. When used within brackets, the character `^` has the meaning *complement of* if it immediately follows the open bracket (example: `[^c]`); elsewhere between brackets (example: `[c^]`) it stands for the ordinary character `^`.

The special meaning of the `\` operator can be escaped only by preceding it with another `\`, e.g. `\\`.

Programs must have the following five macros declared before the `#include <regexp.h>` statement. These macros are used by the `compile` routine. The macros `GETC`, `PEEKC`, and `UNGETC` operate on the regular expression given as input to `compile`. *NOTE:* If any of the macros below consist of more than 1 statement, then they should be surrounded with curly braces (`{, }`) or unexpected results will occur.

<code>GETC</code>	This macro returns the value of the next character (byte) in the regular expression pattern. Successive calls to <code>GETC</code> should return successive characters of the regular expression.
<code>PEEKC</code>	This macro returns the next character (byte) in the regular expression. Immediately successive calls to <code>PEEKC</code> should return the same character, which should also be the next character returned by <code>GETC</code> .
<code>UNGETC</code>	This macro causes the argument <code>c</code> to be returned by the next call to <code>GETC</code> and <code>PEEKC</code> . No more than one character of pushback is ever needed and this character is guaranteed to be the last character read by <code>GETC</code> . The return value of the macro <code>UNGETC(c)</code> is always ignored.
<code>RETURN(ptr)</code>	This macro is used on normal exit of the <code>compile</code> routine. The value of the argument <code>ptr</code> is a pointer to the character after the last character of the compiled regular expression. This is useful to programs which have memory allocation to manage.
<code>ERROR(val)</code>	This macro is the abnormal return from the <code>compile</code> routine. The argument <code>val</code> is an error number [see <code>ERRORS</code> below for meanings]. This call should never return.

The syntax of the `compile` routine is as follows:

```
compile(instring, expbuf, endbuf, eof)
```

The first parameter, *istring*, is never used explicitly by the `compile` routine but is useful for programs that pass down different pointers to input characters. It is sometimes used in the `INIT` declaration (see below). Programs which call functions to input characters or have characters in an external array can pass down a value of `(char *)0` for this parameter.

The next parameter, *expbuf*, is a character pointer. It points to the place where the compiled regular expression will be placed.

The parameter *endbuf* is one more than the highest address where the compiled regular expression may be placed. If the compiled expression cannot fit in `(endbuf-expbuf)` bytes, a call to `ERROR(50)` is made.

The parameter *eof* is the character which marks the end of the regular expression. This character is usually a `/`.

Each program that includes the `<regexp.h>` header file must have a `#define` statement for `INIT`. It is used for dependent declarations and initializations. Most often it is used to set a register variable to point to the beginning of the regular expression so that this register variable can be used in the declarations for `GETC`, `PEEKC`, and `UNGETC`. Otherwise it can be used to declare external variables that might be used by `GETC`, `PEEKC` and `UNGETC`. [See EXAMPLE below.]

The first parameter to the `step` and `advance` functions is a pointer to a string of characters to be checked for a match. This string should be null terminated.

The second parameter, *expbuf*, is the compiled regular expression which was obtained by a call to the function `compile`.

The function `step` returns non-zero if some substring of *string* matches the regular expression in *expbuf* and zero if there is no match. If there is a match, two external character pointers are set as a side effect to the call to `step`. The variable `loc1` points to the first character that matched the regular expression; the variable `loc2` points to the character after the last character that matches the regular expression. Thus if the regular expression matches the entire input string, `loc1` will point to the first character of *string* and `loc2` will point to the null at the end of *string*.

The function `advance` returns non-zero if the initial substring of *string* matches the regular expression in *expbuf*. If there is a match, an external character pointer, `loc2`, is set as a side effect. The variable `loc2` points to the next character in *string* after the last character that matched.

When `advance` encounters a `*` or `\{ \}` sequence in the regular expression, it will advance its pointer to the string to be matched as far as possible and will recursively call itself trying to match the rest of the string to the rest of the regular expression. As long as there is no match, `advance` will back up along the string until it finds a match or reaches the point in the string that initially matched the `*` or `\{ \}`. It is sometimes desirable to stop this backing up before the initial point in the string is reached. If the external character pointer `locs` is equal to the point in the string at sometime during the backing up process, `advance` will break out of the loop that backs up and will return zero.

The external variables `circf`, `sed`, and `nbra` are reserved.

DIAGNOSTICS

The function `compile` uses the macro `RETURN` on success and the macro `ERROR` on failure (see above). The functions `step` and `advance` return non-zero on a successful match and zero if there is no match. Errors are:

11 range endpoint too large.
 16 bad number.
 25 \ *digit* out of range.
 36 illegal or missing delimiter.
 41 no remembered search string.
 42 \(\) imbalance.
 43 too many \(.
 44 more than 2 numbers given in \[\].
 45] expected after \.
 46 first number exceeds second in \[\].
 49 [] imbalance.
 50 regular expression overflow.

EXAMPLE

The following is an example of how the regular expression macros and calls might be defined by an application program:

```

#define INIT          register char *sp = instring;
#define GETC          (*sp++)
#define PEEKC         (*sp)
#define UNGETC(c)     (--sp)
#define RETURN(*c)    return;
#define ERROR(c)      regerr

#include <regexp.h>

. . .
(void) compile(*argv, expbuf, &expbuf[ESIZE], '\0');
. . .
if (step(linebuf, expbuf))
    succeed;
  
```

SEE ALSO

regcmp(1), regcmp(3X).

NAME

sde – software development environment

DESCRIPTION

A *software development environment* (SDE) is a set of tools, libraries and system definitions that are specifically designed to work together to build an application that has certain qualities.

The environments provided in the DG/UX 5.4 release are:

m88kdguxelf	Used to create ELF objects and executables that make use of full DG/UX 5.4 release features.
m88kocs	Used for creating COFF objects and executables that can be linked and run on other vendors' 88open OCS- (and BCS-) conforming platforms.
m88kbcsc	Differs from the m88kocs because it allows the use of certain features (such as Berkeley signals) and optimizations (such as the macro implementation of <code>getc</code>) that are prohibited from the OCS environment. (This is unchanged from the DG/UX 4.3x release.)
m88kdguxcoff	Used to create COFF objects and executables that make use of DG/UX 4.3x level features. This option is interesting to software developers who have COFF-dependent tools, such as third-party debuggers, that they want to use on the DG/UX 5.4 release. (This is the same as m88kdgux on 4.3x.)
m88kdgux	The default for all past and future revisions. It refers to the largest feature set supported by the DG/UX system. In the DG/UX 5.4 release this is equal to m88kdguxelf.

The following table shows the domain of certain standards across the different environments. "Yes" means the environment conforms to that standard.

	BCS	OCS	POSIX	SVID/2	SVID/3	XPG/3	ANSI C
m88kdguxelf	No	No	Yes	No	Yes	Yes	Yes
m88kocs	Yes	Yes	Yes	Yes	No	No	Yes
m88kbcsc	Yes	No	Yes	Yes	No	No	Yes
m88kdguxcoff	No	No	Yes	Yes	No	No	Yes

Support for multiple development environments is handled by the `sde-target(1)` mechanism. It allows you to specify the development environment that is appropriate for your needs, while other users (or you in another context) may be using a different development environment at the same time. You select your environment by setting the environment variable `TARGET_BINARY_INTERFACE` to one of the environment names listed above. The command `sde-target(1)` provides a convenient way to set that variable. (Note that the variable name has changed from `SDE_TARGET` in the DG/UX 4.3x release. The name was changed because additional variables that control the "sde target" in ways other than the binary interface are likely to be introduced in the future. The `sde-target` command will not change, but it might set multiple variables in the future.)

The environment variable set by `sde-target(1)` is used in two contexts. When you invoke a software development tool such as `/bin/cc` or `/bin/ld`, you are actually calling a small program that calls `sde-chooser(1)`, which checks the environment variable and invokes the appropriate target-specific tool. Secondly, tools that read libraries, such as `ld(1)`, use the `elink(5)` mechanism, which uses the environment

variable to find the appropriate system libraries.

The commands, libraries, and other files that support a specific environment are placed in the directory `/usr/sde/<s>`, where `<s>` is the value of the environment variable `TARGET_BINARY_INTERFACE`. If `TARGET_BINARY_INTERFACE` is not set, the default (`m88kdgux`) is used.

Different environments need different header information at compile time. The DG/UX system has one set of include files that are customized by the use of conditional preprocessing under the control of target-specific macro names. The C compiler commands `cc(1)`, `gcc(1)`, and `ghcc(1)` predefine the following macro names according to the value of `TARGET_BINARY_INTERFACE`. (If you use another C compiler, you will need to do this manually with a `-D` option.)

sde target	Target Macro Name
<code>m88kdguxelf</code>	<code>_DGUX_TARGET</code>
<code>m88kocs</code>	<code>_M88KOCS_TARGET</code>
<code>m88kbc</code>	<code>_M88KBCS_TARGET</code>
<code>m88kdguxcoff</code>	<code>_DGUXCOFF_TARGET</code>

The above mechanism using `sde-chooser` and `elinks` was chosen over a more “traditional” method of using the `PATH` environment variable to find the right tools because many sources that people maintain, such as `make` files and shell scripts, contain fully specified path names. Such references would ignore the path specification and perhaps invoke the wrong tool or read the wrong library.

SEE ALSO

`sde-target(1)`, `sde-chooser(1)`, `sdetab(4)`, `elink(5)`.

NAME

siginfo – signal generation information

SYNOPSIS

```
#include <signal.h>
```

or

```
#include <sys/siginfo.h>
```

DESCRIPTION

When a process has caught a signal, it may have access to additional information describing why the system generated the signal. This information may be passed as an argument to the invoked signal handler, depending upon the system call that was used to establish the handler and current software development environment [see `sde-target(1)`].

There are two versions of the signal information structure, `struct siginfo`, which is defined in `signal.h` and `siginfo_t` which is defined in `sys/signal.h`.

If an application is compiled with the software development environment set to `m88kdgux` or `m88kdguxelf`, then the `siginfo_t` style of signal information may be reported to the signal handler. This information will be reported if `sigaction(2)` was used to establish the signal handler and the `SA_SIGINFO` bit is set in the signal action structure at the time the call was made [see `sigaction(2)`]. In this case, an object of type `siginfo_t` will be passed as the second argument to the invoking signal handler, and also an object of type `ucontext_t` will be sent as the third argument [see `sys/ucontext.h`]. The fields of the `siginfo_t` structure are discussed in more detail below.

If, however, an application is compiled with the software development environment set to `m88kbc`, `m88kocs`, or `m88kcoff` then the first argument available to the handler will be the caught signal number and an object of type `struct siginfo` will always be available as the second argument.

In addition, if a process is monitoring its children, it may receive information that tells why a child changed state [see `waitid(2)`]. In either case, the system returns the information in a structure of type `siginfo_t`, which includes the following information:

```
int si_signo    /* signal number */
int si_errno    /* error number */
int si_code     /* signal code */
```

`si_signo` contains the system-generated signal number. (For the `waitid(2)` function, `si_signo` is always `SIGCHLD`.)

If `si_errno` is non-zero, it contains an error number associated with this signal, as defined in `errno.h`.

`si_code` contains a code identifying the cause of the signal. If the value of `si_code` is less than or equal to 0, then the signal was generated by a user process [see `kill(2)` and `sigsend(2)`] and the `siginfo` structure contains the following additional information:

```
pid_t si_pid    /* sending process ID */
uid_t si_uid    /* sending user ID */
```

Otherwise, `si_code` contains a signal-specific reason why the signal was generated, as follows:

Signal	Code	Reason
SIGILL	ILL_ILLOPC	illegal opcode
	ILL_ILLOPN	illegal operation number
	ILL_ILLADR	illegal address
	ILL_ILLTRP	illegal trap
	ILL_PRVOPC	privileged opcode
	ILL_PRVREG	privileged register
	ILL_COPROC	
	ILL_BADSTK	bad stack
SIGFPE	FPE_INTDIV	integer divide by zero
	FPE_INTOVF	integer overflow
	FPE_FLTDIV	floating point divide by zero
	FPE_FLTOVF	floating point overflow
	FPE_FLTUND	floating point underflow
	FPE_FLTRES	floating point inexact result
	FPE_FLTINV	invalid floating point operation
	FPE_FLTSUB	subscript out of range
SIGSEGV	SEGV_MAPERR	address not mapped to object
	SEGV_ACCERR	invalid permissions for mapped object
SIGBUS	BUS_ADRALN	invalid address alignment
	BUS_ADRERR	address error
	BUS_OBJERR	object error
SIGTRAP	TRAP_BRKPT	process breakpoint
	TRAP_TRACE	process trace trap
SIGCHLD	CLD_EXITED	child has exited
	CLD_KILLED	child was killed
	CLD_DUMPED	child terminated abnormally
	CLD_TRAPPED	traced child has trapped
	CLD_STOPPED	child has stopped
	CLD_CONTINUED	stopped child had continued
SIGPOLL	POLL_IN	data input available
	POLL_OUT	output buffers available
	POLL_MSG	input message available
	POLL_ERR	I/O error
	POLL_PRI	high priority input available
	POLL_HUP	device disconnected

In addition, the following signal-dependent information is available for kernel-generated signals:

Signal	Field	Value	Comment
SIGILL SIGFPE	caddr_t si_addr		address of faulting instruction
SIGSEGV SIGBUS	caddr_t si_addr		address of faulting memory reference
SIGCHLD	pid_t si_pid int si_status		child process ID exit value or signal
SIGPOLL	long si_band		band event for POLL_IN, POLL_OUT, or POLL_MSG

SEE ALSO

sde-target(1), sigaction(2), waitid(2), signal(5).

NOTES

For SIGCHLD signals, if si_code is equal to CLD_EXITED, then si_status is equal to the exit value of the process; otherwise, it is equal to the signal that caused the process to change state.

NAME

signal - base signals

SYNOPSIS

```
#include <signal.h>
```

DESCRIPTION

A signal is an asynchronous notification of an event. A signal is said to be generated for a process when the event associated with that signal first occurs. Examples of such events include hardware faults, timer expiration and terminal activity, as well as the invocation of the `kill`, `sigsend`, or `sigsendset` system calls. In some circumstances, the same event generates signals for multiple processes. A process may request a detailed notification of the source of the signal and the reason why it was generated [see `siginfo(5)`].

A signal is said to be delivered to a process when the appropriate action for the process and signal is taken. During the time between the generation of a signal and its delivery, the signal is said to be pending [see `sigpending(2)`]. Ordinarily, this interval cannot be detected by an application. However, a signal can be blocked from delivery to a process [see `signal(2)` and `sigprocmask(2)`]. If the action associated with a blocked signal is anything other than to ignore the signal, and if that signal is generated for the process, the signal remains pending until either it is unblocked or the signal's disposition requests that the signal be ignored. If the signal disposition of a blocked signal requests that the signal be ignored, and if that signal is generated for the process, the signal is discarded immediately upon generation.

Each process may specify a system action to be taken in response to each signal delivered to it, called the signal's disposition. The set of system signal actions for a process is initialized from that of its parent. Once an action is installed for a specific signal, it usually remains installed until another disposition is explicitly requested by a call to either `sigaction` or any of its associated calls or until the process execs [see `sigaction(2)` and `signal(2)`]. When a process execs, all signals whose dispositions have been set to catch the signal will be set to `SIG_DFL`. Alternatively, a process may request that the system automatically reset the disposition of a signal to `SIG_DFL` after it has been caught [see `sigaction(2)`].

Each process has a signal mask that defines the set of signals currently blocked from delivery to it [see `sigprocmask(2)`]. The signal mask for a process is initialized from that of its parent.

The determination of which action is taken in response to a signal is made at the time the signal is delivered, allowing for any changes since the time of generation. This determination is independent of the means by which the signal was originally generated.

For a list of the signals supported by DG/UX, see `<sys/signal.h>`.

SEE ALSO

`kill(2)`, `pause(2)`, `sigaction(2)`, `sigset(2)`, `sigaltstack(2)`, `signal(2)`, `sigprocmask(2)`, `sigsend(2)`, `sigsuspend(2)`, `wait(2)`, `psiginfo(3C)`, `psignal(3C)`, `sigsetops(3C)`, `siginfo(5)`, `ucontext(5)`.

NAME

stat - data returned by stat system call

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
```

DESCRIPTION

The system calls `stat`, `fstat`, `lstat`, and `dg_mstat` return data whose structure is defined by this include file. The encoding of the field `st_mode` is also defined in this file.

```
/*
 * Structure of the result of stat
 */

struct stat
{
    dev_t          st_dev;
    ino_t          st_ino;
    mode_t        st_mode;
    nlink_t       st_nlink;
    uid_t         st_uid;
    gid_t         st_gid;
    dev_t         st_rdev;
    off_t         st_size;
    time_t        st_atime;
    unsigned long st_ausec;
    time_t        st_mtime;
    unsigned long st_musec;
    time_t        st_ctime;
    unsigned long st_cusec;
    timestruc_t   st_atim;
    timestruc_t   st_mtim;
    timestruc_t   st_ctim;
    long          st_blksize;
    long          st_blocks;
    char          st_fstype[16];
    char          st_pad5[408];
};

#define S_IFMT      0170000 /* type of file */
#define S_IFDIR    0040000 /* directory */
#define S_IFCHR    0020000 /* character special */
#define S_IFBLK    0060000 /* block special */
#define S_IFREG    0100000 /* regular */
#define S_IFLNK    0120000 /* symbolic link */
#define S_IFIFO    0010000 /* fifo */
#define S_IFSOCK   0140000 /* socket special file */
#define S_ISUID    04000   /* set user id on execution */
#define S_ISGID    02000   /* set group id on execution */
#define S_ISVTX    01000   /* save swapped text even after use */
#define S_IRREAD   00400   /* read permission, owner */
#define S_IWRITE   00200   /* write permission, owner */
```

```
#define S_IEXEC 00100 /* execute/search permission, owner */
#define S_ENFMT 02000 /* record locking enforcement flag */
#define S_IRWXU 00700 /* read, write, execute search
                        permission, owner */
#define S_IRUSR 00400 /* read permission, owner */
#define S_IWUSR 00200 /* write permission, owner */
#define S_IXUSR 00100 /* execute/search permission, owner */
#define S_IRWXG 00070 /* read, write, execute/search
                        permission, group */
#define S_IRGRP 00040 /* read permission, group */
#define S_IWGRP 00020 /* write permission, group */
#define S_IXGRP 00010 /* execute/search permission, group */
#define S_IRWXO 00007 /* read, write, execute/search
                        permission, other */
#define S_IROTH 00004 /* read permission, other */
#define S_IWOTH 00002 /* write permission, other */
#define S_IXOTH 00001 /* execute/search permission, other */
```

FILES

```
/usr/include/sys/stat.h
/usr/include/sys/types.h
```

SEE ALSO

stat(2), types(5).

NAME

`statfs` – data returned by the `statfs` system call

DESCRIPTION

The system call `statfs` takes a parameter that is a pointer to the structure defined by this include file. This structure returns file system device statistics.

```

struct statfs
{
    short    f_fstyp;
    long     f_bsize;
    long     f_frsize;
    long     f_blocks;
    long     f_bfree;
    long     f_bavail;
    long     f_files;
    long     f_ffree;
    char     f_fname [6];
    char     f_fpack [6];
    long     f_favail;
    long     fs_blocks;
    long     fs_bfree;
    long     fs_bavail;
    long     fs_files;
    long     fs_ffree;
    long     fs_favail;
};

```

The fields of this structure are defined as follows:

- `f_fstyp` The type of the file system.
- `f_bsize` The file system block size, in bytes.
- `f_frsize` The file system fragment size, in bytes.
- `f_blocks` The maximum number of blocks that may exist in the control-point directory containing the pathname passed to `statfs`, taking into account the block limits of all CPDs on the path. If the pathname is a CPD, its own block limit is also taken into account. If the pathname is the root of a file system, this field is the maximum that applies to superusers, so it is the same as `fs_blocks`. If the pathname is not a file system root, the maximum applies to both superusers and non-superusers.
- `f_bfree` The number of free blocks in the control-point directory containing the pathname passed to `statfs`, taking into account the block limits of all CPDs on the path. If the pathname is a CPD, its own block limit is also taken into account. If the pathname is the root of a file system, this field is the number of blocks that can still be allocated by superusers, so it is the same as `fs_bfree`. If the pathname is not a file system root, the free count applies to both superusers and non-superusers.
- `f_bavail` This field is the same as `f_bfree` unless the pathname is the root of a file system. In that case it gives the number of blocks that can still be allocated by non-superusers.
- `f_files` The total number of files that may exist in the control-point directory containing the pathname passed to `statfs`, i.e. the number allocated plus

the number that still may be created, taking into account the file limits of all CPDs on the path. If the pathname is a CPD, its own file limit is also taken into account. If the pathname is the root of a file system, this field is the maximum that applies to superusers, so it is the same as *fs_files*. If the pathname is not a file system root, the maximum applies to both superusers and non-superusers.

- f_ffree** The number of files that still may be created in the control-point directory containing the pathname passed to *statfs*, taking into account the files limits of all CPDs on the path. If the pathname is a CPD, its own file limit is also taken into account. If the pathname is the root of a file system, this field is the number of files that can still be created by superusers, so it is the same as *fs_ffree*. If the pathname is not a file system root, the file count applies to both superusers and non-superusers.
- f_fname** The file system name. This field will be null unless a label has been added to the file system with *labelit*.
- f_fpack** The file system pack name. This field will be null unless a label has been added to the file system with *labelit*.
- f_favail** This field is the same as *f_ffree*.
- fs_blocks** The file system size, in blocks.
- fs_bfree** The total number of free blocks on the file system.
- fs_bavail** The number of free blocks on the file system available to non-superusers.
- fs_files** The total number of files that may exist on the file system, i.e. the number allocated plus the number that still may be created.
- fs_ffree** The number of files that still may be created on the file system.
- fs_favail** The number of files that still may be created on the file system by non-superusers.

FILES

`/usr/include/sys/statfs.h`

SEE ALSO

`statfs(2)`.

NAME

stdarg – handle variable argument list

SYNOPSIS

```
#include <stdarg.h>

va_list pvar;

void va_start(va_list pvar, parmN);

type va_arg(va_list pvar, type);

void va_end(va_list pvar);
```

DESCRIPTION

This set of macros allows portable procedures that accept variable numbers of arguments of variable types to be written. Routines that have variable argument lists [such as `printf`] but do not use *stdarg* are inherently non-portable, as different machines use different argument-passing conventions.

`va_list` is a type defined for the variable used to traverse the list.

The `va_start()` macro is invoked before any access to the unnamed arguments and initializes `pvar` for subsequent use by `va_arg()` and `va_end()`. The parameter *parmN* is the identifier of the rightmost parameter in the variable parameter list in the function definition (the one just before the `, ...`). If this parameter is declared with the `register` storage class or with a function or array type, or with a type that is not compatible with the type that results after application of the default argument promotions, the behavior is undefined.

The `va_arg()` macro expands to an expression that has the type and value of the next argument in the call. The parameter `pvar` should have been previously initialized by `va_start()`. Each invocation of `va_arg()` modifies `pvar` so that the values of successive arguments are returned in turn. The parameter *type* is the type name of the next argument to be returned. The type name must be specified in such a way so that the type of a pointer to an object that has the specified type can be obtained simply by postfixing a `*` to *type*. If there is no actual next argument, or if *type* is not compatible with the type of the actual next argument (as promoted according to the default argument promotions), the behavior is undefined.

The `va_end()` macro is used to clean up.

Multiple traversals, each bracketed by `va_start` and `va_end`, are possible.

EXAMPLE

This example gathers into an array a list of arguments that are pointers to strings (but not more than `MAXARGS` arguments) with function `f1`, then passes the array as a single argument to function `f2`. The number of pointers is specified by the first argument to `f1`.

```
#include <stdarg.h>
#define MAXARGS 31

void f1(int n_ptrs, ...)
{
    va_list ap;
    char *array[MAXARGS];
    int ptr_no = 0;

    if (n_ptrs > MAXARGS)
        n_ptrs = MAXARGS;
    va_start(ap, n_ptrs);
    while (ptr_no < n_ptrs)
        array[ptr_no++] = va_arg(ap, char*);
    va_end(ap);
    f2(n_ptrs, array);
}
```

Each call to `f1` shall have visible the definition of the function or a declaration such as

```
void f1(int, ...)
```

SEE ALSO

`vprintf(3S)`.

NOTES

It is up to the calling routine to specify in some manner how many arguments there are, since it is not always possible to determine the number of arguments from the stack frame. For example, `execl` is passed a zero pointer to signal the end of the list. `printf` can tell how many arguments there are by the format. It is non-portable to specify a second argument of `char`, `short`, or `float` to `va_arg`, because arguments seen by the called function are not `char`, `short`, or `float`. `C` converts `char` and `short` arguments to `int` and converts `float` arguments to `double` before passing them to a function.

NAME

syslog.conf – configuration file for syslogd system log server

SYNOPSIS

/etc/syslog.conf

DESCRIPTION

The file /etc/syslog.conf contains information used by the system log server (daemon), syslogd(1M), to forward a system message to appropriate log files and/or users.

A configuration entry is composed of two TAB-separated fields:

selector *action*

The *selector* field contains a semicolon-separated list of priority specifications of the form:

facility.level[;facility.level]

where *facility* is a system facility, or comma-separated list of facilities, and *level* is an indication of the severity of the condition being logged. Recognized values for *facility* include:

user	Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.
kern	Messages generated by the kernel.
mail	Reserved for the mail system.
daemon	System servers, such as ftpd(1M).
auth	Reserved for the auth system; it does not currently use the syslog mechanism.
lpr	Messages generated by the lpr/lpd line printer spooling system.
news	Reserved for the USENET network news system.
uucp	Reserved for the UUCP system; it does not currently use the syslog mechanism.
cron	Reserved for the cron system; it does not currently use the syslog mechanism.
local0-7	Reserved for local use.
mark	For timestamp messages produced internally by syslogd.
*	An asterisk indicates all facilities except for the mark facility.

Recognized values for *level* are (in descending order of severity):

emerg	For panic conditions that would normally be broadcast to all users.
alert	For conditions that should be corrected immediately, such as a corrupted system database.
crit	For warnings about critical conditions, such as hard device errors.
err	For other errors.
warning	For warning messages.
notice	For conditions that are not error conditions, but may require special handling.

info Informational messages.
debug For messages that are normally used only when debugging a program.
none Do not send messages from the indicated *facility* to the selected file. For example, a *selector* of
 `*.debug;mail.none`
 will send all messages *except* mail messages to the selected file.

The *action* field indicates where to forward the message. Values for this field can have one of four forms:

- A filename, beginning with a leading slash, which indicates that messages specified by the *selector* are to be written to the specified file. The file will be opened in append mode.
- The name of a remote host, prefixed with an @, as with: @server, which indicates that messages specified by the *selector* are to be forwarded to the syslogd on the named host.
- A comma-separated list of usernames, which indicates that messages specified by the *selector* are to be written to the named users if they are logged in.
- An asterisk, which indicates that messages specified by the *selector* are to be written to all logged-in users.

Blank lines are ignored. Lines for which the first nonwhite character is a '#' are treated as comments.

EXAMPLE

With the following configuration file:

```

*.notice;mail.info    /usr/adm/notice
*.crit                /usr/adm/critical
kern,mark.debug      /dev/console
kern.err              @server
*.emerg               *
*.alert               root,operator
*.alert;auth.warning /usr/adm/auth

```

syslogd will log all mail system messages except debug messages and all notice (or higher) messages into a file named /usr/adm/notice. It logs all critical messages into /usr/adm/critical, and all kernel messages and 20-minute marks onto the system console.

Kernel messages of err (error) severity or higher are forwarded to the machine named server. Emergency messages are forwarded to all users. The users root and operator are informed of any alert messages. All messages from the authorization system of warning level or higher are logged in the file /usr/adm/auth.

SEE ALSO

logger(1), syslogd(1M), syslog(3C).

NAME

tar – tape archive file format

DESCRIPTION

tar (the tape archive command) dumps several files into one, in a medium suitable for transportation.

A “tar tape” or file is a series of blocks. Each block is of size TBLOCK. A file on the tape is represented by a header block which describes the file, followed by zero or more blocks which give the contents of the file. At the end of the tape are two blocks filled with binary zeros, as an end-of-file indicator.

The blocks are grouped for physical I/O operations. Each group of *n* blocks (where *n* is set by the *b* keyletter on the tar(1) command line — default is 32 blocks) is written with a single system call; on nine-track tapes, the result of this write is a single tape record. The last group is always written at the full size, so blocks after the two zero blocks contain random data. On reading, the specified or default group size is used for the first read, but if that read returns less than a full tape block, the reduced block size is used for further reads.

The header block looks like:

```
#define TBLOCK 512      /* length of tar header and data blocks */
#define TNAMLEN 100    /* maximum length for tar file names */
#define TMODLEN 8      /* length of mode field */
#define TUIDLEN 8      /* length of uid field */
#define TGIDLEN 8      /* length of gid field */
#define TSIZLEN 12     /* length of size field */
#define TTIMLEN 12     /* length of modification time field */
#define TCRLEN 8       /* length of header checksum field */

union tblock {
    char dummy[TBLOCK];
    struct tar_hdr {
        char t_name[TNAMLEN],      /* name of file */
            t_mode[TMODLEN],      /* mode of file */
            t_uid[TUIDLEN],       /* uid of file */
            t_gid[TGIDLEN],       /* gid of file */
            t_size[TSIZLEN],      /* size of file in bytes */
            t_mtime[TTIMLEN],     /* modification time of file */
            t_cksum[TCRLEN],      /* checksum of header */
            t_typeflag,
            t_linkname[TNAMLEN], /* file this file linked with */
            t_magic[TMAGLEN],
            t_version[TVERSLEN],
            t_uname[32],
            t_gname[32],
            t_devmajor[8],
            t_devminor[8],
            t_prefix[155];
    } tbuf;
};
```

The fields `t_magic`, `t_uname`, and `t_gname` are null-terminated strings. The fields `t_name`, `t_linkname`, and `t_prefix` are null-terminated except when all characters in the field, including the last character, are used for the name.

The `t_name` and `t_prefix` fields are used to construct the pathname of the file. If the `t_prefix` field contains non-null characters, a pathname is formed by concatenating the `t_prefix` field, a slash character, and the `t_name` field; otherwise, the pathname is formed using only the value in the `t_name` field.

`T_mode` is the file mode, with the top bit masked off.

`Uid` and `gid` are the user and group numbers for the file.

`T_size` is the size of the file in bytes (or the size of the current extension of the file if the file has been split over multiple volumes). Links, symbolic links, directories, and device files are dumped with this field specified as zero.

`T_mtime` is the modification time of the file at the time it was dumped.

`T_chksum` is an octal ASCII value which represents the sum of all the bytes in the header block. When calculating the checksum, the `t_chksum` field is treated as if it were all blanks.

`T_typeflag` is a one-character field which specifies the type of the file. The valid values for `t_typeflag` are:

```

null   Regular file (supplied for backward compatibility)
`0'   Regular file
`1'   Link
`2'   Symbolic link
`3'   Character special
`4'   Block special
`5'   Directory
`6'   FIFO special
`7'   Reserved

```

If `typeflag` is ASCII '1' (hard link) or ASCII '2' (symbolic link), the name linked-to, is in `t_linkname`, with a trailing null. The `t_linkname` field does not use the `t_prefix`; hence, linknames are limited to 99 characters.

`T_magic` indicates that the archive was output in this archive format. If the `t_magic` field contains the value `TMAGIC` (defined above), then the `t_uname` and `t_gname` fields contain the ASCII names of the owner and group, respectively, for the file. If necessary, the owner and group names will be truncated to fit in these fields.

`T_version` should contain the value `TVERSION` (defined above).

`T_devmajor` and `T_devminor` contain the major and minor device codes, respectively, for device files and are meaningful only if `T_typeflag` is ASCII '3' (character special) or ASCII '4' (block special).

The fields `t_extno`, `exttotal` and `efsize` are used for files which are split over multiple volumes. The extensions (pieces) of the file are labeled separately on each volume and assigned sequential extension numbers. `t_extno` contains the extension number of the current extension and is null if the file is not split; `t_exttotal` contains the total number of extensions for the file; and `t_efsize` contains the total size of the file.

Unused fields of the header are set to binary zeros and are included in the checksum.

The first time a given i-node number is dumped, it is dumped as a regular file. The second and subsequent times, it is dumped as a link instead. Upon retrieval, if a link entry is retrieved, but not the file it was linked to, an error message is printed and the tape must be manually re-scanned to retrieve the linked-to file.

The encoding of the header is designed to be portable across machines.

SEE ALSO

tar(1).

NOTE

Linknames longer than NAMSIZ produce error reports and cannot be dumped.

NAME

termcap – terminal capability data base

DESCRIPTION

Termcap is a data base of terminal descriptions used by the `termcap(3X)` library. All terminals are described in a file called `/etc/termcap`. Termcap entries describe, in special code, how basic operations are performed on a terminal. They also describe padding requirements, initialization sequences, and so on. The section entitled "Preparing a Termcap Description" that appears later explains how to build a termcap source description.

Entries in Termcap consist of a number of ':'-separated fields. The first line names the terminal, and the remaining lines describe its capabilities.

Terminal Names

The first line of for each terminal description gives the names that are known for the terminal, separated by vertical bar (|) characters. The first name is always two characters long for compatibility with older systems which store the terminal type in a 16-bit word in a system-wide data base. The second name is the most common abbreviation for the terminal, the last name should be a long name fully identifying the terminal, and all others are understood as synonyms for the terminal name. All names but the first and last should be in lower case and contain no blanks; the last name may well contain upper case letters and blanks for readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions. First, the vendor and model of the terminal should be specified in the root name, for example, `hp2621`. This name should not contain hyphens. Terminal modes or user preferences should be indicated by appending a hyphen and an indicator of the mode. Therefore, a `vt100` in 132-column mode would be `vt100-w`. The following suffixes should be used where possible:

Suffix	Meaning	Example
-w	Wide mode (more than 80 columns)	vt100-w
-am	With automatic margins (usually default)	vt100-am
-nam	Without automatic margins	vt100-nam
-n	Number of lines on the screen	aaa-60
-na	No arrow keys (leave them in local mode)	concept100-na
-np	Number of pages of memory	concept100-4p
-rv	Reverse video	concept100-rv

Terminal Capabilities

Lines after the first line of a terminal description describe the terminal's capabilities. Capabilities in termcap are of three general types: Boolean capabilities, which indicate a terminal's particular features; numeric capabilities, which give the size of the display or other attributes; and string capabilities, which give character sequences that can be used to perform particular terminal operations.

The table below lists termcap capabilities alphabetically by name. The second field of the table indicates capability type. The characters in the Notes field in the table have the following meanings (more than one may apply to a capability):

- N indicates numeric parameter(s)
- P indicates that padding may be specified
- * indicates that padding may be based on the number of lines affected
- o indicates that the capability is obsolete

“Obsolete” capabilities have no `terminfo(4)` equivalents; either they were considered useless, or they have been subsumed by other capabilities. New software should not rely on them at all. The last field in the table gives a short description of the terminal capability.

Name	Type	Notes	Description
ae	str	(P)	End alternate character set mode
AL	str	(NP*)	Add <i>n</i> new blank lines
al	str	(P*)	Add one new blank line
am	bool		Terminal has automatic margins
as	str	(P)	Start alternate character set mode
bc	str	(o)	Backspace if not \sim H
bl	str	(P)	Audible signal (bell)
bs	bool	(o)	Terminal can backspace with \sim H
bt	str	(P)	Back tab
bw	bool		1e (backspace) wraps from column 0 to last column
CC	str		Terminal settable command character in prototype
cd	str	(P*)	Clear to end of display
ce	str	(P)	Clear to end of line
ch	str	(NP)	Set cursor column (horizontal position)
cl	str	(P*)	Clear screen and home cursor
CM	str	(NP)	Memory-relative cursor addressing (motion)
cm	str	(NP)	Screen-relative cursor addressing (motion)
co	num		Number of columns in a line
cr	str	(P)	Carriage return
cs	str	(NP)	Change scrolling region (VT100)
ct	str	(P)	Clear all tab stops
cv	str	(NP)	Set cursor row (vertical position)
da	bool		Display may be retained above screen
dB	num	(o)	Milliseconds of bs delay needed (default 0)
db	bool		Display may be retained below screen
DC	str	(NP*)	Delete <i>n</i> characters
dC	num	(o)	Milliseconds of cr delay needed (default 0)
dc	str	(P*)	Delete one character
dF	num	(o)	Milliseconds of ff delay needed (default 0)
DL	str	(NP*)	Delete <i>n</i> lines
dL	str	(P*)	Delete one line
dm	str		Enter delete mode
dN	num	(o)	Milliseconds of nL delay needed (default 0)
DO	str	(NP*)	Move cursor down <i>n</i> lines
do	str		Move cursor down one line
ds	str		Disable status line
dT	num	(o)	Milliseconds of horizontal tab delay needed (default 0)
dV	num	(o)	Milliseconds of vertical tab delay needed (default 0)
ec	str	(NP)	Erase <i>n</i> characters
ed	str		End delete mode
ei	str		End insert mode
eo	bool		Terminal can erase overstrikes with a blank
EP	bool	(o)	Terminal uses even parity
es	bool		Escape sequences can be used on status line
ff	str	(P*)	Hardcopy terminal page eject
fs	str		Return from status line
gn	bool		Generic line type (<i>e.g.</i> dialup, switch)

hc	bool		Hardcopy terminal
HD	bool	(o)	Half-duplex
hd	str		Move a half-line down (forward 1/2 linefeed)
ho	str	(P)	Home cursor
hs	bool		Terminal has extra "status line"
hu	str		Move a half-line up (reverse 1/2 linefeed)
hz	bool		Terminal cannot print tildes (Hazeltine)
IC	str	(NP*)	Insert <i>n</i> blank characters
ic	str	(P*)	Insert one blank character
if	str		Name of file containing initialization string
im	str		Enter insert mode
in	bool		Insert mode distinguishes nulls
ip	str	(P*)	Insert padding after character inserted
is	str		Terminal initialization string
it	num		Tabs are initially every <i>n</i> positions
K1	str		Sent by keypad upper left key
K2	str		Sent by keypad upper right key
K3	str		Sent by keypad center key
K4	str		Sent by keypad lower left key
K5	str		Sent by keypad lower right key
k0-k9	str		Sent by function keys 0-9
kA	str		Sent by insert-line key
ka	str		Sent by clear-all-tabs key
kb	str		Sent by backspace key
kC	str		Sent by clear-screen or erase key
kD	str		Sent by delete-character key
kd	str		Sent by down-arrow key
kE	str		Sent by clear-to-end-of-line key
ke	str		Out of "keypad transmit" mode
kF	str		Sent by scroll-forward/down key
kH	str		Sent by home-down key
kh	str		Sent by home key
kI	str		Sent by insert-character or enter-insert-mode key
kL	str		Sent by delete-line key
kL	str		Sent by left-arrow key
kM	str		Sent by insert key while in insert mode
km	bool		Terminal has a "meta" key (sets eighth bit)
kN	str		Sent by next-page key
kn	num	(o)	Number of function (k0-k9) keys (default 0)
ko	str	(o)	Termcap entries for other non-function keys
kP	str		Sent by previous-page key
kR	str		Sent by scroll-backward/up key
kr	str		Sent by right-arrow key
kS	str		Sent by clear-to-end-of-screen key
ks	str		Put terminal in "keypad transmit" mode
kT	str		Sent by set-tab key
kt	str		Sent by clear-tab key
ku	str		Sent by up-arrow key
l0-19	str		Labels on function keys if not "fn"
LC	bool	(o)	Terminal is lowercase only
LE	str	(NP)	Move cursor left <i>n</i> positions
le	str	(P)	Move cursor left one position
li	num		Number of lines on screen or page

ll	str		Move cursor to last line, first column
lm	num		Lines of memory if > li (0 means varies)
ma	str	(o)	Arrow key map
mb	str		Turn on blinking attribute
md	str		Turn on bold (extra bright) attribute
me	str		Turn off all attributes
mh	str		Turn on half-bright (dim) attribute
mi	bool		Safe to move while in insert mode
mk	str		Turn on blank attribute (characters invisible)
ml	str	(o)	Turn on memory lock above cursor
mm	str		Turn on "meta mode" (transmit eighth bit)
mo	str		Turn off "meta mode"
mp	str		Turn on protected attribute
mr	str		Turn on reverse-video attribute
ms	bool		Safe to move in standout modes
mu	str	(o)	Memory unlock (turn off memory lock)
nc	bool	(o)	No correctly-working cr (Datamedia 2500, Hazeltine 2000)
nd	str		Move cursor right one (non-destructive) space
NL	bool	(o)	\n is newline, not line feed
nl	str	(o)	Newline character if not \n
ns	bool	(o)	Terminal is a CRT but doesn't scroll
nw	str	(P)	Newline (behaves like cr followed by do)
OP	bool	(o)	Terminal uses odd parity
os	bool		Terminal overstrikes
pb	num		Lowest baud rate where delays are required
pc	str		Pad character (default NUL)
pf	str		Turn off printer
pO	str	(N)	Turn on printer for <i>n</i> bytes
po	str		Turn on printer
ps	str		Print contents of screen
pt	bool	(o)	Has hardware tabs (may need to be set with is)
rc	str	(P)	Restore cursor to position of last sc
rf	str		Name of file containing reset string
RI	str	(NP)	Move cursor right <i>n</i> positions
rp	str	(NP*)	Repeat character <i>c</i> <i>n</i> times
rs	str		Reset terminal completely to sane modes
sa	str	(NP)	Define video attributes
sc	str	(P)	Save cursor position
se	str		End standout mode
SF	str	(NP*)	Scroll forward (up) <i>n</i> lines
sf	str	(P)	Scroll forward (up) one line
sg	num		Number of garbage chars left by so or se (default 0)
so	str		Begin standout mode
SR	str	(NP*)	Scroll backward (down) <i>n</i> lines
sr	str	(P)	Scroll backward (down) one line
st	str		Set a tab in all rows, current column
ta	str	(P)	Tab to next hardware tab stop
tc	str		Entry of similar terminal – must be last entry
te	str		String to end programs that use termcap
ti	str		String to begin programs that use termcap
ts	str	(N)	Go to status line, column <i>n</i>
UC	bool	(o)	Terminal is uppercase only
uc	str		Underscore one character and move past it

ue	str		End underscore mode
ug	num		Number of garbage chars left by us or ue (default 0)
ul	bool		Underline character overstrikes
UP	str	(NP*)	Move cursor up <i>n</i> lines
up	str		Move cursor up one line
us	str		Start underscore mode
vb	str		Visible bell (must not move cursor)
ve	str		Make cursor appear normal (undo vs/vi)
vi	str		Make cursor invisible
vs	str		Make cursor very visible
vt	num		Virtual terminal number (not supported on all systems)
wi	str	(N)	Set current window
ws	num		Number of columns in status line
xb	bool		Beehive (f1=ESC, f2=^C)
xn	bool		Newline ignored after column 80 (Concept)
xo	bool		Terminal uses XOFF/XON (DC3/DC1) handshaking
xr	bool	(o)	Return acts like ce cr nl (Delta Data)
xs	bool		Standout not erased by overwriting (Hewlett-Packard)
xt	bool		Destructive tabs, magic so char (Telaray 1061)
xx	bool	(o)	Tektronix 4025 insert-line

PREPARING A TERMCAP DESCRIPTION

The most effective way to prepare a terminal description is by imitating the description of a similar terminal in `termcap` and building up your description gradually, using partial descriptions to check that they are correct.

To easily test a new terminal description, set the environment variable `TERMCAP` to the absolute pathname of a file containing the description you are working on and programs will look there rather than in `/etc/termcap`. `TERMCAP` can also be set to the `termcap` entry itself to avoid reading the file when starting up a program.

Be aware that a very unusual terminal may expose deficiencies in the ability of the `termcap` conventions to describe it.

Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability `tc` can be given with the name of the similar terminal. This capability must be specified last, and the combined length of the entries must not exceed 1024 characters. The capabilities given before `tc` override those in the terminal type included by `tc`. A capability can be canceled by placing `xx@` to the left of the `tc` invocation, where `xx` is the capability. For example, the entry

```
hn || 2621-nl:ks@:ke@:tc=2621:
```

defines a "2621-nl" that does not have the `ks` or `ke` capabilities, and hence does not turn on the function key labels when in visual mode. This is useful for different modes of a terminal, or for different user preferences.

Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with `printf(3S)`-like escapes `%x` in it, while other characters are passed through unchanged. The `%` encodings have the following meanings:

```
%%      output %
%d      output value as in printf(%d)
```


Numeric capabilities are followed by a pound sign (#) and then the value. On the third line of the example above, `co`, which indicates the number of columns in the display, gives the value "80" for the Concept.

Finally, string-valued capabilities, such as `ce` (the sequence to clear-to-end-of-line), are given by the two-letter code, an equals sign (=), then a string ending at the next following colon (:). A delay in milliseconds may appear after the = in such a capability, and causes padding characters to be supplied by `tputs(3X)` to provide this delay after the remainder of the string is sent. The delay can be either a number, for example, 20, or a number followed by an asterisk (*), for example, 3*. An * indicates that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-line padding required. (In the case of insert-character, the factor is still the number of lines affected; this is always 1 unless the terminal has `in` and the software uses it.) When an * is specified, it is sometimes useful to give a delay containing a decimal point, for example 3.5 to specify a delay per line to tenths of milliseconds. (Only one decimal place is allowed.)

A number of escape sequences are provided in the string-valued capabilities for easy encoding of control characters there. `\E` maps to an ESC character, `^X` maps to a control-*X* for any appropriate *X*, and the sequences `\n`, `\r`, `\t`, `\b`, and `\f` map to linefeed, return, tab, backspace, and formfeed, respectively. Finally, characters may be given as three octal digits after a `\`, and the characters `^` and `\` may be given as `\^` and `\\`. If it is necessary to place a `:` in a capability it must be escaped in octal as `\072`. If it is necessary to place a NUL character in a string capability it must be encoded as `\200`. (The routines that deal with `termcap` use C strings and strip the eighth bit of the output very late, so that a `\200` comes out as a `\000` would.)

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the first `cr` and `ta` in the preceding example.

TERMCAP TERMINAL CAPABILITIES

The following subsections describe `termcap` capabilities in detail.

Basic Capabilities

The number of columns on each line of the display is given by the `co` numeric capability. If the display is a CRT, then the number of lines on the screen is given by the `li` capability. If the cursor wraps around to the beginning of the next line when it reaches the right margin, then it should have the `am` capability. If the terminal can clear its screen, the code to do this is given by the `cl` string capability. If the terminal overstrikes (rather than clearing the position when a character is overwritten), it should have the `os` capability. If the terminal is a printing terminal, with no soft copy unit, give it both `hc` and `os`. (`os` applies to storage scope terminals, such as the Tektronix 4010 series, as well as to hard copy and APL terminals.) If there is a code to move the cursor to the left edge of the current row, give this as `cr`. (Normally this will be carriage-return, `^M`.) If there is a code to produce an audible signal (bell, beep, for example), give this as `b1`.

If there is a code (such as backspace) to move the cursor one position to the left, that capability should be given as `le`. Similarly, codes to move to the right, up, and down should be given as `nd`, `up`, and `do`, respectively. These local cursor motions should not alter the text they pass over; for example, you would not normally give `"nd= "` unless the terminal has the `os` capability, because the space would erase the character moved over.

A very important point here is that the local cursor motions encoded in `termcap` have undefined behavior at the left and top edges of a display. Programs should never attempt to backspace around the left edge, unless `bw` is given, and never attempt to move the cursor up off the top line using local cursor motions.

In order to scroll text up, a program moves the cursor to the bottom left corner of the screen and sends the `sf` (index) string. To scroll text down, a program moves the cursor to the top left corner of the screen and sends the `sr` (reverse index) string. The strings `sf` and `sr` have undefined behavior when the cursor is not on their respective corners of the screen. Parameterized versions of the scrolling sequences are `SF` and `SR`, which have the same semantics as `sf` and `sr` except that they take one parameter and scroll that many lines. They also have undefined behavior except at the appropriate corners of the screen.

The `am` capability tells whether the cursor sticks at the right edge of the screen when text is output there, but this does not necessarily apply to `nd` from the last column. Leftward local motion is defined from the left edge only when `bw` is given; then an `le` from the left edge will move to the right edge of the previous row. This is useful for drawing a box around the edge of the screen, for example. If the terminal has switch-selectable automatic margins, the `termcap` description usually assumes that this feature is on, that is, `am`. If the terminal has a command that moves to the first column of the next line, that command can be given as `nw` (newline). It is permissible for this to clear the remainder of the current line, so if the terminal has no correctly-working `CR` and `LF` it may still be possible to craft a working `nw` out of one or both of them.

These capabilities suffice to describe hardcopy and “glass-tty” terminals. Thus the Teletype model 33 is described as

```
T3|tty33|33|tty|Teletype model 33:\
:bl=^G:co#72:cr=^M:do=^J:hc:os:
```

and the Lear Siegler ADM-3 is described as

```
l3|adm3|3|LSI ADM-3:\
:am:bl=^G:cl=^Z:co#80:cr=^M:do=^J:le=^H:li#24:sf=^J:
```

Cursor Motions

If the terminal has a fast way to home the cursor (to the very upper left corner of the screen), this can be given as `ho`. Similarly, a fast way of getting to the lower left-hand corner can be given as `ll`; this may involve going up with `up` from the home position, but a program should never do this itself (unless `ll` does), because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as cursor address (0,0): to the top left corner of the screen, not of memory. (Therefore, the “\EH” (memory home) sequence on Hewlett-Packard terminals cannot be used for `ho`.)

To address the cursor (move it to an absolute position), the `cm` capability is given. `cm` takes two parameters: the row and column to move the cursor to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory. If the terminal has memory-relative cursor addressing, that can be indicated by an analogous `CM` boolean capability.)

Row or column absolute cursor addressing can be given as single parameter capabilities `ch` (horizontal position absolute) and `cv` (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to `cm`. If there are

parameterized local motions (for example, move n positions to the right) these can be given as `DO`, `LE`, `RI`, and `UP` with a single parameter indicating how many positions to move. These are primarily useful if the terminal does not have `cm`, as with the Tektronix 4025.

Area Clears

If the terminal can clear from the current cursor position to the end of the line, leaving the cursor where it is, this should be given as `ce`. If the terminal can clear from the current cursor position to the end of the display, this should be given as `cd`. Programs must output `cd` only from the first column of a line. (Therefore, it can be simulated by a request to delete a large number of lines, if a true `cd` is not available.)

Insert/Delete Line

If the terminal can open a new blank line before the line containing the cursor, this should be given as `a1`; programs must output this only from the first position of a line. The cursor must then appear at the left of the newly blank line. If the terminal can delete the line that the cursor is on, this should be given as `d1`; programs must output this only from the first position on the line to be deleted. Versions of `a1` and `d1` which take a single parameter and insert or delete that many lines can be given as `AL` and `DL`. If the terminal has a settable scrolling region (like the VT100), the command to set this can be described with the `cs` capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is undefined after using this command. The program must reset the cursor position using other `termcap` capabilities such as `cm` or `rc`. It is possible to get the effect of insert or delete line using this command — the `sc` and `rc` (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using `sr` or `sf` on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

If the terminal has the ability to define a window as part of memory which all commands affect, it should be given as the parameterized string `wi`. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above the screen, then the `da` capability should be given; if display memory can be retained below, then `db` should be given. These indicate that deleting a line or scrolling may bring non-blank lines up from below, or that scrolling back with `sr` may bring down non-blank lines.

Insert/Delete Character

There are two basic kinds of intelligent terminals with respect to insert/delete character that can be described using `termcap`. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept-100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen, and then typing text separated by cursor motions. Type `abc def` using local cursor motions (not spaces) between the `abc` and the `def`. Then position the cursor before the `abc` and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the `abc` shifts over to the `def` which then move together around the end of the current line and onto the next as you insert, then you have the second type of terminal and should give the capability `in`, which stands for “insert null”. While these are two logically separate attributes (one line versus multi-line

insert mode, and special treatment of untyped spaces), we have seen no terminals whose insert mode cannot be described with the single attribute.

Termcap can describe both terminals that have an insert mode and terminals that have a sequence to open a blank position on the current line. Give as `im` the sequence to get into insert mode. Give as `ei` the sequence to leave insert mode. Now give as `ic` any sequence that needs to be sent just before each character to be inserted. Most terminals with a true insert mode will not require `ic`; it is mainly intended for terminals that use a sequence to open a screen position. (If your terminal has both, insert mode is usually preferable to `ic`. Do not give both unless the terminal actually requires both to be used in combination.) If post-insert padding is needed, give this as a number of milliseconds in `ip` (a string capability). Any other sequence that may need to be sent after insertion of a single character can also be given in `ip`. The `IC` capability, with one parameter `n`, will repeat the effects of `ic` `n` times.

It is occasionally necessary to move the cursor around while in insert mode to delete characters on the same line (for example, if there is a tab after the insertion position). If your terminal allows motion while in insert mode, you can give the Boolean capability `mi` to speed up inserting in this case. Omitting `mi` will affect only speed. Some terminals (notably Datamedia) must not have `mi` because of the way their insert mode works.

Finally, you can specify `dc` to delete a single character, `DC` with one parameter `n` to delete `n` characters, and delete mode by giving `dm` and `ed` to enter and exit delete mode (which is any mode the terminal needs to be placed into for `dc` to work).

Highlighting, Underlining, and Visible Bells

If your terminal has one or more kinds of display attributes, these can be represented in a number of different ways. You should choose one display form as standout mode, representing a good, high-contrast, easy-on-the-eyes format for highlighting error messages and other attention getters. (If you have a choice, reverse video plus half-bright is good, or reverse video alone.) The sequences to enter and exit standout mode are given as `so` and `se`, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces or garbage characters on the screen, as the TVI 912 and Teleray 1061 do, then the numeric capability `sg` should be given to tell how many characters are left.

Codes to begin and end underlining can be given as `us` and `ue`, respectively. If changing the underlining mode leaves blank spaces or garbage characters on the screen, specify `ug`, analogous to `sg`. If the terminal has a code to underline the current character and move the cursor one position to the right, such as the Microterm Mime, this can be given as `uc`.

Other capabilities to enter various highlighting modes include `mb` (blinking), `md` (bold or extra bright), `mh` (dim or half-bright), `mk` (blanking or invisible text), `mp` (protected), `mr` (reverse video), `me` (turn off all attribute modes), `as` (enter alternate character set mode), and `ae` (exit alternate character set mode). Turning on any of these modes singly may or may not turn off other modes.

If there is a sequence to set arbitrary combinations of attributes, this should be given as `sa` (set attributes), taking 9 parameters. Each parameter is either 0 or 1, as the corresponding attribute is on or off. The 9 parameters are, in order: standout, underline, reverse, blink, dim, bold, blank, protect, and alternate character set. Not all modes need be supported by `sa`, only those for which corresponding attribute commands exist.

Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when the cursor is moved to a new line or is addressed. Programs should exit standout mode on such terminals before moving the cursor or sending a newline. On terminals where this is not a problem, the Boolean capability `ms` should be given to indicate that this overhead is unnecessary.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), this can be given as `vb`; it must not move the cursor.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to change, for example, a non-blinking underline into an easier-to-find block or blinking underline), give this sequence as `vs`. If there is a way to make the cursor completely invisible, give that as `vi`. The capability `ve`, which undoes the effects of both `vs` and `vi` should also be given.

If your terminal correctly displays underlined characters (with no special codes needed) even though it does not overstrike, then you should give the Boolean capability `ul`. If overstrikes are erasable with a blank, this should be indicated by giving the Boolean capability `eo`.

Keypad

If the terminal has a keypad that transmits codes when the keys are pressed, `termcap` can represent. Note that it is not possible to handle terminals where the keypad only works in local mode (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, give these sequences as `ks` and `ke`. Otherwise the keypad is assumed to always transmit. The codes sent by the left-arrow, right-arrow, up-arrow, down-arrow, and home keys can be given as `k1`, `kr`, `ku`, `kd`, and `kh`, respectively. If there are function keys such as `f0`, `f1`, ..., `f9`, the codes they send can be given as `k0`, `k1`, ..., `k9`. If these keys have labels other than the default `f0` through `f9`, the labels can be given as `l0`, `l1`, ..., `l9`. The codes transmitted by certain other special keys can be given: `kH` (home down), `kb` (backspace), `ka` (clear all tabs), `kt` (clear the tab stop in the current column), `kC` (clear screen or erase), `kD` (delete character), `kL` (delete line), `kM` (exit insert mode), `kE` (clear to end of line), `kS` (clear to end of screen), `kI` (insert character or enter insert mode), `kA` (insert line), `kN` (next page), `kP` (previous page), `kF` (scroll forward/down), `kR` (scroll backward/up), and `kT` (set a tab stop in the current column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, then the other five keys can be given as `K1`, `K2`, `K3`, `K4`, and `K5`. These keys are useful when the effects of a 3 by 3 directional pad are needed. The obsolete `ko` capability formerly used to describe "other" function keys has been completely supplanted by the above capabilities.

The `ma` entry is also used to indicate arrow keys that send single-character codes. This field is obsolete and redundant with `k1`, `kr`, `ku`, `kd`, and `kh`. It consists of groups of two characters. In each group, the first character is what an arrow key sends, and the second character is the corresponding cursor movement from `vi(1)`. These commands are `h` for `k1`, `j` for `kd`, `k` for `ku`, `l` for `kr`, and `H` for `kh`. For example, the Mime would have `ma=^Hh^Kj^Zk^Xl` indicating arrow keys left (^H), down (^K), up (^Z), and right (^X). (There is no home key on the Mime.)

Tabs and Initialization

If the terminal needs to be in a special mode when running a program that uses `termcap` capabilities, the codes to enter and exit this mode can be given as `ti` and `te`. This is needed, for example, on terminals like the Concept with more than one page of memory. If the terminal has only memory-relative cursor addressing, a screen-sized window must be fixed into the display for cursor addressing to work

properly. This is also used for the Tektronix 4025, where `ti` sets the command character to be the one used by `termcap`.

Other capabilities include `is`, an initialization string for the terminal, and `if`, the name of a file containing long initialization strings. These strings are expected to set the terminal into modes consistent with the rest of the `termcap` description. They should be printed in the following order: `is`; setting tabs using `ct` and `st`; and finally `if`. A pair of sequences that does a harder reset from a totally unknown state can be analogously given as `rs` and `if`. Commands are normally placed in `rs` and `rf` only if they produce annoying effects on the screen and are usually unnecessary. For example, the command to set the VT100 into 80-column mode would normally be part of `is`, but it causes an annoying glitch of the screen and is not normally needed since the terminal is usually in 80-column mode already.

If the terminal has hardware tabs, the command to advance to the next tab stop can be given as `ta` (usually `^I`). A “backtab” command which moves leftward to the previous tab stop can be given as `bt`. By convention, if the terminal driver modes indicate that tab stops are being expanded by the computer rather than being sent to the terminal, programs should not use `ta` or `bt` even if they are present, since the user may not have the tab stops properly set. If the terminal has hardware tabs that are initially set every n positions when the terminal is powered up, then the numeric parameter `it` should be given, showing the number of positions between tab stops. If the terminal has tab stops that can be saved in nonvolatile memory, the `termcap` description can assume that they are properly set.

If there are commands to set and clear tab stops, they can be given as `ct` (clear all tab stops) and `st` (set a tab stop in the current column of every row). If a more complex sequence is needed to set the tabs than can be described by this, the sequence can be placed in `is` or `if`.

Delays

Certain capabilities control padding in the terminal driver. These are primarily needed by hardcopy terminals. The delays should be embedded as padding information in the `cr`, `sf`, `le`, `ff`, and `ta` capabilities. If the numeric capability `pb` (padding baud rate) is given, these values can be ignored at baud rates below the value of `pb`. The delays can also be given as (obsolete) numeric capabilities instead: `dc`, `dn`, `db`, `df`, and `dt`.

Miscellaneous

If the terminal requires other than a NUL (zero) character as padding, this can be given as `pc`. Only the first character of the `pc` string is used.

If the terminal has commands to save and restore the position of the cursor, give them as `sc` and `rc`.

If the terminal has an extra “status line” that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, then the Boolean capability `hs` should be given. Special strings to go to a position in the status line and to return from the status line can be given as `ts` and `fs`. (`fs` must leave the cursor position in the same place that it was before `ts`. If necessary, the `sc` and `rc` strings can be included in `ts` and `fs` to get this effect.) The capability `ts` takes one parameter, which is the column number of the status line to which the cursor is to be moved. If escape sequences and other special commands such as tab work while in the status line, the flag `es` can be given. A string that turns off the status line (or otherwise erases its contents) should be given as `ds`. The status line is normally assumed to be the same width as the rest of the screen, that is, `co`. If the status line is a different width (possibly because the terminal does not

allow an entire line to be loaded), then its width in columns can be indicated with the numeric parameter `ws`.

If the terminal can move up or down half a line, this can be indicated with `hu` (half-line up) or `hd` (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as `ff` (usually `^L`).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters), this can be indicated with the parameterized string `rp`. The first parameter is the character to be repeated and the second is the number of times to repeat it.

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with `cc`. A prototype command character is chosen which is used in all capabilities. This character is given in the `cc` capability to identify it. The following convention is supported on some UNIX systems: The environment is searched for a `CC` variable, and if found, all occurrences of the prototype character are replaced by the character in the environment variable. This use of the `CC` environment variable is a very bad idea, however, because it conflicts with `make(1)`.

Terminal descriptions that do not represent a specific kind of known terminal, such as *switch*, *dialup*, *patch*, and *network*, should include the `gn` (generic) Boolean capability so that programs can complain that they do not know how to work with that terminal. (This capability does not apply to virtual terminal descriptions for which the escape sequences are known.)

If the terminal uses XOFF/XON (DC3/DC1) handshaking for flow control, give `xo`. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted.

If the terminal has a “meta key” which acts as a shift key, setting the eighth bit of any character transmitted, then this fact can be indicated with `km`. Otherwise, software will assume that the eighth bit is parity and it will usually be cleared. If strings exist to turn this “meta mode” on and off, they can be given as `mm` and `mo`.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with `lm`. An explicit value of 0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

If the terminal is one of those supported by the UNIX system virtual terminal protocol, the terminal number can be given as `vt`.

Media copy strings which control an auxiliary printer connected to the terminal can be given as `ps`: print the contents of the screen; `pf`: turn off the printer; and `po`: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. It is undefined whether the text is also displayed on the terminal screen when the printer is on. A variation `p0` takes one parameter and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. All text, including `pf`, is transparently passed to the printer while `p0` is in effect.

Glitches and Braindamage

Hazeltine terminals, which do not allow tilde (~) characters to be displayed, should indicate `hz`.

The `nc` capability, now obsolete, formerly indicated Datamedia terminals, which echo `\r \n` for carriage return then ignore a following linefeed.

Terminals that ignore a linefeed immediately after an `am` wrap, such as the Concept, should indicate `xn`.

If `ce` is required to get rid of standout text (instead of merely writing normal text on top of it), `xs` should be given.

Teleray terminals, where tabs turn all characters moved over to blanks, should indicate `xt` (destructive tabs). This glitch is also taken to mean that it is not possible to position the cursor on top of a “magic cookie”, and that to erase standout mode it is necessary to use delete and insert line.

The Beehive Superbee, which is unable to correctly transmit the `ESC` or `^C` characters, should specify `xb`, indicating that the “f1” key is used for `ESC` and “f2” for `^C`. (Only certain Superbees have this problem, depending on the ROM.)

You may correct other specific terminal problems by adding more capabilities of the form `xx`.

FILES

`/etc/termcap` file containing terminal descriptions

SEE ALSO

`make(1)` and `vi(1)` in the *User's Reference for the DG/UX System*.

`termcap(3X)`, `curses(3X)`, `printf(3S)`, `term(5)`, `terminfo(4)`, in the *Programmer's Reference for the DG/UX System*.

`captainfo(1M)` and `infocmp(1M)` in *System Manager's Reference for the DG/UX System*.

CAVEATS AND BUGS

Note: `termcap` is made obsolete by `terminfo(4)`. The transition will be relatively painless if capabilities flagged as “obsolete” are avoided.

Lines and columns are now stored by the kernel as well as in the `termcap` entry.

The total length of a single entry (excluding only escaped newlines) may not exceed 1024 characters.

Not all programs support all entries.

NAME

types - primitive system data types

SYNOPSIS

```
#include <sys/types.h>
```

DESCRIPTION

The data types defined in the include file are used in DG/UX system code; some data of these types are accessible to user code:

```
typedef struct { int r[1]; } *physadr;
typedef long  clock_t;
typedef long  daddr_t;
typedef char *  caddr_t;
typedef unsigned char  unchar;
typedef unsigned short ushort;
typedef unsigned int  uint;
typedef unsigned long  ulong;
typedef unsigned long  ino_t;
typedef int  pid_t;
typedef int  uid_t;
typedef int  gid_t;
typedef ulong  nlink_t;
typedef ulong  mode_t;
typedef short  cnt_t;
typedef long  time_t;
typedef int  label_t[10];
typedef ulong  dev_t;
typedef long  off_t;
typedef long  pid_t;
typedef long  paddr_t;
typedef int  key_t;
typedef unsigned char  use_t;
typedef short  sysid_t;
typedef short  index_t;
typedef short  lock_t;
typedef unsigned int  size_t;
```

The form *daddr_t* is used for disk addresses except in an i-node on disk; see *fs(4)*. Times are encoded in seconds since 00:00:00 GMT, January 1, 1970. The major and minor parts of a device code specify kind and unit number of a device and are installation-dependent. Offsets are measured in bytes from the beginning of a file. The *label_t* variables are used to save the processor state while another process is running.

SEE ALSO

fs(4).

NAME

ucontext - user context

SYNOPSIS

```
#include <ucontext.h>
```

DESCRIPTION

The `ucontext` structure defines the context of a thread of control within an executing process.

The `ucontext_t` structure is defined in `<sys/ucontext.h>`.

SEE ALSO

`getcontext(2)`, `setcontext(2)`, `sigaction(2)`, `sigprocmask(2)`,
`sigaltstack(2)`,

NAME

ustat - data returned by the ustat system call

SYNOPSIS

```
#include <sys/types.h>
```

DESCRIPTION

The system call `ustat` takes a parameter that is a pointer to the structure defined by this include file. This structure returns file system device statistics.

```
struct ustat
{
    daddr_t      f_tfree;
    ino_t        f_tinode;
    char         f_fname [6];
    char         f_fpack [6];
};
```

The fields of this structure are defined as follows:

f_tfree

The number of blocks with a size of `DEV_BSIZ` bytes that are available for allocation on the file system.

f_tinode

The number of additional files that can be created on the file system.

f_fname

The file system name. This field will be null unless a label has been added to the file system with `labelit`.

f_fpack

The file system pack name. This field will be null unless a label has been added to the file system with `labelit`.

FILES

`/usr/include/sys/ustat.h`

`/usr/include/sys/types.h`

SEE ALSO

`labelit(1M)`, `ustat(2)`, `types(5)`.

NAME

values – machine-dependent values

SYNOPSIS

```
#include <values.h>
```

DESCRIPTION

This file contains a set of manifest constants, conditionally defined for particular processor architectures.

The model assumed for integers is binary representation (one's or two's complement), where the sign is represented by the value of the high-order bit.

- BITS**(*type*) The number of bits in a specified type (e.g., `int`).
- HIBITS** The value of a short integer with only the high-order bit set.
- HIBITL** The value of a long integer with only the high-order bit set.
- HIBITI** The value of a regular integer with only the high-order bit set.
- MAXSHORT** The maximum value of a signed short integer.
- MAXLONG** The maximum value of a signed long integer.
- MAXINT** The maximum value of a signed regular integer.
- MAXFLOAT**, **LN_MAXFLOAT**
The maximum value of a single-precision floating-point number, and its natural logarithm.
- MAXDOUBLE**, **LN_MAXDOUBLE**
The maximum value of a double-precision floating-point number, and its natural logarithm.
- MINFLOAT**, **LN_MINFLOAT**
The minimum positive value of a single-precision floating-point number, and its natural logarithm.
- MINDOUBLE**, **LN_MINDOUBLE**
The minimum positive value of a double-precision floating-point number, and its natural logarithm.
- FSIGNIF** The number of significant bits in the mantissa of a single-precision floating-point number.
- DSIGNIF** The number of significant bits in the mantissa of a double-precision floating-point number.

SEE ALSO

`intro(3)`, `math(5)`, `limits(4)`.

NAME

varargs – handle variable argument list

SYNOPSIS

```
#include <varargs.h>

va_alist
va_dcl
va_list pvar;

void va_start(va_list pvar);
type va_arg(va_list pvar, type);
void va_end(va_list pvar);
```

DESCRIPTION

This set of macros allows portable procedures that accept variable argument lists to be written. Routines that have variable argument lists [such as `printf(3S)`] but do not use `varargs` are inherently non-portable, as different machines use different argument-passing conventions.

`va_alist` is used as the parameter list in a function header.

`va_dcl` is a declaration for `va_alist`. No semicolon should follow `va_dcl`.

`va_list` is a type defined for the variable used to traverse the list.

`va_start` is called to initialize `pvar` to the beginning of the list.

`va_arg` will return the next argument in the list pointed to by `pvar`. `type` is the type the argument is expected to be. Different types can be mixed, but it is up to the routine to know what type of argument is expected, as it cannot be determined at run-time.

`va_end` is used to clean up.

Multiple traversals, each bracketed by `va_start` and `va_end`, are possible.

EXAMPLE

This example is a possible implementation of `execl` [see `exec(2)`].

```
#include <unistd.h>
#include <varargs.h>
#define MAXARGS 100

/*    execl is called by
        execl(file, arg1, arg2, ..., (char *)0);
*/
execl(va_alist)
va_dcl
{
    va_list ap;
    char *file;
    char *args[MAXARGS];          /* assumed big enough*/
    int argno = 0;

    va_start(ap);
    file = va_arg(ap, char *);
    while ((args[argno++] = va_arg(ap, char *)) != 0)
        ;
}
```

```
        va_end(ap);  
        return execv(file, args);  
    }
```

SEE ALSO

exec(2), printf(3S), vprintf(3S), stdarg(5).

NOTES

It is up to the calling routine to specify in some manner how many arguments there are, since it is not always possible to determine the number of arguments from the stack frame. For example, `execl` is passed a zero pointer to signal the end of the list. `printf` can tell how many arguments are there by the format.

It is non-portable to specify a second argument of `char`, `short`, or `float` to `va_arg`, since arguments seen by the called function are not `char`, `short`, or `float`. C converts `char` and `short` arguments to `int` and converts `float` arguments to `double` before passing them to a function.

`stdarg` is the preferred interface.

NAME

wstat - wait status

SYNOPSIS

```
#include <sys/wait.h>
```

DESCRIPTION

When a process waits for status from its children via either the `wait` or `waitpid` function, the status returned may be evaluated with macros, defined in `sys/wait.h`. These macros evaluate to integral expressions. The *stat* argument to these macros is the integer value returned from `wait` or `waitpid`.

See the `wait` man page for complete descriptions of these macros.

SEE ALSO

`exit(2)`, `wait(2)`, `waitpid(3C)`.

End of Chapter

Chapter 6

Communications Protocols

This chapter contains in printed form the online manual entries for DG/UX, TCP/IP, and ONC/NFS communications protocols. Some entries in this chapter are generic to the DG/UX system; others relate specifically to TCP/IP or ONC/NFS.

Table 6-1 summarizes the man pages in this chapter:

Table 6-1 Summary of Communications Protocol Manual Pages

Product	Name	Description
DG/UX	dot3(6P)	Describes the IEEE 802.3 protocol
	snap(6P)	Describes the Subnetwork Access Protocol
	unix_ipc(6F)	Describes piping communications within a host
TCP/IP	intro(6)	Introduces the TCP/IP protocol family
	inet(6F)	Provides more detail about the TCP/IP protocol family
	ip(6)	Internet protocol
	tcp(6P)	Transport control protocol
	udp(6P)	User datagram protocol
ONC/NFS	nfs(6P)	Describes the Network File System protocol

NAME

intro - Communications Protocols introduction to networking facilities

SYNOPSIS

```
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <net/if.h>
```

DESCRIPTION

This section briefly describes the DG/UX system networking facilities. Documentation in this section covers three areas: the Internet protocol family, the available protocols, and the network interfaces. The Internet protocol family is described on the `inet(6F)` manual page, whereas entries describing the protocols are on manual pages marked *6P*. Network interfaces are described on manual pages marked *6*.

The Internet family includes the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Protocol (IP), and Internet Control Message Protocol (ICMP). These protocols are communications facilities implemented in the DG/UX system kernel that transfer information from user programs to the network and back. Programmers writing user-level programs can access TCP, IP, and UDP with the `socket(2)` family of system calls and the Transport Layer Interface (TLI) library routines.

The Transmission Control Protocol (TCP) fits into the layered networking architecture just above IP. Application programs, such as remote terminal agents and file transfer agents, usually run on top of TCP, using its services.

TCP assures reliable end-to-end delivery of a data byte stream. TCP deals with user data copied to the protocol's buffers. It packages the data into segments and passes this information to IP, which then breaks the information into packets that can be easily transmitted across the network. IP then determines the next hop on a path through the network for the packet being transmitted and transfers the packet to the first host on the path. A gateway host would receive the packet and route it to the destination host. When packets arrive at the destination host, TCP reconstructs the entire message, checking to ensure that the data is complete and correctly ordered before sending it to application programs. If there is a problem, TCP requests that the message be retransmitted.

Like TCP, the User Datagram Protocol (UDP) fits into the layered networking architecture just above IP. It provides procedures for application programs to send messages to other programs with a minimum of protocol mechanism. UDP is a simple datagram protocol. Unlike TCP, it neither guarantees reliable delivery nor does it provide protection from duplicate messages.

The Internet Protocol (IP) is primarily concerned with getting a *datagram* to the next host on the route to the datagram's final destination. A datagram is a self contained package of data carrying sufficient information for hosts to deliver it to its destination. Since host availability changes, the packets that make up a complete message may have different routes and may end up at the destination out of their original order. The TCP layer is responsible for re-ordering the packets correctly. Some packets may be lost or garbled in transmission. IP frequently notifies higher level protocols when packets are lost or damaged, but sometimes does not.

The Internet Control Message Protocol (ICMP) is used to report errors in datagram processing. ICMP is an integral part of IP and must be implemented by every IP

module. ICMP messages are sent to report problems in the communication environment, not to make IP a reliable protocol.

Addressing

Associated with each protocol family is an address format. The following address formats are used by the system:

```
#define AF_UNIX 1    /* local to host (pipes) */
#define AF_INET 2    /* internetwork: UDP, TCP, etc. */
```

Interfaces

Each network interface in a system corresponds to a path through which messages may be sent and received. A network interface usually has a hardware device associated with it, though certain interfaces such as the loopback interface, `loop(7)`, do not.

The following `ioctl` calls may be used to manipulate network interfaces. See *Programming with TCP/IP on the DG/UX™ System* for details.

SIOCSIFADDR

Set interface address. Following the address assignment, the "initialization" routine for the interface is called.

SIOCGIFADDR

Get interface address.

SIOCSIFBRDADDR

Set interface broadcast address. This address is used to send IP broadcast packets on broadcast capable interfaces.

SIOCGIFBRDADDR

Get interface broadcast address.

SIOCSIFDSTADDR

Set the destination address for point-to-point network interfaces.

SIOCGIFDSTADDR

Get interface destination address.

SIOCSIFMETRIC

Set the interface routing metric. This information is used by routing applications.

SIOCGIFMETRIC

Get the interface routing metric.

SIOCSIFNETMASK

Set the interface subnetwork mask.

SIOCGIFNETMASK

Get the interface subnetwork mask.

SIOCSIFFLAGS

Set interface flags field. If the interface is marked as down, any processes currently routing packets through the interface are notified.

SIOCGIFFLAGS

Get interface flags.

SIOCGIFCONF

Get interface configuration list.

SEE ALSO

ioctl(2), socket(2), Programming with TCP/IP on the DG/UX System.

NAME

dot3 – IEEE 802.3 carrier sense multiple access with collision detection

DESCRIPTION

IEEE project 802 has defined specifications for the lowest two layers of an OSI model network architecture, the physical layer and the data link layer. Project 802 focuses on the implementation of these layers for a local area network. It divides the data link layer into a logical link control (LLC) sublayer and a media access control (MAC) sublayer.

802.3 is a MAC sublayer that operates below the LLC sublayer. 802.3 is a standard for providing carrier sense multiple access with collision detection (CSMA/CD).

Data General's dot3 is a Streams module that sits on top of the Ethernet device driver to provide 802.3 functionality to upper-level protocols. This module, when used with the llc(7) multiplexor, allows the dgen, hken, or inen Ethernet device driver to be used to provide IEEE 802.3 functionality to one stream while providing Ethernet functionality to other streams.

Because of the fact that the 802.3 standard uses a length field instead of a type field in its header (as in Ethernet), it is not possible for more than one stream to the ethernet device to have the dot3 module in it. The reason for this is that it is not possible to demultiplex based on the length field of the 802.3 packet. The implication of this is that if a stream is open to the inen device that has llc and dot3 in the stream, another stream to the same inen device that tries to push the dot3 module on it will never receive packets upstream, until the first stream with the dot3 is closed.

SEE ALSO

netinit(1M), llc(7), dgen(7), hken(7), inen(7).

NAME

inet – Communications Protocol Internet protocol family

SYNOPSIS

```
#include <netinet/in.h>
```

DESCRIPTION

The Internet protocol family is a collection of protocols based on and including the Internet Protocol (IP), the Transmission Control Protocol (TCP), and the User Datagram Protocol (UDP). Each of these protocols uses the Internet address format.

Addressing

Internet addresses are four-byte quantities, stored in network standard format. The include file `netinet/in.h` defines this address as a discriminated union.

Sockets bound to the Internet protocol family utilize the following addressing structure:

```
struct sockaddr_in {
    short    sin_family;
    u_short  sin_port;
    struct in_addr  sin_addr;
    char     sin_zero[8];
};
```

Sockets may be created with the address `INADDR_ANY` to affect “wildcard” matching on incoming messages.

Protocols

The Internet protocol family consists of the Internet Protocol (IP), Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). TCP is used to support the `SOCK_STREAM` socket type, while UDP is used to support the `SOCK_DGRAM` socket type. A raw interface to IP is available by creating an Internet socket of type `SOCK_RAW`. The ICMP is not directly accessible.

SEE ALSO

`ip(6P)`, `tcp(6P)`, `udp(6P)`.

NAME

IP – Communications Protocol Internet Protocol

INCLUDE FILES

```
#include <sys/socket.h>
#include <netinet/ip.h>
```

SYNOPSIS

This is an example of how you would create an endpoint for the IP connection.

```
s = socket(AF_INET, SOCK_RAW, 0);
```

DESCRIPTION

IP is the network/internetwork layer protocol used by the Internet protocol family. It may be accessed through a “raw socket” when developing special-purpose applications. A raw socket can be opened only by the superuser.

IP sockets are connectionless, and are normally used with the `sendto` and `recvfrom` calls, though the `connect(2)` call may also be used to fix the destination for future packets (in which case the `read(2)` or `recv(2)` and `write(2)` or `send(2)` system calls may be used).

Outgoing packets must have an IP header prepended to them.

OPTIONS

IPPROTO_IP level options recognized by IP:

- | | |
|---------------|--|
| IP_TX_OPTIONS | IP transmit options. When setting, the system will verify that the option string is well formed. |
| IP_RX_OPTIONS | IP receive options. When setting, the system will verify that the option string is well formed. |
| IP_TOS | IP Type Of Service. |
| IP_TTL | IP Time To Live. Number of routing hops a packet may make before reaching its destination. |
| IP_DONTFRAG | IP Dont Fragment flag. When non-zero, IP will try to send a packet without fragmenting. If a packet is too large to send without fragmenting, the packet is dropped. |

SEE ALSO

`connect(2)`, `recv(2)`, `send(2)`.
`intro(6)`, `inet(6F)`, *Programming with TCP/IP on the DG/UX System*.

NAME

nfs, NFS - network file system

CONFIG

options NFS

DESCRIPTION

The Network File System, or NFS, allows a client workstation to perform transparent file access over the network. Using it, a client workstation can operate on files that reside on a variety of servers, server architectures and across a variety of operating systems. Client file access calls are converted to NFS protocol requests, and are sent to the server system over the network. The server receives the request, performs the actual file system operation, and sends a response back to the client.

The Network File System operates in a stateless fashion using remote procedure (RPC) calls built on top of external data representation (XDR) protocol. These protocols are documented in *Managing ONCTM/NFS[®] and Its Facilities on the DG/UXTM System*. The RPC protocol provides for version and authentication parameters to be exchanged for security over the network.

A server can grant access to a specific filesystem to certain clients by adding an entry for that filesystem to the server's `/etc/exports` file and executing the `exportfs -a` command.

A client gains access to that filesystem with the `mount(2)` system call, which requests a file handle for the filesystem itself. Once the filesystem is mounted by the client, the server issues a file handle to the client for each file (or directory) the client accesses. If the file is somehow removed on the server side, the file handle becomes stale (dissociated with a known file).

A server may also be a client with respect to filesystems it has mounted over the network, but its clients cannot gain access to those filesystems. Instead, the client must mount a filesystem directly from the server on which it resides.

The user ID and group ID mappings must be the same between client and server. However, the server maps uid 0 (the super-user) to uid 65534 before performing access checks for a client. This inhibits super-user privileges on remote filesystems. A server can, however, allow root access for specific clients by making an entry in the `/etc/exports` file.

DIAGNOSTICS

Generally physical disk I/O errors detected at the server are returned to the client for action. If the server is down or inaccessible, the client will see the console message:

```
NFS: file server not responding: still trying.
```

For hard-mounted file systems, the client resends the request until it receives an acknowledgement from the server. This means the server can crash or power down, and come back up, without any special action required by the client.

FILES

`/etc/exports`

SEE ALSO

`exportfs(1M)`, `mount(1M)`, `mountd(1M)`, `nfsd(1M)`, `mount(2)`, `exports(4)`, `fstab(4)`.

NAME

snap – Subnetwork Access Protocol

DESCRIPTION

SNAP is part of the 802.1 layer. SNAP provides a way for protocols that run over Ethernet to run over 802.x media. The SNAP sub-layer contains a five byte header that can be used to specify additional information for upper layers. The first three bytes of the header represents an organizationally unique identifier (OUI) while the last two bytes are locally administered. This structure allows different vendors the flexibility to use the additional bytes of the header as they wish.

One use of SNAP that has been commonly agreed upon is to use an OUI of 0 to represent upper layer protocols that run over an Ethernet based media. The two bytes that are locally administered are then used to represent the two byte ether type field. In this manner, protocols that run over Ethernet are able to use the SNAP layer to run over 802.2 (LLC) and 802.x MAC layers. Data General provides the SNAP functionality in the form of the SNAP pseudo-driver.

SEE ALSO

ifconfig(1M), netinit(1M), dgen(7), hken(7), inen(7), llc(7).

NAME

TCP – Network Protocol Internet Transmission Control Protocol

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/tcp.h>
```

This is an example of how you would create an endpoint for the TCP connection:

```
s = socket(AF_INET, SOCK_STREAM, 0);
```

DESCRIPTION

Transmission Control Protocol (TCP) provides reliable, flow-controlled, two-way transmission of data. It is a byte-stream protocol used to support the `SOCK_STREAM` abstraction. TCP provides a per-host collection of port addresses on top of the standard Internet address format. Thus, each address is composed of an Internet address specifying the host and network, with a specific TCP port on the host identifying the peer entity.

Sockets utilizing the TCP are either “active” or “passive”. Active sockets initiate connections to passive sockets. By default TCP sockets are created active; only active sockets may use the `connect(2)` call to initiate connections. To create a passive socket, the `listen(2)` system call must be used after binding the socket with the `bind(2)` system call. Only passive sockets may use the `accept(2)` call to accept incoming connections.

Passive sockets may “underspecify” their location to match incoming connection requests from multiple networks. This technique, termed “wildcard addressing”, allows a single server to provide service to clients on multiple networks. To create a socket that listens on all networks, the Internet address `INADDR_ANY` must be bound to the socket. The TCP port may still be specified at this time; if the port is not specified, the system will assign one. Once a connection has been established, the socket’s address is fixed by the peer entity’s location. The address assigned to the socket is the address associated with the network interface through which packets are being transmitted and received.

OPTIONS

IPPROTO_TCP level options recognized by TCP:

TCP_NODELAY When the option value is non-zero, the system does not delay sending data to coalesce small packets. When the option value is zero, the system may defer sending data to coalesce small packets to conserve network bandwidth.

TCP_MAXSEG When set prior to a `connect(2)` call, TCP will use the option value to negotiate the maximum size of TCP packets sent and received during the life of the connection. Values for the TCP Maximum Segment Size are between 1 and 65,535. This option is only valid prior to establishing a connection. The result of segment size negotiation is less than or equal to the option value.

TCP_URGENT_INLINE

This option has no effect in the DG/UX system. Use the `SO_OOINLINE` socket level option.

TCP_PEER_ADDRESS

Restricts the passive TCP endpoint to only accept connections

initiated by the address supplied in the option value. The option value must contain a pointer to a `sockaddr_in` structure.

TCP_ACCEPT_QUEUE_LENGTH

Sets the number of outstanding connections allowed at the TCP passive endpoint.

SEE ALSO

`intro(6)`, `inet(6F)`, *Programming with TCP/IP on the DG/UX System*.
`getsockopt(2)`, `setsockopt(2)`.

NAME

UDP – Communications Protocol Internet User Datagram Protocol

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/udp.h>
```

This is an example of how you would create an endpoint for the UDP connection:

```
s = socket(AF_INET, SOCK_DGRAM, 0);
```

DESCRIPTION

UDP is a simple, unreliable datagram protocol that is used to support the `SOCK_DGRAM` abstraction for the Internet protocol family.

UDP sockets are connectionless, and are normally used with the `sendto(2)` and `recvfrom(2)` calls. The `connect(2)` and `bind(2)` calls may also be used to fix the destination for future packets (in which case the `recv(2)` or `read(2)` and `send(2)` or `write(2)` system calls may be used). `listen(2)` and `accept(2)` are not valid operations on datagram sockets.

SEE ALSO

`send(2)`, `recv(2)`, `sendto(2)`, `recvfrom(2)`.
`intro(6)`, `inet(6F)`, *Programming with TCP/IP on the DG/UX™ System*.

NAME

unix_ipc - piping communications within a host

SYNOPSIS

```
#include <sys/types.h>
#include sys/un.h
```

DESCRIPTION

The `unix_ipc` protocol is used for interprocess communications within a single host. It supports stream and datagram interfaces.

Addressing

Endpoints can be named by entries in the file system:

```
struct sockaddr_un {
    short sun_family;    /* AF_UNIX */
    char sun_path[SOCKADDR_UN_MAXLEN]; /* pathname */
};
```

SEE ALSO

`bind(2)`, `pipe(2)`.

NOTE

This implementation uses names in the file system; this is subject to change. See **NOTES** in `bind(2)`.

End of Chapter

Appendix A

Contents and Permuted Index Man Pages

This is a printed copy of the table of contents and the permuted keyword in context index contained in the online `contents(0)` and `index(0)` manual pages. These man pages contain information extracted from the man pages in the *DG/UX Programmer's Reference* (Volumes 1 and 2), *System Manager's Reference*, and *User's Reference*.

The permuted index is a list of keywords, given in the second of three columns, together with the context in which the keyword is found. Keywords are either topical keywords or the names of manual entries. Entries are identified with their chapter numbers shown in parentheses. The right column lists the name of the manual page on which each keyword may be found. The left column contains useful information about the keyword.

TABLE OF CONTENTS

This manual page contains the following sections:

1. Commands and Application Programs
2. System Calls
3. Subroutines and Libraries
4. File Formats
5. Miscellaneous Features
6. Communications Protocols
7. System Special Files
8. System Maintenance Procedures

1. Commands and Application Programs

intro	introduction to system maintenance commands and application programs
intro	introduction to commands and application programs
intro	introduction to commands and application programs
accept	accept or reject print requests
acct	overview of accounting and miscellaneous accounting commands
acctcms	command summary from per-process accounting records
acctcom	search and print process accounting file(s)
acctcon	connect-time accounting
acctmerg	merge or add total accounting files
acctprc	process accounting
acctsh	shell procedures for accounting
admaccounting	manage accounting system
admalias	manage mail alias information in the aliases database
admbackup	manage backup and recovery of file systems
admclient	manage operating system clients
admdate	manipulate the system date, time and time zone
admdefault	provide an interface to named default sets
admdumpcycle	manage dump cycle tables
admdumpdevice	manage the dump device table
admether	manage ether database
admfilesystem	manage file systems
admfsinfo	display information about files and directories
admgroup	manage group information in the group database
admhost	manage hosts database
admin	create and administer SCCS files
admipinterface	manage the TCP/IP network interfaces database
admkernel	manipulate the system's kernel
admlock	manage simple process synchronization
admnetwork	manage network database
admnl	manipulate national language variables
admpackage	manage DG/UX-style software packages
admportmonitor	manage port monitors
admportservice	manage port monitor services
admprocess	manage processes
admrelease	manage software release areas
admresolve	manage DNS resolver's domain name and nameservers database
admroute	manage routing databases
admrsell	manage the remote and restricted shell names
admsar	manage system activity monitoring and reporting
admsservice	manage service database
admsnmpcommunity	manage the SNMP community database
admsnmpobject	manage the snmpd object database
admsnmptrap	manage the SNMP traps database
admsvcorder	manage search order for /etc/hosts, NIS, and DNS databases
admswap	manage swap areas
admtape	manipulate the default parameters for tapes
admtcpipdaemon	manage the TCP/IP servers
admtcpipparams	manage the TCP/IP host parameters
admterminal	manage terminal ports
admtrustedhost	manage the trusted hosts database
admuser	manage user information in the password database

admxterminal manage serving of X display terminals
 alp query the ALP STREAMS module
 apropos locate commands by keyword lookup
 ar archive and library maintainer for portable archives
 as MC88000 assembler
 asa interpret ASA carriage control characters
 at execute commands at a later time
 atq display the jobs queued to run at specified times
 atrm remove jobs spooled by at or batch
 att_dump dump parts of an object or object archive file
 autopush configure automatically pushed STREAMS modules
 banner make posters
 basename deliver portions of path names
 bc arbitrary-precision arithmetic language
 bcs_cat type hosts, networks, passwd, protocols, group or services information
 bdiff big diff
 berk_diff Berkeley differential file and directory comparator
 berk_diff3 Berkeley 3-way differential file comparison
 bfs big file scanner
 biod start block I/O servers
 cal print calendar
 calendar reminder service
 captoinfo convert a TERMCAP entry into a TERMINFO entry
 cat concatenate and type files to standard output
 catexstr extract strings from source files, replace with catgets calls.
 catgets print message from message catalog
 cb C program beautifier
 cc C language compiler
 cd change working directory
 cdc change the delta commentary of an SCCS delta
 cflow generate a C flow graph
 chgrp change the group ownership of a file
 chgtinfo create a temporary version of a TERMINFO entry
 chmod change file mode
 chown change file owner
 chroot change root directory for a command
 chrtbl generate character classification and conversion tables
 ci check in RCS revisions
 ckbinarsys determine whether remote system can accept binary messages
 ckdate prompt for and validate a date
 ckgid prompt for and validate a group id
 ckint display a prompt; verify and return an integer value
 ckitem build a menu; prompt for and return a menu item
 ckkeywd prompt for and validate a keyword
 ckpath display a prompt; verify and return a pathname
 ckrange prompt for and validate an integer
 ckstr display a prompt; verify and return a string answer
 cktime display a prompt; verify and return a time of day
 ckuid prompt for and validate a user ID
 ckyorn prompt for and validate yes/no
 clear clear terminal screen
 clri clear inode
 cmp compare two files
 co check out RCS revisions
 cof2elf translate object file from COFF to ELF
 col filter reverse line-feeds
 colltbl create collation database
 comb combine SCCS deltas
 comm select or reject lines common to two sorted files
 compress compress, expand or display expanded files
 config configure a system
 cp copy files
 cpd change or view the allocation limits for a control point directory
 cpio copy file archives in and out

cpp the C language preprocessor
 cprs compress a common object file
 crash examine system images
 cron clock agent
 crontab user crontab file
 crypt encode/decode
 cscope interactively examine a C program
 csh invoke a shell (command interpreter) having a C-like syntax
 csplit context split
 ct spawn login to a remote terminal
 ctags create a tags file
 ctl COFF-to-legend translator
 ctrace trace a C program to debug it
 cu call another UNIX system
 cut cut out selected fields of each line of a file
 cxref generate C program cross-reference
 date print and set the date
 dbx source level debugger
 dc desk calculator
 dd convert and copy a file
 deblock change blocking size
 default-gcc set or query default version of GNU C
 delta make a delta (change) to an SCCS file
 deroff remove nroff/troff, tbl, and eqn constructs
 devattr lists device attributes
 devfree release devices from exclusive use
 devnm device name
 devreserv reserve devices for exclusive use
 df report number of free disk blocks and inodes
 dg_fsdb file system debugger
 dg_kill test for or terminate a process
 dg_sysctl display or modify boot and dump parameters
 diff differential file comparator
 diff3 3-way differential file comparison
 dircmp compare two directories
 dis object code disassembler
 diskman menu interface for managing physical and logical disks
 diskusg generate disk accounting data by user id
 dispgid display a list of all valid group names
 dispuid display a list of all valid user names
 dkctl control special disk operations
 download download host resident PostScript fonts
 dpost troff postprocessor for PostScript printers
 du summarize disk usage
 dump incremental file system dump
 dump2 incremental file system backup
 dump2label read and write labels for dump tapes
 dumpfs dump file system information
 echo echo arguments
 ed text editor
 edit text editor (variant of ex for casual users)
 egrep search a file for a pattern using full regular expressions
 enable enable/disable LP printers
 env set environment for command execution
 eucset set or get EUC code set widths
 ex text editor
 expr evaluate arguments as an expression
 exstr extract strings from source files
 factor factor a number
 fez display file element sizes
 fgrep search a file for a character string
 file determine file type
 filesave daily/weekly file system backup
 find find files

finger	display information about local and remote users
fingerd	remote user information server
fmt	simple text formatter
fmtmsg	display a message on stderr or system console
fold	fold long lines for finite width output device
freec	recover files from a backup tape
fsck	check file systems for consistency and repair them
fsdb	file system debugger
fsplit	split f77 or ratfor files
fuser	identify processes using a file or file structure
fwtmp	manipulate connect accounting records
gcc	GNU C language compiler
gencat	generate a formatted message catalogue
get	check out a version of an SCCS file
getdev	lists devices based on criteria
getdgrp	lists device groups which contain devices that match criteria
getopt	parse command options
getopts	parse command options
gettext	retrieve a text string from a message data base
getty	set terminal type, modes, speed, and line discipline
glossary	definitions of common terms and symbols
grep	search a file for a pattern
gridman	menu interface for maintaining a High Availability Disk Array subsystem
groupadd	add (create) a new group definition on the system
groupdel	delete a group definition from the system
groupmod	modify a group definition on the system
groups	show group memberships
halt	stop the system processor
head	give the first few lines
help	help facility
helpadm	make changes to the help facility database
iconv	code set conversion
id	print the user name and ID, and group name and ID
idc	interface description compiler
ident	identify files
idi	interface description interpreter
idi_tools	tools for use with the interface description interpreter
info	documentation browser
infocmp	compare or print out TERMINFO descriptions
init	process control initialization
install	install commands
installf	add a file to the software installation database
installman	manage system installation
ipcrm	remove a message queue, semaphore set, or shared memory ID
ipcs	report inter-process communication facilities status
join	relational database operator
kbdcomp	compile att_kbd tables
kbdload	load or link att_kbd tables
kbdpipe	use the KBD module in a pipeline
kbdset	attach to att_kbd mapping tables, set modes
kill	terminate a process by default
killall	kill all active processes
ksh	KornShell, a standard/restricted command and programming language
last	indicate last user or terminal logins
ld	link editor for object files
ld-coff	link editor for common object files
ldd	list dynamic dependencies
lex	generate programs for simple lexical tasks
line	read one line
link	exercise link and unlink system calls
lint	a C program checker
listdgrp	lists members of a device group
listen	network listener server
listusers	list user login information

ln	link files
locate	identify a command using keywords
logger	make entries in the system log
login	sign on
logins	list user and system login information
logname	get login name
lorder	find ordering relation for an object library
lp	send/cancel requests to an LP print service
lpadmin	configure the LP print service
lpc	line printer control program
lpd	line printer spooler
lpfilter	administer filters used with the LP print service
lpforms	administer forms used with the LP print service
lpprint	menu-driven lp interface
lpq	examine the spool queue
lpr	send print requests to a line printer spooler
lprm	remove jobs from the line printer spooling queue
lpsched	start/stop the LP print service and move requests
lpstat	print information about the status of the LP print service
lpssystem	register remote systems with the print service
lptermprinter	start printer session with 40014A Terminal Server
lpusers	set printing queue priorities
ls	list contents of directory
lsd	load a system dump from tape
m4	macro processor
machid	provide truth value about your processor type
mail	read mail or send mail to users
mailalias	translate mail alias names
mailx	interactive message processing system
mail_pipe	invoke recipient command for incoming mail
make	maintain, update, and regenerate groups of programs
makekey	generate encryption key
man	locate and print entries from the reference manuals
mcs	manipulate the comment section of an object file.
merge	three-way file merge
mesg	permit or deny messages
mkdir	make a directory
mkfifo	make FIFO special file
mkfs	create a file system
mkmsgs	create message files for use by gettxt
mknod	build a special file
mkstr	create an error message file by massaging C source
montbl	create monetary database
more	display file one screenful at a time
mount	mount and dismount filesystems
mt	magnetic tape control
mv	move files
mmdir	move a directory
nawk	pattern scanning and processing language
ncheck	generate names from i-numbers
newform	change the format of a text file
newgrp	log in to a new group
news	print news items
nice	run a command at a higher or lower priority
nl	line numbering filter
nlsadmin	network listener service administration
nm	print name list of common object file
nohup	run a command immune to hangups and quits
notify	notify user of the arrival of new mail
oawk	old pattern scanning and processing language
od	octal dump
osysadm	menu-driven system administration program
pack	compress and expand files
passmgt	password files management

passwd change login password
paste merge lines
pg display file forward or backward one screenful at a time
pkgadd transfer software package to the system
pkgask stores answers to a request script
pkgchk check accuracy of installation
pkginfo display software package information
pkgmk produce an installable package
pkgparam displays package parameter values
pkgproto generate a prototype file
pkgrm removes a package from the system
pkgtrans translate package format
pmadm port monitor administration
postdaisy PostScript translator for Diablo 630 files
postdmd PostScript translator for DMD bitmap files
postio serial interface for PostScript printers
postmd matrix display program for PostScript printers
postplot PostScript translator for plot(4) graphics files
postprint translate text files into PostScript
postreverse reverse the page order in a PostScript file
posttek PostScript translator for tektronix 4014 files
pr print files
printenv print out the environment
printf print formatted output
probedev probe system for devices
prof display profile data
profiler operating system profiler
prs print an SCCS file
ps report process status
putdev edit device table
putdgrp edit device group table
pwck check password or group file
pwd print working directory name
ratfor rational FORTRAN dialect
rcs change RCS file attributes
rcsclean clean up working files
rcsdiff compare RCS revisions
rcsfreeze freeze a configuration of sources checked in under RCS
rcsintro introduction to RCS commands
rcsmerge merge RCS revisions
reboot restart the operating system
reelexchange_intro commands for reading and writing IBM and ANSI tapes
regcmp regular expression compile
removef remove a file from software database
renice alter priority of running processes
reset reset the teletype bits to a sensible state
restore incrementally restore a file system
rev reverse order of characters in each line of file
rlog print log messages and other information about RCS files
rm remove, delete files or directories
rmdel remove a delta from an SCCS file
rmt start the remote mag tape server
runacct run daily accounting
sac service access controller
sacadm service access controller administration
sact print current SCCS file editing activity
sar system activity report package
sar system activity reporter
sccsdiff compare two versions of an SCCS file
sccstorcs build RCS file from SCCS file
script make typescript of a terminal session
sdb symbolic debugger
sde-target print commands to reset software development environment target
sdiff side-by-side difference program

sed stream editor
 setmnt establish mount table
 setuname changes machine information
 sh shell, the command programming language
 shl shell layer manager
 shutdown shut down system, change system state
 sifilter preprocess MC88100 assembly language
 size print section sizes of object files
 sleep suspend execution for an interval
 sno SNOBOL interpreter and compiler
 sort sort and/or merge files
 spell find spelling errors
 spline interpolate smooth curve
 split split a file into pieces
 srchtxt display contents of, or search for a text string in, message data bases
 starter information for beginning users
 strace print STREAMS trace messages
 strchg change or query stream configuration
 strclean STREAMS error logger cleanup program
 strerr STREAMS error logger server
 strings find the printable strings in an object or other binary file
 strip strip non-executable information from an object file
 stty set the options for a terminal
 sttydefs maintain line and hunt settings for TTY ports
 su become super-user or another user
 sum print checksum and block count of a file
 swapon specify additional devices for system paging
 syacdb syac debugger utility program
 syacdmp dump syac memory to a file
 syac_routes Change SYAC routing information
 syac_ttyaddr set tty specific internet addresses
 sync update the super-block
 sysadm menu-driven system administration interface
 sysdef output system definition
 syslogd log systems messages
 systemid display the unique system identifier
 tabs set tabs on a terminal
 taccess initiate access to labeled tape
 tail deliver the last part of a file
 tar tape file archiver
 tcload load terminal controller devices
 tdisplay display label and record translation settings
 tee pipe fitting
 termprinter print a file using the 40014A Terminal Server
 test condition evaluation command
 testlocale test locale definition
 tic TERMINFO compiler
 time time a command
 timex time a command; report process data and system activity
 tkey set label and data translation parameters
 tlabel initialize a tape with a volume label
 touch update access and modification times of a file
 tposn position tape to specified file
 tput initialize a terminal or query terminfo database
 tr translate characters
 tread read file(s) from tape
 trelease terminate access to a tape
 true provide truth values
 tsniff summary report of tape contents
 tsort topological sort
 tty get the name of the terminal
 ttyadm format and output TTY port monitor information
 ttymon monitor terminal ports
 tune tune an existing file system

twrite writes a file to tape
 ul do underlining
 umask set file-creation mode mask
 uname print name of current system
 unget undo a previous get of an SCCS file
 uniq report repeated lines in a file
 units conversion program
 usage retrieve a command description and usage examples
 useradd administer a new user login on the system
 userdel delete a user's login from the system
 usermod modify a user's login information on the system
 uucheck check the uucp directories and permissions file
 uucico file transport program for the uucp system
 uucleanup uucp spool directory clean-up
 uucp UNIX-to-UNIX system copy
 uuencode encode/decode a binary file for transmission via mail
 uusched the scheduler for the uucp file transport program
 uustat uucp status inquiry and job control
 uuto public UNIX-to-UNIX system file copy
 uutry try to contact remote system with debugging on
 uux UNIX-to-UNIX system command execution
 uuxqt execute remote command requests
 vacation automatically respond to incoming mail messages
 val validate SCCS file
 valtools introduction to validation tools
 vc version control
 vi screen-oriented (visual) display editor based on ex
 vipw edit the system password file
 volcopy copy file systems with label checking
 vsccheck verify that the VSC synchronous controller is operable
 vscload download board resident software onto VSC synchronous controller
 wait await completion of process
 wall write to all users
 wc word count
 wchrtbl generate character classification and conversion tables
 what identify SCCS files
 whatis display a one-line summary about a topic
 whereis locate source, binary, and or manual for program
 which locate a program file for csh(1) users
 who who is on the system
 whodo who is doing what
 wntd start the WORM magnetic tape device server
 write write to another user
 xargs construct argument list(s) and execute command
 xref generate cross reference table from C, Fortran and Pascal sources
 xstr extract strings from C programs to implement shared strings
 yacc yet another compiler-compiler
 zdump time zone dumper
 zic time zone compiler

2. System Calls

intro introduction to system calls and error numbers
 accept accept a connection on a socket
 access determine the accessibility of a file
 acct enable or disable process accounting
 adjtime correct the time to allow synchronization of the system clock
 alarm set a process alarm clock
 async_daemon start a BIOD server for asynchronous I/O requests
 berk_sigpause set blocked signals and suspend process until a signal is caught
 bind bind a name to a socket
 brk change data segment space allocation
 chdir change the working directory of the calling process
 chmod change mode of file
 chown change user id and group id of a file

chroot change the root directory of the calling process

close close an object associated with a file descriptor

connect initiate a connection on a socket

creat create a new file or rewrite an existing one

csync synchronize hardware caches for execute access

dg_allow_shared_descriptor_attach let processes attach shared descriptor array

dg_attach_to_shared_descriptors attach another process's shared descriptor array

dg_decryptsessionkey decrypt conversation key with the client/server common key

dg_devctl perform device-control functions

dg_encryptsessionkey encrypt conversation key with the client/server common key

dg_ext_errno return the extended errno for the current process

dg_file_info get file usage information for process identified by process key

dg_fstat get extended file status information

dg_getrootkey get root's secret key

dg_ipc_info get information about current IPCs state

dg_lcntl process a record lock request on a filehandle

dg_lock_kill remove locks held by remote lock clients

dg_lock_reset reset remote file lock database, start lock reclaim grace period

dg_lock_wait wait for previously delayed lock requests to complete

dg_mknod create a file system node

dg_mount mount a file system

dg_mstat get file status

dg_paging_info determine residency of memory pages

dg_process_info get information about the system's currently active processes

dg_setsecretkey store a client's secret key in the keyserver

dg_set_cpd_limits change the resource limits of a control point directory

dg_stat get extended file status information

dg_sysctl perform system configuration and control functions

dg_sys_info get system information

dg_unbuffered_read synchronously read data from a file without system buffering

dg_unbuffered_write synchronously write data to a file without system buffering

dg_xtrace extended process trace

dup duplicate an open file descriptor

dup2 duplicate an open file descriptor onto a specific descriptor

exec execute a file

exit terminate process

exportfs make a directory available for mounting via NFS

fchdir change the working directory of the calling process

fchmod change mode of file

fchown change user id and group id of a file

fcntl file descriptor control

fetch_and_add indivisible fetch and add to memory location

fork create a new process

fstat get file status

fstatfs get information about a mounted file system

fstatvfs return information about a file system

fsync synchronize a file's in-core state with that on disk

getcontext get and set current user context

getdents get directory entries in a filesystem-independent format

getdomainname get name of current domain

getdtablesize return the number of open files the current process can have

getegid get the effective-group-id

geteuid get the effective-user-id

getfh return the file handle of the export entry containing filename

getgid get the real-group-id

getgroups get or set supplementary group access list IDs

gethostid get unique identifier of current host

gethostname get name of current host

getitimer get or set value of interval timer

getmsg get a message from a stream

getpagesize get the system page size

getpeername get name of connected peer

getpgrp get process group ID

getpgid get process group

getpid	get process, process group, and parent process IDs
getppid	get parent process-id
getpriority	get process scheduling priority
getpsr	return the current contents of the processor status register
getrlimit	control maximum system resource consumption
getrusage	get information about resource utilization
getsid	get session ID
getsockname	get socket name
getsockopt	get options on a socket
gettimeofday	get date and time
getuid	get the real-user-id
ioctl	control a device
kill	send a signal to a process
killpg	send signal to a process or a process group
link	create a new link to a file
listen	listen for connections on a socket
lseek	change object pointer's current position
lstat	get file status
memcntl	memory management control
memctl	set memory access for mapping
mincore	determine residency of memory pages
mkdir	create a directory file
mknod	create a file entry in the file system
mmap	map pages of memory
mount	mount a file system
mprotect	set memory access for mapping
msgctl	get or set message queue attributes or destroy a message queue
msgget	get message queue identifier
msgrcv	receive a message
msgsnd	send a message
msgsys	perform a message queue operation
munmap	unmap pages of memory
nfssvc	start an NFS server on a specified socket
nice	change priority of a process
open	open file for reading or writing
pathconf	get configurable pathname values
pause	suspend process until a signal is caught
pipe	create an interprocess channel
plock	lock data, text, or both into memory
poll	input/output multiplexing
profil	set up execution time profiling for a process
ptrace	process trace
putmsg	pass a message down a stream
read	read from an object
readlink	read the contents of a symbolic link
readv	read from file
reboot	reboot halts and optionally reboots the system processor(s)
recv	receive a message from a socket
recvfrom	receive a message from a socket
recvmsg	receive a message from a socket
rename	change the name of a file
rmdir	remove a directory file
sbrk	change data segment space allocation
select	examine file descriptors for I/O readiness
semctl	semaphore control operations
semget	get a set of semaphores
semop	semaphore operations
semsys	perform a semaphore operation
send	send a message from a socket
sendmsg	send a message from a socket
sendto	send a message from a socket
setdomainname	set name of current domain
setegid	set the effective group id of the current process
seteuid	set the effective user id of the current process

setgid set the real-, effective-, and saved-group-ids
sethostid set unique identifier of current host
sethostname set name of current host
setpgid set process group ID for job control
setpgrp set process-group-id
setpgrp2 set process-group-id
setpriority set process scheduling priority
setpsr set the processor status register
setregid set the real-, effective-, and saved-group-ids
setreuid set the real-, effective-, and saved-user-ids
setsid create session and set process group ID
setsockopt set options on sockets
settimeofday set date and time
setuid set the real-, effective-, and saved-user-ids
shmat attach a shared memory segment
shmctl shared memory control operations
shmdt detach a shared memory segment
shmget get shared memory segment
shmsys perform a shared memory operation
shutdown shut down part of a full-duplex connection
sigaction examine and change signal action
sigaltstack set or get signal alternate stack context
sigblock add to set of blocked signals
sigfillset fill in the set of implementation-defined signals
sighold add a signal to the calling process's set of blocked signals
sigignore set the signal action of a signal to 'ignore'
signal specify what to do upon presentation of a signal
sigpause clear a blocked signal and suspend the process until a signal is caught
sigpending examine pending signals
sigprocmask examine and change blocked signals
sigrelse remove a signal from the calling process's set of blocked signals
sigret restore the process state to that contained in a signal frame
sigsend send a signal to a process or a group of processes
sigset specify what to do upon presentation of a signal
sigsetmask specify set of blocked signals
sigstack set and/or get signal stack context
sigsuspend wait for a signal
sigvec specify what to do upon presentation of a signal
socket create an endpoint for communication
socketpair create a pair of connected sockets
stat get file status
statfs get information about a mounted file system
statvfs return information about a file system
stime set time
stkexec set stack memory access
stkprotect set access for future stack extensions
store_conditional indivisible compare and swap
swapon add a swap device for demand paging
symlink create a symbolic link file
sync synchronize disk and memory resident file system information
sysconf get configurable system values
sysfs returns information about file system types
sysinfo get and set system information strings
sys_local invoke an extended system call
time get system time
times get process and child process times
truncate truncate a file to a specified length
uadmin request administrative shutdown and reboot options
ulimit get or set process limits
umask set and get file creation mask
umount remove a file system device
uname get name of current UNIX system
unlink remove a directory entry
ustat get file system device statistics

utime set file access and modification times
 utimes set file access and modification times
 vfork spawn new process in a virtual memory efficient way
 vhangup virtually hang up the current control terminal
 wait wait for process termination
 wait3 wait for child process to stop or terminate
 wait4 wait for the specified child process to stop or terminate
 waitid wait for child process to change state
 write write to an object
 writev write on a file

3. Subroutines and Libraries

intro introduction to subroutines and libraries
 intro introduction to math libraries
 intro introduction to network library functions
 a64l convert between long integer and base-64 ASCII string
 abort generate an abnormal termination signal
 abs return integer absolute value
 addseverity build list of severity levels for application to be used with ffmtmsg
 assert verify program assertion
 atexit add program termination routine
 basename return the last element of a path name
 bcmp compare two areas of memory
 bcopy copy bytes from one area to another
 berk_regex handle regular expressions
 berk_signal simplified software signal facilities
 bessel Bessel functions
 bgets read stream up to next delimiter
 bsearch binary search a sorted table
 bufsplit split buffer into fields
 byteorder convert values between host and network byte order
 bzero zero a portion of memory
 catgets read a program message
 catopen open/close a message catalogue
 clock report CPU time used
 conv translate characters
 copylist copy a file into memory
 crypt generate encryption
 crypt password and file encryption functions
 ctermid generate file name for terminal
 ctime convert date and time to string
 ctype character handling
 curses CRT screen handling and optimization package
 curs_addch add a character (with attributes) to a curses window
 curs_addchst add string of characters (and attributes) to a curses window
 curs_addchstr add string of characters (and attributes) to a curses window
 curs_addstr add a string of characters to a curses window and advance cursor
 curs_addwch add a wchar_t character to a curses window
 curs_addwchstr add string of wchar_t characters to a curses window
 curs_addwstr add a string of wchar_t characters to a curses window
 curs_attr curses character and window attribute control routines
 curs_beep curses bell and screen flash routines
 curs_bkgd curses window background manipulation routines
 curs_border create curses borders, horizontal and vertical lines
 curs_clear clear all or part of a curses window
 curs_color curses color manipulation routines
 curs_delch delete character under cursor in a curses window.
 curs_deleteln delete and insert lines in a curses window
 curs_getch get (or push back) characters from curses terminal keyboard
 curs_getstr get character strings from curses terminal keyboard
 curs_getwch get (or push back) wchar_t characters from curses terminal keyboard
 curs_getwstr get wchar_t character strings from curses terminal keyboard
 curs_getyx get curses cursor and window coordinates
 curs_inch get a character and its attributes from a curses window

curs_inchstr get a string of characters (and attributes) from a curses window
 curs_initscr curses screen initialization and manipulation routines
 curs_inopts curses terminal input option control routines
 curs_insch insert a character before the character under the cursor in a curses window
 curs_insstr insert string before character under the cursor in a curses window
 curs_instr get a string of characters from a curses window
 curs_inswcinsert a wchar_t character before the character under the cursor in a curses window
 curs_inswstr insert wchar_t string before character under the cursor in a curses window
 curs_inwch get a wchar_t character from a curses window
 curs_inwchstr get a string of wchar_t characters from a curses window
 curs_inwstr get a string of wchar_t characters from a curses window
 curs_kernel low-level curses routines
 curs_move move curses window cursor
 curs_outopts curses terminal output option control routines
 curs_overlay overlap and manipulate overlapped curses windows
 curs_pad create and display curses pads
 curs_printw print formatted output in curses windows
 curs_refresh refresh curses windows and lines
 curs_scanw convert formatted input from a curses widow
 curs_scroll scroll a curses window
 curs_scr_dump read (write) a curses screen from (to) a file
 curs_slk curses soft label routines
 curs_termattrs curses environment query routines
 curs_termcap curses interfaces (emulated) to the termcap library
 curs_terminfo curses interfaces to terminfo database
 curs_touch curses refresh control routines
 curs_util miscellaneous curses utility routines
 curs_window create curses windows
 cuserid get character login name or user name associated with effective UID
 dbm data base subroutines
 dg_flock apply or remove an advisory lock on an open DG/UX file
 dg_seek extended seek functions
 dg_strsignal get message string describing the given signal
 dial establish an out-going terminal line connection
 difftime computes the difference between two calendar times
 directory directory operations
 dirname report the parent directory name of a file path name
 div compute the quotient and remainder
 dlclose close a shared object
 dlerror get diagnostic information
 dlopen open a shared object
 dlsym get the address of a symbol in shared object
 doconfig execute a configuration script
 drand48 generate uniformly distributed pseudo-random numbers
 drem IEEE floating-point remainder
 ecvt convert floating-point number to string
 elf object file access library
 elf_begin make a file descriptor
 elf_cntl control a file descriptor
 elf_end finish using an object file
 elf_error error handling
 elf_fill set fill byte
 elf_flag manipulate flags
 elf_fsize return the size of an object file type
 elf_getarhdr retrieve archive member header
 elf_getarsym retrieve archive symbol table
 elf_getbase get the base offset for an object file
 elf_getdata get section data
 elf_getehdr retrieve class-dependent object file header
 elf_getident retrieve file identification data
 elf_getphdr retrieve class-dependent program header table
 elf_getscn get section information
 elf_getshdr retrieve class-dependent section header
 elf_hash compute hash value

elf_kind determine file type
elf_next sequential archive member access
elf_rand random archive member access
elf_rawfile retrieve uninterpreted file contents
elf_strptr make a string pointer
elf_update update an ELF descriptor
elf_version coordinate library and application versions
elf_xlate class-dependent data translation
end last locations in program
erf error function and complementary error function
ethers Ethernet address mapping operations
exp exponential, logarithm, power, square root functions
exportent get exported file system information
extended_perror print an error message to standard error
extended_strerror get extended error message string
fattach attach STREAMS-based file descriptor to object in file system name space
fclose close or flush a stream
fdetach detach a name from a STREAMS-based file descriptor
ferror stream status inquiries
ffs find first set bit
floor floor, ceiling, remainder, absolute value functions
fmtmsg display a message on stderr or system console
fopen open a stream
forms character based forms package
form_cursor position forms window cursor
form_data tell if forms field has off-screen data ahead or behind
form_driver command processor for the forms subsystem
form_field connect fields to forms
form_fieldtype forms fieldtype routines
form_field_attributes format the general display attributes of forms
form_field_buffer set and get forms field attributes
form_field_info get forms field characteristics
form_field_just format the general appearance of forms
form_field_new create and destroy forms fields
form_field_opts forms field option routines
form_field_userptr associate application data with forms
form_field_validation forms field data type validation
form_hook assign application-specific routines for invocation by forms
form_new create and destroy forms
form_new_page forms pagination
form_opts forms option routines
form_page set forms current page and field
form_post write or erase forms from associated subwindows
form_userptr associate application data with forms
form_win forms window and subwindow association routines
fpgetround IEEE floating-point environment control
fread binary input/output
frexp manipulate parts of floating-point numbers
fseek reposition a file pointer in a stream
fsetpos reposition a file pointer in a stream
ftime get date and time
ftruncate set a file to a specified length
ftw walk a file tree
gamma log gamma function
getc get character or word from a stream
getcwd get pathname of current working directory
getdate convert user format date and time
getenv return value for environment name
getfsent get filesystem descriptor file entry
getgrent get group file entry
gethostent get network host entry
getlogin get login name
getmntent get file system descriptor file entry
getnetconfig get network configuration database entry

getnetent get network entry
getnetgrent get network group entry
getnetpath get /etc/netconfig entry corresponding to NETPATH component
getopt get option letter from argument vector
getpass read a password
getprotoent get protocol entry
getpw get name from UID
getpwent manipulate password file entry
getrpcent get RPC entry
getrpcport get RPC port number
gets get a string from a stream
getservent get service entry
getspent manipulate shadow password file entry
getsubopt parse suboptions from a string
gettext retrieve a text string
getut access utmp file entry
getwc get wchar_t character from a stream
getwd get current working directory pathname
getwidth get information of supplementary code sets
getws get a wchar_t string from a stream
gmatch shell global pattern matching
grantpt grant access to the slave pseudo-terminal device
hsearch manage hash search tables
hypot Euclidean distance function
index search for the first occurrence of a character in a string
inet Internet address manipulation routines
initgroups initialize the supplementary group access list
insque insert/remove element from a queue
isalphanum determine if a character is alphanumeric
isastream test a file descriptor
isencrypt determine whether a character buffer is encrypted
ishex determine if a character is hexadecimal
isnan determine type of floating-point number
itoa convert an integer to an ASCII character string
jobs summary of DG/UX job control facilities
l3tol convert between 3-byte integers and long integers
ldahread read the archive header of a member of a COFF archive file
ldclose close a common object file
ldfhread read the file header of a common object file
ldgetname retrieve symbol name for object file symbol table entry
ldhread manipulate line number entries of a common object file function
ldlseek seek to line number entries of a section of a common object file
ldohseek seek to the optional file header of an object file
ldopen open an object file for reading
ldrseek seek to relocation entries of a section of a common object file
ldshread read an indexed/named section header of a common object file
ldsseek seek to an indexed/named section of a common object file
ldtbindex compute index of symbol table entry of an object file
ldtbread read an indexed symbol table entry of an object file
ldtbseek seek to the symbol table of an object file
localeconv get numeric formatting information
lockf record locking on files
logname return login name of user
lsearch linear search and update
main enter a C main program
malloc memory allocator
mlock memory allocator
matherr error-handling function
mbchar multibyte character conversion
mbchar multibyte character handling
mbstring multibyte string conversion
mbstring multibyte string functions
memory memory operations
menus character based menus package

menu_attributes control menus display attributes
 menu_cursor correctly position a menus cursor
 menu_driver command processor for the menus subsystem
 menu_format set and get maximum numbers of rows and columns in menus
 menu_hook assign application-specific routines for automatic invocation by menus
 menu_items connect and disconnect items to and from menus
 menu_item_current set and get current menus items
 menu_item_name get menus item name and description
 menu_item_new create and destroy menus items
 menu_item_opts menus item option routines
 menu_item_userptr associate application data with menus items
 menu_item_value set and get menus item values
 menu_item_visible tell if menus item is visible
 menu_mark menus mark string routines
 menu_new create and destroy menus
 menu_opts menus option routines
 menu_pattern set and get menus pattern match buffer
 menu_post write or erase menus from associated subwindows
 menu_userptr associate application data with menus
 menu_win menus window and subwindow association routines
 mkdirp create, remove directories in a path
 mkfifo create a new FIFO
 mkstemp make a unique file name
 mktemp make a unique file name
 mktime converts a tm structure to a calendar time
 mlock lock (or unlock) pages in memory
 mlockall lock or unlock address space
 monitor prepare execution profile
 mp multiple precision integer arithmetic
 msync synchronize memory with physical storage
 ndbm data base subroutines
 netdir generic transport name-to-address translation
 nlist get entries from name list
 nlsgetcall get client's data passed via the listener
 nlsprovider get name of transport provider
 nlsrequest format and send listener service request message
 nl_langinfo language information
 offsetof offset of structure member
 p2open open, close pipes to and from a command
 panels character based panels package
 panel_above panels deck traversal primitives
 panel_move move a panels window on the virtual screen
 panel_new create and destroy panels
 panel_show panels deck manipulation routines
 panel_top panels deck manipulation routines
 panel_update panels virtual screen refresh routine
 panel_userptr associate application data with a panels panel
 panel_window get or set the current window of a panels panel
 pathfind search for named file in named directories
 perror print system error messages
 popen initiate pipe to/from a process
 printf print formatted output
 printf print formatted output
 psignal system signal messages
 ptsname get name of the slave pseudo-terminal device
 putc put character or word on a stream
 putenv change or add value to environment
 putpwent write password file entry
 puts put a string on a stream
 putspent write shadow password file entry
 putwc put wchar_t character on a stream
 putws put a wchar_t string on a stream
 qsort quicker sort
 raise send signal to program

rand simple random-number generator
random generate random numbers better, or change the generator
rcmd routines for returning a stream to a remote command
realpath returns the real file name
regcmp compile and execute regular expression
regcmp compile and execute regular expression
regexpr regular expression compile and match routines
remove remove file
resolver make, send, and interpret packets to Internet domain name servers
rexec return stream to a remote command
rindex search for the last occurrence of a character in a string
rpc library routines for remote procedure calls
rtime get remote time
scandir scan a directory
scanf convert formatted input
scanf convert formatted input
setbuf assign buffering to a stream
setbuffer assign a buffer to a specified stream
setjmp non-local goto
setlinebuf assign line buffering for a specified stream
setlocale modify and query a program's locale
sigsetjmp a non-local goto with signal state
sigsetops manipulate sets of signals.
sinh hyperbolic functions
sleep suspend execution for interval
sputl access long integer data in a machine-independent fashion
ssignal software signals
stdio standard buffered input/output package
stdipc standard interprocess communication package
str string manipulations
strccpy copy strings, compressing or expanding escape codes
strcoll string collation
strerror get error message string
strftime convert date and time to string
string string operations
strsave allocate area large enough to hold string and move string into it
strtod convert string to double-precision number
strtol convert string to integer
strxfrm string transformation
swab swap bytes
swapcontext manipulate user contexts
syslog control system log
system issue a shell command
tcsetpgrp set terminal foreground process group id
termcap terminal independent operation routines
termios general terminal interface
tmpfile create a temporary file
tmpnam create a name for a temporary file
trig trigonometric functions
tsearch manage binary search trees
ttyname find name of a terminal
ttyslot find the slot in the utmp file of the current user
t_accept accept a connect request
t_alloc allocate a library structure
t_bind bind an address to a transport endpoint
t_close close a transport endpoint
t_connect establish a connection with another transport user
t_error produce error message
t_free free a library structure
t_getinfo get protocol-specific service information
t_getstate get the current state
t_listen listen for a connect request
t_look look at the current event on a transport endpoint
t_open establish a transport endpoint

t_optmgmt	manage options for a transport endpoint
t_rcv	receive data or expedited data sent over a connection
t_rcvconnect	receive the confirmation from a connect request
t_rcvdis	retrieve information from disconnect
t_rcvrel	acknowledge receipt of an orderly release indication
t_rcvudata	receive a data unit
t_rcvuderr	receive a unit data error indication
t_snd	send data or expedited data over a connection
t_snddis	send user-initiated disconnect request
t_sndrel	initiate an orderly release
t_sndudata	send a data unit
t_sync	synchronize transport library
t_unbind	disable a transport endpoint
ungetc	push character back onto input stream
ungetwc	push wchar_t character back into input stream
unlockpt	unlock a pseudo-terminal master/slave pair
vlimit	control maximum system resource consumption
vprintf	print formatted output of a variable argument list
vprintf	print formatted output of a variable argument list
vscanf	convert formatted input using varargs argument list
vtimes	get information about resource usage
wconv	translate characters
wctype	classify ASCII and supplementary code set characters
widec	multibyte character I/O routines
wstring	wchar_t string operations and type transformation
xdr	library routines for external data representation
ypclnt	Network Information Service client interface

4. File Formats

intro	introduction to file formats
intro	introduction to file formats
a.out	assembler and link editor output
acct	per-process accounting file format
ar	DG/UX common archive file format
checklist	list of file systems processed by fsck and ncheck
compver	compatible versions file
copyright	copyright information file
core	format of core image file
cpio	format of cpio archive
cpz	compose-key maps
depend	software dependencies files
dfm	DOS file manager
dialups	devices requiring a dial-up password.
dirent	file system independent directory entry
dumpcycle	dump cycle file for backups
dumptab	tape table file for dump2
d_passwd	log-in programs and passwords for dial-up devices
filehdr	file header for common object files
fs	file system format
fspec	format specification in text files
fstab	static information about file systems
group	group file
hfm	high sierra file manager
holidays	accounting information used to distinguish prime and non-prime days
idl	interface description language
inittab	script for init
inode	file node structure
issue	issue identification file
ldfcn	COFF executable file access routines
limits	header file for implementation-specific constants
linenum	line number entries in a common object file
mailcnfg	initialization information for mail and rmail
mailsurr	surrogate commands for routing and transport of mail
master	format of a master file

mfs memory file system
mnttab mounted file system table
netconfig network configuration database
passwd password file
pkginfo package characteristics file
pkgmap package contents description file
profile setting up an environment at login time
prototype package information file
rcsfile format of RCS file
reloc relocation information for a common object file
sccsfile format of SCCS file
scr_dump format of curses screen image file
sde-chooser execute environment-sensitive tool
sdetab software development environment data base
space disk space requirement file
strftime language specific strings
syms common object file symbol table format
system format of a kernel description file
terminfo terminal and printer capability database
timezone set default system time zone and locale
ttydefs terminal line settings information for ttymon
ttypsych directory search list for ttyname
utmp utmp and wtmp entry formats
vtc.addr SYAC VTC configuration file

5. Miscellaneous Features

intro introduction to miscellany
ascii map of ASCII character set
dg_mknod data returned by the dg_mknod system call
dg_stat data returned by dg_stat and dg_fstat system call
editread command line editor
elink Environment variable sensitive file link
environ user environment
eucliocl generic interface to EUC handling TTY drivers and modules
fcntl file control options
hier DG/UX file system hierarchy
langinfo language information constants
legend Debugging information technology
math math functions and constants
misalign handle misaligned memory access faults
nl_types native language data types
printcap printer capability data base
prof profile within a function
regexp regular expression compile and match routines
sde software development environment
siginfo signal generation information
signal base signals
stat data returned by stat system call
statfs data returned by the statfs system call
stdarg handle variable argument list
syslog.conf configuration file for syslogd system log server
tar tape archive file format
term conventional names for terminals
termcap terminal capability data base
types primitive system data types
ucontext user context
ustat data returned by the ustat system call
values machine-dependent values
varargs handle variable argument list
wstat wait status

6. Communications Protocols

dot3 IEEE 802.3 carrier sense multiple access with collision detection
 snap Subnetwork Access Protocol
 unix_ipc piping communications within a host

7. System Special Files

intro introduction to DG/UX System special files
 alp Algorithm Pool management module
 att_kbd generalized string translation module
 cied AViiON family disk subsystem
 cimd AViiON family disk subsystem
 cird AViiON family disk subsystem
 cisc AViiON family SCSI adapter subsystem
 clone open any minor device on a STREAMS driver
 connld line discipline for unique stream connections
 da AViiON family disk array subsystem
 devtty control terminal pseudo-device
 dgen second generation integrated Ethernet interface
 dsk block special disk interface
 duart Dual Asynchronous Receiver/Transmitter
 err error-logging interface
 filesystem file system organization
 grfx AViiON series workstation graphics processor
 hada AViiON family High Availability Disk Array adapter subsystem
 hken Hawk Ethernet interface
 inen integrated Ethernet interface
 insc AViiON family SCSI adapter subsystem
 iscd Integrated Synchronous Chip Driver
 kbd AViiON series workstation system keyboard
 kmem kernel logical memory
 ldterm standard STREAMS terminal line discipline module
 log interface to STREAMS error logging and event tracing
 lp DGC AViiON family line printer special files
 mem main system memory
 mouse mouse device
 ncsc AViiON family SCSI adapter subsystem
 null the null file
 pkt STREAMS Packet Mode module
 plm pseudo lock manager device interface
 prf operating system profiler
 ptem STREAMS Pseudo Terminal Emulation module
 pty pseudo-terminal master/slave pseudo-device pair
 rdsk character special disk interface
 rmt character special magnetic tape interface
 sad STREAMS Administrative Driver
 sd AViiON family disk subsystem
 ssid Streams Synchronous Interface Driver
 st AViiON family tape subsystem
 streamio STREAMS ioctl commands
 syac AViiON family intelligent asynchronous controller
 syscon DG/UX operating system console pseudo-device
 termio general terminal interface
 termiox extended general terminal interface
 timod Transport Interface cooperating STREAMS module
 tirdwr Transport Interface read/write interface STREAMS module
 ttcompat V7, 4BSD and XENIX STREAMS compatibility module
 vitr Vilya TokenRing Controller interface
 wmt pseudo WORM (Write Once Read Multiple optical device) as magtape interface
 zero source of zeroes

8. System Maintenance Procedures

intro	introduction to system maintenance procedures
crash	what to do when the DG/UX system crashes

PERMUTED INDEX

collision detection dot3: IEEE 802.3 carrier sense multiple access with
 l3tol, ltol3: convert between 3-byte integers and long integers
 berk_diff3: Berkeley 3-way differential file comparison
 diff3: 3-way differential file comparison
 PostScript translator for tektronix 4014 files /posttek:
 module tcompat: V7, 4BSD and XENIX STREAMS compatibility
 PostScript translator for Diablo 630 files /postdaisy:
 collision detection /dot3: IEEE 802.3 carrier sense multiple access with
 and base-64 ASCII string a64l, l64a: convert between long integer
 abort: generate an abnormal termination signal
 abort(3C)
 signal
 abs, labs: return integer absolute value
 absolute value
 absolute value functions /fabsf, rint, floor(3M)
 remainder: floor, ceiling, remainder, accept a connect request
 t_accept(3N)
 accept: accept a connection on a socket
 accept(2)
 accept: accept a connection on a socket
 accept(2)
 accept binary messages /ckbinarsys: ckbinarsys(1M)
 accept or reject print requests
 accept(1M)
 accept, reject: accept or reject print
 accept(1M)
 /touch: update access and modification times of a file
 touch(1)
 utime: set file access and modification times
 utime(2)
 utimes: set file access and modification times
 utimes(2)
 sacadm: service access controller administration
 sacadm(1M)
 sac: service access controller
 sac(1M)
 synchronize hardware caches for execute access /csync:
 csync(2)
 file
 access: determine the accessibility of a
 access(2)
 elf_next: sequential archive member
 elf_next(3E)
 elf_rand: random archive member
 elf_rand(3E)
 misalign: handle misaligned memory
 misalign(5)
 stkprotect: set access for future stack extensions
 stkprotect(2)
 memctl: set memory access for mapping
 memctl(2)
 mprotect: set memory access for mapping
 mprotect(2)
 elf: object file
 elf(3E)
 get or set supplementary group access list IDs /getgroups, setgroups:
 getgroups(2)
 initialize the supplementary group access list /initgroups:
 initgroups(3C)
 machine-independent/ spctl, sgetl: access long integer data in a
 spctl(3X)
 snap: Subnetwork Access Protocol
 snap(6P)
 ldfcn: COFF executable file access routines
 ldfcn(4)
 stkexec: set stack memory
 stkexec(2)
 trelease: terminate
 trelease(1)
 taccess: initiate
 taccess(1)
 device grantpt: grant access to the slave pseudo-terminal
 grantpt(3C)
 pututline, setutent, endutent, utmpname: access utmp file entry /getutline,
 getut(3C)
 dot3: IEEE 802.3 carrier sense multiple access with collision detection
 dot3(6P)
 access: determine the accessibility of a file
 access(2)
 acct: enable or disable process
 acct(2)
 acctcon1, acctcon2: connect-time
 acctcon(1M)
 acctprc1, acctprc2: process
 acctprc(1M)
 startup, turnacct: shell procedures for
 acctsh(1M)
 /acctdusg, accton, acctwtmp: overview of accounting and miscellaneous
 acct(1M)
 overview of accounting and miscellaneous accounting commands /accton, acctwtmp:
 acct(1M)
 diskusg: generate disk accounting data by user id
 diskusg(1M)
 acct: per-process accounting file format
 acct(4)
 acctcom: search and print process
 acctcom(1)
 acctmerg: merge or add total accounting files
 acctmerg(1M)
 distinguish prime and/ holidays: accounting information used to
 holidays(4)
 command summary from per-process accounting records /acctcms:
 acctcms(1M)
 fwtmp, wtmpfix: manipulate connect accounting records
 fwtmp(1M)
 runacct: run daily
 runacct(1M)
 /admaccounting: manage accounting system
 admaccounting(1M)
 acct: enable or disable process
 acct(2)
 acct: per-process accounting file format
 acct(4)
 acctcms: command summary from
 acctcms(1M)
 acctcom: search and print process
 acctcom(1)
 acctcon1, acctcon2: connect-time
 acctcon(1M)
 acctcon2: connect-time accounting
 acctcon(1M)
 acctdisk, acctdusg, accton, acctwtmp:
 acct(1M)
 acctdusg, accton, acctwtmp: overview of
 acct(1M)

	admether: manage ether database	admether(1M)
	admfilesystem: manage file systems	admfilesystem(1M)
files and directories	admfinfo: display information about	admfinfo(1M)
the group database	admgroup: manage group information in	admgroup(1M)
	admhost: manage hosts database	admhost(1M)
	admin: create and administer SCCS files	admin(1)
	administer a new user login on the	useradd(1M)
system useradd:	administer filters used with the LP	lpfilter(1M)
print service lpfilter:	administer forms used with the LP print	lpforms(1M)
service /lpforms:	administer SCCS files	admin(1)
admin: create and	administration interface /sysadm,	sysadm(1M)
asysadm, xsysadm: menu-driven system	administration	nlsadmin(1M)
nlsadmin: network listener service	administration	pmadm(1M)
pmadm: port monitor	administration program	osysadm(1M)
osysadm: menu-driven system	administration	sacadm(1M)
sacadm: service access controller	Administrative Driver	sad(7)
sad: STREAMS	administrative shutdown and reboot	uadmin(2)
options uadmin: request	admpinterface: manage the TCP/IP	admpinterface(1M)
network interfaces database	admkernel: manipulate the system's	admkernel(1M)
kernel	admlock: manage simple process	admlock(1M)
synchronization	admnetwork: manage network database	admnetwork(1M)
	admnl: manipulate national language	admnl(1M)
variables	admpackage: manage DG/UX-style software	admpackage(1M)
packages	admportmonitor: manage port monitors	admportmonitor(1M)
	admportservice: manage port monitor	admportservice(1M)
services	admprocess: manage processes	admprocess(1M)
	admrelease: manage software release	admrelease(1M)
areas	admresolve: manage DNS resolver's domain	admresolve(1M)
name and nameservers database	admroute: manage routing databases	admroute(1M)
	admshell: manage the remote and	admshell(1M)
restricted shell names	admsar: manage system activity	admsar(1M)
monitoring and reporting	admservice: manage service database	admservice(1M)
	admsnmpcommunity: manage the SNMP	admsnmpcommunity(1M)
community database	admsnmpobject: manage the snmpd object	admsnmpobject(1M)
database	admsnmptrap: manage the SNMP traps	admsnmptrap(1M)
database	admsvcorder: manage search order for	admsvcorder(1M)
/etc/hosts, NIS, and DNS databases	admswap: manage swap areas	admswap(1M)
	admtape: manipulate the default	admtape(1M)
parameters for tapes	admtcpipdaemon: manage the TCP/IP	admtcpipdaemon(1M)
servers	admtcpiiparams: manage the TCP/IP host	admtcpiiparams(1M)
parameters	admterminal: manage terminal ports	admterminal(1M)
	admtrustedhost: manage the trusted hosts	admtrustedhost(1M)
database	admuser: manage user information in the	admuser(1M)
password database	admxtterminal: manage serving of X	admxtterminal(1M)
display terminals	advance cursor /mvwaddnstr: add a string	cursor_addstr(3X)
of characters to a curses window and	advance: regular expression compile and	regexp(5)
match routines /regexp: compile, step,	advance: regular expression compile and	regexpr(3G)
match routines /regexpr: compile, step,	advisory lock on an open DG/UX file	dg_flock(3C)
dg_flock: apply or remove an	agent	cron(1M)
cron: clock	ahead or behind /data_behind:	form_data(3X)
tell if forms field has off-screen data	alarm clock	alarm(2)
alarm: set a process	alarm: set a process alarm clock	alarm(2)
	alp: Algorithm Pool management module	alp(7)
alp:	alias information in the aliases	admalias(1M)
database admalias: manage mail	alias names	mailalias(1)
mailalias: translate mail	aliases database /admalias:	admalias(1M)
manage mail alias information in the	allocate a library structure	t_alloc(3N)
t_alloc:	allocate area large enough to hold	strsave(3C)
string and move/ strsave, strnsave:	allocation	brk(2)
brk: change data segment space	allocation limits for a control point	cpd(1)
directory /cpd: change or view the	allocation	sbrk(2)
sbrk: change data segment space	allocator /malloc, free, realloc,	malloc(3C)
calloc, memalign, valloc,: memory	allocator /malloc, free, realloc,	malloc(3X)
calloc, mallopt, mallinfo: memory	alp: Algorithm Pool management module	alp(7)
	ALP STREAMS module	alp(1)
alpq: query the	alphanumeric	isalphanum(3C)
isalphanum: determine if a character is	alphasort: scan a directory	scandir(3C)
scandir,	alp: query the ALP STREAMS module	alp(1)
	alter priority of running processes	renice(1)
renice:	alternate stack context	sigaltstack(2)
sigaltstack: set or get signal	(and attributes) from a curses window	cursor_inchstr(3X)
/mvwinchnstr: get a string of characters		

/mvwaddchnstr: add string of characters	(and attributes) to a curses window	curs_addchst(3X)
/mvwaddchnstr: add string of characters	(and attributes) to a curses window	curs_addchstr(3X)
sigstack: set	and/or get signal stack context	sigstack(2)
sort: sort	and/or merge files	sort(1)
commands for reading and writing IBM and	ANSI tapes /REELexchange:	reelexchange_intro(1)
a prompt; verify and return a string	answer /ckstr: display	ckstr(1)
pkgask: stores	answers to a request script	pkgask(1M)
field_just: format the general	appearance of forms /set_field_just,	form_field_just(3X)
/panel_userptr: associate	application data with a panels panel	panel_userptr(3X)
/field_userptr: associate	application data with forms	form_field_userptr(3X)
/form_userptr: associate	application data with forms	form_userptr(3X)
/item_userptr: associate	application data with menu items	menu_item_userptr(3X)
/menu_userptr: associate	application data with menus	menu_userptr(3X)
intro: introduction to commands and	application programs	intro(1)
intro: introduction to commands and	application programs	intro(1)
to system maintenance commands and	application programs /introduction	intro(1M)
/build list of severity levels for	application to be used with fmtmsg	addseverity(3C)
/elf_version: coordinate library and	application versions	elf_version(3E)
/set_menu_term, menu_term: assign	application-specific routines for/	menu_hook(3X)
/set_field_term, field_term: assign	application-specific routines for/	form_hook(3X)
open DG/UX file /dg_flock:	apply or remove an advisory lock on an	dg_flock(3C)
lookup	apropos: locate commands by keyword	apropos(1)
portable archives	ar: archive and library maintainer for	ar(1)
	ar: DG/UX common archive file format	ar(4)
	arbitrary-precision arithmetic language	bc(1)
/bc:	archive and library maintainer for	ar(1)
portable archives ar:	archive	cpio(4)
cpio: format of cpio	archive file /att_dump:	att_dump(1)
dump parts of an object or object	archive file format	ar(4)
ar: DG/UX common	archive file format	tar(5)
tar: tape	archive file /ldahread: read	ldahread(3X)
the archive header of a member of a COFF	archive header of a member of a COFF	ldahread(3X)
archive file ldahread: read the	archive member access	elf_next(3E)
elf_next: sequential	archive member access	elf_rand(3E)
elf_rand: random	archive member header	elf_getarhdr(3E)
/elf_getarhdr: retrieve	archive symbol table	elf_getarsym(3E)
/elf_getarsym: retrieve	archiver	tar(1)
tar: tape file	archives /ar: archive	ar(1)
and library maintainer for portable	archives in and out	cpio(1)
cpio: copy file	area large enough to hold string and	strsave(3C)
move string/ strsave, strnsave: allocate	area to another	bcopy(3C)
bcopy: copy bytes from one	areas	admrelease(1M)
admrelease: manage software release	areas	admswap(1M)
admswap: manage swap	areas of memory	bcmp(3C)
bcmp: compare two	argument list	stdarg(5)
stdarg: handle variable	argument list	varargs(5)
varargs: handle variable	argument list /vfprintf, vsprintf:	vprintf(3S)
print formatted output of a variable	argument list /vfprintf, vsprintf:	vprintf(3W)
print formatted output of a variable	argument list /vscanf, vscanf, vsscanf:	vscanf(3S)
convert formatted input using varargs	argument list(s) and execute command	xargs(1)
xargs: construct	argument vector	getopt(3C)
getopt: get option letter from	arguments as an expression	expr(1)
expr: evaluate	arguments	echo(1)
echo: echo	arithmetic language	bc(1)
bc: arbitrary-precision	arithmetic /mout, omout, fmout, m_out,	mp(3X)
sdiv, itom: multiple precision integer	Array adapter subsystem /hada:	hada(7)
AViiON family High Availability Disk	array	dg_allow_shared_descriptor_attach(
/let processes attach shared descriptor	array /attach	dg_attach_to_shared_descriptors(2)
another process's shared descriptor	array subsystem	da(7)
da: AViiON family disk	Array subsystem /gridman: menu interface	gridman(1M)
for maintaining a High Availability Disk	arrival of new mail	notify(1)
notify: notify user of the	as an expression	expr(1)
expr: evaluate arguments	as magtape interface /pseudo WORM (Write	wmt(7)
Once Read Multiple optical device)	as: MC88000 assembler	asa(1)
	ASA carriage control characters	asa(1)
asa: interpret	asa: interpret ASA carriage control	asa(1)
characters	ascftime: convert date and time to	strftime(3C)
string strftime, cftime,	ASCII and supplementary code set/	wctype(3W)
/isenglish, isnumber, isspecial: classify	ASCII character set	ascii(5)
ascii: map of	ASCII character string	itoa(3C)
itoa: convert an integer to an	ascii: map of ASCII character set	ascii(5)

convert between long integer and base-64 string /ctime, localtime, gmtime, /trig: sin, sinf, cos, cosf, tan, tanf, /sin, sinf, cos, cosf, tan, tanf, asin, sinh, cosh, coshf, tanh, tanhf, a.out: as: MC88000 sifilter: preprocess MC88100	ASCII string /a64l, l64a: a64l(3C) asctime, tzset: convert date and time to ctime(3C) asin, asinf, acos, acosf, atan, atanf, / trig(3M) asinh, acosh, atanh: hyperbolic/ sinh(3M) assembler and link editor output a.out(4) assembler as(1) assembly language sifilter(1) assert: verify program assertion assert(3X) assertion assert(3X) assign a buffer to a specified stream setbuffer(3C) assign application-specific routines for/ menu_hook(3X) assign application-specific routines for/ form_hook(3X) assign buffering to a stream setbuf(3S) assign line buffering for a specified setlinebuf(3C) associate application data with a panels panel_userptr(3X) associate application data with forms form_field_userptr(3X) associate application data with forms form_userptr(3X) associate application data with menus menu_item_userptr(3X) associate application data with menus menu_userptr(3X) associated subwindows /post_form, form_post(3X) associated subwindows /post_menu, menu_post(3X) associated with a file descriptor close(2) associated with effective UID /cuserid: cuserid(3S) association routines /form_sub, form_win(3X) association routines /menu_sub, menu_win(3X) async_daemon: start a BIOD server for async_daemon(2) asynchronous controller syac(7) asynchronous I/O requests async_daemon(2) Asynchronous Receiver/Transmitter duart(7) asysadm, xsysadm: menu-driven system sysadm(1M) at a higher or lower priority nice(1) at a later time at(1) at a time more(1) at a time /pg: display pg(1) at, batch: execute commands at a later at(1) at login time profile(4) at or batch atrm(1) at specified times atq(1) at the current event on a transport t_look(3N) atan, atanf, atan2, atan2f: /cos, cosf, trig(3M) atan2, atan2f: trigonometric functions trig(3M) atan2f: trigonometric functions /asin, trig(3M) atanf, atan2, atan2f: trigonometric/ trig(3M) atanh: hyperbolic functions /sinhf, sinh(3M) atexit: add program termination routine atexit(3C) atof,: convert string to strtod(3C) atoi: convert string to integer strtol(3C) atol, atoi: convert string to integer strtol(3C) atq: display the jobs queued to run at atq(1) atrm: remove jobs spooled by at or batch atrm(1) attach a shared memory segment shmat(2) /dg_attach_to_shared_descriptors: dg_attach_to_shared_descriptors(2) /let processes dg_allow_shared_descriptor_attach(2) object in file system name/ /fattach: fattach(3C) modes kbdset: kbdset(1) object archive file att_dump(1) module att_kbd(7) kbdset: attach to kbdset(1) kbdcomp: compile kbdcomp(1M) kbdload: load or link kbdload(1M) wstandout: curses character and window curs_attr(3X) devattr: lists device devattr(1M) set_max_field: set and get forms field form_field_buffer(3X) mvinch, mvwinch: get a character and its curs_inch(3X) /get a string of characters (and curs_inchstr(3X) menu_pad: control menus display menu_attributes(3X) field_pad: format the general display form_field_attributes(3X) msgctl: get or set message queue msgctl(2) rcs: change RCS file rcs(1) /wechochar: add a character (with curs_addch(3X)
--	---

/add string of characters (and	attributes) to a curses window	curs_addchst(3X)
/add string of characters (and	attributes) to a curses window	curs_addchstr(3X)
attrset, wattrset, standend,/ curs_attr:	attroff, wattroff, attron, wattron,	curs_attr(3X)
curs_attr: attrroff, wattrroff,	attron, wattron, attrset, wattrset,/	curs_attr(3X)
/attroff, wattroff, attron, wattron,	attrset, wattrset, standend, wstandend,/	curs_attr(3X)
auth_destroy, authnone_create,	authdes_create, authdes_getucured,/	rpc(3N)
/authnone_create, authdes_create,	authdes_getucured, authunix_create,/	rpc(3N)
authdes_create, authdes_getucured,/	auth_destroy, authnone_create,	rpc(3N)
authdes_getucured,/ auth_destroy,	authnone_create, authdes_create,	rpc(3N)
authdes_create, authdes_getucured,	authunix_create,/ /authnone_create,	rpc(3N)
/authdes_getucured, authunix_create,	authunix_create_default, callrpc,/	rpc(3N)
/assign application-specific routines for	automatic invocation by menus	menu_hook(3X)
autopush: configure	automatically pushed STREAMS modules	autopush(1M)
messages /vacation:	automatically respond to incoming mail	vacation(1)
STREAMS modules	autopush: configure automatically pushed	autopush(1M)
subsystem hada: AViiON family High	Availability Disk Array adapter	hada(7)
/menu interface for maintaining a High	Availability Disk Array subsystem	gridman(1M)
exportfs: make a directory	available for mounting via NFS	exportfs(2)
da:	AViiON family disk array subsystem	da(7)
cied:	AViiON family disk subsystem	cied(7)
cimd:	AViiON family disk subsystem	cimd(7)
cird:	AViiON family disk subsystem	cird(7)
sd:	AViiON family disk subsystem	sd(7)
Array adapter subsystem hada:	AViiON family High Availability Disk	hada(7)
controller /syac:	AViiON family intelligent asynchronous	syac(7)
/lp: DGC	AViiON family line printer special files	lp(7)
cisc:	AViiON family SCSI adapter subsystem	cisc(7)
insc:	AViiON family SCSI adapter subsystem	insc(7)
ncsc:	AViiON family SCSI adapter subsystem	ncsc(7)
st:	AViiON family tape subsystem	st(7)
processor grfx:	AViiON series workstation graphics	grfx(7)
keyboard kbd:	AViiON series workstation system	kbd(7)
wait:	await completion of process	wait(1)
language nawk,	awk: pattern scanning and processing	nawk(1)
/mvgetch, mvwgetch, ungetch: get (or push	back) characters from curses terminal/	curs_getch(3X)
ungetwc: push wchar_t character	back into input stream	ungetwc(3W)
ungetc: push character	back onto input stream	ungetc(3S)
/mvwgetwch, ungetwch: get (or push	back) wchar_t characters from curses/	curs_getwch(3X)
/wbkgdset, bkgd, wbkgd: curses window	background manipulation routines	curs_bkgd(3X)
admbackup: manage	backup and recovery of file systems	admbackup(1M)
dump2: incremental file system	backup	dump2(1M)
tapesave: daily/weekly file system	backup /filesave,	filesave(1M)
frec: recover files from a	backup tape	frec(1M)
dumpcycle: dump cycle file for	backups	dumpcycle(4M)
pg: display file forward or	backward one screenful at a time	pg(1)
a text string from a message data	banner: make posters	banner(1)
/elf_getbase: get the	base /gettxt: retrieve	gettxt(1)
printcap: printer capability data	base offset for an object file	elf_getbase(3E)
software development environment data	base	printcap(5)
signal:	base /sdetab:	sdetab(4)
store, delete, firstkey, nextkey: data	base signals	signal(5)
dbm_error, dbm_clearerr: data	base subroutines /dbm_init, fetch,	dbm(3X)
termcap: terminal capability data	base subroutines /dbm_nextkey,	ndbm(3C)
164a: convert between long integer and	base	termcap(5)
forms: character	base-64 ASCII string /a64l,	a64l(3C)
menus: character	based forms package	forms(3X)
getdev: lists devices	based menu package	menus(3X)
screen-oriented (visual) display editor	based on criteria	getdev(1M)
panels: character	based on ex /vi, view, vedit:	vi(1)
path names	based panels package	panels(3X)
path name	basename, dirname: deliver portions of	basename(1)
for a text string in, message data	basename: return the last element of a	basename(3G)
atrm: remove jobs spooled by at or	bases /display contents of, or search	srchtxt(1)
/at,	batch	atrm(1)
killchar, longname,/ /curs_termattrs:	batch: execute commands at a later time	at(1)
language	baudrate, erasechar, has_ic, has_il,	curs_termattrs(3X)
another	bc: arbitrary-precision arithmetic	bc(1)
protocols, group or services/	bcmp: compare two areas of memory	bcmp(3C)
	bcopy: copy bytes from one area to	bcopy(3C)
	bcs_cat: type hosts, networks, passwd,	bcs_cat(1M)
	bdiff: big diff	bdiff(1)

cb: C program	beautifier	cb(1)
su:	become super-user or another user	su(1)
flash routines	beep, flash: curses bell and screen	curs_bEEP(3X)
/mvwinsstr, mvwinsnstr: insert string	before character under the cursor in a/	curs_Insstr(3X)
/mvwinswstr: insert wchar_t string	before character under the cursor in a/	curs_InsWSTR(3X)
a/ /mvinsch, mvwinsch: insert a character	before the character under the cursor in	curs_Insch(3X)
a/ /mvwinswch: insert a wchar_t character	before the character under the cursor in	curs_InsWch(3X)
starter: information for	beginning users	starter(1)
forms field has off-screen data ahead or	behind /data_ahead, data_behind: tell if	form_data(3X)
curs_bEEP: beep, flash: curses	bell and screen flash routines	curs_bEEP(3X)
and directory comparator	berk_diff: Berkeley differential file	berk_diff(1)
file comparison	berk_diff3: Berkeley 3-way differential	berk_diff3(1)
comparison	berk_diff3: Berkeley 3-way differential file	berk_diff3(1)
comparator /berk_diff:	Berkeley differential file and directory	berk_diff(1)
handle regular expressions	berk_regex, regex, re_comp, re_exec:	berk_regex(3C)
signal facilities	berk_signal, signal: simplified software	berk_signal(3C)
suspend process until a signal is/	berk_sigpause: set blocked signals and	berk_sigpause(2)
bessel: j0, j1, jn, y0, y1, yn:	Bessel functions	bessel(3M)
functions	bessel: j0, j1, jn, y0, y1, yn: Bessel	bessel(3M)
/setstate: generate random numbers	better, or change the generator	random(3C)
	bfs: big file scanner	bfs(1)
	bgets: read stream up to next delimiter	bgets(3G)
	big diff	bdiff(1)
bdiff:	big file scanner	bfs(1)
whereis: locate source,	binary, and or manual for program	whereis(1)
uencode, undecode: encode/decode a	binary file for transmission via mail	uencode(1)
printable strings in an object or other	binary file /strings: find the	strings(1)
fread, fwrite:	binary input/output	fread(3S)
whether remote system can accept	binary messages /ckbinarsys: determine	ckbinarsys(1M)
bsearch:	binary search a sorted table	bsearch(3C)
tsearch, tfind, tdelete, twalk: manage	binary search trees	tsearch(3C)
bind:	bind a name to a socket	bind(2)
/t_bind:	bind an address to a transport endpoint	t_bind(3N)
	bind: bind a name to a socket	bind(2)
requests /async_daemon: start a	BIOD server for asynchronous I/O	async_daemon(2)
	biod: start block I/O servers	biod(1M)
ffs: find first set	bit	ffs(3C)
postdmd: PostScript translator for DMD	bitmap files	postdmd(1)
reset: reset the teletype	bits to a sensible state	reset(1)
curs_bkgd: bkgdset, wbkgdset,	bkgd, wbkgd: curses window background/	curs_bkgd(3X)
window background/ /curs_bkgd:	bkgdset, wbkgdset, bkgd, wbkgd: curses	curs_bkgd(3X)
sum: print checksum and	block count of a file	sum(1)
biod: start	block I/O servers	biod(1M)
dsk:	block special disk interface	dsk(7)
until a signal is/ /sigpause: clear a	blocked signal and suspend the process	sigpause(2)
until a signal is/ /berk_sigpause: set	blocked signals and suspend process	berk_sigpause(2)
sigblock: add to set of	blocked signals	sigblock(2)
a signal to the calling process's set of	blocked signals /sighold: add	sighold(2)
sigprocmask: examine and change	blocked signals	sigprocmask(2)
signal from the calling process's set of	blocked signals /sigrelse: remove a	sigrelse(2)
sigsetmask: specify set of	blocked signals	sigsetmask(2)
deblock: change	blocking size	deblock(1)
df: report number of free disk	blocks and inodes	df(1M)
synchronous/ vsload: download	board resident software onto VSC	vsload(1M)
dg_sysctl: display or modify	boot and dump parameters	dg_sysctl(1M)
create curses borders,/ /curs_border:	border, wborder, box, whline, wvline:	curs_border(3X)
/box, whline, wvline: create curses	borders, horizontal and vertical lines	curs_border(3X)
plock: lock data, text, or	both into memory	plock(2)
routines /panel_top: top_panel,	bottom_panel: panels deck manipulation	panel_top(3X)
borders,/ /curs_border: border, wborder,	box, whline, wvline: create curses	curs_border(3X)
allocation	brk: change data segment space	brk(2)
info: documentation	browser	info(1)
	bsearch: binary search a sorted table	bsearch(3C)
bufsplit: split	buffer into fields	bufsplit(3G)
isencrypt: determine whether a character	buffer is encrypted	isencrypt(3G)
set and get menus pattern match	buffer /set_menu_pattern, menu_pattern:	menu_pattern(3X)
setbuffer: assign a	buffer to a specified stream	setbuffer(3C)
stdio: standard	buffered input/output package	stdio(3S)
read data from a file without system	buffering /synchronously	dg_unbuffered_read(2)
write data to a file without system	buffering /synchronously	dg_unbuffered_write(2)
setlinebuf: assign line	buffering for a specified stream	setlinebuf(3C)

setbuf, setvbuf: assign	buffering to a stream	setbuf(3S)
menu item ckitem:	bufsplit: split buffer into fields	bufsplit(3G)
mknod:	build a menu; prompt for and return a	ckitem(1)
application to be used/ /addseverity:	build a special file	mknod(1M)
sccstorcs:	build list of severity levels for	addseverity(3C)
elf_fill: set fill	build RCS file from SCCS file	sccstorcs(1)
convert values between host and network	byte	elf_fill(3E)
bcopy: copy	byte order /htonl, htons, ntohl, ntohs:	byteorder(3N)
swab: swap	bytes from one area to another	bcopy(3C)
	bytes	swab(3C)
	bzero: zero a portion of memory	bzero(3C)
set or query default version of GNU	C /default-gcc:	default-gcc(1)
cflow: generate a	C flow graph	cflow(1)
generate cross reference table from	C, Fortran and Pascal sources /xref:	xref(1)
cc:	C language compiler	cc(1)
gcc: GNU	C language compiler	gcc(1)
cpp: the	C language preprocessor	cpp(1)
main: enter a	C main program	main(3C)
cb:	C program beautifier	cb(1)
lint: a	C program checker	lint(1)
cxref: generate	C program cross-reference	cxref(1)
cscope: interactively examine a	C program	cscope(1)
ctrace: trace a	C program to debug it	ctrace(1)
/xstr: extract strings from	C programs to implement shared strings	xstr(1)
an error message file by massaging	C source /mkstr: create	mkstr(1)
csync: synchronize hardware	cache for execute access	csync(2)
	cal: print calendar	cal(1)
	calculator	dc(1)
	calendar	cal(1)
	calendar: reminder service	calendar(1)
	calendar time	mktime(3C)
mktime: converts a tm structure to a	calendar times /difftime:	difftime(3C)
computes the difference between two	call another UNIX system	cu(1)
cu:	call /dg_mknod:	dg_mknod(5)
data returned by the dg_mknod system	call /dg_stat: data	dg_stat(5)
returned by dg_stat and dg_fstat system	call	stat(5)
stat: data returned by stat system	call /statfs:	statfs(5)
data returned by the statfs system	call	sys_local(2)
sys_local: invoke an extended system	call	ustat(5)
ustat: data returned by the ustat system	calling process /chdir:	chdir(2)
change the working directory of the	calling process	chroot(2)
chroot: change the root directory of the	calling process /fchdir:	fchdir(2)
change the working directory of the	calling process's set of blocked signals	sighold(2)
/sighold: add a signal to the	calling process's set of blocked signals	sigrelse(2)
/sigrelse: remove a signal from the	caller, malloc, mallinfo: memory	malloc(3X)
allocator malloc, free, realloc,	caller, memalign, valloc,: memory	malloc(3C)
allocator malloc, free, realloc,	callrpc, clnt_broadcast, clnt_call,	rpc(3N)
clnt_destroy,/ /authunix_create_default,	calls and error numbers	intro(2)
intro: introduction to system	calls. /catexstr: extract strings	catexstr(1)
from source files, replace with catgets	calls /link,	link(1M)
unlink: exercise link and unlink system	calls /xprt_register, xprt_unregister:	rpc(3N)
library routines for remote procedure	can accept binary messages /ckbinarsys:	ckbinarsys(1M)
determine whether remote system	can have /getdtablesize: return the	getdtablesize(2)
number of open files the current process	cancel: send/cancel requests to an LP	lp(1)
print service /lp,	can_change_color, color_content,/	curs_color(3X)
/init_pair, init_color, has_colors,	capability data base	printcap(5)
printcap: printer	capability data base	termcap(5)
termcap: terminal	capability database	terminfo(4)
terminfo: terminal and printer	captainfo: convert a TERMCAP entry into	captainfo(1M)
a TERMINFO entry	carriage control characters	asa(1)
asa: interpret ASA	carrier sense multiple access with	dot3(6P)
collision detection dot3: IEEE 802.3	casual users)	edit(1)
edit: text editor (variant of ex for	cat: concatenate and type files to	cat(1)
standard output	catalog	catgets(1)
catgets: print message from message	catalogue	catopen(3C)
catopen, catclose: open/close a message	catalogue	gencat(1)
gencat: generate a formatted message	catclose: open/close a message catalogue	catopen(3C)
/catopen,	catexstr: extract strings from source	catexstr(1)
files, replace with catgets calls.	catgets calls. /catexstr: extract	catexstr(1)
strings from source files, replace with	catgets: print message from message	catgets(1)
catalog	catgets: read a program message	catgets(3C)

catalogue	catopen, catclose: open/close a message	catopen(3C)
and suspend process until a signal is	caught /set blocked signals	berk_sigppause(2)
pause: suspend process until a signal is	caught	pause(2)
suspend the process until a signal is	caught /clear a blocked signal and	sigppause(2)
halfdelay, intrflush,/ /curs_inopts:	cb: C program beautifier	cb(1)
powf, sqrt, sqrtf:/ exp, expf,	cbreak, nocbreak, echo, noecho,	curs_inopts(3X)
	cbt, log, logf, log10, log10f, pow,	exp(3M)
	cc: C language compiler	cc(1)
	cd: change working directory	cd(1)
	cdc: change the delta commentary of an	cdc(1)
SCCS delta	ceil, ceilf, copysign, fmod, fmodf,	floor(3M)
fabs, fabsf, rint,/ floor, floorf,	ceilf, copysign, fmod, fmodf, fabs,	floor(3M)
fabsf, rint,/ floor, floorf, ceil,	ceiling, remainder, absolute value/	floor(3M)
/fabs, fabsf, rint, remainder: floor,	cfgetspeed, cfsetispeed, cfsetospeed,/	termios(3C)
/tcdrain, tcflush, tcflow, cfgetospeed,	cfgetospeed, cfgetispeed, cfsetispeed,/	termios(3C)
/tcsendbreak, tcdrain, tcflush, tcflow,	cfsetispeed, cfsetospeed, tcgetpgrp,/	termios(3C)
	cfsetospeed, tcgetpgrp, tcsetpgrp,/	termios(3C)
/tcflow, cfgetospeed, cfgetispeed,	cftime, ascftime: convert date and time	strftime(3C)
/cfgetospeed, cfgetispeed, cfsetispeed,	change blocked signals	sigprocmask(2)
to string /strftime,	change blocking size	deblock(1)
sigprocmask: examine and	brk: change data segment space allocation	brk(2)
deblock:	sbrk: change data segment space allocation	sbrk(2)
brk:	chmod: change file mode	chmod(1)
sbrk:	chown: change file owner	chown(1)
chmod:	passwd: change login password	passwd(1)
chown:	chmod: change mode of file	chmod(2)
passwd:	fchmod: change mode of file	fchmod(2)
chmod:	/lseek: change object pointer's current position	lseek(2)
fchmod:	putenv: change or add value to environment	putenv(3C)
/lseek:	strchg, strconf: change or query stream configuration	strchg(1)
putenv:	a control point directory /cpd:	change or view the allocation limits for
strchg, strconf:	nice: change priority of a process	cpd(1)
change or query stream configuration	rcs: change RCS file attributes	nice(2)
change or view the allocation limits for	chroot: change root directory for a command	rcs(1)
nice:	change signal action	chroot(1M)
rcs:	change state	sigaction(2)
chroot:	Change SYAC routing information	waitid(2)
change root directory for a command	change system state	syac_routes(1M)
change signal action	change the delta commentary of an SCCS	shutdown(1M)
change state	newform: change the format of a text file	cdc(1)
Change SYAC routing information	change the generator /setstate:	newform(1)
change system state	change the group ownership of a file	random(3C)
change the delta commentary of an SCCS	change the name of a file	chgrp(1)
change the format of a text file	change the resource limits of a control	rename(2)
change the generator /setstate:	chroot: change the root directory of the calling	dg_set_cpd_limits(2)
change the group ownership of a file	change the working directory of the	chroot(2)
change the name of a file	(change) to an SCCS file	chdir(2)
change the resource limits of a control	chown, lchown: change user id and group id of a file	fchdir(2)
chroot:	fchown: change user id and group id of a file	delta(1)
change the root directory of the calling	cd: change working directory	chown(2)
change the working directory of the	setuname: changes machine information	fchown(2)
change the working directory of the	helpadm: make	cd(1)
(change) to an SCCS file	pipe: create an interprocess	setuname(1M)
change user id and group id of a file	/inch, winch, mvinch, mvwinch: get a	helpadm(1M)
change user id and group id of a file	/wstandend, standout, wstandout: curses	pipe(2)
change working directory	ungetwc: push wchar_t	cursor_inch(3X)
changes machine information	ungetc: push	cursor_attr(3X)
changes to the help facility database	forms: character based forms package	ungetwc(3W)
channel	menus: character based menus package	ungetc(3S)
character and its attributes from a/	panels: character based panels package	forms(3X)
character and window attribute control/	/wansch, mvwansch, mvwansch: insert a	menus(3X)
character back into input stream	/mvwansch, mvwansch: insert a wchar_t	panels(3X)
character back onto input stream	isencrypt: determine whether a	cursor_insch(3X)
character based forms package	tables /chrtbl: generate	cursor_inswch(3X)
character based menus package	tables /wchrtbl: generate	isencrypt(3G)
character based panels package	mbchar: mbtowc, wctomb, mblen: multibyte	chrtbl(1M)
character before the character under the/	winch, mvinch, mvwinch: get a wchar_t	wchrtbl(1M)
character before the character under the/	getwc, getwchar, fgetwc: get wchar_t	mbchar(3W)
character buffer is encrypted	ispunct, isprint, isgraph, isascii:	winch(3X)
character classification and conversion	catopen, catclose: open/close a message	cursor_inwch(3X)
character classification and conversion	caught /set blocked signals	getwc(3W)
character conversion	caught	ctype(3C)
character from a curses window /inwch,	caught /clear a blocked signal and	
character from a stream	cb: C program beautifier	
character handling /isspace, iscntrl,	cbreak, nocbreak, echo, noecho,	

mbchar:	mbtowc, mblen, wctomb: multibyte search for the first occurrence of a search for the last occurrence of a widec: multibyte isalphanum: determine if a ishex: determine if a associated with effective/ cuserid: get putwc, putwchar, fputwc: put wchar_t getc, getchar, fgetc, getw: get putc, putchar, fputc, putw: put ascii: map of ASCII rdsd: interface rmt: fgrep: search a file for a itoa: convert an integer to an ASCII /mvgetstr, mvwgetstr, mvwgetstr: get /mvwgetwstr, mvwgetwstr: get wchar_t echowchar, wechowchar: add a wchar_t /delch, wdelch, mvdelch, mvwdelch: delete /mvwvinsch: insert a character before the window /mvwvinsnstr: insert string before /insert a wchar_t character before the window /insert wchar_t string before /mvwaddch, echochar, wechochar: add a pkginfo: package dynamic_field_info: get forms field /mvwvinschstr, mvwvinschnstr: get a string of /mvwaddchstr, mvwaddchnstr: add string of /mvwaddchstr, mvwaddchnstr: add string of asa: interpret ASA carriage control _toupper, _tolower, toascii: translate /mvwvinsstr, mvwvinsnstr: get a string of /mvwvinschnstr: get a string of wchar_t /mvwvinnwstr: get a string of wchar_t /mvwvgetch, ungetch: get (or push back) /ungetwch: get (or push back) wchar_t rev: reverse order of /mvwaddstr, mvwaddnstr: add a string of /mvwaddwchnstr: add string of wchar_t /mvwaddnwstr: add a string of wchar_t tr: translate wconv: toupper, tolower: translate classify ASCII and supplementary code set monacct, nulladm, prctmp, prdaily, / the calling process pkgchk: check accuracy of installation repair them /fsck: check file systems for consistency and ci: check in RCS revisions get: check out a version of an SCCS file co: check out RCS revisions pwck, grpck: check password or group file permissions file ucheck: check the uucp directories and freeze a configuration of sources lint: a C program checker labelit: copy file systems with label processed by fsck and ncheck sum: print file chgrp: change the group ownership of a chgtnfo: create a temporary version of child process times waitid: wait for wait3: wait for wait4: wait for the specified iscd: Integrated Synchronous character handling character in a string /index: character in a string /rindex: character I/O routines character is alphanumeric character is hexadecimal character login name or user name character on a stream character or word from a stream character or word on a stream character set character special disk interface character special magnetic tape character string character string character strings from curses terminal/ character strings from curses terminal/ character to a curses window /mvwaddwch, character under cursor in a curses/ character under the cursor in a curses/ character under the cursor in a curses character under the cursor in a curses/ character under the cursor in a curses character (with attributes) to a curses/ characteristics file characteristics /field_info, characters (and attributes) from a/ characters (and attributes) to a curses/ characters (and attributes) to a curses/ characters /conv: toupper, tolower, characters from a curses window characters from curses terminal keyboard characters from curses terminal keyboard characters in each line of file characters to a curses window and/ characters to a curses window characters to a curses window characters characters characters /isnumber, isspecial: chargefee, ckpacct, dodisk, lastlogin, chdir: change the working directory of check accuracy of installation check file systems for consistency and ci(1) get(1) co(1) pwck(1M) ucheck(1M) rcsfreeze(1) lint(1) volcopy(1M) checklist(4) sum(1) chgrp(1) chgtnfo(1) times(2) waitid(2) wait3(2) wait4(2) iscd(7) chmod(1) chmod(2) chown(1) chown(2) chroot(1M) chroot(2) chrtbl(1M)	mbchar(3C) index(3C) rindex(3C) widec(3W) isalphanum(3C) ishex(3C) cuserid(3S) putwc(3W) getc(3S) putc(3S) ascii(5) rdsd(7) rmt(7) fgrep(1) itoa(3C) curs_getstr(3X) curs_getwstr(3X) curs_addwch(3X) curs_delch(3X) curs_insch(3X) curs_insstr(3X) curs_inswch(3X) curs_inswstr(3X) curs_addch(3X) pkginfo(4) form_field_info(3X) curs_inchstr(3X) curs_addchst(3X) curs_addchstr(3X) asa(1) conv(3C) curs_instr(3X) curs_inwchstr(3X) curs_inwstr(3X) curs_getch(3X) curs_getwch(3X) rev(1) curs_addstr(3X) curs_addwchstr(3X) curs_addwstr(3X) tr(1) wconv(3W) wctype(3W) acctsh(1M) chdir(2) pkgchk(1M) fsck(1M) ci(1) get(1) co(1) pwck(1M) ucheck(1M) rcsfreeze(1) lint(1) volcopy(1M) checklist(4) sum(1) chgrp(1) chgtnfo(1) times(2) waitid(2) wait3(2) wait4(2) iscd(7) chmod(1) chmod(2) chown(1) chown(2) chroot(1M) chroot(2) chrtbl(1M)
---------	--	---

	ci: check in RCS revisions	ci(1)
	cied: AViiON family disk subsystem	cied(7)
	cimd: AViiON family disk subsystem	cimd(7)
	cird: AViiON family disk subsystem	cird(7)
	cisc: AViiON family SCSI adapter	cisc(7)
	ckbinarsys: determine whether remote	ckbinarsys(1M)
	ckdate, errdate, helpdate, valdate:	ckdate(1)
	ckgid, errgid, helpgid, valgid: prompt	ckgid(1)
	ckint: display a prompt; verify and	ckint(1)
	ckitem: build a menu; prompt for and	ckitem(1)
	ckkeywd: prompt for and validate a	ckkeywd(1)
	ckpacct, dodisk, lastlogin, monacct,	acctsh(1M)
	ckpath: display a prompt; verify and	ckpath(1)
	ckrange: prompt for and validate an	ckrange(1)
	ckstr: display a prompt; verify and	ckstr(1)
	cktime: display a prompt; verify and	cktime(1)
	ckuid: prompt for and validate a user ID	ckuid(1)
	ckyorn: prompt for and validate yes/no	ckyorn(1)
	class-dependent data translation	elf_xlate(3E)
	class-dependent object file header	elf_getehdr(3E)
	class-dependent program header table	elf_getphdr(3E)
	class-dependent section header	elf_getshdr(3E)
	classification and conversion tables	chrtbl(1M)
	classification and conversion tables	wchrtbl(1M)
	classify ASCII and supplementary code set/	wctype(3W)
	clean up working files	rcsclean(1)
	cleanup program	strclean(1M)
	clean-up	uucleanup(1M)
	clear a blocked signal and suspend the	sigpause(2)
	clear all or part of a curses window	clear(1)
	clear: clear terminal screen	clear(1)
	clear inode	clear(1)
	clear terminal screen	clear(1)
	clear, wclear, clrtoeol, wclrtoeol,	clear(1)
	clearerr, fileno: stream status	clearerr(3S)
	clearok, idlok, idcok immediok, leaveok,	clearok(3S)
	client interface /yperr_string,	ypclnt(3N)
	clients	admlnt(1M)
	client's data passed via the listener	nlsgtcall(3N)
	clients /dg_lock_kill:	dg_lock_kill(2)
	client's secret key in the keyserver	dg_setsecretkey(2)
	client/server common key	dg_decryptsessionkey(2)
	client/server common key	dg_encryptsessionkey(2)
	C-like syntax /csh: invoke	csh(1)
	clnt_broadcast, clnt_call, clnt_destroy,/	rpc(3N)
	clnt_call, clnt_destroy, clnt_create,	rpc(3N)
	clnt_control, clnt_freeres, clnt_geterr,/	rpc(3N)
	clnt_create, clnt_control, clnt_freeres,/	rpc(3N)
	clnt_destroy, clnt_create, clnt_control,/	rpc(3N)
	clnt_freeres, clnt_geterr,/ /clnt_call,	rpc(3N)
	clnt_geterr, clnt_pcreateerror,/	rpc(3N)
	clnt_pcreateerror, clnt_perrno,/	rpc(3N)
	clnt_perrno, clnt_perror,/ /clnt_freeres,	rpc(3N)
	clnt_perror, clnt_spcreateerror,/	rpc(3N)
	clntraw_create, clnttcp_create,/	rpc(3N)
	clnt_spcreateerror, clnt_sperrno,	rpc(3N)
	clnt_sperrno, clnt_sperror,/	rpc(3N)
	clnt_sperror, clntraw_create,/	rpc(3N)
	clnttcp_create, clntudp_create,/	rpc(3N)
	clntudp_create, host2netname,/	rpc(3N)
	clock /adjtime: correct the time	adjtime(2)
	clock agent	cron(1M)
	clock	alarm(2)
	clock: report CPU time used	clock(3C)
	clone: open any minor device on a	clone(7)
	close a common object file	ldclose(3X)
	close a shared object	dlclose(3X)
	close a transport endpoint	t_close(3N)
	close an object associated with a file	close(2)
	close: close an object associated with a	close(2)
	close or flush a stream	fclose(3S)
	subsystem	
	system can accept binary messages	
	prompt for and validate a date	
	for and validate a group id	
	return an integer value	
	return a menu item	
	keyword	
	nulladm, prctmp, prdaily,/ chargefee,	
	return a pathname	
	integer	
	return a string answer	
	return a time of day	
	/elf32_xlatetof, elf32_xlatetom:	
	/elf32_getehdr, elf32_newehdr: retrieve	
	/elf32_getphdr, elf32_newphdr: retrieve	
	/elf_getshdr: elf32_getshdr: retrieve	
	chrtbl: generate character	
	wchrtbl: generate character	
	/isenglish, isnumber, isspecial:	
	rcsclean:	
	strclean: STREAMS error logger	
	uucleanup: uucp spool directory	
	process until a signal is/ /sigpause:	
	/wclrtoeol, clrtoeol, wclrtoeol:	
	clri:	
	clear:	
	clrtoeol,/ curs_clear: erase, werase,	
	inquiries ferror, feof,	
	setscreg, wsetscreg,/ /curs_outopts:	
	ypprot_err: Network Information Service	
	admlnt: manage operating system	
	nlsgtcall: get	
	remove locks held by remote lock	
	/dg_setsecretkey: store a	
	/decrypt conversation key with the	
	/encrypt conversation key with the	
	a shell (command interpreter) having a	
	/authunix_create_default, callrpc,	
	clnt_control,/ /callrpc, clnt_broadcast,	
	/clnt_call, clnt_destroy, clnt_create,	
	/clnt_broadcast, clnt_call, clnt_destroy,	
	/callrpc, clnt_broadcast, clnt_call,	
	clnt_destroy, clnt_create, clnt_control,	
	/clnt_create, clnt_control, clnt_freeres,	
	/clnt_control, clnt_freeres, clnt_geterr,	
	clnt_geterr, clnt_pcreateerror,	
	/clnt_pcreateerror, clnt_perrno,	
	clnt_sperrno, clnt_sperror,	
	clnt_sperror,/ /clnt_perrno, clnt_perror,	
	/clnt_perror, clnt_spcreateerror,	
	/clnt_sperror, clntraw_create,	
	/clntraw_create, clnttcp_create,	
	to allow synchronization of the system	
	cron:	
	alarm: set a process alarm	
	STREAMS driver	
	ldclose, ldaclose:	
	dlclose:	
	t_close:	
	descriptor /close:	
	file descriptor	
	fclose, fflush:	

p2open, p2close: open,	close pipes to and from a command	p2open(3G)
readdir, telldir, seekdir, rewinddir,	closedir: directory operations /opendir,	directory(3X)
/syslog, openlog,	closelog, setlogmask: control system log	syslog(3C)
/erase, werase, clear, wclear,	clri: clear inode	clri(1M)
of/ /clear, wclear, clrtoeol, wclrtoeol,	clrtoeol, wclrtoeol: clear all or part	clrtoeol(3X)
	clrtoeol, wclrtoeol: clear all or part	clrtoeol(3X)
	cmp: compare two files	cmp(1)
	co: check out RCS revisions	co(1)
dis: object	code disassembler	dis(1)
classify ASCII and supplementary	code set characters /isspecial:	wctype(3W)
iconv:	code set conversion	iconv(1)
eucset: set or get EUC	code set widths	eucset(1)
get information of supplementary	code sets /getwidth:	getwidth(3W)
strings, compressing or expanding escape	codes /streadd, strcadd, strecpy: copy	strccpy(3G)
to ELF	cof2elf: translate object file from COFF	cof2elf(1)
read the archive header of a member of a	COFF archive file /ldahread:	ldahread(3X)
ldfcn:	COFF executable file access routines	ldfcn(4)
cof2elf: translate object file from	COFF to ELF	cof2elf(1)
ctl:	COFF-to-legend translator	ctl(1)
	col: filter reverse line-feeds	col(1)
colltbl: create	collation database	colltbl(1M)
strcoll: string	collation	strcoll(3C)
802.3 carrier sense multiple access with	collision detection /dot3: IEEE	dot3(6P)
	colltbl: create collation database	colltbl(1M)
/color_content, pair_content: curses	color manipulation routines	color(3X)
color/ /has_colors, can_change_color,	color_content, pair_content: curses	color(3X)
set and get maximum numbers of rows and	columns in menus /menu_format:	menu_format(3X)
	comb: combine SCCS deltas	comb(1)
comb:	combine SCCS deltas	comb(1)
two sorted files	comm: select or reject lines common to	comm(1)
rksh: KornShell, a standard/restricted	command and programming language /ksh,	ksh(1)
nice: run a	command at a higher or lower priority	nice(1)
chroot: change root directory for a	command	chroot(1M)
/usage: retrieve a	command description and usage examples	usage(1)
env: set environment for	command execution	env(1)
uux: UNIX-to-UNIX system	command execution	uux(1)
mail_pipe: invoke recipient	command for incoming mail	mail_pipe(1M)
nohup: run a	command immune to hangups and quits	nohup(1)
syntax /csh: invoke a shell	(command interpreter) having a C-like	csh(1)
editread:	command line editor	editread(5)
getopt: parse	command options	getopt(1)
getopts, getoptcv: parse	command options	getopts(1)
p2close: open, close pipes to and from a	command /p2open,	p2open(3G)
subsystem /form_driver:	command processor for the forms	form_driver(3X)
subsystem /menu_driver:	command processor for the menus	menu_driver(3X)
sh, jsh, rsh, restsh: shell, the	command programming language	sh(1)
for returning a stream to a remote	command /rresvport, ruserok: routines	rcmd(3X)
activity /timex: time a	command; report process data and system	timex(1)
uuxqt: execute remote	command requests	uuxqt(1M)
rexec: return stream to a remote	command	rexec(3X)
accounting records acctcms:	command summary from per-process	acctcms(1M)
system: issue a shell	command	system(3S)
test: condition evaluation	command	test(1)
time: time a	command	time(1)
locate: identify a	command using keywords	locate(1)
construct argument list(s) and execute	command /xargs:	xargs(1)
accounting and miscellaneous accounting	commands /accton, acctwtmp: overview of	acct(1M)
intro: introduction to	commands and application programs	intro(1)
intro: introduction to	commands and application programs	intro(1)
/introduction to system maintenance	commands and application programs	intro(1M)
at, batch: execute	commands at a later time	at(1)
apropos: locate	commands by keyword lookup	apropos(1)
ANSI tapes /REELexchange:	commands for reading and writing IBM and	reelexchange_intro(1)
mail mailsurr: surrogate	commands for routing and transport of	mailsurr(4M)
install: install	commands	install(1M)
rcsintro: introduction to RCS	commands	rcsintro(1)
streamio: STREAMS ioctl	commands	streamio(7)
environment target /sde-target: print	commands to reset software development	sde-target(1)
mcs: manipulate the	comment section of an object file.	mcs(1)
cdc: change the delta	commentary of an SCCS delta	cdc(1)
ar: DG/UX	common archive file format	ar(4)

conversation key with the client/server	common key /decrypt	dg_decryptsessionkey(2)
conversation key with the client/server	common key /encrypt	dg_encryptsessionkey(2)
cprs: compress a	common object file	cprs(1)
manipulate line number entries of a	common object file function /ldlitem:	ldlread(3X)
ldclose, ldaclose: close a	common object file	ldclose(3X)
ldfhread: read the file header of a	common object file	ldfhread(3X)
to line number entries of a section of a	common object file /ldnlseek: seek	ldlseek(3X)
to relocation entries of a section of a	common object file /ldnrseek: seek	ldrseek(3X)
an indexed/named section header of a	common object file /ldnshread: read	ldshread(3X)
seek to an indexed/named section of a	common object file /ldsseek, ldnsseek:	ldsseek(3X)
linenum: line number entries in a	common object file	linenum(4)
nm: print name list of	common object file	nm(1)
reloc: relocation information for a	common object file	reloc(4)
/syms:	common object file symbol table format	syms(4)
filehdr: file header for	common object files	filehdr(4)
ld: link editor for	common object files	ld-coff(1)
glossary: definitions of	common terms and symbols	glossary(1)
comm: select or reject lines	common to two sorted files	comm(1)
ipcs: report inter-process	communication facilities status	ipcs(1)
stdipc: ftok: standard interprocess	communication package	stdipc(3C)
socket: create an endpoint for	communication	socket(2)
unix_ipc: piping	communications within a host	unix_ipc(6F)
/admsnmpcommunity: manage the SNMP	community database	admsnmpcommunity(1M)
Berkeley differential file and directory	comparator /berk_diff:	berk_diff(1)
diff: differential file	comparator	diff(1)
/store_conditional: indivisible	compare and swap	store_conditional(2)
descriptions	compare or print out TERMINFO	infocmp(1M)
infocmp:	compare RCS revisions	rcsdiff(1)
rcsdiff:	compare two areas of memory	bcmp(3C)
bcmp:	compare two directories	dircmp(1)
dircmp:	compare two files	cmp(1)
cmp:	compare two versions of an SCCS file	sccsdiff(1)
sccsdiff:	comparison /berk_diff3:	berk_diff3(1)
Berkeley 3-way differential file	comparison	diff3(1)
diff3: 3-way differential file	compatibility module	ttcompat(7)
ttcompat: V7, 4BSD and XENIX STREAMS	compatible versions file	compver(4)
compver:	compile and execute regular expression	regcmp(3G)
/regcmp, regex:	compile and execute regular expression	regcmp(3X)
/regcmp, regex:	compile and match routines /compile,	regexp(5)
step, advance: regular expression	compile and match routines /compile,	regexpr(3G)
step, advance: regular expression	compile att_kbd tables	kbdcomp(1M)
kbdcomp:	compile	regcmp(1)
regcmp: regular expression	compile, step, advance: regular	regexp(5)
expression compile and match/ regexp:	compile, step, advance: regular	regexpr(3G)
expression compile and match/ regexpr:	compiler	cc(1)
cc: C language	compiler	gcc(1)
gcc: GNU C language	compiler	idc(1)
idc: interface description	compiler	sno(1)
sno: SNOBOL interpreter and	compiler	tic(1M)
tic: TERMINFO	compiler	zic(1M)
zic: time zone	compiler-compiler	yacc(1)
yacc: yet another	complementary error function	erf(3M)
erf, erfc: error function and	complete /dg_lock_wait: wait	dg_lock_wait(2)
for previously delayed lock requests to	completion of process	wait(1)
wait: await	component /get /etc/netconfig	getnetpath(3N)
entry corresponding to NETPATH	compose-key maps	cpz(4M)
cpz:	compress a common object file	cprs(1)
cprs:	compress and expand files	pack(1)
pack, pcat, unpack:	compress, expand or display expanded	compress(1)
files compress, uncompress, zcat:	compress, uncompress, zcat: compress,	compress(1)
expand or display expanded files	compressing or expanding escape codes	strccpy(3G)
/streadd, strcadd, strecpy: copy strings,	compute hash value	elf_hash(3E)
elf_hash:	compute index of symbol table entry of	ldtbindex(3X)
an object file /ldtbindex:	compute the quotient and remainder	div(3C)
div, ldiv:	computes the difference between two	difftime(3C)
calendar times difftime:	compver: compatible versions file	compver(4)
output /cat:	concatenate and type files to standard	cat(1)
test:	condition evaluation command	test(1)
system log server syslog:	conf: configuration file for syslogd	syslog.conf(5)
pathconf, fpathconf: get	config: configure a system	config(1M)
	configurable pathname values	pathconf(2)

sysconf: get	configurable system values	sysconf(2)
dg_sysctl: perform system	configuration and control functions	dg_sysctl(2)
/getnetconfig: get network	configuration database entry	getnetconfig(3N)
netconfig: network	configuration database	netconfig(4)
log server syslog.conf:	configuration file for syslogd system	syslog.conf(5)
vtc.addr: SYAC VTC	configuration file	vtc.addr(4M)
under RCS rcsfreeze: freeze a	configuration of sources checked in	rcsfreeze(1)
doconfig: execute a	configuration script	doconfig(3N)
strchg, strconf: change or query stream	configuration	strchg(1)
config:	configure a system	config(1M)
modules /autopush:	configure automatically pushed STREAMS	autopush(1M)
lpadmin:	configure the LP print service	lpadmin(1M)
/t_rcvconnect: receive the	confirmation from a connect request	t_rcvconnect(3N)
fwtmp, wtmpfx: manipulate	connect accounting records	fwtmp(1M)
/set_menu_items, menu_items, item_count:	connect and disconnect items to and from/	menu_items(3X)
/form_fields, field_count, move_field:	connect fields to forms	form_field(3X)
socket	connect: initiate a connection on a	connect(2)
t_accept: accept a	connect request	t_accept(3N)
t_listen: listen for a	connect request	t_listen(3N)
receive the confirmation from a	connect request /t_rcvconnect:	t_rcvconnect(3N)
getpeername: get name of	connected peer	getpeername(2)
socketpair: create a pair of	connected sockets	socketpair(2)
establish an out-going terminal line	connection /dial:	dial(3C)
accept: accept a	connection on a socket	accept(2)
connect: initiate a	connection on a socket	connect(2)
shut down part of a full-duplex	connection /shutdown:	shutdown(2)
data or expedited data sent over a	connection /t_rcv: receive	t_rcv(3N)
send data or expedited data over a	connection /t_snd:	t_snd(3N)
/t_connect: establish a	connection with another transport user	t_connect(3N)
line discipline for unique stream	connections /connld:	connld(7)
listen: listen for	connections on a socket	listen(2)
acctcon1, acctcon2:	connect-time accounting	acctcon(1M)
stream connections	connld: line discipline for unique	connld(7)
fsck: check file systems for	consistency and repair them	fsck(1M)
display a message on stderr or system	console /fmtmsg:	fmtmsg(1)
display a message on stderr or system	console /fmtmsg:	fmtmsg(3C)
console, systty: DG/UX operating system	console pseudo-device /syscon,	syscon(7)
console pseudo-device /syscon,	console, systty: DG/UX operating system	syscon(7)
langinfo: language information	constants	langinfo(5)
header file for implementation-specific	constants /limits:	limits(4)
math: math functions and	constants	math(5)
command /xargs:	construct argument list(s) and execute	xargs(1)
deroff: remove nroff/troff, tbl, and eqn	constructs	deroff(1)
control maximum system resource	consumption /getrlimit, setrlimit:	getrlimit(2)
vlimit: control maximum system resource	consumption	vlimit(3C)
/Uutry: try to	contact remote system with debugging on	uutry(1M)
getdgrp: lists device groups which	contain devices that match criteria	getdgrp(1M)
restore the process state to that	contained in a signal frame /sigret:	sigret(2)
the file handle of the export entry	containing filename /getfh: return	getfh(2)
pkgmap: package	contents description file	pkgmap(4)
/elf_rawfile: retrieve uninterpreted file	contents	elf_rawfile(3E)
readlink: read the	contents of a symbolic link	readlink(2)
ls: list	contents of directory	ls(1)
in, message data bases /srchtxt: display	contents of, or search for a text string	srchtxt(1)
register getpsr: return the current	contents of the processor status	getpsr(2)
tsniff: summary report of tape	contents	tsniff(1)
setcontext: get and set current user	context /getcontext,	getcontext(2)
set or get signal alternate stack	context /sigaltstack:	sigaltstack(2)
sigstack: set and/or get signal stack	context	sigstack(2)
csplit:	context split	csplit(1)
ucontext: user	context	ucontext(5)
/swapcontext: manipulate user	contexts	swapcontext(3C)
ioctl:	control a device	ioctl(2)
elf_cntl:	control a file descriptor	elf_cntl(3E)
asa: interpret ASA carriage	control characters	asa(1)
jobs: summary of DG/UX job	control facilities	jobs(3C)
fcntl: file descriptor	control	fcntl(2)
IEEE floating-point environment	control /fpgetsticky, fpsetsticky:	fpgetround(3C)
perform system configuration and	control functions /dg_sysctl:	dg_sysctl(2)
init, telinit: process	control initialization	init(1M)
consumption getrlimit, setrlimit:	control maximum system resource	getrlimit(2)

consumption vlimit:	control maximum system resource	vlimit(3C)
memcntl: memory management	control	memcntl(2)
/menu_grey, set_menu_pad, menu_pad:	control menus display attributes	menu_attributes(3X)
mt: magnetic tape	control	mt(1)
semctl: semaphore	control operations	semctl(2)
shmctl: shared memory	control operations	shmctl(2)
fcntl: file	control options	fcntl(5)
or view the allocation limits for a	control point directory /cpd: change	cpd(1)
/change the resource limits of a	control point directory	dg_set_cpd_limits(2)
lpc: line printer	control program	lpc(1M)
curses character and window attribute	control routines /standout, wstandout:	curs_attr(3X)
typeahead: curses terminal input option	control routines /timeout, wtimeout,	curs_inopts(3X)
nl, nonl: curses terminal output option	control routines /wsetscrrg, scrollok,	curs_outopts(3X)
is_wintouched: curses refresh	control routines /is_linetouched,	curs_touch(3X)
setpgid: set process group ID for job	control	setpgid(2)
dkctl:	control special disk operations	dkctl(1M)
syslog, openlog, closelog, setlogmask:	control system log	syslog(3C)
devtty:	control terminal pseudo-device	devtty(7)
vhangup: virtually hang up the current	control terminal	vhangup(2)
uustat: uucp status inquiry and job	control	uustat(1)
vc: version	control	vc(1)
sacadm: service access	controller administration	sacadm(1M)
tload: load terminal	controller devices	tload(1M)
vitr: Vilya TokenRing	Controller interface	vitr(7)
verify that the VSC synchronous	controller is operable /vsccheck:	vsccheck(1M)
sac: service access	controller	sac(1M)
AViiON family intelligent asynchronous	controller /syac:	syac(7)
resident software onto VSC synchronous	controller /vsload: download board	vsload(1M)
_tolower, toascii: translate characters	conv: toupper, tolower, _toupper,	conv(3C)
term:	conventional names for terminals	term(5)
common/ /dg_decryptsessionkey: decrypt	conversation key with the client/server	dg_decryptsessionkey(2)
common/ /dg_encryptsessionkey: encrypt	conversation key with the client/server	dg_encryptsessionkey(2)
iconv: code set	conversion	iconv(1)
wctomb, mblen: multibyte character	conversion /mbchar: mbtowc,	mbchar(3W)
mbstowcs, wctombs: multibyte string	conversion /mbstring:	mbstring(3W)
units:	conversion program	units(1)
generate character classification and	conversion tables /chrtbl:	chrtbl(1M)
generate character classification and	conversion tables /wchrtbl:	wchrtbl(1M)
entry /captinfo:	convert a TERMCAP entry into a TERMINFO	captinfo(1M)
string /itoa:	convert an integer to an ASCII character	itoa(3C)
dd:	convert and copy a file	dd(1)
integers /l3tol, ltol3:	convert between 3-byte integers and long	l3tol(3C)
ASCII string /a64l, l64a:	convert between long integer and base-64	a64l(3C)
localtime, gmtime, asctime, tzset:	convert date and time to string /ctime,	ctime(3C)
strftime, cftime, asctime:	convert date and time to string	strftime(3C)
/ecvt, fcvt, gcvt:	convert floating-point number to string	ecvt(3C)
/wscanw, mvscanw, mvwscanw, vwscanw:	convert formatted input from a curses/	curs_scanw(3X)
scanf, fscanf, sscanf:	convert formatted input	scanf(3S)
scanf, fscanf, sscanf:	convert formatted input	scanf(3W)
argument list vscanf, vfscanf, vsscanf:	convert formatted input using varargs	vscanf(3S)
number strtod, atof,:	convert string to double-precision	strtod(3C)
strtoul, strtoul, atol, atoi:	convert string to integer	strtoul(3C)
getdate, getdate_err:	convert user format date and time	getdate(3C)
byte order /htonl, htons, ntohs, ntohs:	convert values between host and network	byteorder(3N)
time mktime:	converts a tm structure to a calendar	mktime(3C)
timod: Transport Interface	cooperating STREAMS module	timod(7)
versions /elf_version:	coordinate library and application	elf_version(3E)
getmaxyx: get curses cursor and window	coordinates /getyx, getparyx, getbegyx,	curs_getyx(3X)
dd: convert and	copy a file	dd(1)
copylist:	copy a file into memory	copylist(3G)
bcopy:	copy bytes from one area to another	bcopy(3C)
cpio:	copy file archives in and out	cpio(1)
volcopy, labelit:	copy file systems with label checking	volcopy(1M)
cp:	copy files	cp(1)
/strccpy: streadd, strcadd, Strecpy:	copy strings, compressing or expanding/	strccpy(3G)
uucp, uulog, uuname: UNIX-to-UNIX system	copy	uucp(1)
uupick: public UNIX-to-UNIX system file	copy /uuto,	uuto(1)
	copylist: copy a file into memory	copylist(3G)
	copyright: copyright information file	copyright(4)
copyright:	copyright information file	copyright(4)
rint,/ floor, floorf, ceil, ceilf,	copysign, fmod, fmodf, fabs, fabsf,	floor(3M)

/curs_overlay: overlay, overwrite,	copywin: overlap and manipulate/	curs_overlay(3X)
core: format of	core: format of core image file	core(4)
synchronization of the system/ adjtime:	core image file	core(4)
/menu_cursor: pos_menu_cursor:	correct the time to allow	adjtime(2)
getnetpath: get /etc/netconfig entry	correctly position a menus cursor	menu_cursor(3X)
acosf, atan, atanf,/ /trig: sin, sinf,	corresponding to NETPATH component	getnetpath(3N)
acosf, atan,/ trig: sin, sinf, cos,	cos, cosf, tan, tanf, asin, asinf, acos,	trig(3M)
atanh: hyperbolic/ /sinh, sinhf,	cosf, tan, tanf, asin, asinf, acos,	trig(3M)
hyperbolic functions /sinh, sinhf, cosh,	cosh, coshf, tanh, tanhf, asinh, acosh,	sinh(3M)
sum: print checksum and block	coshf, tanh, tanhf, asinh, acosh, atanh:	sinh(3M)
wc: word	count of a file	sum(1)
limits for a control point directory	count	wc(1)
cpio: format of	cp: copy files	cp(1)
cpio archive	cpd: change or view the allocation	cpd(1)
cpio: copy file archives in and out	cpio archive	cpio(4)
cpio: format of cpio archive	cpio: copy file archives in and out	cpio(1)
cpp: the C language preprocessor	cpio: format of cpio archive	cpio(4)
cprs: compress a common object file	cpr: the C language preprocessor	cpp(1)
clock: report	cprs: compress a common object file	cprs(1)
crashes	CPU time used	clock(3C)
crash: what to do when the DG/UX system	cpz: compose-key maps	cpz(4M)
existing one	crash: examine system images	crash(1M)
mkdir: create a directory file	crash: what to do when the DG/UX system	crash(8)
/mknod: create a file entry in the file system	crashes	crash(8)
mkfs, newfs: create a file system	creat: create a new file or rewrite an	creat(2)
dg_mknod: create a file system node	creat: create a new file or rewrite an	creat(2)
tmpnam, tempnam: create a name for a temporary file	creat: create a new file or rewrite an	creat(2)
mkfifo: create a new FIFO	creat: create a new file or rewrite an	creat(2)
one /creat: create a new file or rewrite an existing	creat: create a new file or rewrite an	creat(2)
system /groupadd: add	(create) a new group definition on the	groupadd(1M)
link: create a new link to a file	link: create a new link to a file	link(2)
fork: create a new process	fork: create a new process	fork(2)
socketpair: create a pair of connected sockets	socketpair: create a pair of connected sockets	socketpair(2)
symlink: create a symbolic link file	symlink: create a symbolic link file	symlink(2)
ctags: create a tags file	ctags: create a tags file	ctags(1)
tmpfile: create a temporary file	tmpfile: create a temporary file	tmpfile(3S)
entry /chgtinfo: create a temporary version of a TERMINFO	chgtinfo: create a temporary version of a TERMINFO	chgtinfo(1)
socket: create an endpoint for communication	socket: create an endpoint for communication	socket(2)
massaging C source mkstr: create an error message file by	mkstr: create an error message file by	mkstr(1)
pipe: create an interprocess channel	pipe: create an interprocess channel	pipe(2)
admin: create and administer SCCS files	admin: create and administer SCCS files	admin(1)
/dup_field, link_field, free_field,:	create and destroy forms fields	form_field_new(3X)
form_new: new_form, free_form:	create and destroy forms	form_new(3X)
/menu_item_new: new_item, free_item:	create and destroy menus items	menu_item_new(3X)
menu_new: new_menu, free_menu:	create and destroy menus	menu_new(3X)
panel_new: new_panel, del_panel:	create and destroy panels	panel_new(3X)
/pnoutrefresh, pechochar, pechowchar:	create and display curses pads	curs_pad(3X)
colltbl: create collation database	colltbl: create collation database	colltbl(1M)
/border, wborder, box, whline, wvline:	create curses borders, horizontal and/	curs_border(3X)
wsyncup, syncok, wcursyncup, wsyncdown :	create curses windows /mvderwin, dupwin,	curs_window(3X)
/mkmsgs: create message files for use by gettxt	mkmsgs: create message files for use by gettxt	mkmsgs(1)
montbl: create monetary database	montbl: create monetary database	montbl(1M)
mkdirp, rmdirp: create, remove directories in a path	mkdirp, rmdirp: create, remove directories in a path	mkdirp(3G)
/setuid: create session and set process group ID	setuid: create session and set process group ID	setuid(2)
umask: set and get file	creation mask	umask(2)
getdev: lists devices based on	criteria	getdev(1M)
groups which contain devices that match	criteria /getdgrp: lists device	getdgrp(1M)
crontab: user	cron: clock agent	cron(1M)
and Pascal sources xref: generate	crontab file	crontab(1)
cxref: generate C program	crontab: user crontab file	crontab(1)
package curses:	CRT screen handling and optimization	curses(3X)
functions	crypt: encode/decode	crypt(1)
encryption	crypt: password and file encryption	crypt(3X)
program	crypt, setkey, encrypt: generate	crypt(3C)
interpreter) having a C-like syntax	cscope: interactively examine a C	cscope(1)
	csh: invoke a shell (command	csh(1)

which: locate a program file for execute access	which(1)	users	which(1)
	csplit: context split	csplit(1)	csplit(1)
	csync: synchronize hardware caches for	csync(2)	csync(2)
	ct: spawn login to a remote terminal	ct(1)	ct(1)
	ctags: create a tags file	ctags(1)	ctags(1)
	ctermid: generate file name for terminal	ctermid(3S)	ctermid(3S)
tzset: convert date and time to string	ctime, localtime, gmtime, asctime,	ctime(3C)	ctime(3C)
	ctl: COFF-to-legend translator	ctl(1)	ctl(1)
	ctrace: trace a C program to debug it	ctrace(1)	ctrace(1)
	ctype: isdigit, isxdigit, islower,	ctype(3C)	ctype(3C)
isupper, isalpha, isalnum, isspace,/ register /getpsr: return the vhangup: virtually hang up the /getdomainname: get name of /setdomainname: set name of t_look: look at the gethostid: get unique identifier of gethostname: get name of sethostid: set unique identifier of sethostname: set name of dg_ipc_info: get information about top_row, item_index: set and get /current_field, field_index: set forms lseek: change object pointer's return the number of open files the return the extended errno for the set the effective group id of the set the effective user id of the sact: print t_getstate: get the uname: print name of uname, nuname: get name of getcontext, setcontext: get and set find the slot in the utmp file of the /replace_panel: get or set the getcwd: get pathname of getwd: get current/ /form_page, set_current_field, /menu_item_current: set_current_item, /get information about the system's mvwaddch, echochar, wechochar: add a/ waddchstr, waddchnstr, mvaddchstr,/ waddchstr, waddchnstr, mvaddchstr,/ waddnstr, mvaddstr, mvaddnstr,/ mvwaddwch, echowchar, wechowchar: add a/ waddwchstr, waddwchnstr, mvaddwchstr,/ waddwstr, waddnwstr, mvaddwstr,/ watron, attrset, wattrset, standend,/ screen flash routines wbkgd: curses window background/ whline, wvline: create curses borders/ wclear, clrtoeol, wclrtoeol, clrtoeol,/ init_color, has_colors,/ mvwdelch: delete character under cursor/ insdelln, windsdelln, insertln,/ curs_beep: beep, flash: /wborder, box, whline, wvline: create control/ /standend, standout, wstandout: /color_content, pair_content: optimization package /getyx, getparyx, getbegyx, getmaxyx: get /killchar, longname, termattrs, termname: termcap/ /tgetnum, tgetstr, tgoto, tputs: /mvcur, tigetflag, tigetnum, tigetstr: pechowchar: create and display /wtouchln, is_linetouched, is_wintouched: riporffline, curs_set, napms: low-level /scr_init, scr_set: read (write) a scr_dump: format of /endwin, isendwin, set_term, delscreen: slk_attron, slk_attrset, slk_attroff:	current contents of the processor status current control terminal current domain current domain current event on a transport endpoint current host current IPCs state current menu items /set_top_row, current page and field current position current process can have /getdtablesize: current process /dg_ext_errno: current process /setgid: current process /seteuid: current SCCS file editing activity current state current system current UNIX system current user context current user /ttypslot: current window of a panels panel current working directory current working directory pathname current_field, field_index: set forms current_item, set_top_row, top_row,/ currently active processes curs_addch: addch, waddch, mvaddch, curs_addchstr: addchstr, addchnstr, curs_addchstr: addchstr, addchnstr, curs_addstr: addstr, addnstr, waddstr, curs_addwch: addwch, waddwch, mvaddwch, curs_addwchstr: addwchstr, addwchnstr, curs_addwstr: addwstr, addnwstr, curs_attr: attroff, wattroff, attron, curs_beep: beep, flash: curses bell and curs_bkgd: bkgdset, wbkgdset, bkgd, curs_border: border, wborder, box, curs_clear: erase, werase, clear, curs_color: start_color, init_pair, curs_delch: delch, wdelch, mvdelch, curs_deleteln: deleteln, wdeleteln, curses bell and screen flash routines curses borders, horizontal and vertical/ curses character and window attribute curses color manipulation routines curses: CRT screen handling and curses cursor and window coordinates curses environment query routines curses interfaces (emulated) to the curses interfaces to terminfo database curses pads /pnoutrefresh, pechowchar, curses refresh control routines curses routines /getsyx, setsyx, curses screen from (to) a file curses screen image file curses screen initialization and/ curses soft label routines /slk_touch,	getpsr(2) vhangup(2) getdomainname(2) setdomainname(2) t_look(3N) gethostid(2) gethostname(2) sethostid(2) sethostname(2) dg_ipc_info(2) menu_item_current(3X) form_page(3X) lseek(2) getdtablesize(2) dg_ext_errno(2) setegid(2) seteuid(2) sact(1) t_getstate(3N) uname(1) uname(2) getcontext(2) ttypslot(3C) panel_window(3X) getcwd(3C) getwd(3C) form_page(3X) menu_item_current(3X) dg_process_info(2) curs_addch(3X) curs_addchstr(3X) curs_addchstr(3X) curs_addstr(3X) curs_addwch(3X) curs_addwchstr(3X) curs_addwstr(3X) curs_attr(3X) curs_beep(3X) curs_bkgd(3X) curs_border(3X) curs_clear(3X) curs_color(3X) curs_delch(3X) curs_deleteln(3X) curs_beep(3X) curs_border(3X) curs_attr(3X) curs_color(3X) curs_deleteln(3X) curs_beep(3X) curs_border(3X) curs_attr(3X) curs_color(3X) curs_deleteln(3X) curs_beep(3X) curs_kernel(3X) curs_scr_dump(3X) scr_dump(4) curs_initscr(3X) curs_slk(3X)	

/qiflush, timeout, wtimeout, typeahead:
 get (or push back) characters from
 mvwgetnstr: get character strings from
 (or push back) wchar_t characters from
 get wchar_t character strings from
 /wssetscreg, scrollok, nl, nonl:
 delay_output, flushinp: miscellaneous
 vwscanw: convert formatted input from a
 /add a string of characters to a
 /bkgdset, wbkgdset, bkgd, wbkgd:
 add a character (with attributes) to a
 of characters (and attributes) to a
 of characters (and attributes) to a
 wechowchar: add a wchar_t character to a
 add string of wchar_t characters to a
 add a string of wchar_t characters to a
 wclrtoeol: clear all or part of a
 delete character under cursor in a
 winsertln: delete and insert lines in a
 a character and its attributes from a
 of characters (and attributes) from a
 the character under the cursor in a
 before character under the cursor in a
 get a string of characters from a
 the character under the cursor in a
 before character under the cursor in a
 mvwinwch: get a wchar_t character from a
 a string of wchar_t characters from a
 a string of wchar_t characters from a
 curs_move: move, wmove: move
 scroll, srcl, wscl: scroll
 douupdate, redrawwin, wredrawln: refresh
 overlap and manipulate overlapped
 vwprintw: print formatted output in
 syncok, wcursyncup, wsyncdown: create
 mvwgetch, wgetch: get (or push back)/
 wgetnstr, mvgetstr, mvgetnstr,/
 mvwgetwch, wgetwch: get (or push back)/
 wgetwstr, wgetnwstr, mvgetwstr,/
 getmaxyx: get curses cursor and window/
 mvwinch: get a character and its/
 winchstr, winchnstr, mvwinchstr,/
 isendwin, set_term, delscreen: curses/
 noecho, halfdelay, intrflush, keypad,/
 mvwinsch: insert a character before the/
 winsnstr, mvinsstr, mvinsnstr,/
 winnstr, mvinstr, mvinnstr, mvwinstr,/
 winswstr, winswstr, mvinswstr,/
 mvwinswch: insert a wchar_t character/
 mvwinwch: get a wchar_t character from/
 winwchstr, winwchnstr, mvwinwchstr,/
 winnwstr, mvinnwstr, mvinnwstr,/
 def_shell_mode, reset_prog_mode,/
 window cursor
 getparyx, getbegyx, getmaxyx: get curses
 to a curses window and advance
 move, wmove: move curses window
 pos_form_cursor: position forms window
 mvwdelch: delete character under
 character before the character under the
 insert string before character under the
 character before the character under the
 string before character under the
 correctly position a menus
 immedok, leaveok, setscreg,/
 copywin: overlap and manipulate/
 pnoutrefresh, pechochar, pechowchar:/
 mvvprintw, vwprintw: print formatted/
 wnoutrefresh, douupdate, redrawwin,/
 mvwscanw, vwscanw: convert formatted/
 scr_init, scr_set: read (write) a
 curses terminal input option control/ curs_inopts(3X)
 curses terminal keyboard /ungetch: curs_getch(3X)
 curses terminal keyboard /mvwgetstr, curs_getstr(3X)
 curses terminal keyboard /ungetwch: get curs_getwch(3X)
 curses terminal keyboard /mvwgetnwstr: curs_getwstr(3X)
 curses terminal output option control/ curs_outopts(3X)
 curses utility routines /putwin, getwin, curs_util(3X)
 curses widow /wscanw, mvscanw, mvwscanw, curs_scanw(3X)
 curses window and advance cursor curs_addstr(3X)
 curses window background manipulation/ curs_bkgd(3X)
 curses window /echochar, wechochar: curs_addch(3X)
 curses window /mvwaddchnstr: add string curs_addchstr(3X)
 curses window /mvwaddchnstr: add string curs_addchstr(3X)
 curses window /mvwaddwch, echowchar, curs_addwch(3X)
 curses window /mvwaddwchnstr: curs_addwchstr(3X)
 curses window /mvwaddwstr, mvwaddnwstr: curs_addwstr(3X)
 curses window /wclrtoeol, clrtoeol, curs_clear(3X)
 curses window. /mvdclch, mvwdelch: curs_delch(3X)
 curses window /windsdelln, insertln, curs_deleteln(3X)
 curses window /mvinch, mvwinch: get curs_inch(3X)
 curses window /mvwinchnstr: get a string curs_inchstr(3X)
 curses window /insert a character before curs_insch(3X)
 curses window /mvwinsnstr: insert string curs_insstr(3X)
 curses window /mvwinstr, mvwinstr: curs_instr(3X)
 curses window /wchar_t character before curs_inswch(3X)
 curses window /insert wchar_t string curs_inswstr(3X)
 curses window /inwch, winwch, mvinwch, curs_inwch(3X)
 curses window /mvwinwchnstr: get curs_inwchstr(3X)
 curses window /mvwinnwstr: get curs_innwstr(3X)
 curses window cursor curs_move(3X)
 curses window /curs_scroll: curs_scroll(3X)
 curses windows and lines /wnoutrefresh, curs_refresh(3X)
 curses windows /overwrite, copywin: curs_overlay(3X)
 curses windows /mvprintw, mvvprintw, curs_printw(3X)
 curses windows /dupwin, wsyncup, curs_window(3X)
 curs_getch: getch, wgetch, mvgetch, curs_getch(3X)
 curs_getstr: getstr, getnstr, wgetstr, curs_getstr(3X)
 curs_getwch: getwch, wgetwch, mvgetwch, curs_getwch(3X)
 curs_getwstr: getwstr, getnwstr, curs_getwstr(3X)
 curs_getyx: getyx, getparyx, getbegyx, curs_getyx(3X)
 curs_inch: inch, winch, mvinch, curs_inch(3X)
 curs_inchstr: inchstr, inchnstr, curs_inchstr(3X)
 curs_initscr: initscr, newterm, endwin, curs_initscr(3X)
 curs_inopts: cbreak, nocbreak, echo, curs_inopts(3X)
 curs_insch: insch, winsch, mvinsch, curs_insch(3X)
 curs_instr: instr, insnstr, winsstr, curs_instr(3X)
 curs_instr: instr, innstr, winstr, curs_instr(3X)
 curs_instr: inswstr, insnwstr, curs_inswstr(3X)
 curs_inswch: inswch, winswch, mvinswch, curs_inswch(3X)
 curs_inwch: inwch, winwch, mvinwch, curs_inwch(3X)
 curs_inwchstr: inwchstr, inwchnstr, curs_inwchstr(3X)
 curs_innwstr: innwstr, innwstr, winnwstr, curs_innwstr(3X)
 curs_kernel: def_prog_mode, curs_kernel(3X)
 curs_move: move, wmove: move curses curs_move(3X)
 cursor and window coordinates /getyx, curs_getyx(3X)
 cursor /add a string of characters curs_addstr(3X)
 cursor /curs_move: curs_move(3X)
 cursor /form_cursor: form_cursor(3X)
 cursor in a curses window. /mvdclch, curs_delch(3X)
 cursor in a curses window /insert a curs_insch(3X)
 cursor in a curses window /mvwinsnstr: curs_insstr(3X)
 cursor in a curses window /a wchar_t curs_inswch(3X)
 cursor in a curses window /wchar_t curs_inswstr(3X)
 cursor /menu_cursor: pos_menu_cursor: menu_cursor(3X)
 curs_outopts: clearok, idlok, idcok curs_outopts(3X)
 curs_overlay: overlay, overwrite, curs_overlay(3X)
 curs_pad: newpad, subpad, prefresh, curs_pad(3X)
 curs_printw: printw, wprintw, mvprintw, curs_printw(3X)
 curs_refresh: refresh, wrefresh, curs_refresh(3X)
 curs_scanw: scanw, wscanw, mvscanw, curs_scanw(3X)
 curs_scr_dump: scr_dump, scr_restore, curs_scr_dump(3X)

scroll a curses window	curs_scroll: scroll, srcl, wscr:	curs_scroll(3X)
/savetty, getsyx, setsyx, ripoffline,	curs_set, napms: low-level curses/	curs_kernel(3X)
slk_refresh, slk_noutrefresh,/	curs_slk: slk_init, slk_set,	curs_slk(3X)
has_ic, has_il, killchar, longname,/	curs_termattrs: baudrate, erasechar,	curs_termattrs(3X)
tgetnum, tgetstr, tgoto, tputs: curses/	curs_termcap: tgetent, tgetflag,	curs_termcap(3X)
set_curterm, del_curterm, restartterm,/	curs_terminfo: setupterm, setterm,	curs_terminfo(3X)
untouchwin, wtouchln, is_linetouched,/	curs_touch: touchwin, touchline,	curs_touch(3X)
use_env, putwin, getwin, delay_output,/	curs_util: unctrl, keyname, filter,	curs_util(3X)
subwin, derwin, mvderwin, dupwin,/	curs_window: newwin, delwin, mvwin,	curs_window(3X)
spline: interpolate smooth	curve	spline(1G)
user name associated with effective UID	cuserid: get character login name or	cuserid(3S)
line of a file	cut: cut out selected fields of each	cut(1)
a file /cut:	cut out selected fields of each line of	cut(1)
cross-reference	cxref: generate C program	cxref(1)
dumpcycle: dump	cycle file for backups	dumpcycle(4M)
/admdumpcycle: manage dump	cycle tables	admdumpcycle(1M)
runacct: run	da: AViiON family disk array subsystem	da(7)
filesave, tapesave:	daily accounting	runacct(1M)
tell if forms field has off-screen	daily/weekly file system backup	filesave(1M)
timex: time a command; report process	data ahead or behind /data_behind:	form_data(3X)
retrieve a text string from a message	data and system activity	timex(1)
printcap: printer capability	data base /gettxt:	gettxt(1)
sdetab: software development environment	data base	printcap(5)
fetch, store, delete, firstkey, nextkey:	data base	sdetab(4)
dbm_nextkey, dbm_error, dbm_clearerr:	data base subroutines /dbmimit,	dbm(3X)
termcap: terminal capability	data base subroutines /dbm_firstkey,	ndbm(3C)
or search for a text string in, message	data base	termcap(5)
diskusg: generate disk accounting	data bases /display contents of,	srchtxt(1)
elf_newdata, elf_rawdata: get section	data by user id	diskusg(1M)
retrieve file identification	data /elf_getdata,	elf_getdata(3E)
t_rcvuderr: receive a unit	data /elf_getident:	elf_getident(3E)
/dg_unbuffered_read: synchronously read	data error indication	t_rcvuderr(3N)
sputl, sgetl: access long integer	data from a file without system/	dg_unbuffered_read(2)
/t_snd: send	data in a machine-independent fashion	sputl(3X)
connection t_rcv: receive	data or expedited data over a connection	t_snd(3N)
t_snd: send data or expedited	data or expedited data sent over a	t_rcv(3N)
nlsgetcall: get client's	data over a connection	t_snd(3N)
prof: display profile	data passed via the listener	nlsgetcall(3N)
library routines for external	data	prof(1)
system call dg_stat:	data representation /xdr_wrapstring:	xdr(3N)
stat:	data returned by dg_stat and dg_fstat	dg_stat(5)
call dg_mknod:	data returned by stat system call	stat(5)
/statfs:	data returned by the dg_mknod system	dg_mknod(5)
/ustat:	data returned by the statfs system call	statfs(5)
brk: change	data returned by the ustat system call	ustat(5)
sbrk: change	data segment space allocation	brk(2)
t_rcv: receive data or expedited	data segment space allocation	sbrk(2)
plock: lock	data sent over a connection	t_rcv(3N)
/dg_unbuffered_write: synchronously write	data, text, or both into memory	plock(2)
elf32_xlatetom: class-dependent	data to a file without system buffering	dg_unbuffered_write(2)
tkey: set label and	data translation /elf32_xlatetof,	elf_xlate(3E)
field_type, field_arg: forms field	data translation parameters	tkey(1)
nl_types: native language	data type validation /set_field_type,	form_field_validation(3X)
types: primitive system	data types	nl_types(5)
t_rcvudata: receive a	data types	types(5)
t_sndudata: send a	data unit	t_rcvudata(3N)
/panel_userptr: associate application	data unit	t_sndudata(3N)
field_userptr: associate application	data with a panels panel	panel_userptr(3X)
form_userptr: associate application	data with forms /set_field_userptr,	form_field_userptr(3X)
item_userptr: associate application	data with forms /set_form_userptr,	form_userptr(3X)
menu_userptr: associate application	data with menu items /set_item_userptr,	menu_item_userptr(3X)
field has off-screen data/ /form_data:	data with menus /set_menu_userptr,	menu_userptr(3X)
mail alias information in the aliases	data_ahead, data_behind: tell if forms	form_data(3X)
admmether: manage ether	database /admalias: manage	admalias(1M)
manage group information in the group	database	admmether(1M)
admhost: manage hosts	database /admgroup:	admgroup(1M)
manage the TCP/IP network interfaces	database	admhost(1M)
admnetwork: manage network	database /admpinterface:	admpinterface(1M)
resolver's domain name and nameservers	database	admnetwork(1M)
admresolver: manage DNS	database /admresolve: manage DNS	admresolve(1M)
admresolver: manage service	database	admresolver(1M)
		admresolver(1M)

manage the SNMP community	database /admsnmpcommunity:	admsnmpcommunity(1M)
/admsnmpobject: manage the snmpd object	database	admsnmpobject(1M)
/admsnmptrap: manage the SNMP traps	database	admsnmptrap(1M)
/admtrustedhost: manage the trusted hosts	database	admtrustedhost(1M)
manage user information in the password	database /admuser:	admuser(1M)
colltbl: create collation	database	colltbl(1M)
tigetstr: curses interfaces to terminfo	database /mvcurl, tigetflag, tigetnum,	curs_terminfo(3X)
/getnetconfig: get network configuration	database entry	getnetconfig(3N)
make changes to the help facility	database /helpadm:	helpadm(1M)
add a file to the software installation	database /installf:	installf(1M)
monttbl: create monetary	database	monttbl(1M)
netconfig: network configuration	database	netconfig(4)
join: relational	database operator	join(1)
removef: remove a file from software	database	removef(1M)
/dg_lock_reset: reset remote file lock	database, start lock reclaim grace/	dg_lock_reset(2)
terminal and printer capability	database /terminfo:	terminfo(4)
initialize a terminal or query terminfo	database /tput:	tput(1)
admroute: manage routing	databases	admroute(1M)
order for /etc/hosts, NIS, and DNS	databases /admsvcorder: manage search	admsvcorder(1M)
off-screen data/ form_data: data Ahead,	data_behind: tell if forms field has	form_data(3X)
ftime: get	date and time	ftime(3C)
getdate_err: convert user format	date and time /getdate,	getdate(3C)
/gettimeofday: get	date and time	gettimeofday(2)
/settimeofday: set	date and time	settimeofday(2)
gmtime, asctime, tzset: convert	date and time to string /localtime,	ctime(3C)
strftime, cftime, asctime: convert	date and time to string	strftime(3C)
valdate: prompt for and validate a	date /ckdate, errdate, helpdate,	ckdate(1)
date: print and set the	date	date(1)
admdate: manipulate the system	date: print and set the date	date(1)
a prompt; verify and return a time of	date, time and time zone	admdate(1M)
used to distinguish prime and non-prime	day /cktime: display	cktime(1)
/dbm_firstkey, dbm_nextkey, dbm_error,	days /holidays: accounting information	holidays(4)
dbm_delete,/ ndbm: dbm_open,	dbm_clearerr: data base subroutines	ndbm(3C)
/dbm_close, dbm_fetch, dbm_store,	dbm_close, dbm_fetch, dbm_store,	ndbm(3C)
/dbm_delete, dbm_firstkey, dbm_nextkey,/	dbm_delete, dbm_firstkey, dbm_nextkey,/	ndbm(3C)
ndbm: dbm_open, dbm_close,	dbm_error, dbm_clearerr: data base/	ndbm(3C)
/dbm_fetch, dbm_store, dbm_delete,	dbm_fetch, dbm_store, dbm_delete,/	ndbm(3C)
nextkey: data base subroutines	dbm_firstkey, dbm_nextkey, dbm_error,/	ndbm(3C)
/dbm_store, dbm_delete, dbm_firstkey,	dbm_nextkey, fetch, store, delete, firstkey,	dbm(3X)
dbm_store, dbm_delete,/ ndbm:	dbm_nextkey, dbm_error, dbm_clearerr:/	ndbm(3C)
ndbm: dbm_open, dbm_close, dbm_fetch,	dbm_open, dbm_close, dbm_fetch,	ndbm(3C)
	dbm_store, dbm_delete, dbm_firstkey,/	ndbm(3C)
	dbx: source level debugger	dbx(1)
	dc: desk calculator	dc(1)
	dd: convert and copy a file	dd(1)
	deblock: change blocking size	deblock(1)
	debug it	ctrace(1)
ctrace: trace a C program to	debugger	dbx(1)
dbx: source level	debugger	dg_fsdb(1M)
dg_fsdb: file system	debugger	fsdb(1M)
fsdb: file system	debugger	sdb(1)
sdb: symbolic	debugger utility program	syacdb(1M)
syacdb: syac	Debugging information technology	legend(5)
legend:	debugging on	uutry(1M)
Uutry: try to contact remote system with	deck manipulation routines /show_panel,	panel_show(3X)
hide_panel, panel_hidden: panels	deck manipulation routines /panel_top:	panel_top(3X)
top_panel, bottom_panel: panels	deck traversal primitives /panel_above:	panel_above(3X)
panel_above, panel_below: panels	decrypt conversation key with the	dg_decryptsessionkey(2)
client/server/ /dg_decryptsessionkey:	default	kill(1)
kill: terminate a process by	default parameters for tapes	admtape(1M)
admtape: manipulate the	default sets /admdefault:	admdefault(1M)
provide an interface to named	default system time zone and locale	timezone(4)
timezone: set	default version of GNU C	default-gcc(1)
default-gcc: set or query	default-gcc: set or query default	default-gcc(1)
version of GNU C	definition from the system	groupdel(1M)
groupdel: delete a group	definition on the system	groupadd(1M)
groupadd: add (create) a new group	definition on the system	groupmod(1M)
groupmod: modify a group	definition	sysdef(1M)
sysdef: output system	definition	testlocale(1M)
testlocale: test locale	definitions of common terms and symbols	glossary(1)
/glossary:	def_prog_mode, def_shell_mode,	curs_kernel(3X)
reset_prog_mode,/ /curs_kernel:		

/curs_kernel: def_prog_mode,	def_shell_mode, reset_prog_mode,/	curs_kernel(3X)
/dg_lock_wait: wait for previously	delayed lock requests to complete	dg_lock_wait(2)
curses/ /filter, use_env, putwin, getwin,	delay_output, flushinp: miscellaneous	curs_util(3X)
character under cursor in a/ /curs_delch:	delch, wdelch, mvdelch; mvwdelch: delete	curs_delch(3X)
putp,/ /setupterm, setterm, set_curterm,	del_curterm, restartterm, tparm, tputs,	curs_terminfo(3X)
system groupdel:	delete a group definition from the	groupdel(1M)
userdel:	delete a user's login from the system	userdel(1M)
window /winsdelln, insertln, winsertln:	delete and insert lines in a curses	curs_deleteln(3X)
/delch, wdelch, mvdelch, mvwdelch:	delete character under cursor in a/	curs_delch(3X)
rm, rmdir: remove,	delete files or directories	rm(1)
subroutines dbmunit, fetch, store,	delete, firstkey, nextkey: data base	dbm(3X)
winsdelln, insertln,/ /curs_deleteln:	deleteln, wdeleteln, insdelln,	curs_deleteln(3X)
bgets: read stream up to next	delimiter	bgets(3G)
basename, dirname:	deliver portions of path names	basename(1)
tail:	deliver the last part of a file	tail(1)
panel_new: new_panel,	del_panel: create and destroy panels	panel_new(3X)
/newterm, endwin, isendwin, set_term,	delscreen: curses screen initialization/	curs_initscr(3X)
change the delta commentary of an SCCS	delta /cdc:	cdc(1)
delta: make a	delta (change) to an SCCS file	delta(1)
cdc: change the	delta commentary of an SCCS delta	cdc(1)
rmdel: remove a	delta from an SCCS file	rmdel(1)
file	delta: make a delta (change) to an SCCS	delta(1)
comb: combine SCCS	deltas	comb(1)
dupwin, wsyncup,/ /curs_window: newwin,	delwin, mvwin, subwin, derwin, mvderwin,	curs_window(3X)
swapon: add a swap device for	demand paging	swapon(2)
mesg: permit or	deny messages	mesg(1)
depend: software	depend: software dependencies files	depend(4)
ldd: list dynamic	dependencies files	depend(4)
constructs	dependencies	ldd(1)
syncok,/ /newwin, delwin, mvwin, subwin,	deroff: remove nroff/troff, tbl, and eqn	deroff(1)
/dg_strsignal: get message string	derwin, mvderwin, dupwin, wsyncup,	curs_window(3X)
usage: retrieve a command	describing the given signal	dg_strsignal(3C)
idc: interface	description and usage examples	usage(1)
pkgmap: package contents	description compiler	idc(1)
system: format of a kernel	description file	pkgmap(4)
idi: interface	description file	system(4)
tools for use with the interface	description interpreter	idi(1)
idl: interface	description interpreter /idi_warning:	idi_tools(1)
get menus item name and	description language	idl(4)
infocmp: compare or print out TERMINFO	description /item_description:	menu_item_name(3X)
/let processes attach shared	descriptions	infocmp(1M)
/attach another process's shared	descriptor array	dg_allow_shared_descriptor_attach(2)
close an object associated with a file	descriptor array	dg_attach_to_shared_descriptors(2)
fcntl: file	descriptor /close:	close(2)
dup: duplicate an open file	descriptor control	fcntl(2)
an open file descriptor onto a specific	descriptor	dup(2)
elf_begin: make a file	descriptor /dup2: duplicate	dup2(2)
elf_cntl: control a file	descriptor	elf_begin(3E)
elf_update: update an ELF	descriptor	elf_cntl(3E)
detach a name from a STREAMS-based file	descriptor	elf_update(3E)
setfsent, endfsent: get filesystem	descriptor /fdetach:	fdetach(3C)
endmntent, hasmntopt: get file system	descriptor file entry /getfstype,	getfsent(3C)
isastream: test a file	descriptor file entry /addmntent,	getmntent(3C)
dup2: duplicate an open file	descriptor	isastream(3C)
/fattach: attach STREAMS-based file	descriptor onto a specific descriptor	dup2(2)
select: examine file	descriptor to object in file system name/	fattach(3C)
dc:	descriptors for I/O readiness	select(2)
get or set message queue attributes or	desk calculator	dc(1)
link_field, free_field,: create and	destroy a message queue /msgctl:	msgctl(2)
new_form, free_form: create and	destroy forms fields /dup_field,	form_field_new(3X)
new_item, free_item: create and	destroy forms /form_new:	form_new(3X)
new_menu, free_menu: create and	destroy menus items /menu_item_new:	menu_item_new(3X)
new_panel, del_panel: create and	destroy menus /menu_new:	menu_new(3X)
descriptor /fdetach:	destroy panels /panel_new:	panel_new(3X)
shmdt:	detach a name from a STREAMS-based file	fdetach(3C)
sense multiple access with collision	detach a shared memory segment	shmdt(2)
elf_kind:	detection /dot3: IEEE 802.3 carrier	dot3(6P)
file:	determine file type	elf_kind(3E)
/isalphanum:	determine file type	file(1)
/ishex:	determine if a character is alphanumeric	isalphanum(3C)
	determine if a character is hexadecimal	ishex(3C)

/dg_paging_info:	determine residency of memory pages	dg_paging_info(2)
mincore:	determine residency of memory pages	mincore(2)
access:	determine the accessibility of a file	access(2)
/isnanf, finite, fpclass, unordered:	determine type of floating-point number	isnan(3C)
encrypted /isencrypt:	determine whether a character buffer is	isencrypt(3G)
accept binary messages ckbinarsys:	determine whether remote system can	ckbinarsys(1M)
	devattr: lists device attributes	devattr(1M)
	sdetab: software development environment data base	sdetab(4)
	sde: software development environment	sde(5)
/print commands to reset software	development environment target	sde-target(1)
use	devfree: release devices from exclusive	devfree(1M)
WORM (Write Once Read Multiple optical	device) as magtape interface /pseudo	wmt(7)
devattr: lists	device attributes	devattr(1M)
fold long lines for finite width output	device /fold:	fold(1)
swapon: add a swap	device for demand paging	swapon(2)
access to the slave pseudo-terminal	device /grantpt: grant	grantpt(3C)
listdgrp: lists members of a	device group	listdgrp(1M)
putdgrp: edit	device group table	putdgrp(1M)
match criteria /getdgrp: lists	device groups which contain devices that	getdgrp(1M)
plm: pseudo lock manager	device interface	plm(7)
ioctl: control a	device	ioctl(2)
mouse: mouse	device	mouse(7)
devnm:	device name	devnm(1M)
clone: open any minor	device on a STREAMS driver	clone(7)
get name of the slave pseudo-terminal	device /ptsname:	ptsname(3C)
wmtd: start the WORM magnetic tape	device server	wmtd(1M)
ustat: get file system	device statistics	ustat(2)
/admdumpdevice: manage the dump	device table	admdumpdevice(1M)
putdev: edit	device table	putdev(1M)
umount: remove a file system	device	umount(2)
dg_devctl: perform	device-control functions	dg_devctl(2)
getdev: lists	devices based on criteria	getdev(1M)
programs and passwords for dial-up	devices /d_passwd: log-in	d_passwd(4)
devreserv: reserve	devices for exclusive use	devreserv(1M)
swapon: specify additional	devices for system paging	swapon(1M)
devfree: release	devices from exclusive use	devfree(1M)
probedev: probe system for	devices	probedev(1M)
dialups:	devices requiring a dial-up password.	dialups(4)
tload: load terminal controller	devices	tload(1M)
lists device groups which contain	devices that match criteria /getdgrp:	getdgrp(1M)
	devnm: device name	devnm(1M)
use	devreserv: reserve devices for exclusive	devreserv(1M)
	devtty: control terminal pseudo-device	devtty(7)
and inodes	df: report number of free disk blocks	df(1M)
	dfm: DOS file manager	dfm(4M)
processes attach shared descriptor/ another process's shared descriptor/ /dg_seek, files /lp:	dg_allow_shared_descriptor_attach: let	dg_allow_shared_descriptor_attach(2)
conversation key with the client/server/ functions	dg_attach_to_shared_descriptors: attach	dg_attach_to_shared_descriptors(2)
Ethernet interface	dg_block_seek: extended seek functions	dg_seek(3C)
conversation key with the client/server/ for the current process	DGC AViiON family line printer special	lp(7)
for process identified by process key	dg_decryptsessionkey: decrypt	dg_decryptsessionkey(2)
lock on an open DG/UX file	dg_devctl: perform device-control	dg_devctl(2)
	dgen: second generation integrated	dgen(7)
information	dg_encryptsessionkey: encrypt	dg_encryptsessionkey(2)
dg_stat: data returned by dg_stat and	dg_ext_errno: return the extended errno	dg_ext_errno(2)
	dg_file_info: get file usage information	dg_file_info(2)
u3b5, vax: provide truth value/ machid:	dg_flock: apply or remove an advisory	dg_flock(3C)
current IPCs state	dg_fsdb: file system debugger	dg_fsdb(1M)
on a filehandle	dg_fstat: get extended file status	dg_fstat(2)
remote lock clients	dg_fstat system call	dg_stat(5)
database, start lock reclaim grace/ delayed lock requests to complete	dg_getrootkey: get root's secret key	dg_getrootkey(2)
	dghost, m68k, m88k, i386, pdp11, u3b, dg_ipc_info: get information about	machid(1)
system call	dg_kill: test for or terminate a process	dg_ipc_info(2)
dg_mknod: data returned by the	dg_lock: process a record lock request	dg_kill(1)
	dg_lock_kill: remove locks held by	dg_lock(2)
	dg_lock_reset: reset remote file lock	dg_lock_kill(2)
	dg_lock_wait: wait for previously	dg_lock_reset(2)
	dg_mknod: create a file system node	dg_lock_wait(2)
	dg_mknod: data returned by the dg_mknod	dg_mknod(2)
	dg_mknod system call	dg_mknod(5)
	dg_mount: mount a file system	dg_mknod(5)
		dg_mount(2)

	dg_mstat: get file status	dg_mstat(2)
memory pages	dg_paging_info: determine residency of	dg_paging_info(2)
the system's currently active processes	dg_process_info: get information about	dg_process_info(2)
functions	dg_seek, dg_block_seek: extended seek	dg_seek(3C)
limits of a control point directory	dg_set_cpd_limits: change the resource	dg_set_cpd_limits(2)
key in the keyserver	dg_setsecretkey: store a client's secret	dg_setsecretkey(2)
dg_stat: data returned by	dg_stat and dg_fstat system call	dg_stat(5)
dg_fstat system call	dg_stat: data returned by dg_stat and	dg_stat(5)
information	dg_stat: get extended file status	dg_stat(2)
describing the given signal	dg_strsignal: get message string	dg_strsignal(3C)
dump parameters	dg_sysctl: display or modify boot and	dg_sysctl(1M)
and control functions	dg_sysctl: perform system configuration	dg_sysctl(2)
	dg_sys_info: get system information	dg_sys_info(2)
data from a file without system/	dg_unbuffered_read: synchronously read	dg_unbuffered_read(2)
data to a file without system buffering	dg_unbuffered_write: synchronously write	dg_unbuffered_write(2)
ar:	DG/UX common archive file format	ar(4)
or remove an advisory lock on an open	DG/UX file /dg_flock: apply	dg_flock(3C)
hier:	DG/UX file system hierarchy	hier(5)
jobs: summary of	DG/UX job control facilities	jobs(3C)
pseudo-device syscon, console, systty:	DG/UX operating system console	syscon(7)
crash: what to do when the	DG/UX system crashes	crash(8)
intro: introduction to	DG/UX System special files	intro(7)
admpackage: manage	DG/UX-style software packages	admpackage(1M)
	dg_xtrace: extended process trace	dg_xtrace(2)
postdaisy: PostScript translator for	Diablo 630 files	postdaisy(1)
dlerror: get	diagnostic information	dlerror(3X)
line connection	dial: establish an out-going terminal	dial(3C)
ratfor: rational FORTRAN	dialect	ratfor(1)
log-in programs and passwords for	dial-up devices /d_passwd:	d_passwd(4)
dialups: devices requiring a	dial-up password.	dialups(4)
password.	dialups: devices requiring a dial-up	dialups(4)
bdiff: big	diff	bdiff(1)
comparison	diff: differential file comparator	diff(1)
difftime: computes the	diff3: 3-way differential file	diff3(1)
sdiff: side-by-side	difference between two calendar times	difftime(3C)
comparator berk_diff: Berkeley	difference program	sdiff(1)
diff:	differential file and directory	berk_diff(1)
berk_diff3: Berkeley 3-way	differential file comparator	diff(1)
diff3: 3-way	differential file comparison	berk_diff3(1)
between two calendar times	differential file comparison	diff3(1)
	difftime: computes the difference	difftime(3C)
display information about files and	dircmp: compare two directories	dircmp(1)
uuccheck: check the uucp	directories /admfinfo:	admfinfo(1M)
dircmp: compare two	directories and permissions file	uuccheck(1M)
mkdirp, rmdirp: create, remove	directories	dircmp(1)
pathfind: search for named file in named	directories in a path	mkdirp(3G)
rm, rmdir: remove, delete files or	directories	pathfind(3G)
/exportfs: make a	directories	rm(1)
cd: change working	directory available for mounting via NFS	exportfs(2)
uucleanup: uucp spool	directory	cd(1)
Berkeley differential file and	directory clean-up	uucleanup(1M)
allocation limits for a control point	directory comparator /berk_diff:	berk_diff(1)
the resource limits of a control point	directory /cpd: change or view the	cpd(1)
filesystem-independent/ getdents: get	directory /dg_set_cpd_limits: change	dg_set_cpd_limits(2)
dirent: file system independent	directory entries in a	getdents(2)
unlink: remove a	directory entry	dirent(4)
mkdir: create a	directory entry	unlink(2)
rmdir: remove a	directory file	mkdir(2)
chroot: change root	directory file	rmdir(2)
getcwd: get pathname of current working	directory for a command	chroot(1M)
ls: list contents of	directory	getcwd(3C)
mkdir: make a	directory	ls(1)
mvdire: move a	directory	mkdir(1)
dirname: report the parent	directory	mvdire(1M)
pwd: print working	directory name of a file path name	dirname(3G)
chdir: change the working	directory name	pwd(1)
chroot: change the root	directory of the calling process	chdir(2)
fchdir: change the working	directory of the calling process	chroot(2)
seekdir, rewinddir, closedir: directory/	directory of the calling process	fchdir(2)
telldir, seekdir, rewinddir, closedir:	directory: opendir, readdir, telldir,	directory(3X)
	directory operations /opendir, readdir,	directory(3X)

getwd: get current working	directory pathname	getwd(3C)
scandir, alphasort: scan a	directory	scandir(3C)
tysrch:	directory search list for ttyname	tysrch(4M)
directory entry	dirent: file system independent	dirent(4)
/basename,	dirname: deliver portions of path names	basename(1)
name of a file path name	dirname: report the parent directory	dirname(3G)
	dis: object code disassembler	dis(1)
t_unbind:	disable a transport endpoint	t_unbind(3N)
enable,	disable: enable/disable LP printers	enable(1)
acct: enable or	disable process accounting	acct(2)
dis: object code	disassembler	dis(1)
/connld: line	discipline for unique stream connections	connld(7)
terminal type, modes, speed, and line	discipline /getty: set	getty(1M)
ldterm: standard STREAMS terminal line	discipline module	ldterm(7)
/menu_items, item_count: connect and	disconnect items to and from menus	menu_items(3X)
t_snddis: send user-initiated	disconnect request	t_snddis(3N)
t_rcvdis: retrieve information from	disconnect	t_rcvdis(3N)
diskusg: generate	disk accounting data by user id	diskusg(1M)
information sync: synchronize	disk and memory resident file system	sync(2)
hada: AViiON family High Availability	Disk Array adapter subsystem	hada(7)
da: AViiON family	disk array subsystem	da(7)
for maintaining a High Availability	Disk Array subsystem /menu interface	gridman(1M)
df: report number of free	disk blocks and inodes	df(1M)
a file's in-core state with that on	disk /fsync: synchronize	fsync(2)
dsk: block special	disk interface	dsk(7)
rdsd: character special	disk interface	rdsd(7)
dkctl: control special	disk operations	dkctl(1M)
space:	disk space requirement file	space(4)
cied: AViiON family	disk subsystem	cied(7)
cimd: AViiON family	disk subsystem	cimd(7)
cird: AViiON family	disk subsystem	cird(7)
sd: AViiON family	disk subsystem	sd(7)
du: summarize	disk usage	du(1)
physical and logical disks	diskman: menu interface for managing	diskman(1M)
for managing physical and logical	disks /diskman: menu interface	diskman(1M)
by user id	diskusg: generate disk accounting data	diskusg(1M)
mount, umount: mount and	dismount filesystems	mount(1M)
group names	dispgid: display a list of all valid	dispgid(1)
/dispgid:	display a list of all valid group names	dispgid(1)
/dispuid:	display a list of all valid user names	dispuid(1)
console fmtmsg:	display a message on stderr or system	fmtmsg(1)
console fmtmsg:	display a message on stderr or system	fmtmsg(3C)
/whatis:	display a one-line summary about a topic	whatis(1)
pathname ckpath:	display a prompt; verify and return a	ckpath(1)
string answer ckstr:	display a prompt; verify and return a	ckstr(1)
time of day cktime:	display a prompt; verify and return a	cktime(1)
integer value /ckint:	display a prompt; verify and return an	ckint(1)
set_menu_pad, menu_pad: control menus	display attributes /menu_grey,	menu_attributes(3X)
/field_pad: format the general	display attributes of forms	form_field_attributes(3X)
text string in, message data/ srchtxt:	display contents of, or search for a	srchtxt(1)
pechochar, pechowchar: create and	display curses pads /pnoutrefresh,	curs_pad(3X)
view, vedit: screen-oriented (visual)	display editor based on ex /vi,	vi(1)
uncompress, zcat: compress, expand or	display expanded files /compress,	compress(1)
fez:	display file element sizes	fez(1)
screenful at a time pg:	display file forward or backward one	pg(1)
more, page:	display file one screenful at a time	more(1)
directories admfsinfo:	display information about files and	admfsinfo(1M)
remote users finger:	display information about local and	finger(1)
settings tdisplay:	display label and record translation	tdisplay(1)
parameters dg_sysctl:	display or modify boot and dump	dg_sysctl(1M)
prof:	display profile data	prof(1)
/postmd: matrix	display program for PostScript printers	postmd(1)
pkginfo:	display software package information	pkginfo(1)
/admterminal: manage serving of X	display terminals	admterminal(1M)
specified times atq:	display the jobs queued to run at	atq(1)
systemid:	display the unique system identifier	systemid(1M)
pkgparam:	displays package parameter values	pkgparam(1)
user names	dispuid: display a list of all valid	dispuid(1)
hypot: Euclidean	distance function	hypot(3M)
holidays: accounting information used to	distinguish prime and non-prime days	holidays(4)
/seed48, lcong48: generate uniformly	distributed pseudo-random numbers	drand48(3C)

remainder	div, ldiv: compute the quotient and	div(3C)
	dkctl: control special disk operations	dkctl(1M)
	dlclose: close a shared object	dlclose(3X)
	dlerror: get diagnostic information	dlerror(3X)
	dlopen: open a shared object	dlopen(3X)
shared object	dlsym: get the address of a symbol in	dlsym(3X)
postdmd: PostScript translator for	DMD bitmap files	postdmd(1)
/res_mkquery, res_send, res_init,	dn_comp, dn_expand: make, send, and/	resolver(3C)
packets to/ /res_send, res_init, dn_comp,	dn_expand: make, send, and interpret	resolver(3C)
search order for /etc/hosts, NIS, and	DNS databases /admsvcorder: manage	admsvcorder(1M)
nameservers database	DNS resolver's domain name and	admresolve(1M)
admresolve: manage	doconfig: execute a configuration script	doconfig(3N)
info:	documentation browser	info(1)
prctmp, prdaily,/ chargefee, ckpacct,	dodisk, lastlogin, monacct, nulladm,	acctsh(1M)
whodo: who is	doing what	whodo(1M)
/getdomainname: get name of current	domain	getdomainname(2)
admresolve: manage DNS resolver's	domain name and nameservers database	admresolve(1M)
send, and interpret packets to Internet	domain name servers /dn_expand: make,	resolver(3C)
/setdomainname: set name of current	domain	setdomainname(2)
dfm:	DOS file manager	dfm(4M)
access with collision detection	dot3: IEEE 802.3 carrier sense multiple	dot3(6P)
strtod, atof,: convert string to	double-precision number	strtod(3C)
curses/ /refresh, wrefresh, wnoutrefresh,	doupdate, redrawwin, wredrawin: refresh	curs_refresh(3X)
putmsg, putpmsg: pass a message	down a stream	putmsg(2)
shutdown: shut	down part of a full-duplex connection	shutdown(2)
shutdown: shut	down system, change system state	shutdown(1M)
VSC synchronous controller	download board resident software onto	vsload(1M)
vsload:	download: download host resident	download(1)
PostScript fonts	download host resident PostScript fonts	download(1)
/download:	d_passwd: log-in programs and passwords	d_passwd(4)
for dial-up devices	dpost: troff postprocessor for	dpost(1)
PostScript printers	drand48, erand48, lrand48, nrand48,	drand48(3C)
mrand48, jrand48, srand48, seed48,/	drem: IEEE floating-point remainder	drem(3M)
open any minor device on a STREAMS	driver /clone:	clone(7)
iscd: Integrated Synchronous Chip	Driver	iscd(7)
sad: STREAMS Administrative	Driver	sad(7)
ssid: Streams Synchronous Interface	Driver	ssid(7)
generic interface to EUC handling TTY	drivers and modules /eucioctl:	eucioctl(5)
	dsk: block special disk interface	dsk(7)
	du: summarize disk usage	du(1)
	Dual Asynchronous Receiver/Transmitter	duart(7)
/duart:	duart: Dual Asynchronous	duart(7)
Receiver/Transmitter	dump cycle file for backups	dumpcycle(4M)
dumpcycle:	dump cycle tables	admdumpcycle(1M)
/admdumpcycle: manage	dump device table	admdumpdevice(1M)
/admdumpdevice: manage the	dump	dump(1M)
dump: incremental file system	dump file system information	dumpfs(1M)
dumpfs:	dump from tape	lsd(1M)
lsd: load a system	dump: incremental file system dump	dump(1M)
	dump	od(1)
od: octal	dump parameters	dg_sysctl(1M)
dg_sysctl: display or modify boot and	dump parts of an object or object	att_dump(1)
archive file att_dump:	dump syac memory to a file	syacdump(1M)
syacdump:	dump tapes	dump2label(1M)
dump2label: read and write labels for	dump2	dump2label(1M)
dump2label: tape table file for	dump2: incremental file system backup	dump2label(1M)
	dump2label: read and write labels for	dump2label(1M)
dump tapes	dumpcycle: dump cycle file for backups	dumpcycle(4M)
	dumper	zdump(1M)
zdump: time zone	dumpfs: dump file system information	dumpfs(1M)
	dumptab: tape table file for dump2	dump2label(1M)
	dup: duplicate an open file descriptor	dup(2)
onto a specific descriptor	dup2: duplicate an open file descriptor	dup2(2)
create and/ /form_field_new: new_field,	dup_field, link_field, free_field,:	form_field_new(3X)
dup:	duplicate an open file descriptor	dup(2)
specific descriptor /dup2:	duplicate an open file descriptor onto a	dup2(2)
/delwin, mvwin, subwin, derwin, mvderwin,	dupwin, wsyncup, syncok, wcursyncup,/	curs_window(3X)
ldd: list	dynamic dependencies	ldd(1)
/form_field_info: field_info,	dynamic_field_info: get forms field/	form_field_info(3X)
cut: cut out selected fields of	each line of a file	cut(1)
rev: reverse order of characters in	each line of file	rev(1)

	echo:	echo arguments	echo(1)
		echo: echo arguments	echo(1)
	keypad,/ /curs_inopts:	cbreak, nocbreak,	curs_inopts(3X)
	(with/ /addch, waddch, mvaddch, mvwaddch,	echochar, wechochar: add a character	curs_addch(3X)
	/addwch, waddwch, mvaddwch, mvwaddwch,	echowchar, wechowchar: add a wchar_t/	curs_addwch(3X)
	number to string	ecvt, fcvt, gcvt: convert floating-point	ecvt(3C)
		ed, red: text editor	ed(1)
	end, etext,	edata: last locations in program	end(3C)
	putdgrp:	edit device group table	putdgrp(1M)
	putdev:	edit device table	putdev(1M)
	casual users)	edit: text editor (variant of ex for	edit(1)
	vipw:	edit the system password file	vipw(1M)
	sact: print current SCCS file	editing activity	sact(1)
	vedit: screen-oriented (visual) display	editor based on ex /vi, view,	vi(1)
	ed, red: text	editor	ed(1)
	editread: command line	editor	editread(5)
	ex: text	editor	ex(1)
	ld: link	editor for common object files	ld-coff(1)
	ld: link	editor for object files	ld(1)
	a.out: assembler and link	editor output	a.out(4)
	sed: stream	editor	sed(1)
	/edit: text	editor (variant of ex for casual users)	edit(1)
		editread: command line editor	editread(5)
	setgid: set the real-,	effective-, and saved-group-ids	setgid(2)
	setregid: set the real-,	effective-, and saved-group-ids	setregid(2)
	setreuid: set the real-,	effective-, and saved-user-ids	setreuid(2)
	setuid: set the real-,	effective-, and saved-user-ids	setuid(2)
	process setgid: set the	effective group id of the current	setgid(2)
	login name or user name associated with	effective UID /cuserid: get character	cuserid(3S)
	/seteuid: set the	effective user id of the current process	seteuid(2)
	getegid: get the	effective-group-id	getegid(2)
	geteuid: get the	effective-user-id	geteuid(2)
	spawn new process in a virtual memory	efficient way /vfork:	vfork(2)
	full regular expressions	egrep: search a file for a pattern using	egrep(1)
	insque, remque: insert/remove	element from a queue	insque(3C)
	basename: return the last	element of a path name	basename(3G)
	fez: display file	element sizes	fez(1)
	translate object file from COFF to	ELF /cof2elf:	cof2elf(1)
	elf_update: update an	ELF descriptor	elf_update(3E)
		elf: object file access library	elf(3E)
	object file type elf_fsize:	elf32_fsize: return the size of an	elf_fsize(3E)
	class-dependent object/ /elf_getehdr:	elf32_getehdr, elf32_newehdr: retrieve	elf_getehdr(3E)
	class-dependent program/ /elf_getphdr:	elf32_getphdr, elf32_newphdr: retrieve	elf_getphdr(3E)
	section header /elf_getshdr:	elf32_getshdr: retrieve class-dependent	elf_getshdr(3E)
	object file/ /elf_getehdr: elf32_getehdr,	elf32_newehdr: retrieve class-dependent	elf_getehdr(3E)
	program/ /elf_getphdr: elf32_getphdr,	elf32_newphdr: retrieve class-dependent	elf_getphdr(3E)
	class-dependent data/ elf_xlate:	elf32_xlatetof, elf32_xlatetom:	elf_xlate(3E)
	translation elf_xlate: elf32_xlatetof,	elf32_xlatetom: class-dependent data	elf_xlate(3E)
		elf_begin: make a file descriptor	elf_begin(3E)
		elf_cntl: control a file descriptor	elf_cntl(3E)
		elf_end: finish using an object file	elf_end(3E)
		elf_errmsg, elf_errno: error handling	elf_error(3E)
	elf_errmsg,	elf_errno: error handling	elf_error(3E)
		elf_fill: set fill byte	elf_fill(3E)
	elf_flagphdr, elf_flagscn,/	elf_flagdata, elf_flaghdr, elf_flagelf,	elf_flag(3E)
	elf_flagscn,/ /elf_flagdata,	elf_flaghdr, elf_flagelf, elf_flagphdr,	elf_flag(3E)
	/elf_flagdata, elf_flagehdr,	elf_flagelf, elf_flagphdr, elf_flagscn,/	elf_flag(3E)
	/elf_flagdata, elf_flagehdr, elf_flagelf,	elf_flagphdr, elf_flagscn, elf_flagshdr:/	elf_flag(3E)
	/elf_flagehdr, elf_flagelf, elf_flagphdr,	elf_flagscn, elf_flagshdr: manipulate/	elf_flag(3E)
	/elf_flagelf, elf_flagphdr, elf_flagscn,	elf_flagshdr: manipulate flags	elf_flag(3E)
	of an object file type	elf_fsize: elf32_fsize: return the size	elf_fsize(3E)
	header	elf_getarhdr: retrieve archive member	elf_getarhdr(3E)
	table	elf_getarsym: retrieve archive symbol	elf_getarsym(3E)
	object file	elf_getbase: get the base offset for an	elf_getbase(3E)
	get section data	elf_getdata, elf_newdata, elf_rawdata:	elf_getdata(3E)
	elf32_newehdr: retrieve class-dependent/	elf_getehdr: elf32_getehdr,	elf_getehdr(3E)
	identification data	elf_getident: retrieve file	elf_getident(3E)
	elf32_newphdr: retrieve class-dependent/	elf_getphdr: elf32_getphdr,	elf_getphdr(3E)
	elf_nextscn: get section information	elf_getscn, elf_ndxscn, elf_newscn,	elf_getscn(3E)
	class-dependent section header	elf_getshdr: elf32_getshdr: retrieve	elf_getshdr(3E)
		elf_hash: compute hash value	elf_hash(3E)

section information	/elf_getscn, data /elf_getdata, information elf_getscn, elf_ndxscn, access elf_getscn, elf_ndxscn, elf_newscn, /elf_getdata, elf_newdata, contents	elf_kind: determine file type elf_kind(3E) elf_ndxscn, elf_newscn, elf_nextscn: get elf_getscn(3E) elf_newdata, elf_rawdata: get section elf_getdata(3E) elf_newscn, elf_nextscn: get section elf_getscn(3E) elf_next: sequential archive member elf_next(3E) elf_nextscn: get section information elf_getscn(3E) elf_rand: random archive member access elf_rand(3E) elf_rawdata: get section data elf_getdata(3E) elf_rawfile: retrieve uninterpreted file elf_rawfile(3E) elf_strptr: make a string pointer elf_strptr(3E) elf_update: update an ELF descriptor elf_update(3E) elf_version: coordinate library and elf_version(3E) elf_xlate: elf32_xlatetof, elf_xlate(3E) elink: Environment variable sensitive elink(5) (emulated) to the termcap library curs_termcap(3X) Emulation module ptem(7) enable, disable: enable/disable LP enable(1) enable or disable process accounting acct(2) enable, disable: enable/disable LP printers enable(1) encode/decode a binary file for uuencode(1) crypt: crypt(1) client/server/ /dg_encryptsessionkey: encrypt conversation key with the dg_encryptsessionkey(2) crypt, setkey, encrypt: generate encryption crypt(3C) encrypted /isencrypt: isencrypt(3G) crypt, setkey, encrypt: generate crypt(3C) crypt: password and file encryption functions crypt(3X) makekey: generate encryption key makekey(1) program end, etext, edata: last locations in end(3C) file system/ /addexportent, remexportent, endexportent, getexportent: get exported exportent(3C) entry /getfsfile, getfstype, setfsent, endfsent: get filesystem descriptor file getfsent(3C) getgrent, getgrgid, getgrnam, setgrent, endgrent, fgetgrent: get group file/ getgrent(3C) /gethostbyname, sethostent, endhostent: get network host entry gethostent(3N) getmntent, setmntent, addmntent, endmntent, hasmntopt: get file system/ getmntent(3C) getnetbyaddr, getnetbyname, setnetent, endnetent: get network entry /getnetent, getnetent(3N) entry /getnetgrent, setnetgrent, endnetgrent, innetgr: get network group getnetgrent(3N) socket: create an endpoint for communication socket(2) t_bind: bind an address to a transport t_bind(3N) t_close: close a transport t_close(3N) look at the current event on a transport t_look(3N) t_open: establish a transport t_open(3N) manage options for a transport t_optmgmt(3N) t_unbind: disable a transport t_unbind(3N) /getprotobyname, setprotoent, endprotoent: get protocol entry getprotoent(3N) getpwent, getpwuid, getpwnam, setpwent, endpwent, setpwnfile, fgetpwent:/ getpwent(3C) getrpcbyname, getrpcnumber, setrpcent, endrpcent: get RPC entry /getrpcent, getrpcent(3N) /getservbyname, setservent, endservent: get service entry getservent(3N) /getspent, getspnam, setspent, endspent, fgetspent, lckpwwdf, ulckpwwdf:/ getspent(3C) /getutid, getutline, pututline, setutent, endutent, utmpname: access utmp file/ getut(3C) curses/ /curs_initscr: isendwin, set_term, delscreen: curs_initscr(3X) strsave, strnsave: allocate area large enough to hold string and move string/ strsave(3C) main: enter a C main program main(3C) nlist: get entries from name list nlist(3C) man: locate and print entries from the reference manuals man(1) linenum: line number entries in a common object file linenum(4) format getdents: get directory entries in a filesystem-independent getdents(2) logger: make entries in the system log logger(1) /ldlinit, ldlitem: manipulate line number entries of a common object file function ldldread(3X) /ldlseek, ldlnseek: seek to line number entries of a section of a common object/ ldldseek(3X) /ldrseek, ldnrseek: seek to relocation entries of a section of a common object/ ldldrseek(3X) convert a TERMCAP entry into a TERMINFO entry /captinfo: captinfo(1M) create a temporary version of a TERMINFO entry /chgtinfo: chgtinfo(1) return the file handle of the export entry containing filename /getfh: getfh(2) /getnetpath: get /etc/netconfig entry corresponding to NETPATH component getnetpath(3N) file system independent directory entry /dirent: dirent(4) utmp, wtmp: utmp and wtmp entry formats utmp(4) endfsent: get filesystem descriptor file entry /getfsfile, getfstype, setfsent, getfsent(3C) endgrent, fgetgrent: get group file entry /getgrgid, getgrnam, setgrent, getgrent(3C) sethostent, endhostent: get network host entry /gethostbyaddr, gethostbyname, gethostent(3N) get file system descriptor file entry /addmntent, endmntent, hasmntopt: getmntent(3C) get network configuration database entry /getnetconfig: getnetconfig(3N) setnetent, endnetent: get network entry /getnetbyaddr, getnetbyname, getnetent(3N)
---------------------	--	--

endnetgrent, innetr: get network group	entry /getnetgrent, setnetgrent,	getnetgrent(3N)
setprotoent, endprotoent: get protocol	entry /getprotobyname, getprotobyname,	getprotoent(3N)
fgtppwnt: manipulate password file	entry /setpwent, endpwent, setpwife,	getpwent(3C)
setrpcnt, endrpcnt: get RPC	entry /getrpcbyname, getrpcbynumber,	getrpcnt(3N)
setservent, endservent: get service	entry /getservbyport, getservbyname,	getservent(3N)
manipulate shadow password file	entry /getspent, lckpwdf, ulckpwdf:	getspent(3C)
endutent, utmpname: access utmp file	entry /getutline, pututline, setutent,	getut(3C)
mknod: create a file	entry in the file system	mknod(2)
captainfo: convert a TERMCAP	entry into a TERMINFO entry	captainfo(1M)
symbol name for object file symbol table	entry /ldgetname: retrieve	ldgetname(3X)
ldtbindx: compute index of symbol table	entry of an object file	ldtbindx(3X)
ldtbread: read an indexed symbol table	entry of an object file	ldtbread(3X)
putpwent: write password file	entry	putpwent(3C)
putspent: write shadow password file	entry	putspent(3C)
unlink: remove a directory	entry	unlink(2)
execution	env: set environment for command	env(1)
profile: setting up an	environ: user environment	environ(5)
fpsetsticky: IEEE floating-point	environment at login time	profile(4)
sdetab: software development	environment control /fpgetsticky,	fpgetround(3C)
environ: user	environment data base	sdetab(4)
env: set	environment	environ(5)
getenv: return value for	environment for command execution	env(1)
printenv: print out the	environment name	getenv(3C)
putenv: change or add value to	environment	printenv(1)
longname, termattr, termname: curses	environment query routines /killchar,	putenv(3C)
sde: software development	environment	term_attr(3X)
commands to reset software development	environment target /sde-target: print	sde(5)
/link:	Environment variable sensitive file link	sde-target(1)
sde-chooser: execute	environment-sensitive tool	elink(5)
deroff: remove nroff/troff, tbl, and	eqn constructs	sde-chooser(4)
jrand48, srand48, seed48, /drand48,	erand48, lrand48, nrand48, mrand48,	deroff(1)
/post_form, unpost_form: write or	erase forms from associated subwindows	drand48(3C)
/post_menu, unpost_menu: write or	erase menus from associated subwindows	form_post(3X)
wclrtoebot, clrtoeol, /curs_clear:	erase, werase, clear, wclear, clrtoebot,	menu_post(3X)
longname, /curs_termattr: baudrate,	erasechar, has_ic, has_il, killchar,	curs_clear(3X)
complementary error function	erf, erfc: error function and	curs_termattr(3X)
error function /erf,	erfc: error function and complementary	erf(3M)
and validate a date /ckdate,	err: error-logging interface	erf(3M)
validate a group id /ckgid,	errdate, helpdate, valdate: prompt for	err(7)
/dg_ext_errno: return the extended	errgid, helpgid, valgid: prompt for and	ckdate(1)
print an error message to standard	errno for the current process	ckgid(1)
function /erf, erfc:	error /extended_perror:	dg_ext_errno(2)
erfc: error function and complementary	error function and complementary error	extended_perror(3C)
elf_errmsg, elf_errno:	error function /erf,	erf(3M)
t_rcvuderr: receive a unit data	error handling	erf(3M)
strclean: STREAMS	error indication	elf_error(3E)
strerr: STREAMS	error logger cleanup program	t_rcvuderr(3N)
log: interface to STREAMS	error logger server	strclean(1M)
/mkstr: create an	error logging and event tracing	strerr(1M)
/extended_strerror: get extended	error message file by messaging C source	log(7)
strerror: get	error message string	mkstr(1)
t_error: produce	error message string	extended_strerror(3C)
/extended_perror: print an	error message	strerror(3C)
perror: print system	error message to standard error	t_error(3N)
intro: introduction to system calls and	error messages	extended_perror(3C)
matherr:	error numbers	perror(3C)
err:	error-handling function	intro(2)
spellin, hashcheck: find spelling	error-logging interface	matherr(3M)
copy strings, compressing or expanding	errors /spell, hashmake,	err(7)
transport user t_connect:	escape codes /streadd, strcadd, streccpy:	spell(1)
t_open:	establish a connection with another	strccpy(3G)
connection dial:	establish a transport endpoint	t_connect(3N)
setmnt:	establish an out-going terminal line	t_open(3N)
/admsvcorder: manage search order for	establish mount table	dial(3C)
NETPATH component getnetpath: get	/etc/hosts, NIS, and DNS databases	setmnt(1M)
/end,	/etc/netconfig entry corresponding to	admsvcorder(1M)
admether: manage	etext, edata: last locations in program	getnetpath(3N)
ether_hostton, /ethers, ether_ntoa,	ether database	end(3C)
/ether_ntoa, ether_aton, ether_ntohost,	ether_aton, ether_ntohost,	admether(1M)
ether_hostton, ether_line: Ethernet/	ether_hostton, ether_line: Ethernet/	ethers(3N)
		ethers(3N)

/ether_ntohost, ether_hostton,	ether_line: Ethernet address mapping/	ethers(3N)
/ether_hostton, ether_line:	Ethernet address mapping operations	ethers(3N)
dgen: second generation integrated	Ethernet interface	dgen(7)
hken: Hawk	Ethernet interface	hken(7)
inen: integrated	Ethernet interface	inen(7)
ether_hostton, ether_line:/	ether_ntoa, ether_aton, ether_ntohost,	ethers(3N)
ethers, ether_ntoa, ether_aton,	ether_ntohost, ether_hostton,/	ethers(3N)
ether_ntohost, ether_hostton,/	ethers, ether_ntoa, ether_aton,	ethers(3N)
eucset: set or get	EUC code set widths	eucset(1)
eucioctl: generic interface to	EUC handling TTY drivers and modules	eucioctl(5)
handling TTY drivers and modules	eucioctl: generic interface to EUC	eucioctl(5)
hypot:	Euclidean distance function	hypot(3M)
expr:	eucset: set or get EUC code set widths	eucset(1)
test: condition	evaluate arguments as an expression	expr(1)
t_look: look at the current	evaluation command	test(1)
interface to STREAMS error logging and	event on a transport endpoint	t_look(3N)
edit: text editor (variant of	event tracing /log:	log(7)
(edit) display editor based on	ex for casual users)	edit(1)
cscope: interactively	ex: text editor	ex(1)
sigprocmask:	ex /vi, view, vedit: screen-oriented	vi(1)
sigaction:	examine a C program	cscope(1)
readiness select:	examine and change blocked signals	sigprocmask(2)
sigpending:	examine and change signal action	sigaction(2)
crash:	examine file descriptors for I/O	select(2)
lpq:	examine pending signals	sigpending(2)
retrieve a command description and usage	examine system images	crash(1M)
devfree: release devices from	examine the spool queue	lpq(1)
devreserv: reserve devices for	examples /usage:	usage(1)
execlp, execvp: execute a file	exclusive use	devfree(1M)
execvp: execute a file	exclusive use	devreserv(1M)
a file /exec: execl, execv,	exec: execl, execv, execl, execve,	exec(2)
exec: execl, execv, execl, execve,	execl, execv, execl, execve, execlp,	exec(2)
ldfcn: COFF	execl, execv, execlp, execvp: execute	exec(2)
doconfig:	execlp, execvp: execute a file	exec(2)
execv, execl, execve, execlp, execvp:	executable file access routines	ldfcn(4)
csync: synchronize hardware caches for	execute a configuration script	doconfig(3N)
xargs: construct argument list(s) and	execute a file /exec: execl,	exec(2)
at, batch:	execute access	csync(2)
sde-chooser:	execute command	xargs(1)
regcmp, regex: compile and	execute commands at a later time	at(1)
regcmp, regex: compile and	execute environment-sensitive tool	sde-chooser(4)
uuxqt:	execute regular expression	regcmp(3G)
env: set environment for command	execute regular expression	regcmp(3X)
sleep: suspend	execute remote command requests	uuxqt(1M)
sleep: suspend	execution	env(1)
monitor: prepare	execution for an interval	sleep(1)
/profil: set up	execution for interval	sleep(3C)
uux: UNIX-to-UNIX system command	execution profile	monitor(3C)
execute a file /exec: execl,	execution time profiling for a process	profil(2)
/exec: execl, execv, execl,	execution	uux(1)
execl, execv, execl, execve, execlp,	execv, execl, execve, execlp, execvp:	exec(2)
link, unlink:	execve, execlp, execvp: execute a file	exec(2)
tunefs: tune an	execvp: execute a file /exec:	exec(2)
creat: create a new file or rewrite an	exercise link and unlink system calls	link(1M)
exit,	existing file system	tunefs(1M)
log10f, pow, powf, sqrt, sqrtf:/	existing one	creat(2)
pack, pcat, unpack: compress and	exit, _exit: terminate process	exit(2)
compress, uncompress, zcat: compress,	_exit: terminate process	exit(2)
zcat: compress, expand or display	exp, expf, cbrt, log, logf, log10,	exp(3M)
strecpy: copy strings, compressing or	expand files	pack(1)
t_snd: send data or	expand or display expanded files	compress(1)
t_rcv: receive data or	expanded files /compress, uncompress,	compress(1)
pow, powf, sqrt, sqrtf:/	expanding escape codes /strcadd,	strncpy(3G)
/log10, log10f, pow, powf, sqrt, sqrtf:	expedited data over a connection	t_snd(3N)
getfh: return the file handle of the	expedited data sent over a connection	t_rcv(3N)
/endexportent, getexportopt: get	expf, cbrt, log, logf, log10, log10f,	exp(3M)
addexportent, remexportent,/	exponential, logarithm, power, square/	exp(3M)
mounting via NFS	export entry containing filename	getfh(2)
	exported file system information	exportent(3C)
	exportent, getexportent, setexportent,	exportent(3C)
	exportfs: make a directory available for	exportfs(2)

	expression	expr: evaluate arguments as an	expr(1)
regex: compile, step, advance:	regular	expression compile and match routines	regexp(5)
regexpr: compile, step, advance:	regular	expression compile and match routines	regexpr(3G)
	regcmp: regular	expression compile	regcmp(1)
	expr: evaluate arguments as an	expression	expr(1)
	regex: compile and execute regular	expression /regcmp,	regcmp(3G)
	regex: compile and execute regular	expression /regcmp,	regcmp(3X)
regex, re_comp, re_exec: handle regular	a file for a pattern using full regular	expressions /berk_regex,	berk_regex(3C)
		expressions /egrep: search	egrep(1)
		exstr: extract strings from source files	exstr(1)
	/dg_ext_errno: return the	extended errno for the current process	dg_ext_errno(2)
	/extended_strerror: get	extended error message string	extended_strerror(3C)
	dg_fstat: get	extended file status information	dg_fstat(2)
	dg_stat: get	extended file status information	dg_stat(2)
	termiox:	extended general terminal interface	termiox(7)
	dg_xtrace:	extended process trace	dg_xtrace(2)
	dg_seek, dg_block_seek:	extended seek functions	dg_seek(3C)
	sys_local: invoke an	extended system call	sys_local(2)
	to standard error	extended_perror: print an error message	extended_perror(3C)
	message string	extended_strerror: get extended error	extended_strerror(3C)
stkprotect: set access for future stack		extensions	stkprotect(2)
xdr_wrapstring: library routines for	implement shared strings	external data representation /xdr_void,	xdr(3N)
	xstr:	extract strings from C programs to	xstr(1)
	exstr:	extract strings from source files	exstr(1)
replace with catgets calls.	catexstr:	extract strings from source files,	catexstr(1)
	fsplit: split	f77 or ratfor files	fsplit(1)
/ceil, ceilf, copysign, fmod, fmodf,	/ceilf, copysign, fmod, fmodf, fabs,	fabs, fabsf, rint, remainder: floor,/	floor(3M)
signal: simplified software signal	jobs: summary of DG/UX job control	fabsf, rint, remainder: floor, ceiling,/	floor(3M)
ipcs: report inter-process communication	helpadm: make changes to the help	facilities /berk_signal,	berk_signal(3C)
	help: help	facilities	jobs(3C)
	factor:	facilities status	ipcs(1)
	true,	facility database	helpadm(1M)
	da: AViiON	facility	help(1)
	cied: AViiON	factor a number	factor(1)
	cimd: AViiON	factor: factor a number	factor(1)
	cird: AViiON	false: provide truth values	true(1)
	sd: AViiON	family disk array subsystem	da(7)
adapter subsystem	hada: AViiON	family disk subsystem	cied(7)
controller	syac: AViiON	family disk subsystem	cimd(7)
	lp: DGC	family disk subsystem	cird(7)
	cisc: AViiON	family disk subsystem	sd(7)
	insc: AViiON	family High Availability Disk Array	hada(7)
	ncsc: AViiON	family intelligent asynchronous	syac(7)
	st: AViiON	family line printer special files	lp(7)
integer data in a machine-independent	descriptor to object in file system/	family SCSI adapter subsystem	cisc(7)
handle misaligned memory access	the calling process	family SCSI adapter subsystem	insc(7)
	file	family SCSI adapter subsystem	ncsc(7)
	number to string	family tape subsystem	st(7)
STREAMS-based file descriptor	fdetach: detach a name from a	fashion /sputl, sgetl: access long	sputl(3X)
fdopen, freopen,	fdopen: open a stream	fattach: attach STREAMS-based file	fattach(3C)
inquiries	feof, clearerr, fileno: stream status	faults /misalign:	misalign(5)
status inquiries	ferror, feof, clearerr, fileno: stream	fchdir: change the working directory of	fchdir(2)
/fetch_and_add: indivisible	fetch, store, delete, firstkey, nextkey:	fchmod: change mode of file	fchmod(2)
data base subroutines /dbmimit,	to memory location	fchown: change user id and group id of a	fchown(2)
head: give the first	few lines	fclose, fflush: close or flush a stream	fclose(3S)
	fclose,	fcntl: file control options	fcntl(5)
	a stream /getc, getchar,	fcntl: file descriptor control	fcntl(2)
/getgrgid, getgrnam, setgrent, endgrent,		fcvt, gcvt: convert floating-point	ecvt(3C)
		fdetach: detach a name from a	fdetach(3C)
		fdopen: open a stream	fdopen(3S)
		feof, clearerr, fileno: stream status	ferror(3S)
		ferror, feof, clearerr, fileno: stream	ferror(3S)
		fetch and add to memory location	fetch_and_add(2)
		fetch, store, delete, firstkey, nextkey:	dbm(3X)
		fetch_and_add: indivisible fetch and add	fetch_and_add(2)
		few lines	head(1)
		fez: display file element sizes	fez(1)
		fclose, fflush: close or flush a stream	fclose(3S)
		ffs: find first set bit	ffs(3C)
		getc, getw: get character or word from	getc(3S)
		getgrent: get group file entry	getgrent(3C)

stream /fsetpos,	fgetpos: reposition a file pointer in a	fsetpos(3C)
/getpwnam, setpwent, endpwent, setpfile,	fgetpwent: manipulate password file/	getpwent(3C)
gets,	fgets: get a string from a stream	gets(3S)
/getspent, getspname, setspent, endspent,	fgetspent, lckpword, ulckpword: manipulate/	getspent(3C)
stream getwc, getwchar,	fgetwc: get wchar_t character from a	getwc(3W)
stream getws,	fgetws: get a wchar_t string from a	getws(3W)
string	fgrep: search a file for a character	fgrep(1)
set_max_field: set and get forms	field attributes /field_status,	form_field_buffer(3X)
dynamic_field_info: get forms	field characteristics /field_info,	form_field_info(3X)
/field_type, field_arg: forms	field data type validation	form_field_validation(3X)
field_index: set forms current page and	field /set_current_field, current_field,	form_page(3X)
/data_ahead, data_behind: tell if forms	field has off-screen data ahead or/	form_data(3X)
field_opts_off, field_opts: forms	field option routines /field_opts_on,	form_field_opts(3X)
validation /set_field_type, field_type,	field_arg: forms field data type	form_field_validation(3X)
format the/ /field_fore, set_field_back,	field_back, set_field_pad, field_pad:	form_field_attributes(3X)
/form_field_buffer: set_field_buffer,	field_buffer, set_field_status,/	form_field_buffer(3X)
to forms /set_form_fields, form_fields,	field_count, move_field: connect fields	form_field(3X)
/form_field_attributes: set_field_fore,	field_fore, set_field_back, field_back,/	form_field_attributes(3X)
field /set_current_field, current_field,	field_index: set forms current page and	form_page(3X)
forms field/ /form_field_info:	field_info, dynamic_field_info: get	form_field_info(3X)
assign/ /form_term, set_field_init,	field_init, set_field_term, field_term:	form_hook(3X)
/form_field_just: set_field_just,	field_just: format the general/	form_field_just(3X)
/field_opts_on, field_opts_off,	field_opts: forms field option routines	form_field_opts(3X)
option/ /set_field_opts, set_field_opts_on,	field_opts_off, field_opts: forms field	form_field_opts(3X)
/form_field_opts: set_field_opts,	field_opts_on, field_opts_off,/	form_field_opts(3X)
attributes/ /field_back, set_field_pad,	field_pad: format the general display	form_field_attributes(3X)
bufsplit: split buffer into	fields	bufsplit(3G)
free_field,: create and destroy forms	fields /dup_field, link_field,	form_field_new(3X)
cut: cut out selected	fields of each line of a file	cut(1)
field_count, move_field: connect	fields to forms /form_fields,	form_field(3X)
forms/ /field_buffer, set_field_status,	field_status, set_max_field: set and get	form_field_buffer(3X)
routines/ /field_init, set_field_term,	field_term: assign application-specific	form_hook(3X)
/form_field_validation: set_field_type,	field_type, field_arg: forms field data/	form_field_validation(3X)
/link_fieldtype: forms	fieldtype routines	form_fieldtype(3X)
/form_field_userptr: set_field_userptr,	field_userptr: associate application/	form_field_userptr(3X)
mkfifo: create a new	FIFO	mkfifo(3C)
mkfifo: make	FIFO special file	mkfifo(1M)
utime: set	file access and modification times	utime(2)
utimes: set	file access and modification times	utimes(2)
elf: object	file access library	elf(3E)
ldfcn: COFF executable	file access routines	ldfcn(4)
access: determine the accessibility of a	file	access(2)
berk_diff: Berkeley differential	file and directory comparator	berk_diff(1)
tar: tape	file archiver	tar(1)
cpio: copy	file archives in and out	cpio(1)
parts of an object or object archive	file /att_dump: dump	att_dump(1)
rcs: change RCS	file attributes	rcs(1)
mkstr: create an error message	file by massaging C source	mkstr(1)
chgrp: change the group ownership of a	file	chgrp(1)
chmod: change mode of	file	chmod(2)
lchown: change user id and group id of a	file /chown,	chown(2)
diff: differential	file comparator	diff(1)
berk_diff3: Berkeley 3-way differential	file comparison	berk_diff3(1)
diff3: 3-way differential	file comparison	diff3(1)
compver: compatible versions	file	compver(4)
/elf_rawfile: retrieve uninterpreted	file contents	elf_rawfile(3E)
fcntl:	file control options	fcntl(5)
uuto, uupick: public UNIX-to-UNIX system	file copy	uuto(1)
copyright: copyright information	file	copyright(4)
core: format of core image	file	core(4)
cprs: compress a common object	file	cprs(1)
umask: set and get	file creation mask	umask(2)
crontab: user crontab	file	crontab(1)
ctags: create a tags	file	ctags(1)
read (write) a curses screen from (to) a	file /scr_restore, scr_init, scr_set:	curlscr_dump(3X)
out selected fields of each line of a	file /cut: cut	cut(1)
dd: convert and copy a	file	dd(1)
delta: make a delta (change) to an SCCS	file	delta(1)
close: close an object associated with a	file descriptor	close(2)
fcntl:	file descriptor control	fcntl(2)
dup: duplicate an open	file descriptor	dup(2)

elf_begin: make a	file descriptor	elf_begin(3E)
elf_cntl: control a	file descriptor	elf_cntl(3E)
detach a name from a STREAMS-based	file descriptor /fdetach:	fdetach(3C)
isastream: test a	file descriptor	isastream(3C)
descriptor dup2: duplicate an open	file descriptor onto a specific	dup2(2)
name/ /fattach: attach STREAMS-based	file descriptor to object in file system	fattach(3C)
select: examine	file descriptors for I/O readiness	select(2)
remove an advisory lock on an open DG/UX	file: determine file type	file(1)
sact: print current SCCS	file /dg_flock: apply or	dg_flock(3C)
fez: display	file editing activity	sact(1)
elf_end: finish using an object	file element sizes	fez(1)
get the base offset for an object	file	elf_end(3E)
crypt: password and	file /elf_getbase:	elf_getbase(3E)
endfsent: get filesystem descriptor	file encryption functions	crypt(3X)
setgrent, endgrent, fgetgrent: get group	file entry /getfstype, setfsent,	getfsent(3C)
hasmntopt: get file system descriptor	file entry /getgrgid, getgrnam,	getgrent(3C)
fgetpwent: manipulate password	file entry /addmntent, endmntent,	getmntent(3C)
ulckpwwdf: manipulate shadow password	file entry /endpwent, setpwfile,	getpwent(3C)
endutent, utmpname: access utmp	file entry /fgetspent, lckpwwdf,	getspent(3C)
mknod: create a	file entry /pututline, setutent,	getut(3C)
putpwent: write password	file entry in the file system	mknod(2)
putspent: write shadow password	file entry	putpwent(3C)
execve, execlp, execvp: execute a	file entry	putspent(3C)
fchmod: change mode of	file /exec: execl, execlv, execl,	exec(2)
fchown: change user id and group id of a	file	fchmod(2)
dumptab: tape table	file	fchown(2)
fgrep: search a	file for dump2	dumptab(4)
grep: search a	file for a character string	fgrep(1)
expressions egrep: search a	file for a pattern	grep(1)
dumpcycle: dump cycle	file for a pattern using full regular	egrep(1)
which: locate a program	file for backups	dumpcycle(4M)
constants limits: header	file for csh(1) users	which(1)
ldopen, ldaopen: open an object	file for implementation-specific	limits(4)
open: open	file for reading	ldopen(3X)
syslog.conf: configuration	file for reading or writing	open(2)
/uudecode: encode/decode a binary	file for syslogd system log server	syslog.conf(5)
acct: per-process accounting	file for transmission via mail	uudecode(1)
ar: DG/UX common archive	file format	acct(4)
tar: tape archive	file format	ar(4)
intro: introduction to	file format	tar(5)
intro: introduction to	file formats	intro(4)
at a time /pg: display	file formats	intro(4M)
cof2elf: translate object	file forward or backward one screenful	pg(1)
scstorcs: build RCS	file from COFF to ELF	cof2elf(1)
removef: remove a	file from SCCS file	scstorcs(1)
line number entries of a common object	file from software database	removef(1M)
get: check out a version of an SCCS	file function /lditem: manipulate	ldread(3X)
group: group	file	get(1)
containing filename getfh: return the	file	group(4)
retrieve class-dependent object	file handle of the export entry	getfh(2)
filehdr:	file header /elf32_newehdr:	elf_getehdr(3E)
ldfhead: read the	file header for common object files	filehdr(4)
ldohseek: seek to the optional	file header of a common object file	ldfhead(3X)
/elf_getident: retrieve	file header of an object file	ldohseek(3X)
pathfind: search for named	file identification data	elf_getident(3E)
copylist: copy a	file in named directories	pathfind(3G)
split: split a	file into memory	copylist(3G)
issue: issue identification	file into pieces	split(1)
header of a member of a COFF archive	file	issue(4)
ldclose, ldaclose: close a common object	file /ldahread: read the archive	ldahread(3X)
read the file header of a common object	file	ldclose(3X)
entries of a section of a common object	file /ldfhead:	ldfhead(3X)
to the optional file header of an object	file /ldnlseek: seek to line number	ldnlseek(3X)
entries of a section of a common object	file /ldohseek: seek	ldohseek(3X)
section header of a common object	file /ldnrseek: seek to relocation	ldnrseek(3X)
section of a common object	file /ldnshread: read an indexed/named	ldshread(3X)
index of symbol table entry of an object	file /seek to an indexed/named	ldsseek(3X)
indexed symbol table entry of an object	file /ldtbindex: compute	ldtbindex(3X)
seek to the symbol table of an object	file /ldtbread: read an	ldtbread(3X)
line number entries in a common object	file /ldtbseek:	ldtbseek(3X)
	file /linenum:	linenum(4)

elink: Environment variable sensitive	file link	elink(5)
link: create a new link to a	file	link(2)
grace/ /dg_lock_reset: reset remote	file lock database, start lock reclaim	dg_lock_reset(2)
dfm: DOS	file manager	dfm(4M)
hfm: high sierra	file manager	hfm(4)
master: format of a master	file	master(4)
the comment section of an object	file. /mcs: manipulate	mcs(1)
merge: three-way	file merge	merge(1)
mkdir: create a directory	file	mkdir(2)
mkfifo: make FIFO special	file	mkfifo(1M)
mknod: build a special	file	mknod(1M)
chmod: change	file mode	chmod(1)
ctermid: generate	file name for terminal	ctermid(3S)
mkstemp: make a unique	file name	mkstemp(3C)
mktemp: make a unique	file name	mktemp(3C)
realpath: returns the real	file name	realpath(3C)
newform: change the format of a text	file	newform(1)
nm: print name list of common object	file	nm(1)
inode:	file node structure	inode(4)
null: the null	file	null(7)
ttyslot: find the slot in the utmp	file of the current user	ttyslot(3C)
more, page: display	file one screenful at a time	more(1)
fuser: identify processes using a	file or file structure	fuser(1M)
creat: create a new	file or rewrite an existing one	creat(2)
chown: change	file owner	chown(1)
passwd: password	file	passwd(4)
report the parent directory name of a	file path name /dirname:	dirname(3G)
pkginfo: package characteristics	file	pkginfo(4)
pkgmap: package contents description	file	pkgmap(4)
pkgproto: generate a prototype	file	pkgproto(1)
fseek, rewind, ftell: reposition a	file pointer in a stream	fseek(3S)
fsetpos, fgetpos: reposition a	file pointer in a stream	fsetpos(3C)
reverse the page order in a PostScript	file /postreverse:	postreverse(1)
prototype: package information	file	prototype(4)
prs: print an SCCS	file	prs(1)
pwck, grpck: check password or group	file	pwck(1M)
rcsfile: format of RCS	file	rcsfile(4)
readv: read from	file	readv(2)
information for a common object	file /reloc: relocation	reloc(4)
remove: remove	file	remove(3C)
rename: change the name of a	file	rename(2)
order of characters in each line of	file /rev: reverse	rev(1)
rmDEL: remove a delta from an SCCS	file	rmDEL(1)
rmdir: remove a directory	file	rmdir(2)
bfs: big	file scanner	bfs(1)
compare two versions of an SCCS	file /sccsdiff:	sccsdiff(1)
sccsfile: format of SCCS	file	sccsfile(4)
sccstorcs: build RCS file from SCCS	file	sccstorcs(1)
scr_dump: format of curses screen image	file	scr_dump(4)
space: disk space requirement	file	space(4)
dg_mstat: get	file status	dg_mstat(2)
fstat: get	file status	fstat(2)
dg_fstat: get extended	file status information	dg_fstat(2)
dg_stat: get extended	file status information	dg_stat(2)
lstat: get	file status	lstat(2)
stat: get	file status	stat(2)
strings in an object or other binary	file /strings: find the printable	strings(1)
information from an object	file /strip: strip non-executable	strip(1)
identify processes using a file or	file structure /fuser:	fuser(1M)
sum: print checksum and block count of a	file	sum(1)
syacdump: dump syac memory to a	file	syacdump(1M)
retrieve symbol name for object	file symbol table entry /ldgetname:	ldgetname(3X)
syms: common object	file symbol table format	syms(4)
symlink: create a symbolic link	file	symlink(2)
dump2: incremental	file system backup	dump2(1M)
filesave, tapesave: daily/weekly	file system backup	filesave(1M)
dg_fsdb:	file system debugger	dg_fsdb(1M)
fsdb:	file system debugger	fsdb(1M)
/addmntent, endmntent, hasmntopt: get	file system descriptor file entry	getmntent(3C)
ustat: get	file system device statistics	ustat(2)
umount: remove a	file system device	umount(2)

dg_mount: mount a	file system	dg_mount(2)
dump: incremental	file system dump	dump(1M)
fs:	file system format	fs(4)
fstats: get information about a mounted	file system	fstats(2)
fstatvfs: return information about a	file system	fstatvfs(2)
hier: DG/UX	file system hierarchy	hier(5)
/dirent:	file system independent directory entry	dirent(4)
dumpfs: dump	file system information	dumpfs(1M)
endexportent, getexportopt: get exported	file system information /remexportent,	exportent(3C)
synchronize disk and memory resident	file system information /sync:	sync(2)
mfs: memory	file system	mfs(4)
mkfs, newfs: create a	file system	mkfs(1M)
mknod: create a file entry in the	file system	mknod(2)
mount: mount a	file system	mount(2)
file descriptor to object in	file system name space /STREAMS-based	fattach(3C)
dg_mknod: create a	file system node	dg_mknod(2)
filesystem:	file system organization	filesystem(7)
restore: incrementally restore a	file system	restore(1M)
stats: get information about a mounted	file system	stats(2)
statvfs: return information about a	file system	statvfs(2)
mnttab: mounted	file system table	mnttab(4)
tunefs: tune an existing	file system	tunefs(1M)
sysfs: returns information about	file system types	sysfs(2)
system: format of a kernel description	file	system(4)
admbackup: manage backup and recovery of	file systems	admbackup(1M)
/admfilesystem: manage	file systems	admfilesystem(1M)
them /fsck: check	file systems for consistency and repair	fsck(1M)
fstab: static information about	file systems	fstab(4)
ncheck checklist: list of	file systems processed by fsck and	checklist(4)
volcopy, labelit: copy	file systems with label checking	volcopy(1M)
tail: deliver the last part of a	file	tail(1)
tmpfile: create a temporary	file	tmpfile(3S)
tempnam: create a name for a temporary	file /tmpnam,	tempnam(3S)
ftruncate: set a	file to a specified length	ftruncate(3C)
truncate: truncate a	file to a specified length	truncate(2)
twrite: writes a	file to tape	twrite(1)
database installf: add a	file to the software installation	installf(1M)
access and modification times of a	file /touch: update	touch(1)
tposn: position tape to specified	file	tposn(1)
system uucico:	file transport program for the uucp	uucico(1M)
uusched: the scheduler for the uucp	file transport program	uusched(1M)
ftw, nftw: walk a	file tree	ftw(3C)
return the size of an object	file type /elf_fsize: elf32_fsize:	elf_fsize(3E)
elf_kind: determine	file type	elf_kind(3E)
file: determine	file type	file(1)
unget: undo a previous get of an SCCS	file	unget(1)
uniq: report repeated lines in a	file	uniq(1)
identified by process/ /dg_file_info: get	file usage information for process	dg_file_info(2)
termprinter: print a	file using the 40014A Terminal Server	termprinter(1)
the uucp directories and permissions	file /uuccheck: check	uuccheck(1M)
val: validate SCCS	file	val(1)
vipw: edit the system password	file	vipw(1M)
vtc.addr: SYAC VTC configuration	file	vtc.addr(4M)
/synchronously read data from a	file without system buffering	dg_unbuffered_read(2)
/synchronously write data to a	file without system buffering	dg_unbuffered_write(2)
writev: write on a	file	writev(2)
umask: set	file-creation mode mask	umask(1)
process a record lock request on a	filehandle /dg_lcntl:	dg_lcntl(2)
files	filehdr: file header for common object	filehdr(4)
handle of the export entry containing	filename /getfh: return the file	getfh(2)
ferror, feof, clearerr,	fileno: stream status inquiries	ferror(3S)
search and print process accounting	file(s) /acctcom:	acctcom(1)
acctmerg: merge or add total accounting	files	acctmerg(1M)
admin: create and administer SCCS	files	admin(1)
admfsinfo: display information about	files and directories	admfsinfo(1M)
cmp: compare two	files	cmp(1)
or reject lines common to two sorted	files /comm: select	comm(1)
compress, expand or display expanded	files /compress, uncompress, zcat:	compress(1)
cp: copy	files	cp(1)
depend: software dependencies	files	depend(4)
exstr: extract strings from source	files	exstr(1)

filehdr: file header for common object	files	filehdr(4)
find: find	files	find(1)
mkmsgs: create message	files for use by gettxt	mkmsgs(1)
frec: recover	files from a backup tape	frec(1M)
tread: read	file(s) from tape	tread(1)
fspec: format specification in text	files	fspec(4)
fsplit: split f77 or ratfor	files	fsplit(1)
ident: identify	files	ident(1)
/fsync: synchronize a	file's in-core state with that on disk	fsync(2)
postprint: translate text	files into PostScript	postprint(1)
introduction to DG/UX System special	files /intro:	intro(7)
ld: link editor for object	files	ld(1)
ld: link editor for common object	files	ld-coff(1)
ln: link	files	ln(1)
lockf: record locking on	files	lockf(3C)
DGC AViiON family line printer special	files /lp:	lp(7)
passmgmt: password	files management	passmgmt(1M)
mv: move	files	mv(1)
rm, rmdir: remove, delete	files or directories	rm(1)
pack, pcat, unpack: compress and expand	files	pack(1)
PostScript translator for Diablo 630	files /postdaisy:	postdaisy(1)
PostScript translator for DMD bitmap	files /postdmd:	postdmd(1)
translator for plot(4) graphics	files /postplot: PostScript	postplot(1)
PostScript translator for tektronix 4014	files /posttek:	posttek(1)
pr: print	files	pr(1)
rcsclean: clean up working	files	rcsclean(1)
catexstr: extract strings from source	files, replace with catgets calls.	catexstr(1)
messages and other information about RCS	files /rlog: print log	rlog(1)
size: print section sizes of object	files	size(1)
sort: sort and/or merge	files	sort(1)
/getdtablesize: return the number of open	files the current process can have	getdtablesize(2)
cat: concatenate and type	files to standard output	cat(1)
what: identify SCCS	files	what(1)
system backup	filesave, tapesave: daily/weekly file	filesave(1M)
/getfstype, setfsent, endfsent: get	filesystem descriptor file entry	getfsent(3C)
filesystem: file system organization	filesystem: file system organization	filesystem(7)
filesystem-independent format	filesystem-independent format	getdents(2)
getdents: get directory entries in a	filesystems	mount(1M)
mount, umount: mount and dismount	fill byte	elf_fill(3E)
elf_fill: set	fill in the set of	sigfillset(2)
implementation-defined/ sigfillset:	filter	nl(1)
nl: line numbering	filter reverse line-feeds	col(1)
col:	filter, use_env, putwin, getwin, /	curl_util(3X)
curl_util: unctrl, keyname,	filters used with the LP print service	lpfilter(1M)
/lpfilter: administer	find files	find(1)
find:	find: find files	find(1)
ffs:	find first set bit	ffs(3C)
ttyname, isatty:	find name of a terminal	ttyname(3C)
library lorder:	find ordering relation for an object	lorder(1)
spell, hashmake, spellin, hashcheck:	find spelling errors	spell(1)
or other binary file /strings:	find the printable strings in an object	strings(1)
current user ttyslot:	find the slot in the utmp file of the	ttyslot(3C)
and remote users	finger: display information about local	finger(1)
information server	fingerd, in.fingerd: remote user	fingerd(1M)
/fingerd, in.	fingerd: remote user information server	fingerd(1M)
elf_end:	finish using an object file	elf_end(3E)
type of/ isnan, isnand, isnanf,	finite, fpclass, unordered: determine	isnan(3C)
fold: fold long lines for	finite width output device	fold(1)
head: give the	first few lines	head(1)
string index: search for the	first occurrence of a character in a	index(3C)
ffs: find	first set bit	ffs(3C)
/dbmunit, fetch, store, delete,	firstkey, nextkey: data base subroutines	dbm(3X)
tee: pipe	fitting	tee(1)
elf_flagscn, elf_flagshdr: manipulate	flags /elf_flagelf, elf_flagphdr,	elf_flag(3E)
routines curl_beep: beep,	flash: curses bell and screen flash	curl_beep(3X)
beep, flash: curses bell and screen	flash routines /curl_beep:	curl_beep(3X)
/fpgetsticky, fpsetsticky: IEEE	floating-point environment control	fpgetround(3C)
fpclass, unordered: determine type of	floating-point number /isnanf, finite,	isnan(3C)
ecvt, fcvt, gcvt: convert	floating-point number to string	ecvt(3C)
nextafter, scalb: manipulate parts of	floating-point numbers /modf, modff,	frexp(3C)
drem: IEEE	floating-point remainder	drem(3M)

/fmodf, fabs, fabsf, rint, remainder:	floor, ceiling, remainder, absolute/	floor(3M)
fmod, fmodf, fabs, fabsf, rint,/	floor, floorf, ceil, ceilf, copysign,	floor(3M)
fmodf, fabs, fabsf, rint,/ floor,	floorf, ceil, ceilf, copysign, fmod,	floor(3M)
cfloor: generate a C	flow graph	cflow(1)
fclose, fflush: close or	flush a stream	fclose(3S)
/use_env, putwin, getwin, delay_output,	flushinp: miscellaneous curses utility/	flushinp(3X)
/rpow, msqrt, mcomp, move, min, omin,	fmin, m_in, mout, omout, fmout, m_out,/	mp(3X)
floor, floorf, ceil, ceilf, copysign,	fmod, fmodf, fabs, fabsf, rint,/	floor(3M)
/floorf, ceil, ceilf, copysign, fmod,	fmodf, fabs, fabsf, rint, remainder:/	floor(3M)
/min, omin, fmin, m_in, mout, omout,	fmout, m_out, sdiv, itom: multiple/	mp(3X)
	fmt: simple text formatter	fmt(1)
levels for application to be used with	fmtmsg /build list of severity	addseverity(3C)
system console	fmtmsg: display a message on stderr or	fmtmsg(1)
system console	fmtmsg: display a message on stderr or	fmtmsg(3C)
output device	fold: fold long lines for finite width	fold(1)
device /fold:	fold long lines for finite width output	fold(1)
download host resident PostScript	fonts /download:	download(1)
	fopen, freopen, fdopen: open a stream	fopen(3S)
tcsetpgrp: set terminal	foreground process group id	tcsetpgrp(3C)
	fork: create a new process	fork(2)
acct: per-process accounting file	format	acct(4)
information ttyadm:	format and output TTY port monitor	ttyadm(1M)
message /nlsrequest:	format and send listener service request	nlsrequest(3N)
ar: DG/UX common archive file	format	ar(4)
getdate, getdate_err: convert user	format date and time	getdate(3C)
fs: file system	format	fs(4)
entries in a filesystem-independent	format /getdents: get directory	getdents(2)
system:	format of a kernel description file	system(4)
master:	format of a master file	master(4)
newform: change the	format of a text file	newform(1)
core:	format of core image file	core(4)
cpio:	format of cpio archive	cpio(4)
scr_dump:	format of curses screen image file	scr_dump(4)
rscfile:	format of RCS file	rscfile(4)
sccsfile:	format of SCCS file	sccsfile(4)
pkgtrans: translate package	format	pkgtrans(1)
fspec:	format specification in text files	fspec(4)
syms: common object file symbol table	format	syms(4)
tar: tape archive file	format	tar(5)
/set_field_just, field_just:	format the general appearance of forms	form_field_just(3X)
/field_back, set_field_pad, field_pad:	format the general display attributes of/	form_field_attributes(3X)
intro: introduction to file	formats	intro(4)
intro: introduction to file	formats	intro(4M)
utmp, wtmp: utmp and wtmp entry	formats	utmp(4)
/mvscanw, mvwscanw, vwscanw: convert	formatted input from a curses widow	scanw(3S)
scanf, fscanf, sscanf: convert	formatted input	scanf(3S)
scanf, fscanf, sscanf: convert	formatted input	scanf(3W)
list /vscanf, vfscanf, vsscanf: convert	formatted input using varargs argument	vscanf(3S)
gencat: generate a	formatted message catalogue	gencat(1)
/mvprintw, mvwprintw, vwprintw: print	formatted output in curses windows	printw(3X)
list /vprintf, vfprintf, vsprintf: print	formatted output of a variable argument	vprintf(3S)
list /vprintf, vfprintf, vsprintf: print	formatted output of a variable argument	vprintf(3W)
printf: print	formatted output	printf(1)
printf, fprintf, sprintf: print	formatted output	printf(3S)
printf, fprintf, sprintf: print	formatted output	printf(3W)
fmt: simple text	formatter	fmt(1)
localeconv: get numeric	formatting information	localeconv(3C)
forms window cursor	form_cursor: pos_form_cursor: position	form_cursor(3X)
tell if forms field has off-screen data/	form_data: data_ahead, data_behind:	form_data(3X)
forms subsystem	form_driver: command processor for the	form_driver(3X)
form_fields, field_count, move_field:/	form_field: set_form_fields,	form_field(3X)
field_fore, set_field_back, field_back,/	form_field_attributes: set_field_fore,	form_field_attributes(3X)
field_buffer, set_field_status,/	form_field_buffer: set_field_buffer,	form_field_buffer(3X)
dynamic_field_info: get forms field/	form_field_info: field_info,	form_field_info(3X)
field_just: format the general/	form_field_just: set_field_just,	form_field_just(3X)
link_field, free_field,: create and/	form_field_new: new_field, dup_field,	form_field_new(3X)
field_opts_on, field_opts_off,/	form_field_opts: set_field_opts,	form_field_opts(3X)
connect/ form_field: set_form_fields,	form_fields, field_count, move_field:	form_field(3X)
free_fieldtype, set_fieldtype_arg,/	form_fieldtype: new_fieldtype,	form_fieldtype(3X)
field_userptr: associate application/	form_field_userptr: set_field_userptr,	form_field_userptr(3X)
field_type, field_arg: forms field data/	form_field_validation: set_field_type,	form_field_validation(3X)

set_form_term, form_term,/ form_hook: set_form_init, and destroy forms forms pagination	form_hook: set_form_init, form_init, form_hook(3X) form_init, set_form_term, form_term,/ form_new: new_form, free_form: create form_new(3X) form_new_page: set_new_page, new_page: form_new_page(3X)
form_opts_off, form_opts: forms option/ /form_opts_on, form_opts_off, /form_opts: set_form_opts, form_opts_on, forms option/ /form_opts: set_form_opts, set_current_field, current_field,/ form_page: set_form_page, form_page: set_form_page, write or erase forms from associated/ /current_field, field_index: set /field_status, set_max_field: set and get /field_info, dynamic_field_info: get /set_field_type, field_type, field_arg: behind /data_ahead, data_behind: tell if /field_opts_off, field_opts: free_field,: create and destroy /set_fieldtype_choice, link_fieldtype: move_field: connect fields to format the general display attributes of format the general appearance of associate application data with routines for invocation by new_form, free_form: create and destroy associate application data with /post_form, unpost_form: write or erase form_opts_on, form_opts_off, form_opts: forms: character based	form_opts: set_form_opts, form_opts_on, form_opts(3X) form_opts: forms option routines form_opts(3X) form_opts_off, form_opts: forms option/ form_opts_on, form_opts_off, form_opts: form_page: set_form_page, form_page, form_page(3X) form_page, set_current_field,/ form_post: post_form, unpost_form: form_post(3X) forms: character based forms package forms(3X) forms current page and field form_page(3X) forms field attributes form_field_buffer(3X) forms field characteristics form_field_info(3X) forms field data type validation form_field_validation(3X) forms field has off-screen data ahead or forms field option routines form_data(3X) forms field option routines form_field_opts(3X) forms fields /dup_field, link_field, form_field_new(3X) forms fieldtype routines form_fieldtype(3X) forms /form_fields, field_count, form_field(3X) forms /set_field_pad, field_pad: form_field_attributes(3X) forms /set_field_just, field_just: form_field_just(3X) forms /set_field_userptr, field_userptr: forms /assign application-specific form_field_userptr(3X) forms /form_hook: form_hook(3X) forms /form_new: form_new(3X) forms /set_form_userptr, form_userptr: form_userptr(3X) forms from associated subwindows form_post(3X) forms option routines /set_form_opts, form_opts(3X) forms package forms(3X) forms pagination form_new_page(3X) forms subsystem form_driver(3X) forms used with the LP print service lpforms(1M) forms window and subwindow association/ forms window cursor form_win(3X) form_sub, scale_form: forms window and/ form_term, set_field_init, field_init,/ form_userptr: set_form_userptr, form_cursor(3X) form_userptr: associate application data form_win(3X) form_win: set_form_win, form_win, form_hook(3X) form_win, set_form_sub, form_sub, form_userptr(3X) form_userptr: set_form_userptr, form_userptr(3X) form_userptr: associate application data form_win(3X) form_win: set_form_win, form_win, form_win(3X) form_win, set_form_sub, form_sub, form_win(3X) Fortran and Pascal sources /xref: xref(1) FORTRAN dialect ratfor(1) forward or backward one screenful at a fpathconf: get configurable pathname pg(1) fpathconf: get configurable pathname pathconf(2) fpclass, unordered: determine type of/ fpgetmask, fpsetmask, fpgetsticky, isnan(3C) fpgetmask, fpsetmask, fpgetsticky, fpgetround(3C) fpgetmask, fpsetmask, fpgetsticky: / /fpsetround, fpgetmask, fpsetmask, fpgetround(3C) /fpsetround, fpgetmask, fpsetmask, fpgetround(3C) /printf, printf(3S) /printf, printf(3W) IIEEE/ fpgetround, fpsetround, fpgetmask, fpgetsticky, fpsetsticky: / fpgetround, /fpgetmask, fpsetmask, fpgetsticky, fpgetround(3C) stream /putc, putchar, puts, stream putwc, putwchar, /putws, state to that contained in a signal
t_free: df: report number of mallinfo: memory allocator malloc, valloc,: memory allocator malloc, /new_field, dup_field, link_field, /form_fieldtype: new_fieldtype, field_new: new_form, items /menu_item_new: new_item, menu_new: new_menu, checked in under RCS rcsfreeze:	free a library structure t_free(3N) free disk blocks and inodes df(1M) free, realloc, calloc, mallopt, malloc(3X) free, realloc, calloc, memalign, malloc(3C) free_field,: create and destroy forms/ free_fieldtype, set_fieldtype_arg,/ free_form: create and destroy forms form_field_new(3X) free_item: create and destroy menus form_fieldtype(3X) free_item: create and destroy menus form_new(3X) free_menu: create and destroy menus menu_item_new(3X) freeze a configuration of sources menu_new(3X) rscsfreeze(1)

	fopen, freopen, fdopen: open a stream	fopen(3S)
nextafter, scalb: manipulate parts of/	frexp, ldexp, logb, modf, modff,	frexp(3C)
	fs: file system format	fs(4)
	/scanf, fscanf, sscanf: convert formatted input	scanf(3S)
	/scanf, fscanf, sscanf: convert formatted input	scanf(3W)
list of file systems processed by	fsck and ncheck /checklist:	checklist(4)
and repair them	fsck: check file systems for consistency	fsck(1M)
	fsdb: file system debugger	fsdb(1M)
pointer in a stream	fseek, rewind, ftell: reposition a file	fseek(3S)
pointer in a stream	fsetpos, fgetpos: reposition a file	fsetpos(3C)
files	fspec: format specification in text	fspec(4)
	fsplit: split f77 or ratfor files	fsplit(1)
systems	fstab: static information about file	fstab(4)
	fstat: get file status	fstat(2)
file system	fstats: get information about a mounted	fstats(2)
file system	fstatvfs: return information about a	fstatvfs(2)
state with that on disk	fsync: synchronize a file's in-core	fsync(2)
stream	ftell: reposition a file pointer in a	fseek(3S)
	ftime: get date and time	ftime(3C)
communication package	ftok: standard interprocess	stdipc(3C)
length	ftruncate: set a file to a specified	ftruncate(3C)
	ftw, nftw: walk a file tree	ftw(3C)
egrep: search a file for a pattern using	full regular expressions	egrep(1)
shutdown: shut down part of a	full-duplex connection	shutdown(2)
function	function and complementary error	erf(3M)
error function and complementary error	function /erf, erfc:	erf(3M)
gamma, lgamma: log gamma	function	gamma(3M)
hypot: Euclidean distance	function	hypot(3M)
number entries of a common object file	function /ldlitem: manipulate line	ldread(3X)
matherr: error-handling	function	matherr(3M)
prof: profile within a	function	prof(5)
math: math	functions and constants	math(5)
bessel: j0, j1, jn, y0, y1, yn: Bessel	functions	bessel(3M)
crypt: password and file encryption	functions	crypt(3X)
dg_devctl: perform device-control	functions	dg_devctl(2)
dg_seek, dg_block_seek: extended seek	functions	dg_seek(3C)
perform system configuration and control	functions /dg_sysctl:	dg_sysctl(2)
logarithm, power, square root	functions /sqrt, sqrtf: exponential,	exp(3M)
ceiling, remainder, absolute value	functions /rint, remainder: floor,	floor(3M)
intro: introduction to network library	functions	intro(3N)
mbstowcs, wcstombs: multibyte string	functions /mbstring:	mbstring(3C)
tanhf, asinh, acosh, atanh: hyperbolic	functions /sinhf, cosh, coshf, tanh,	sinh(3M)
atanf, atan2, atan2f: trigonometric	functions /asinf, acos, acosf, atan,	trig(3M)
or file structure	fuser: identify processes using a file	fuser(1M)
stkprotect: set access for	future stack extensions	stkprotect(2)
fread,	fwrite: binary input/output	fread(3S)
accounting records	fwtmp, wtmpfix: manipulate connect	fwtmp(1M)
gamma, lgamma: log	gamma function	gamma(3M)
	gamma, lgamma: log gamma function	gamma(3M)
min,/ /mp: madd, msub, mult, mdiv, pow,	gcc: GNU C language compiler	gcc(1)
string /ecvt, fcvt,	gcd, invert, rpow, msqrt, mcmp, move,	mp(3X)
catalogue	gcvt: convert floating-point number to	ecvt(3C)
/set_field_just, field_just: format the	gencat: generate a formatted message	gencat(1)
/set_field_pad, field_pad: format the	general appearance of forms	form_field_just(3X)
termio:	general display attributes of forms	form_field_attributes(3X)
tcgetpgrp, tcsetpgrp, tcgetsid:	general terminal interface	termio(7)
termiox: extended	general terminal interface /cfsetospeed,	termios(3C)
att_kbd:	general terminal interface	termiox(7)
cflow:	generalized string translation module	att_kbd(7)
/gencat:	generate a C flow graph	cflow(1)
pkgproto:	/gencat: generate a formatted message catalogue	gencat(1)
/abort:	generate a prototype file	pkgproto(1)
cxref:	generate an abnormal termination signal	abort(3C)
conversion tables	generate C program cross-reference	cxref(1)
conversion tables	generate character classification and	chrtbl(1M)
Fortran and Pascal sources	generate character classification and	wchrtbl(1M)
/diskusg:	generate cross reference table from C,	xref(1)
crypt, setkey, encrypt:	generate disk accounting data by user id	diskusg(1M)
makekey:	generate encryption	crypt(3C)
ctermid:	generate encryption key	makekey(1)
	generate file name for terminal	ctermid(3S)

	ncheck:	generate names from i-numbers	ncheck(1M)
	tasks lex:	generate programs for simple lexical	lex(1)
random, srand, initstate, setstate:		generate random numbers better, or/	random(3C)
jrand48, srand48, seed48, lcong48:		generate uniformly distributed/ /mrand48,	drand48(3C)
	siginfo: signal	generation information	siginfo(5)
	/dgen: second	generation integrated Ethernet interface	dgen(7)
rand, srand: simple random-number		generator	rand(3C)
random numbers better, or change the		generator /initstate, setstate: generate	random(3C)
drivers and modules	eucioctl:	generic interface to EUC handling TTY	eucioctl(5)
/netdir_perror, netdir_sperror:		generic transport name-to-address/	netdir(3N)
a curses/ /inch, winch, mvinch, mvwinch:		get a character and its attributes from	curs_inch(3X)
	getmsg, getpmsg:	get a message from a stream	getmsg(2)
	semget:	get a set of semaphores	semget(2)
	gets, fgets:	get a string from a stream	gets(3S)
/mvinchnstr, mvwinchnstr, mvwinchnstr:		get a string of characters (and/	curs_inchstr(3X)
/mvinstr, mvinstr, mvwinstr, mvwinstr:		get a string of characters from a curses/	curs_instr(3X)
/mvinwchnstr, mvwinwchnstr, mvwinwchnstr:		get a string of wchar_t characters from/	curs_inwchstr(3X)
a/ /mvinnwstr, mvwinwstr, mvwinwstr:		get a string of wchar_t characters from	curs_inwstr(3X)
/inwch, winwch, mvwinwch, mvwinwch:		get a wchar_t character from a curses/	curs_inwch(3X)
	getws, fgetws:	get a wchar_t string from a stream	getws(3W)
	getcontext, setcontext:	get and set current user context	getcontext(2)
	/sysinfo:	get and set system information strings	sysinfo(2)
associated with effective UID	cuserid:	get character login name or user name	cuserid(3S)
	getc, getchar, fgetc, getw:	get character or word from a stream	getc(3S)
/mvgetnstr, mvwgetstr, mvwgetnstr:		get character strings from curses/	curs_getstr(3X)
		get: check out a version of an SCCS file	get(1)
	listener nlsgetcall:	get client's data passed via the	nlsgetcall(3N)
	pathconf, fpathconf:	get configurable pathname values	pathconf(2)
	sysconf:	get configurable system values	sysconf(2)
top_row, item_index: set and		get current menu items /set_top_row,	menu_item_current(3X)
	/getwd:	get current working directory pathname	getwd(3C)
/getyx, getparyx, getbegyx, getmaxyx:		get curses cursor and window coordinates	curs_getyx(3X)
	ftime:	get date and time	ftime(3C)
	/gettimeofday:	get date and time	gettimeofday(2)
	dlerror:	get diagnostic information	dlerror(3X)
filesystem-independent format	getdents:	get directory entries in a	getdents(2)
	nlist:	get entries from name list	nlist(3C)
	strerror:	get error message string	strerror(3C)
to NETPATH component	/getnetpath:	get /etc/netconfig entry corresponding	getnetpath(3N)
	eucset: set or	get EUC code set widths	eucset(1)
/endexportent, getexportopt:		get exported file system information	exportent(3C)
/extended_strerror:		get extended error message string	extended_strerror(3C)
	dg_fstat:	get extended file status information	dg_fstat(2)
	dg_stat:	get extended file status information	dg_stat(2)
	umask: set and	get file creation mask	umask(2)
	dg_mstat:	get file status	dg_mstat(2)
	fstat:	get file status	fstat(2)
	lstat:	get file status	lstat(2)
	stat:	get file status	stat(2)
/addmntent, endmntent, hasmntopt:		get file system descriptor file entry	getmntent(3C)
	ustat:	get file system device statistics	ustat(2)
identified by process key	/dg_file_info:	get file usage information for process	dg_file_info(2)
/getfstype, setfsent, endfsent:		get filesystem descriptor file entry	getfsent(3C)
/field_status, set_max_field: set and		get forms field attributes	form_field_buffer(3X)
/field_info, dynamic_field_info:		get forms field characteristics	form_field_info(3X)
getgrnam, setgrent, endgrent, fgetgrent:		get group file entry /getgrgid,	getgrent(3C)
	system fstatfs:	get information about a mounted file	fstatfs(2)
	system statfs:	get information about a mounted file	statfs(2)
	/dg_ipc_info:	get information about current IPCs state	dg_ipc_info(2)
	vtimes:	get information about resource usage	vtimes(3C)
	utilization getrusage:	get information about resource	getrusage(2)
currently active/	/dg_process_info:	get information about the system's	dg_process_info(2)
sets	getwidth:	get information of supplementary code	getwidth(3W)
	getlogin:	get login name	getlogin(3C)
	logname:	get login name	logname(1)
/set_menu_format, menu_format: set and		get maximum numbers of rows and columns/	menu_format(3X)
/item_name, item_description:		get menu item name and description	menu_item_name(3X)
set_item_value, item_value: set and		get menu item values /menu_item_value:	menu_item_value(3X)
/set_menu_pattern, menu_pattern: set and		get menu pattern match buffer	menu_pattern(3X)
	msgget:	get message queue identifier	msgget(2)
	signal /dg_strsignal:	get message string describing the given	dg_strsignal(3C)

getpw:	get name from UID	getpw(3C)
getpeername:	get name of connected peer	getpeername(2)
/getdomainname:	get name of current domain	getdomainname(2)
gethostname:	get name of current host	gethostname(2)
uname, nuname:	get name of current UNIX system	uname(2)
device ptsname:	get name of the slave pseudo-terminal	ptsname(3C)
/nlsprovider:	get name of transport provider	nlsprovider(3N)
/getnetconfig:	get network configuration database entry	getnetconfig(3N)
getnetbyname, setnetent, endnetent:	get network entry /getnetbyaddr,	getnetent(3N)
setnetgrent, endnetgrent, innetgr:	get network group entry /getnetgrent,	getnetgrent(3N)
gethostbyname, sethostent, endhostent:	get network host entry /gethostbyaddr,	gethostent(3N)
localeconv:	get numeric formatting information	localeconv(3C)
unset: undo a previous	get of an SCCS file	unset(1)
/getopt:	get option letter from argument vector	getopt(3C)
getsockopt:	get options on a socket	getsockopt(2)
/wgetch, mvwgetch, mvwgetch, ungetch:	get (or push back) characters from/	curl_getch(3X)
/wgetwch, mvwgetwch, mvwgetwch, ungetwch:	get (or push back) wchar_t characters/	curl_getwch(3X)
destroy a message queue /msgctl:	get or set message queue attributes or	msgctl(2)
ulimit:	get or set process limits	ulimit(2)
list IDs getgroups, setgroups:	get or set supplementary group access	getgroups(2)
panels/ /panel_window, replace_panel:	get or set the current window of a	panel_window(3X)
getitimer, setitimer:	get or set value of interval timer	getitimer(2)
getppid:	get parent process-id	getppid(2)
directory getcwd:	get pathname of current working	getcwd(3C)
times:	get process and child process times	times(2)
getpgrp2:	get process group	getpgrp2(2)
getpgrp:	get process group ID	getpgrp(2)
/getpid, getpgrp, getppid, getpgid:	get process, process group, and parent/	getpid(2)
getpriority:	get process scheduling priority	getpriority(2)
setprotoent, endprotoent:	get protocol entry /getprotobyname,	getprotoent(3N)
information t_getinfo:	get protocol-specific service	t_getinfo(3N)
rtime:	get remote time	rtime(3N)
/dg_getrootkey:	get root's secret key	dg_getrootkey(2)
getrpcbynumber, setrpcent, endrpcent:	get RPC entry /getrpcent, getrpcbyname,	getrpcent(3N)
getrpcport:	get RPC port number	getrpcport(3R)
/elf_getdata, elf_newdata, elf_rawdata:	get section data	elf_getdata(3E)
elf_ndxscn, elf_newscn, elf_nextscn:	get section information /elf_getscn,	elf_getscn(3E)
getservbyname, setservent, endservent:	get service entry /getservbyport,	getservent(3N)
getsid:	get session ID	getsid(2)
shmget:	get shared memory segment	shmget(2)
sigaltstack: set or	get signal alternate stack context	sigaltstack(2)
sigstack: set and/or	get signal stack context	sigstack(2)
getsockname:	get socket name	getsockname(2)
dg_sys_info:	get system information	dg_sys_info(2)
time:	get system time	time(2)
object dlsym:	get the address of a symbol in shared	dlsym(3X)
/elf_getbase:	get the base offset for an object file	elf_getbase(3E)
t_getstate:	get the current state	t_getstate(3N)
getegid:	get the effective-group-id	getegid(2)
geteuid:	get the effective-user-id	geteuid(2)
tty:	get the name of the terminal	tty(1)
getgid:	get the real-group-id	getgid(2)
getuid:	get the real-user-id	getuid(2)
getpagesize:	get the system page size	getpagesize(2)
gethostid:	get unique identifier of current host	gethostid(2)
getwc, getwchar, fgetwc:	get wchar_t character from a stream	getwc(3W)
/mvgetnwstr, mvwgetwstr, mvwgetnwstr:	get wchar_t character strings from/	curl_getwstr(3X)
and/ curs_getyx: getyx, getparyx,	getbegyx, getmaxyx: get curses cursor	curl_getyx(3X)
character or word from a stream	getc, getchar, fgetc, getw: get	getc(3S)
ungetch: get (or push back)/	getc, wgetch, mvwgetch, mvwgetch,	curl_getch(3X)
word from a stream /getc,	getchar, fgetc, getw: get character or	getc(3S)
current user context	getcontext, setcontext: get and set	getcontext(2)
directory	getcwd: get pathname of current working	getcwd(3C)
format date and time	getdate, getdate_err: convert user	getdate(3C)
and time getdate,	getdate_err: convert user format date	getdate(3C)
filesystem-independent format	getdents: get directory entries in a	getdents(2)
	getdev: lists devices based on criteria	getdev(1M)
contain devices that match criteria	getdgrp: lists device groups which	getdgrp(1M)
domain	getdomainname: get name of current	getdomainname(2)
files the current process can have	getdtablesize: return the number of open	getdtablesize(2)
	getegid: get the effective-group-id	getegid(2)

name	getenv: return value for environment	getenv(3C)
	geteuid: get the effective-user-id	geteuid(2)
addexportent, remexportent, / exportent, information /remexportent, endexportent, export entry containing filename	getexportent, setexportent,	exportent(3C)
	getexportopt: get exported file system	exportent(3C)
getfstype, setfsent, endfsent: get/ endfsent: get/ getfsent, getfsspec, setfsent, endfsent: get/ getfsent, getfsent, getfsspec, getfsfile,	getfh: return the file handle of the	getfh(2)
	getfsent, getfsspec, getfsfile,	getfsent(3C)
	getfsfile, getfstype, setfsent,	getfsent(3C)
	getfsspec, getfsfile, getfstype,	getfsent(3C)
	getfstype, setfsent, endfsent: get/	getfsent(3C)
	getgid: get the real-group-id	getgid(2)
endgrent, fgetgrent: get group file/ fgetgrent: get group file/ /getgrent, get group file/ /getgrent, getgrgid, supplementary group access list IDs	getgrent, getgrgid, getgrnam, setgrent,	getgrent(3C)
sethostent, endhostent: get/ gethostent, get network/ /gethostent, gethostbyaddr, gethostbyname, sethostent, endhostent:/ current host	getgrgid, getgrnam, setgrent, endgrent,	getgrent(3C)
	getgrnam, setgrent, endgrent, fgetgrent:	getgrent(3C)
	getgroups, setgroups: get or set	getgroups(2)
	gethostbyaddr, gethostbyname,	gethostent(3N)
	gethostbyname, sethostent, endhostent:	gethostent(3N)
	gethostent, gethostbyaddr,	gethostent(3N)
	gethostid: get unique identifier of	gethostid(2)
	gethostname: get name of current host	gethostname(2)
	getitimer, setitimer: get or set value	getitimer(2)
	getlogin: get login name	getlogin(3C)
	getmaxyx: get curses cursor and window/	curls_getyx(3X)
	getmntent, setmntent, addmntent,	getmntent(3C)
	getmsg, getpmsg: get a message from a	getmsg(2)
	get_myaddress, getnetname, netname2host,/	rpc(3N)
endnetent: get network entry /getnetent, network entry /getnetent, getnetbyaddr, database entry	getnetbyaddr, getnetbyname, setnetent,	getnetent(3N)
setnetent, endnetent: get network entry	getnetbyname, setnetent, endnetent: get	getnetent(3N)
inetnetr: get network group entry	getnetconfig: get network configuration	getnetconfig(3N)
/key_setsecret, get_myaddress,	getnetent, getnetbyaddr, getnetbyname,	getnetent(3N)
corresponding to NETPATH component	getnetgrent, setnetgrent, endnetgrent,	getnetgrent(3N)
mvgetnstr,/ /curls_getstr: getstr, mvgetwstr,/ /curls_getwstr: getwstr, vector	getnetname, netname2host, netname2user,/	rpc(3N)
	getnetpath: get /etc/netconfig entry	getnetpath(3N)
	getnstr, wgetstr, wgetnstr, mvgetstr,	curls_getstr(3X)
	getnwstr, wgetwstr, wgetnwstr,	curls_getwstr(3X)
	getopt: get option letter from argument	getopt(3C)
	getopt: parse command options	getopt(1)
	getoptcv: parse command options	getopts(1)
	getopts, getoptcv: parse command	getopts(1)
	getpagesize: get the system page size	getpagesize(2)
cursor and window/ /curls_getyx: getyx, parent/ /getpid, getppid, process group, and parent/ /getpid, process, process group, and parent/ getmsg, group, and parent/ /getpid, getppid, priority /getprotoent, getprotobynumber, setprotoent, endprotoent:/ /getprotoent, getprotobyname, setprotoent,/	getparyx, getbegyx, getmaxyx: get curses	curls_getyx(3X)
	getpass: read a password	getpass(3C)
	getpeername: get name of connected peer	getpeername(2)
	getpgid: get process, process group, and	getpid(2)
	getppid: get process group ID	getppid(2)
	getppid, getppid, getpgid: get process,	getpid(2)
	getppid2: get process group	getppid2(2)
	getpid, getppid, getppid, getpgid: get	getpid(2)
	getpmsg: get a message from a stream	getpmsg(2)
	getppid: get parent process-id	getppid(2)
	getppid, getpgid: get process, process	getpid(2)
	getpriority: get process scheduling	getpriority(2)
	getprotobyname, setprotoent,/	getprotoent(3N)
	getprotobyname, getprotobyname,	getprotoent(3N)
	getprotoent, getprotobyname,	getprotoent(3N)
	getpsr: return the current contents of	getpsr(2)
	getpw: get name from UID	getpw(3C)
	getpwent, getpwuid, getpwnam, setpwent,	getpwent(3C)
	getpwnam, setpwent, endpwent, setpwfile,	getpwent(3C)
	getpwfile, fgetpwent:/ /getpwent, system resource consumption	getpwent(3C)
endrpcent: get RPC entry /getrpcent, get RPC entry getrpcent, getrpcbyname, setrpcent, endrpcent: get RPC entry	getrlimit, setrlimit: control maximum	getrlimit(2)
	getrpcbyname, getrpcbynumber, setrpcent,	getrpcent(3N)
	getrpcbynumber, setrpcent, endrpcent:	getrpcent(3N)
	getrpcent, getrpcbyname, getrpcbynumber,	getrpcent(3N)
	getrpcport: get RPC port number	getrpcport(3R)
	getrusage: get information about	getrusage(2)
	gets, fgets: get a string from a stream	gets(3S)
	getservbyname, setservent, endservent:	getservent(3N)
get service/ /getservent, getservbyport, setservent, endservent: get/ getservent, getservbyname, setservent, endservent:/	getservbyport, getservbyname,	getservent(3N)
	getservent, getservbyport,	getservent(3N)
	getsid: get session ID	getsid(2)
	getsockname: get socket name	getsockname(2)
	getsockname: get socket name	getsockname(2)

fgetspent, lckpwwdf, ulckpwwdf:/	getsockopt: get options on a socket	getsockopt(2)
lckpwwdf, ulckpwwdf: manipulate/ /getsent,	getspent, getsnam, setspent, endspent,	getspent(3C)
mvgetstr, mvgetnstr,/ /curs_getstr:	getsnam, setspent, endspent, fgetspent,	getspent(3C)
string	getstr, getnstr, wgetstr, wgetnstr,	curs_getstr(3X)
/reset_shell_mode, resetty, savetty,	getsubopt: parse suboptions from a	getsubopt(3C)
	getsyx, setsyx, ripoffline, curs_set,/	curs_kernel(3X)
	gettimeofday: get date and time	gettimeofday(2)
mkmsgs: create message files for use by	gettxt	mkmsgs(1)
message data base	gettxt: retrieve a text string from a	gettxt(1)
	gettxt: retrieve a text string	gettxt(3C)
	getty: set terminal type, modes, speed,	getty(1M)
	getuid: get the real-user-id	getuid(2)
pututline, setutent, endutent,/	getut: getutent, getutid, getutline,	getut(3C)
setutent, endutent, utmpname:/ /getut:	getutent, getutid, getutline, pututline,	getut(3C)
endutent, utmpname:/ /getut: getutent,	getutid, getutline, pututline, setutent,	getut(3C)
endutent,/ getut: getutent, getutid,	getutline, pututline, setutent,	getut(3C)
stream getc, getchar, fgetc,	getw: get character or word from a	getc(3S)
character from a stream	getwc, getwchar, fgetwc: get wchar_t	getwc(3W)
ungetwch: get (or push/ /curs_getwch:	getwch, wgetwch, mvgetwch, mvwgetwch,	curs_getwch(3X)
from a stream /getwc,	getwchar, fgetwc: get wchar_t character	getwc(3W)
pathname	getwd: get current working directory	getwd(3C)
supplementary code sets	getwidth: get information of	getwidth(3W)
keyname, filter, use_env, putwin,	getwin delay_output, flushinp:/ /unctrl,	curs_util(3X)
a stream	getws, fgetws: get a wchar_t string from	ctime(3C)
mvgetwstr, mvgetnwstr,/ /curs_getwstr:	getwstr, getnwstr, wgetwstr, wgetnwstr,	curs_getwstr(3X)
curses cursor and window/ /curs_getyx:	getyx, getparyx, getbegyx, getmaxyx: get	curs_getyx(3X)
	head: give the first few lines	head(1)
get message string describing the	given signal /dg_strsignal:	dg_strsignal(3C)
	global pattern matching	gmatch(3G)
gmatch: shell	glossary: definitions of common terms	glossary(1)
and symbols	gmatch: shell global pattern matching	gmatch(3G)
	gtime, asctime, tzset: convert date and	ctime(3C)
time to string /ctime, localtime,	GNU C /default-gcc:	default-gcc(1)
set or query default version of	GNU C language compiler	gcc(1)
gcc:	goto	setjmp(3C)
setjmp, longjmp: non-local	goto with signal state	sigsetjmp(3C)
sigsetjmp, siglongjmp: a non-local	grace period /reset remote	dg_lock_reset(2)
file lock database, start lock reclaim	grant access to the slave	grantpt(3C)
pseudo-terminal device grantpt:	grantpt: grant access to the slave	grantpt(3C)
pseudo-terminal device	graph	cflow(1)
cflow: generate a C flow	graphics files /postplot:	postplot(1)
PostScript translator for plot(4)	graphics processor	grfx(7)
grfx: AViiON series workstation	grep: search a file for a pattern	grep(1)
	grfx: AViiON series workstation graphics	grfx(7)
	gridman: menu interface for maintaining	gridman(1M)
processor	group access list IDs /getgroups,	getgroups(2)
a High Availability Disk Array/	group access list	initgroups(3C)
setgroups: get or set supplementary	group, and parent process IDs /getpgrp,	getpid(2)
initgroups: initialize the supplementary	group database /admgroup:	admgroup(1M)
getpgid, getpgid: get process, process	group definition from the system	groupdel(1M)
manage group information in the	group definition on the system	groupadd(1M)
groupdel: delete a	group definition on the system	groupmod(1M)
groupadd: add (create) a new	group entry /getnetgrent, setnetgrent,	getnetgrent(3N)
groupmod: modify a	group file entry /getgrgid, getgrnam,	getgrent(3C)
endnetgrent, inneter: get network	group file	group(4)
setgrent, endgrent, fgetgrent: get	group file	pwck(1M)
group:	group	getpgrp(2)
pwck, grpck: check password or	group: group file	group(4)
getpgrp2: get process	group id /ckgid, errgid, helpgid,	ckgid(1)
	group ID for job control	setpgid(2)
	group ID	getpgrp(2)
valgid: prompt for and validate a	group id of a file	chown(2)
setpgid: set process	group id of a file	fchown(2)
getpgrp: get process	group id of the current process	setegid(2)
chown, lchown: change user id and	group ID	setsid(2)
fchown: change user id and	group id /tcsetpgrp:	tcsetpgrp(3C)
setegid: set the effective	group information in the group database	admgroup(1M)
setsid: create session and set process	group /killpg:	killpg(2)
set terminal foreground process	group	listdgrp(1M)
/admgroup: manage	group memberships	groups(1)
send signal to a process or a process	group name and ID	id(1)
listdgrp: lists members of a device		
groups: show		
id: print the user name and ID, and		

dispgid: display a list of all valid
 newgrp: log in to a new
 send a signal to a process or a
 type hosts, networks, passwd, protocols,
 chgrp: change the
 putdgrp: edit device
 definition on the system
 the system
 the system
 make: maintain, update, and regenerate
 criteria /getdgrp: lists device
 pwck,
 ssignal,
 Disk Array adapter subsystem
 /cbreak, nocbreak, echo, noecho,
 processor(s) /reboot: reboot
 /misalign:
 filename getfh: return the file
 berk_regex, regex, re_comp, re_exec:
 stdarg:
 varargs:
 curses: CRT screen
 isprint, isgraph, isascii: character
 elf_errmsg, elf_errno: error
 mblen, wctomb: multibyte character
 eucioctl: generic interface to EUC
 vhangup: virtually
 nohup: run a command immune to
 csync: synchronize
 /start_color, init_pair, init_color,
 hsearch, hcreate, hdestroy: manage
 elf_hash: compute
 spell, hashmake, spellin,
 spelling errors spell
 /curs_termattr: baudrate, erasechar,
 termname:/ /baudrate, erasechar, has_ic,
 file/ /setmntent, addmntent, endmntent,
 hken:
 tables hsearch,
 hsearch, hcreate,
 /elf_getarhdr: retrieve archive member
 retrieve class-dependent object file
 retrieve class-dependent section
 constants /limits:
 filehdr: file
 ldfhead: read the file
 /read an indexed/named section
 file ldahread: read the archive
 ldohseek: seek to the optional file
 retrieve class-dependent program
 /dg_lock_kill: remove locks
 helpadm: make changes to the
 help:
 facility database
 validate a date ckdate, errdate,
 a group id /ckgid, errgid,
 ishex: determine if a character is
 manipulation/ panel_show: show_panel,
 hier: DG/UX file system
 subsystem hada: AViiON family
 /menu interface for maintaining a
 hfm:
 nice: run a command at a
 /strnsave: allocate area large enough to
 group names
 group
 group of processes /sigsend, sigsendset:
 group or services information /bcs_cat:
 group ownership of a file
 group table
 groupadd: add (create) a new group
 groupdel: delete a group definition from
 groupmod: modify a group definition on
 groups of programs
 groups: show group memberships
 groups which contain devices that match
 grpck: check password or group file
 gsignal: software signals
 hada: AViiON family High Availability
 halfdelay, intrflush, keypad, meta,/
 halt: stop the system processor
 halts and optionally reboots the system
 handle misaligned memory access faults
 handle of the export entry containing
 handle regular expressions
 handle variable argument list
 handle variable argument list
 handling and optimization package
 handling /ispace, isctrl, ispunct,
 handling
 handling /mbchar: mbtowc,
 handling TTY drivers and modules
 hang up the current control terminal
 hangups and quits
 hardware caches for execute access
 has_colors, can_change_color,/
 hash search tables
 hash value
 hashcheck: find spelling errors
 hashmake, spellin, hashcheck: find
 has_ic, has_il, killchar, longname,/
 has_il, killchar, longname, termattr,
 hasmntopt: get file system descriptor
 Hawk Ethernet interface
 hcreate, hdestroy: manage hash search
 hdestroy: manage hash search tables
 head: give the first few lines
 header
 header /elf32_getehdr, elf32_newehdr:
 header /elf_getshdr: elf32_getshdr:
 header file for implementation-specific
 header for common object files
 header of a common object file
 header of a common object file
 header of a member of a COFF archive
 header of an object file
 header table /elf32_newphdr:
 held by remote lock clients
 help facility database
 help facility
 help: help facility
 helpadm: make changes to the help
 helpdate, valdate: prompt for and
 helpgid, valgid: prompt for and validate
 hexadecimal
 hfm: high sierra file manager
 hide_panel, panel_hidden: panels deck
 hier: DG/UX file system hierarchy
 hierarchy
 High Availability Disk Array adapter
 High Availability Disk Array subsystem
 high sierra file manager
 higher or lower priority
 hken: Hawk Ethernet interface
 hold string and move string into it
 dispgid(1)
 newgrp(1)
 sigsend(2)
 bcs_cat(1M)
 chgrp(1)
 putdgrp(1M)
 groupadd(1M)
 groupdel(1M)
 groupmod(1M)
 make(1)
 groups(1)
 getdgrp(1M)
 pwck(1M)
 ssignal(3C)
 hada(7)
 curs_inopts(3X)
 halt(1M)
 reboot(2)
 misalign(5)
 getfh(2)
 berk_regex(3C)
 stdarg(5)
 varargs(5)
 curses(3X)
 ctype(3C)
 elf_error(3E)
 mbchar(3C)
 eucioctl(5)
 vhangup(2)
 nohup(1)
 csync(2)
 curs_color(3X)
 hsearch(3C)
 elf_hash(3E)
 spell(1)
 spell(1)
 curs_termattr(3X)
 curs_termattr(3X)
 getmntent(3C)
 hken(7)
 hsearch(3C)
 hsearch(3C)
 head(1)
 elf_getarhdr(3E)
 elf_getehdr(3E)
 elf_getshdr(3E)
 limits(4)
 filehdr(4)
 ldfhead(3X)
 ldshread(3X)
 ldahread(3X)
 ldohseek(3X)
 elf_getphdr(3E)
 dg_lock_kill(2)
 helpadm(1M)
 help(1)
 help(1)
 helpadm(1M)
 ckdate(1)
 ckgid(1)
 ishex(3C)
 hfm(4)
 panel_show(3X)
 hier(5)
 hier(5)
 hada(7)
 gridman(1M)
 hfm(4)
 nice(1)
 hken(7)
 strsave(3C)

distinguish prime and non-prime days
 wline, wvline: create curses borders,
 ntohl, ntohs: convert values between
 sethostent, endhostent: get network
 get unique identifier of current
 gethostname: get name of current
 /admtcpipparams: manage the TCP/IP
 download: download
 set unique identifier of current
 sethostname: set name of current
 unix_ipc: piping communications within a
 /clnttcp_create, clntudp_create,
 admhost: manage
 /admtrustedhost: manage the trusted
 group or services/ bcs_cat: type
 search tables
 values between host and network byte/
 between host and network byte/ htonl,
 sttydefs: maintain line and
 coshf, tanh, tanhf, asinh, acosh, atanh:
 truth value/ machid: dghost, m68k, m88k,
 commands for reading and writing
 chown, lchown: change user
 fchown: change user
 id: print the user name and
 valgid: prompt for and validate a group
 ckuid: prompt for and validate a user
 generate disk accounting data by user
 setpgid: set process group
 getpgrp: get process group
 getsid: get session
 the user name and ID, and group name and
 queue, semaphore set, or shared memory
 chown, lchown: change user id and group
 fchown: change user id and group
 setegid: set the effective group
 seteuid: set the effective user
 group name and ID
 create session and set process group
 set terminal foreground process group
 /curs_ouptopts: clearok, idlok,
 /elf_getident: retrieve file
 issue: issue
 get file usage information for process
 msgget: get message queue
 gethostid: get unique
 sethostid: set unique
 systemid: display the unique system
 locate:
 ident:
 structure /fuser:
 what:
 idi_log, idi_warning: tools/ idi_tools:
 idi_warning:/ idi_tools: idi_confirm,
 for/ /idi_tools: idi_confirm, idi_echo,
 the/ /idi_confirm, idi_echo, idi_error,
 idi_error, idi_log, idi_warning: tools/
 interface/ /idi_echo, idi_error, idi_log,
 setscreg,/ /curs_ouptopts: clearok,
 or set supplementary group access list
 process group, and parent process
 with collision detection /dot3:
 /fpsetmask, fpgetsticky, fpsetsticky:
 drem:
 isalphanum: determine
 holidays: accounting information used to holidays(4)
 horizontal and vertical lines /box, curs_border(3X)
 host and network byte order /htons, byteorder(3N)
 host entry /gethostbyname, gethostent(3N)
 host /gethostid: gethostid(2)
 host gethostname(2)
 host parameters admtcpipparams(1M)
 host resident PostScript fonts download(1)
 host /sethostid: sethostid(2)
 host sethostname(2)
 host unix_ipc(6F)
 host2netname, key_decryptsession,/ rpc(3N)
 hosts database admhost(1M)
 hosts database admtrustedhost(1M)
 hosts, networks, passwd, protocols, bcs_cat(1M)
 hsearch, hcreate, hdestroy: manage hash hsearch(3C)
 htonl, htons, ntohl, ntohs: convert byteorder(3N)
 htons, ntohl, ntohs: convert values byteorder(3N)
 hunt settings for TTY ports sttydefs(1M)
 hyperbolic functions /sinh, sinhf, cosh, sinh(3M)
 hypot: Euclidean distance function hypot(3M)
 i386, pdp11, u3b, u3b5, vax: provide machid(1)
 IBM and ANSI tapes /REELexchange: reelexchange_intro(1)
 iconv: code set conversion iconv(1)
 id and group id of a file chown(2)
 id and group id of a file fchown(2)
 ID, and group name and ID id(1)
 id /ckgid, errgid, helpgid, ckgid(1)
 ID ckuid(1)
 id /diskusg: diskusg(1M)
 ID for job control setpgid(2)
 ID getpgrp(2)
 ID getsid(2)
 ID /id: print id(1)
 ID /ipcrm: remove a message ipcrm(1)
 id of a file chown(2)
 id of a file fchown(2)
 id of the current process setegid(2)
 id of the current process seteuid(2)
 id: print the user name and ID, and id(1)
 ID /setsid: setsid(2)
 id /tcsetpgrp: tcsetpgrp(3C)
 idc: interface description compiler idc(1)
 idcok immedok, leaveok, setscreg,/ curs_ouptopts(3X)
 ident: identify files ident(1)
 identification data elf_getident(3E)
 identification file issue(4)
 identified by process key /dg_file_info: dg_file_info(2)
 identifier msgget(2)
 identifier of current host gethostid(2)
 identifier of current host sethostid(2)
 identifier systemid(1M)
 identify a command using keywords locate(1)
 identify files ident(1)
 identify processes using a file or file fuser(1M)
 identify SCCS files what(1)
 idi: interface description interpreter idi(1)
 idi_confirm, idi_echo, idi_error, idi_tools(1)
 idi_echo, idi_error, idi_log, idi_tools(1)
 idi_error, idi_log, idi_warning: tools idi_tools(1)
 idi_log, idi_warning: tools for use with idi_tools(1)
 idi_tools: idi_confirm, idi_echo, idi_tools(1)
 idi_warning: tools for use with the idi_tools(1)
 idl: interface description language idl(4)
 idlok, idcok immedok, leaveok, curs_ouptopts(3X)
 IDs /getgroups, setgroups: get getgroups(2)
 IDs /getppid, getpgid: get process, getpid(2)
 IEEE 802.3 carrier sense multiple access dot3(6P)
 IEEE floating-point environment control fpgetround(3C)
 IEEE floating-point remainder drem(3M)
 if a character is alphanumeric isalphanum(3C)

ishex: determine if a character is hexadecimal	ishex(3C)
or behind /data_ahead, data_behind: tell if forms field has off-screen data ahead	form_data(3X)
/menu_item_visible: item_visible: tell if menu item is visible	menu_item_visible(3X)
set the signal action of a signal to 'ignore' /sigignore:	sigignore(2)
core: format of core image file	core(4)
scr_dump: format of curses screen image file	scr_dump(4)
crash: examine system images	crash(1M)
/curs_outopts: clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, /	curs_outopts(3X)
nohup: run a command immune to hangups and quits	nohup(1)
xstr: extract strings from C programs to implement shared strings	xstr(1)
sigfillset: fill in the set of implementation-defined signals	sigfillset(2)
limits: header file for implementation-specific constants	limits(4)
character and its attributes/ curs_inch: inch, winch, mvinch, mvwinch: get a	curs_inch(3X)
mvinchstr, /curs_inchstr: inchstr, inchnstr, winchstr, winchnstr,	curs_inchstr(3X)
mvinchstr, mvwinchnstr, /curs_inchstr: inchstr, inchnstr, winchstr, winchnstr,	curs_inchstr(3X)
mail_pipe: invoke recipient command for incoming mail	mail_pipe(1M)
vacation: automatically respond to incoming mail messages	vacation(1)
fsync: synchronize a file's in-core state with that on disk	fsync(2)
dump2: incremental file system backup	dump2(1M)
dump: incremental file system dump	dump(1M)
restore: incrementally restore a file system	restore(1M)
dirent: file system independent directory entry	dirent(4)
/tgetstr, tgoto, tputs: terminal independent operation routines	termcap(3X)
file /ldtbindex: compute index of symbol table entry of an object	ldtbindex(3X)
of a character in a string index: search for the first occurrence	index(3C)
file /ldtbread: read an indexed symbol table entry of an object	ldtbread(3X)
common/ ldshread, ldnsbread: read an indexed/named section header of a	ldshread(3X)
object/ ldsseek, ldnsseek: seek to an indexed/named section of a common	ldsseek(3X)
last: indicate last user or terminal logins	last(1)
receipt of an orderly release indication /trcvrel: acknowledge	trcvrel(3N)
trcvuderr: receive a unit data error indication	trcvuderr(3N)
/store_conditional: indivisible compare and swap	store_conditional(2)
location /fetch_and_add: indivisible fetch and add to memory	fetch_and_add(2)
inen: integrated Ethernet interface	inen(7)
inet_makeaddr, inet_lnaof, inet_netof: /inet_network, inet_ntoa, inet_makeaddr,	inet(3N)
/inet_addr, inet_network, inet_ntoa, inet_lnaof, inet_netof: Internet address/	inet(3N)
/inet_addr, inet_network, inet_ntoa, inet_makeaddr, inet_lnaof, inet_netof: /	inet(3N)
/inet_ntoa, inet_makeaddr, inet_lnaof, inet_netof: Internet address/	inet(3N)
inet_network, inet_ntoa, inet_makeaddr,	inet(3N)
inet_ntoa, inet_makeaddr, inet_lnaof,	inet(3N)
info: documentation browser	info(1)
descriptions infocmp: compare or print out TERMINFO	infocmp(1M)
fstatvfs: return information about a file system	fstatvfs(2)
statvfs: return information about a file system	statvfs(2)
/fstats: get information about a mounted file system	fstats(2)
/stats: get information about a mounted file system	stats(2)
dg_ipc_info: get information about current IPCs state	dg_ipc_info(2)
sysfs: returns information about file system types	sysfs(2)
fstab: static information about file systems	fstab(4)
/admfinfo: display information about files and directories	admfinfo(1M)
/finger: display information about local and remote users	finger(1)
rlog: print log messages and other information about RCS files	rlog(1)
vtimes: get information about resource usage	vtimes(3C)
/getrusage: get information about resource utilization	getrusage(2)
print service /lpstat: print information about the status of the LP	lpstat(1)
active processes /dg_process_info: get information about the system's currently	dg_process_info(2)
passwd, protocols, group or services information /type hosts, networks,	bcs_cat(1M)
langinfo: language information constants	langinfo(5)
dg_fstat: get extended file status information	dg_fstat(2)
dg_stat: get extended file status information	dg_stat(2)
dg_sys_info: get system information	dg_sys_info(2)
dlerror: get diagnostic information	dlerror(3X)
dumpfs: dump file system information	dumpfs(1M)
elf_newscn, elf_nextscn: get section information /elf_getscn, elf_ndxscn,	elf_getscn(3E)
getexportopt: get exported file system information /remexportent, endexportent,	exportent(3C)
copyright: copyright information file	copyright(4)
prototype: package information file	prototype(4)
reloc: relocation information for a common object file	reloc(4)
starter: information for beginning users	starter(1)
mailcnfg: initialization information for mail and rmail	mailcnfg(4M)
process/ /dg_file_info: get file usage information for process identified by	dg_file_info(2)

ttydefs: terminal line settings	information for ttymon	ttydefs(4M)
strip: strip non-executable	information from an object file	strip(1)
_t_rcvdis: retrieve	information from disconnect	_t_rcvdis(3N)
admalias: manage mail alias	information in the aliases database	admalias(1M)
admgroup: manage group	information in the group database	admgroup(1M)
admuser: manage user	information in the password database	admuser(1M)
listusers: list user login	information	listusers(1)
localeconv: get numeric formatting	information	localeconv(3C)
logins: list user and system login	information	logins(1M)
/nl_langinfo: language	information	nl_langinfo(3C)
/getwidth: get	information of supplementary code sets	getwidth(3W)
usermod: modify a user's login	information on the system	usermod(1M)
pkginfo: display software package	information	pkginfo(1)
fingerd, in.fingerd: remote user	information server	fingerd(1M)
/yperr_string, ypprot_err: Network	Information Service client interface	ypclnt(3N)
setuname: changes machine	information	setuname(1M)
siginfo: signal generation	information	siginfo(5)
sysinfo: get and set system	information strings	sysinfo(2)
/syac_routes: Change SYAC routing	information	syac_routes(1M)
disk and memory resident file system	information /sync: synchronize	sync(2)
legend: Debugging	information technology	legend(5)
_t_getinfo: get protocol-specific service	information	_t_getinfo(3N)
format and output TTY port monitor	information /ttyadm:	ttyadm(1M)
and non-prime days	information used to distinguish prime	holidays(4)
inittab: script for	init	inittab(4)
initialization	init, telinit: process control	init(1M)
curs_color: start_color, init_pair,	init_color, has_colors,/	curs_color(3X)
group access list	initgroups: initialize the supplementary	initgroups(3C)
/set_term, delscreen: curses screen	initialization and manipulation routines	curs_initscr(3X)
rmail /mailcnfg:	initialization information for mail and	mailcnfg(4M)
init, telinit: process control	initialization	init(1M)
tlabel:	initialize a tape with a volume label	tlabel(1)
database /tput:	initialize a terminal or query terminfo	tput(1)
access list	initgroups: initialize the supplementary group	initgroups(3C)
connect:	initiate a connection on a socket	connect(2)
taccess:	initiate access to labeled tape	taccess(1)
t_sndrel:	initiate an orderly release	t_sndrel(3N)
popen, pclose:	initiate pipe to/from a process	popen(3S)
curs_color: start_color,	init_pair, init_color, has_colors,/	curs_color(3X)
set_term, delscreen:/ /curs_initscr:	initscr, newterm, endwin, isendwin,	curs_initscr(3X)
numbers better, or/ random, srandom,	initstate, setstate: generate random	random(3C)
	inittab: script for init	inittab(4)
/getnetgrent, setnetgrent, endnetgrent,	innetr: get network group entry	getnetgrent(3N)
mvinnstr, mvwinstr,/ /curs_instr: instr,	instr, winstr, winnstr, mvinstr,	curs_instr(3X)
mvinnwstr,/ /curs_inwstr: inwstr,	innwstr, winwstr, winnwstr, mvinnwstr,	curs_inwstr(3X)
clri: clear	inode	clri(1M)
	inode: file node structure	inode(4)
report number of free disk blocks and	inodes /df:	df(1M)
mvwscanw, vwscanw: convert formatted	input from a curses widow /mvscanw,	curs_scanw(3X)
wtimeout, typeahead: curses terminal	input option control routines /timeout,	curs_inopts(3X)
scanf, fscanf, sscanf: convert formatted	input	scanf(3S)
scanf, fscanf, sscanf: convert formatted	input	scanf(3W)
ungetc: push character back onto	input stream	ungetc(3S)
push wchar_t character back into	input stream /ungetc:	ungetc(3W)
/vfscanf, vsscanf: convert formatted	input using varargs argument list	vscanf(3S)
fread, fwrite: binary	input/output	fread(3S)
poll:	input/output multiplexing	poll(2)
stdio: standard buffered	input/output package	stdio(3S)
feof, clearerr, fileno: stream status	inquiries /ferror,	ferror(3S)
uustat: uucp status	inquiry and job control	uustat(1)
subsystem	insc: AViiON family SCSI adapter	insc(7)
a character before the/ /curs_insch:	insch, winsch, mvinsch, mvwinsch: insert	curs_insch(3X)
/curs_deleteln: deleteln, wdeleteln,	insdelln, winsdelln, insertln,/	curs_deleteln(3X)
under/ /insch, winsch, mvinsch, mvwinsch:	insert a character before the character	curs_insch(3X)
/inswch, winswch, mvinswch, mvwinswch:	insert a wchar_t character before the/	curs_inswch(3X)
/insertln, winsertln: delete and	insert lines in a curses window	curs_deleteln(3X)
/mvinsnstr, mvwinsstr, mvwinsnstr:	insert string before character under the/	curs_insnstr(3X)
/mvinsnwstr, mvwinswstr, mvwinsnwstr:	insert wchar_t string before character/	curs_inswstr(3X)
lines/ /wdeleteln, insdelln, winsdelln,	insertln, winsertln: delete and insert	curs_deleteln(3X)
insque, remque:	insert/remove element from a queue	insque(3C)
mvinsnstr,/ /curs_instr: insstr,	insnstr, winsstr, winsnstr, mvinsstr,	curs_instr(3X)

mvinswstr,/ /curs_instr: inswstr,	insnwstr, winswstr, winsnwstr,	curs_inswstr(3X)
from a queue	insque, remque: insert/remove element	insque(3C)
mvinsstr, mvinsnstr,/ /curs_instr:	insstr, insnstr, winsstr, winsnstr,	curs_insstr(3X)
install:	install commands	install(1M)
pkgmk: produce an	install: install commands	install(1M)
installf: add a file to the software	installable package	pkgmk(1)
installman: manage system	installation database	installf(1M)
pkgchk: check accuracy of	installation	installman(1M)
installation database	installation	pkgchk(1M)
	installf: add a file to the software	installf(1M)
	installman: manage system installation	installman(1M)
mvinnstr, mvwinstr,/ /curs_instr:	instr, innstr, winstr, winnstr, mvinstr,	curs_instr(3X)
insert a wchar_t character/ /curs_inswch:	inswch, winswch, mvinswch, mvwinswch:	curs_inswch(3X)
mvinswstr, mvinsnwstr,/ /curs_instr:	inswstr, insnwstr, winswstr, winsnwstr,	curs_inswstr(3X)
abs, labs: return	integer absolute value	abs(3C)
a64l, l64a: convert between long	integer and base-64 ASCII string	a64l(3C)
m_out, sdiv, itom: multiple precision	integer arithmetic /mout, omout, fmout,	mp(3X)
ckrange: prompt for and validate an	integer	ckrange(1)
fashion sputl, sgetl: access long	integer data in a machine-independent	sputl(3X)
strtoul, atol, atoi: convert string to	integer /strtol,	strtoul(3C)
itoa: convert an	integer to an ASCII character string	itoa(3C)
display a prompt; verify and return an	integer value /ckint:	ckint(1)
l3tol, ltol3: convert between 3-byte	integers and long integers	l3tol(3C)
convert between 3-byte integers and long	integers /l3tol, ltol3:	l3tol(3C)
dgen: second generation	integrated Ethernet interface	dgen(7)
inen:	integrated Ethernet interface	inen(7)
iscd:	Integrated Synchronous Chip Driver	iscd(7)
syac: AViiON family	intelligent asynchronous controller	syac(7)
mailx:	interactive message processing system	mailx(1)
cscope:	interactively examine a C program	cscope(1)
timod: Transport	Interface cooperating STREAMS module	timod(7)
idc:	interface description compiler	idc(1)
idi:	interface description interpreter	idi(1)
/idi_warning: tools for use with the	interface description interpreter	idi_tools(1)
idl:	interface description language	idl(4)
second generation integrated Ethernet	interface /dgen:	dgen(7)
ssid: Streams Synchronous	Interface Driver	ssid(7)
dsk: block special disk	interface	dsk(7)
err: error-logging	interface	err(7)
Availability Disk Array/ gridman: menu	interface for maintaining a High	gridman(1M)
logical disks diskman: menu	interface for managing physical and	diskman(1M)
postio: serial	interface for PostScript printers	postio(1)
hken: Hawk Ethernet	interface	hken(7)
inen: integrated Ethernet	interface	inen(7)
lpprint, xlpprint: menu-driven lp	interface	lpprint(1M)
plm: pseudo lock manager device	interface	plm(7)
rdsd: character special disk	interface	rdsd(7)
module /tirdwr: Transport	Interface read/write interface STREAMS	tirdwr(7)
rmt: character special magnetic tape	interface	rmt(7)
tirdwr: Transport Interface read/write	interface STREAMS module	tirdwr(7)
menu-driven system administration	interface /sysadm, asysadm, xsysadm:	sysadm(1M)
termio: general terminal	interface	termio(7)
tcsetpgrp, tcsetgid: general terminal	interface /cfsetospeed, tcsetpgrp,	termios(3C)
termiox: extended general terminal	interface	termiox(7)
and modules eucioctl: generic	interface to EUC handling TTY drivers	eucioctl(5)
admdefault: provide an	interface to named default sets	admdefault(1M)
event tracing /log:	interface to STREAMS error logging and	log(7)
vitr: Vilya TokenRing Controller	interface	vitr(7)
Read Multiple optical device) as magtape	interface /wmt: pseudo WORM (Write Once	wmt(7)
Network Information Service client	interface /yperr_string, ypprot_err:	ypclnt(3N)
manage the TCP/IP network	interfaces database /admipinterface:	admipinterface(1M)
/tgetnum, tgetstr, tgoto, tputs: curses	interfaces (emulated) to the termcap/	curs_termcap(3X)
tgetflag, tgetnum, tgetstr: curses	interfaces to terminfo database /mvcur,	curs_terminfo(3X)
/inet_makeaddr, inet_lnaof, inet_netof:	Internet address manipulation routines	inet(3N)
/syac_ttyaddr: set tty specific	internet addresses	syac_ttyaddr(1M)
make, send, and interpret packets to	Internet domain name servers /dn_expand:	resolver(3C)
spline:	interpolate smooth curve	spline(1G)
characters asa:	interpret ASA carriage control	asa(1)
/dn_comp, dn_expand: make, send, and	interpret packets to Internet domain/	resolver(3C)
sno: SNOBOL	interpreter and compiler	sno(1)
csh: invoke a shell (command	interpreter) having a C-like syntax	csh(1)

idi: interface description	interpreter	idi(1)
for use with the interface description	interpreter /idi_log, idi_warning: tools	idi_tools(1)
pipe: create an	interprocess channel	pipe(2)
status /ipcs: report	inter-process communication facilities	ipcs(1)
stdipc: ftok: standard	interprocess communication package	stdipc(3C)
sleep: suspend execution for an	interval	sleep(1)
sleep: suspend execution for	interval	sleep(3C)
setitimer: get or set value of	interval timer /getitimer,	getitimer(2)
captainfo: convert a TERMCAP entry	into a TERMINFO entry	captainfo(1M)
bufsplit: split buffer	into fields	bufsplit(3G)
ungetwc: push wchar_t character back	into input stream	ungetwc(3W)
enough to hold string and move string	into it /strnsave: allocate area large	strsave(3C)
copylist: copy a file	into memory	copylist(3G)
plock: lock data, text, or both	into memory	plock(2)
split: split a file	into pieces	split(1)
postprint: translate text files	into PostScript	postprint(1)
/nocbreak, echo, noecho, halfdelay,	inrflush, keypad, meta, nodelay,/	cursor_opts(3X)
application programs	intro: introduction to commands and	intro(1)
application programs	intro: introduction to commands and	intro(1)
special files	intro: introduction to DG/UX System	intro(7)
	intro: introduction to file formats	intro(4)
	intro: introduction to file formats	intro(4M)
	intro: introduction to math libraries	intro(3M)
	intro: introduction to miscellany	intro(5)
functions	intro: introduction to network library	intro(3N)
libraries	intro: introduction to subroutines and	intro(3)
error numbers	intro: introduction to system calls and	intro(2)
	intro: introduction to system	intro(1M)
maintenance commands and application/	intro: introduction to system	intro(8)
maintenance procedures	intro: introduction to commands and application	intro(1)
programs /intro:	intro: introduction to commands and application	intro(1)
programs /intro:	intro: introduction to DG/UX System special	intro(7)
files intro:	intro: introduction to file formats	intro(4)
	intro: introduction to file formats	intro(4M)
	intro: introduction to math libraries	intro(3M)
	intro: introduction to miscellany	intro(5)
functions intro:	intro: introduction to network library	intro(3N)
rcsintro:	intro: introduction to RCS commands	rcsintro(1)
libraries intro:	intro: introduction to subroutines and	intro(3)
numbers /intro:	intro: introduction to system calls and error	intro(2)
commands and application/ intro:	intro: introduction to system maintenance	intro(1M)
procedures intro:	intro: introduction to system maintenance	intro(8)
valtools:	intro: introduction to validation tools	valtools(1)
ncheck: generate names from	i-numbers	ncheck(1M)
/mp: madd, msub, mult, mdiv, pow, gcd,	invert, rpow, msqrt, mcmp, move, min,/	mp(3X)
assign application-specific routines for	invocation by forms /field_term:	form_hook(3X)
/routines for automatic	invocation by menus	menu_hook(3X)
having a C-like syntax csh:	invoke a shell (command interpreter)	csh(1)
sys_local:	invoke an extended system call	sys_local(2)
mail mail_pipe:	invoke recipient command for incoming	mail_pipe(1M)
wchar_t character from a /curs_inwch:	inwch, winwch, mvinwch, mvwinwch: get a	curs_inwch(3X)
mvinwchstr,/ /curs_inwchstr: inwchstr,	inwchnstr, winwchstr, winwchnstr,	curs_inwchstr(3X)
winwchnstr, mvinwchstr,/ /curs_inwchstr:	inwchstr, inwchnstr, winwchstr,	curs_inwchstr(3X)
mvinwstr, mvinnwstr,/ /curs_inwstr:	inwstr, innwstr, winwstr, winnwstr,	curs_inwstr(3X)
select: examine file descriptors for	I/O readiness	select(2)
start a BIOD server for asynchronous	I/O requests /async_daemon:	async_daemon(2)
widec: multibyte character	I/O routines	widec(3W)
biod: start block	I/O servers	biod(1M)
streamio: STREAMS	ioctl commands	streamio(7)
	ioctl: control a device	ioctl(2)
set, or shared memory ID	ipcrm: remove a message queue, semaphore	ipcrm(1)
facilities status	ipcs: report inter-process communication	ipcs(1)
get information about current	IPCs state /dg_ipc_info:	dg_ipc_info(2)
/isxdigit, islower, isupper, isalpha,	isalnum, isspace, isctrl, ispunct,/	ctype(3C)
/isdigit, isxdigit, islower, isupper,	isalpha, isalnum, isspace, isctrl,/	ctype(3C)
alphanumeric	isalphanum: determine if a character is	isalphanum(3C)
isctrl, ispunct, isprint, isgraph,	isascii: character handling /isspace,	ctype(3C)
	isastream: test a file descriptor	isastream(3C)
ttyname,	isatty: find name of a terminal	ttyname(3C)
	iscd: Integrated Synchronous Chip Driver	iscd(7)
/isupper, isalpha, isalnum, isspace,	isctrl, ispunct, isprint, isgraph,/	ctype(3C)

isalpha, isalnum, isspace, / ctype: buffer is encrypted
 /curs_initscr: initscr, newterm, endwin,
 /iswascii, isphonogram, isideogram,
 /isspace, iscntrl, ispunct, isprint,
 hexadecimal
 /iswcntrl, iswascii, isphonogram,
 /touchline, untouchwin, wtouchln,
 isspace, / ctype: isdigit, isxdigit,
 unordered: determine type of/
 unordered: determine type of/ isnan, isnand,
 determine type of/ isnan, isnand,
 /isphonogram, isideogram, isenglish,
 /iswprint, iswgraph, iswcntrl, iswascii,
 /isalnum, isspace, iscntrl, ispunct,
 /isalpha, isalnum, isspace, iscntrl,
 /islower, isupper, isalpha, isalnum,
 /isideogram, isenglish, isnumber,
 system:
 issue:
 ctype: isdigit, isxdigit, islower,
 /iswupper, iswlower, iswdigit, iswxdigit,
 iswxdigit, iswalnum, iswspace, / wctype:
 /iswpunct, iswprint, iswgraph, iswcntrl,
 /iswspace, iswpunct, iswprint, iswgraph,
 /wctype: iswalpha, iswupper, iswlower,
 /iswalnum, iswspace, iswpunct, iswprint,
 /untouchwin, wtouchln, is_linetouched,
 iswspace, / wctype: iswalpha, iswupper,
 /iswxdigit, iswalnum, iswspace, iswpunct,
 /iswdigit, iswxdigit, iswalnum, iswspace,
 /iswlower, iswdigit, iswxdigit, iswalnum,
 iswalnum, iswspace, / wctype: iswalpha,
 /iswalpha, iswupper, iswlower, iswdigit,
 isalnum, isspace, / ctype: isdigit,
 a menu; prompt for and return a menu
 item_visible: tell if menus
 /item_name, item_description: get menus
 item_opts_off, item_opts: menus
 item_value: set and get menus
 /menu_items: set_menu_items, menu_items,
 and/ /menu_item_name: item_name,
 /current_item, set_top_row, top_row,
 menu_hook: set_item_init,
 item name and/ /menu_item_name:
 /item_opts_on, item_opts_off,
 option/ /set_item_opts, item_opts_on,
 menus/ /menu_item_opts: set_item_opts,
 item_index: set and get current menus
 free_item: create and destroy menus
 associate application data with menus
 news: print news
 item_count: connect and disconnect
 /set_item_init, item_init, set_item_term,
 /menu_item_userptr: set_item_userptr,
 /menu_item_value: set_item_value,
 visible /menu_item_visible:
 character string
 m_in, mout, omout, fmout, m_out, sdiv,
 /bessel:
 bessel: j0,
 bessel: j0, j1,
 jobs: summary of DG/UX
 setpgid: set process group ID for
 uustat: uucp status inquiry and
 queue lprm: remove
 /atq: display the
 atrm: remove
 facilities
 isdigit, isxdigit, islower, isupper, ctype(3C)
 isencrypt: determine whether a character isencrypt(3G)
 isendwin, set_term, delscreen: curses/ curs_initscr(3X)
 isenglish, isnumber, isspecial: classify/ wctype(3W)
 isgraph, isascii: character handling ctype(3C)
 ishex: determine if a character is ishex(3C)
 isideogram, isenglish, isnumber, / wctype(3W)
 is_linetouched, is_wintouched: curses/ curs_touch(3X)
 islower, isupper, isalpha, isalnum, ctype(3C)
 isnan, isnand, isnanf, finite, fpclass, isnan(3C)
 isnand, isnanf, finite, fpclass, isnan(3C)
 isnanf, finite, fpclass, unordered: isnan(3C)
 isnumber, isspecial: classify ASCII and/ wctype(3W)
 isphonogram, isideogram, isenglish, / wctype(3W)
 isprint, isgraph, isascii: character/ ctype(3C)
 ispunct, isprint, isgraph, isascii:/ ctype(3C)
 isspace, iscntrl, ispunct, isprint, / ctype(3C)
 isspecial: classify ASCII and/ wctype(3W)
 issue a shell command system(3S)
 issue identification file issue(4)
 issue: issue identification file issue(4)
 isupper, isalpha, isalnum, isspace, / ctype(3C)
 iswalnum, iswspace, iswpunct, iswprint, / wctype(3W)
 iswalpha, iswupper, iswlower, iswdigit, wctype(3W)
 iswascii, isphonogram, isideogram, / wctype(3W)
 iswcntrl, iswascii, isphonogram, / wctype(3W)
 iswdigit, iswxdigit, iswalnum, iswspace, / wctype(3W)
 iswgraph, iswcntrl, iswascii, / wctype(3W)
 is_wintouched: curses refresh control/ curs_touch(3X)
 iswlower, iswdigit, iswxdigit, iswalnum, wctype(3W)
 iswprint, iswgraph, iswcntrl, iswascii, / wctype(3W)
 iswpunct, iswprint, iswgraph, iswcntrl, / wctype(3W)
 iswspace, iswpunct, iswprint, iswgraph, / wctype(3W)
 iswupper, iswlower, iswdigit, iswxdigit, wctype(3W)
 iswxdigit, iswalnum, iswspace, iswpunct, / wctype(3W)
 isxdigit, islower, isupper, isalpha, ctype(3C)
 item /ckitem: build ckitem(1)
 item is visible /menu_item_visible: menu_item_visible(3X)
 item name and description menu_item_name(3X)
 item option routines /item_opts_on, menu_item_opts(3X)
 item values /set_item_value, menu_item_value(3X)
 item_count: connect and disconnect items/ menu_items(3X)
 item_description: get menus item name menu_item_name(3X)
 item_index: set and get current menus/ menu_item_current(3X)
 item_init, set_item_term, item_term, / menu_hook(3X)
 item_name, item_description: get menus menu_item_name(3X)
 item_opts: menus item option routines menu_item_opts(3X)
 item_opts_off, item_opts: menus item menu_item_opts(3X)
 item_opts_on, item_opts_off, item_opts: menu_item_opts(3X)
 items /set_top_row, top_row, menu_item_current(3X)
 items /menu_item_new: new_item, menu_item_new(3X)
 items /set_item_userptr, item_userptr: menu_item_userptr(3X)
 items news(1)
 items to and from menus /menu_items, menu_items(3X)
 item_term, set_menu_init, menu_init, / menu_hook(3X)
 item_userptr: associate application data/ menu_item_userptr(3X)
 item_value: set and get menus item/ menu_item_value(3X)
 item_visible: tell if menus item is menu_item_visible(3X)
 itoa: convert an integer to an ASCII itoa(3C)
 itom: multiple precision integer/ /fmin, mp(3X)
 j0, j1, jn, y0, y1, yn: Bessel functions bessel(3M)
 j1, jn, y0, y1, yn: Bessel functions bessel(3M)
 jn, y0, y1, yn: Bessel functions bessel(3M)
 job control facilities jobs(3C)
 job control setpgid(2)
 job control uustat(1)
 jobs from the line printer spooling lprm(1)
 jobs queued to run at specified times atq(1)
 jobs spooled by at or batch atrm(1)
 jobs: summary of DG/UX job control jobs(3C)
 join: relational database operator join(1)

/erand48, lrand48, nrand48, mrand48, programming language sh, keyboard kbdpipe: use the	jrand48, srand48, seed48, lcong48:/ drand48(3C) jsh, rsh, restsh: shell, the command sh(1) kbd: AViiON series workstation system kbd(7) KBD module in a pipeline kbdpipe(1) kbdcomp: compile att_kbd tables kbdcomp(1M) kbdload: load or link att_kbd tables kbdload(1M) kbdpipe: use the KBD module in a kbdpipe(1) kbdset: attach to att_kbd mapping kbdset(1)
pipeline tables, set modes	kernel admkernel(1M) kernel description file system(4) kernel logical memory kmem(7)
admkernel: manipulate the system's system: format of a	key /decrypt conversation dg_decryptsessionkey(2) key /encrypt conversation dg_encryptsessionkey(2) key /get file usage information dg_file_info(2) key dg_getrootkey(2) key in the keyserver dg_setsecretkey(2) key makekey(1)
kmem: key with the client/server common key with the client/server common for process identified by process /dg_getrootkey: get root's secret /dg_setsecretkey: store a client's secret makekey: generate encryption /decrypt conversation /encrypt conversation	key with the client/server common key dg_decryptsessionkey(2) key with the client/server common key dg_encryptsessionkey(2) keyboard /ungetch: get (or push curs_getch(3X) keyboard /mvwgetstr, mvwgetstr: get curs_getstr(3X) keyboard /ungetwch: get (or push back) curs_getwch(3X) keyboard /mvwgetwstr: get wchar_t curs_getwstr(3X) keyboard kbd(7)
back) characters from curses terminal character strings from curses terminal wchar_t characters from curses terminal character strings from curses terminal kbd: AViiON series workstation system /clntudp_create, host2netname, /host2netname, key_decryptsession, /key_decryptsession, key_encryptsession, getwin,/ curs_util: unctrl, /echo, noecho, halfdelay, intrflush, store a client's secret key in the /key_encryptsession, key_gendes, ckkeywd: prompt for and validate a apropos: locate commands by locate: identify a command using killall:	key_decryptsession, key_encryptsession,/ rpc(3N) key_encryptsession, key_gendes,/ rpc(3N) key_gendes, key_setsecret,/ rpc(3N) keyname, filter, use_env, putwin, curs_util(3X) keypad, meta, nodelay, notimeout, raw,/ curs_inopts(3X) keyserver /dg_setsecretkey: dg_setsecretkey(2) key_setsecret, get_myaddress,/ rpc(3N) keyword ckkeywd(1) keyword lookup apropos(1) keywords locate(1) kill all active processes killall(1M) kill: send a signal to a process kill(2) kill: terminate a process by default kill(1) killall: kill all active processes killall(1M) killchar, longname, termattr, termname:/ curs_termattr(3X) killpg: send signal to a process or a killpg(2) kmem: kernel logical memory kmem(7)
/baudrate, erasechar, has_ic, has_il, process group	KornShell, a standard/restricted command ksh(1) ksh, rksh: KornShell, a ksh(1) l3tol, ltol3: convert between 3-byte l3tol(3C) l64a: convert between long integer and a64l(3C) label routines /slk_touch, slk_attron, curs_slk(3X) label and data translation parameters tkey(1) label and record translation settings tdisplay(1) label checking volcopy(1M) label tlabel(1) labeled tape taccess(1) labelit: copy file systems with label volcopy(1M) labels for dump tapes dump2label(1M) labs: return integer absolute value abs(3C) langinfo: language information constants langinfo(5) language bc(1) language compiler cc(1) language compiler gcc(1) language data types nl_types(5) language idl(4)
and programming language /ksh, rksh: standard/restricted command and/ integers and long integers base-64 ASCII string /a64l, slk_attrset, slk_attroff: curses soft tkey: set tdisplay: display	language information constants langinfo(5) language information nl_langinfo(3C) language /a standard/restricted ksh(1) language /nawk, nawk(1) language /oawk: oawk(1) language preprocessor cpp(1) language /sh, jsh, rsh, sh(1) language sifilter(1) language specific strings strftime(4) language variables admnls(1M) large enough to hold string and move strsave(3C)
volcopy, labelit: copy file systems with tlabel: initialize a tape with a volume taccess: initiate access to checking volcopy, dump2label: read and write abs,	bc: arbitrary-precision arithmetic cc: C gcc: GNU C nl_types: native idl: interface description langinfo: /nl_langinfo: command and programming awk: pattern scanning and processing old pattern scanning and processing cpp: the C restsh: shell, the command programming sifilter: preprocess MC88100 assembly strftime: admnls: manipulate national string/ strsave, strnsave: allocate area

basename: return the	last element of a path name	basename(3G)
logins	last: indicate last user or terminal	last(1)
end, etext, edata:	last locations in program	end(3C)
string rindex: search for the	last occurrence of a character in a	rindex(3C)
tail: deliver the	last part of a file	tail(1)
last: indicate	last user or terminal logins	last(1)
prdaily,/ chargefee, ckpacct, dodisk,	lastlogin, monacct, nulladm, prctmp,	acctsh(1M)
at, batch: execute commands at a	later time	at(1)
shl: shell	layer manager	shl(1)
file /chown,	lchown: change user id and group id of a	chown(2)
/getspnam, setspent, endspent, fgetspent,	lckpwwdf, ulckpwwdf: manipulate shadow/	getspent(3C)
/mrand48, jrand48, srand48, seed48,	lcong48: generate uniformly distributed/	drand48(3C)
	ld: link editor for common object files	ld-coff(1)
	ld: link editor for object files	ld(1)
	ldclose, ldaclose: close a common object file	ldclose(3X)
member of a COFF archive file	ldahread: read the archive header of a	ldahread(3X)
/ldopen,	ldaopen: open an object file for reading	ldopen(3X)
file	ldclose, ldaclose: close a common object	ldclose(3X)
	ldd: list dynamic dependencies	ldd(1)
scalb: manipulate parts of/ frexp,	ldexp, logb, modf, modff, nextafter,	frexp(3C)
routines	ldfcn: COFF executable file access	ldfcn(4)
common object file	ldfhread: read the file header of a	ldfhread(3X)
object file symbol table entry	ldgetname: retrieve symbol name for	ldgetname(3X)
/div,	ldiv: compute the quotient and remainder	div(3C)
entries of a common object/ /ldread,	ldlinit, ldllitem: manipulate line number	ldread(3X)
of a common object/ /ldread, ldlinit,	ldllitem: manipulate line number entries	ldread(3X)
line number entries of a common object/	ldread, ldlinit, ldllitem: manipulate	ldread(3X)
entries of a section of a common object/	ldlseek, ldlnlseek: seek to line number	ldlseek(3X)
a section of a common object/ /ldlseek,	ldnlseek: seek to line number entries of	ldlseek(3X)
a section of a common object/ /ldrseek,	ldnrseek: seek to relocation entries of	ldrseek(3X)
section header of a common/ ldshread,	ldnshread: read an indexed/named	ldshread(3X)
section of a common object/ ldsseek,	ldnsseek: seek to an indexed/named	ldnsseek(3X)
header of an object file	ldohseek: seek to the optional file	ldohseek(3X)
reading	ldopen, ldaopen: open an object file for	ldopen(3X)
entries of a section of a common object/	ldrseek, ldnrseek: seek to relocation	ldrseek(3X)
indexed/named section header of a/	ldshread, ldnshread: read an	ldshread(3X)
indexed/named section of a common/	ldsseek, ldnsseek: seek to an	ldsseek(3X)
entry of an object file	ldtbindex: compute index of symbol table	ldtbindex(3X)
entry of an object file	ldtbread: read an indexed symbol table	ldtbread(3X)
object file	ldtbseek: seek to the symbol table of an	ldtbseek(3X)
discipline module	ldterm: standard STREAMS terminal line	ldterm(7)
/clearok, idlok, idcok immedok,	leaveok, setscreg, wsetscreg,/	cursor_outopts(3X)
	legend: Debugging information technology	legend(5)
ftuncate: set a file to a specified	length	ftuncate(3C)
truncate: truncate a file to a specified	length	truncate(2)
/dg_allow_shared_descriptor_attach:	let processes attach shared descriptor/	dg_allow_shared_descriptor_attach(2)
getopt: get option	letter from argument vector	getopt(3C)
dbx: source	level debugger	dbx(1)
/addseverity: build list of severity	levels for application to be used with/	addseverity(3C)
lexical tasks	lex: generate programs for simple	lex(1)
lex: generate programs for simple	lexical tasks	lex(1)
lsearch,	lfind: linear search and update	lsearch(3C)
gamma,	lgamma: log gamma function	gamma(3M)
intro: introduction to subroutines and	libraries	intro(3)
intro: introduction to math	libraries	intro(3M)
/elf_version: coordinate	library and application versions	elf_version(3E)
interfaces (emulated) to the termcap	library /tgetstr, tgoto, tputs: curses	cursor_termcap(3X)
elf: object file access	library	elf(3E)
intro: introduction to network	library functions	intro(3N)
find ordering relation for an object	library /lorder:	lorder(1)
/ar: archive and	library maintainer for portable archives	ar(1)
/xdr_vector, xdr_void, xdr_wrapstring:	library routines for external data/	xdr(3N)
calls /xpvt_register, xpvt_unregister:	library routines for remote procedure	rpc(3N)
t_alloc: allocate a	library structure	t_alloc(3N)
t_free: free a	library structure	t_free(3N)
t_sync: synchronize transport	library	t_sync(3N)
cpd: change or view the allocation	limits for a control point directory	cpd(1)
implementation-specific constants	limits: header file for	limits(4)
/dg_set_cpd_limits: change the resource	limits of a control point directory	dg_set_cpd_limits(2)
ulimit: get or set process	limits	ulimit(2)
sttydefs: maintain	line and hunt settings for TTY ports	sttydefs(1M)

setlinebuf: assign	line buffering for a specified stream	setlinebuf(3C)
dial: establish an out-going terminal	line connection	dial(3C)
connections connld:	line discipline for unique stream	connld(7)
set terminal type, modes, speed, and	line discipline /getty:	getty(1M)
ldterm: standard STREAMS terminal	line discipline module	ldterm(7)
editread: command	line editor	editread(5)
line: read one	line	line(1)
file /linenum:	line number entries in a common object	linenum(4)
/ldlread, ldllinit, ldllitem: manipulate	line number entries of a common object/	ldlread(3X)
common/ ldllseek, ldllseek: seek to	line number entries of a section of a	ldllseek(3X)
nl:	line numbering filter	nl(1)
cut: cut out selected fields of each	line of a file	cut(1)
rev: reverse order of characters in each	line of file	rev(1)
lpc:	line printer control program	lpc(1M)
lp: DGC AViiON family	line printer special files	lp(7)
lpd:	line printer spooler	lpd(1M)
lpr: send print requests to a	line printer spooler	lpr(1)
lprm: remove jobs from the	line printer spooling queue	lprm(1)
	line: read one line	line(1)
ttydefs: terminal	line settings information for ttymon	ttydefs(4M)
lsearch, lfind:	linear search and update	lsearch(3C)
col: filter reverse	line-feeds	col(1)
object file	linenum: line number entries in a common	linenum(4)
comm: select or reject	lines common to two sorted files	comm(1)
curses borders, horizontal and vertical	lines /box, whline, wvline: create	curs_border(3X)
wredrawln: refresh curses windows and	lines /doupdate, redrawwin,	curs_refresh(3X)
fold: fold long	lines for finite width output device	fold(1)
head: give the first few	lines	head(1)
insertln, winsertln: delete and insert	lines in a curses window /winsdelln,	curs_deleteln(3X)
uniq: report repeated	lines in a file	uniq(1)
paste: merge	lines	paste(1)
link, unlink: exercise	link and unlink system calls	link(1M)
kbdload: load or	link att_kbd tables	kbdload(1M)
	link: create a new link to a file	link(2)
ld:	link editor for common object files	ld-coff(1)
ld:	link editor for object files	ld(1)
a.out: assembler and	link editor output	a.out(4)
Environment variable sensitive file	link /elink:	elink(5)
symlink: create a symbolic	link file	symlink(2)
ln:	link files	ln(1)
read the contents of a symbolic	link /readlink:	readlink(2)
link: create a new	link to a file	link(2)
system calls	link, unlink: exercise link and unlink	link(1M)
/form_field_new: new_field, dup_field,	link_field, free_field,: create and/	form_field_new(3X)
/set_fieldtype_arg, set_fieldtype_choice,	link_fieldtype: forms fieldtype routines	form_fieldtype(3X)
	lint: a C program checker	lint(1)
ls:	list contents of directory	ls(1)
ldd:	list dynamic dependencies	ldd(1)
ttysrch: directory search	list for ttyname	ttysrch(4M)
get or set supplementary group access	list IDs /getgroups, setgroups:	getgroups(2)
the supplementary group access	list /initgroups: initialize	initgroups(3C)
nlist: get entries from name	list	nlist(3C)
dispgid: display a	list of all valid group names	dispgid(1)
dispuid: display a	list of all valid user names	dispuid(1)
nm: print name	list of common object file	nm(1)
and ncheck /checklist:	list of file systems processed by fsck	checklist(4)
to be used with/ /addseverity: build	list of severity levels for application	addseverity(3C)
stdarg: handle variable argument	list	stdarg(5)
/logins:	list user and system login information	logins(1M)
listusers:	list user login information	listusers(1)
varargs: handle variable argument	list	varargs(5)
formatted output of a variable argument	list /vprintf, vfprintf, vsprintf: print	vprintf(3S)
formatted output of a variable argument	list /vprintf, vfprintf, vsprintf: print	vprintf(3W)
formatted input using varargs argument	list /vscanf, vscanf, vsscanf: convert	vscanf(3S)
group	listdgrp: lists members of a device	listdgrp(1M)
t_listen:	listen for a connect request	t_listen(3N)
listen:	listen for connections on a socket	listen(2)
socket	listen: listen for connections on a	listen(2)
	listen: network listener server	listen(1M)
get client's data passed via the	listener /nlsgetcall:	nlsgetcall(3N)
listen: network	listener server	listen(1M)

nlsadmin: network	listener service administration	nlsadmin(1M)
nlsrequest: format and send	listener service request message	nlsrequest(3N)
xargs: construct argument	list(s) and execute command	xargs(1)
devattr:	lists device attributes	devattr(1M)
devices that match criteria	getdgrp: lists device groups which contain	getdgrp(1M)
getdev:	lists devices based on criteria	getdev(1M)
listdgrp:	lists members of a device group	listdgrp(1M)
listusers:	list user login information	listusers(1)
ln: link files	ln(1)	ln(1)
lsd:	load a system dump from tape	lsd(1M)
kbdload:	load or link att_kbd tables	kbdload(1M)
tload:	load terminal controller devices	tload(1M)
finger: display information about	local and remote users	finger(1)
testlocale: test	locale definition	testlocale(1M)
setlocale: modify and query a program's	locale	setlocale(3C)
set default system time zone and	locale /timezone:	timezone(4)
information	localeconv: get numeric formatting	localeconv(3C)
convert date and time to string	localtime, gmtime, asctime, tzset:	ctime(3C)
ctime, /which:	locate a program file for csh(1) users	which(1)
reference manuals	locate and print entries from the	man(1)
man:	locate commands by keyword lookup	apropos(1)
apropos:	locate: identify a command using	locate(1)
keywords	locate source, binary, and or manual for	whereis(1)
program /whereis:	location /fetch_and_add:	fetch_and_add(2)
indivisible fetch and add to memory	locations in program	end(3C)
end, etext, edata: last	lock clients /dg_lock_kill:	dg_lock_kill(2)
remove locks held by remote	lock data, text, or both into memory	plock(2)
plock:	lock database, start lock reclaim grace	dg_lock_reset(2)
period /dg_lock_reset: reset remote file	lock manager device interface	plm(7)
plm: pseudo	lock on an open DG/UX file	dg_flock(3C)
dg_flock: apply or remove an advisory	lock or unlock address space	mlockall(3C)
mlockall, munlockall:	lock (or unlock) pages in memory	mlock(3C)
mlock, munlock:	lock reclaim grace period	dg_lock_reset(2)
/reset remote file lock database, start	lock request on a filehandle	dg_lcntl(2)
dg_lcntl: process a record	lock requests to complete /dg_lock_wait:	dg_lock_wait(2)
wait for previously delayed	lockf: record locking on files	lockf(3C)
lockf: record	locking on files	lockf(3C)
/dg_lock_kill: remove	locks held by remote lock clients	dg_lock_kill(2)
gamma, lgamma:	log gamma function	gamma(3M)
newgrp:	log in to a new group	newgrp(1)
and event tracing	log: interface to STREAMS error logging	log(7)
sqrt, sqrtf:/ exp, expf, cbrt,	log, logf, log10, log10f, pow, powf,	exp(3M)
logger: make entries in the system	log	logger(1)
RCS files /rlog: print	log messages and other information about	rlog(1)
configuration file for syslogd system	log server /syslog.conf:	syslog.conf(5)
closeolog, setlogmask: control system	log /syslog, openlog,	syslog(3C)
syslogd:	log systems messages	syslogd(1M)
/exp, expf, cbrt, log, logf,	log10, log10f, pow, powf, sqrt, sqrtf:/	exp(3M)
exp, expf, cbrt, log, logf, log10,	log10f, pow, powf, sqrt, sqrtf:/	exp(3M)
/pow, powf, sqrt, sqrtf: exponential,	logarithm, power, square root functions	exp(3M)
manipulate parts of/ frexp, ldexp,	logb, modf, modff, nextafter, scalb:	frexp(3C)
sqrtf:/ exp, expf, cbrt, log,	logf, log10, log10f, pow, powf, sqrt,	exp(3M)
strclean: STREAMS error	logger cleanup program	strclean(1M)
strerr: STREAMS error	logger: make entries in the system log	logger(1)
log: interface to STREAMS error	logger server	strerr(1M)
menu interface for managing physical and	logging and event tracing	log(7)
kmem: kernel	logical disks /diskman:	diskman(1M)
userdel: delete a user's	logical memory	kmem(7)
listusers: list user	login from the system	userdel(1M)
logins: list user and system	login information	listusers(1)
usermod: modify a user's	login information	logins(1M)
getlogin: get	login information on the system	usermod(1M)
logname: get	login name	getlogin(3C)
logname: return	login name	logname(1)
effective UID /cuserid: get character	login name of user	logname(3X)
useradd: administer a new user	login name or user name associated with	cuserid(3S)
passwd: change	login on the system	useradd(1M)
dial-up devices d_passwd:	login password	passwd(1)
profile: setting up an environment at	log-in programs and passwords for	d_passwd(4)
login: sign on	login: sign on	login(1)
login time	login time	profile(4)

ct: spawn	login to a remote terminal	ct(1)
last: indicate last user or terminal information	logins	last(1)
	logins: list user and system login	logins(1M)
	logname: get login name	logname(1)
	logname: return login name of user	logname(3X)
a64l, l64a: convert between	long integer and base-64 ASCII string	a64l(3C)
sputl, sgetl: access	long integer data in a/	sputl(3X)
convert between 3-byte integers and device	long integers /l3tol, ltol3:	l3tol(3C)
fold: fold	long lines for finite width output	fold(1)
setjmp,	longjmp: non-local goto	setjmp(3C)
/erasechar, has_ic, has_il, killchar,	longname, termattrs, termname: curses/	curls_termattrs(3X)
endpoint /t_look:	look at the current event on a transport	t_look(3N)
apropos: locate commands by keyword	lookup	apropos(1)
object library	lorder: find ordering relation for an	lorder(1)
nice: run a command at a higher or	lower priority	nice(1)
setsyx, ripoffline, curs_set, napms:	low-level curses routines /getsyx,	curs_kernel(3X)
LP print service	lp, cancel: send/cancel requests to an	lp(1)
special files	lp: DGC AViiON family line printer	lp(7)
lpprint, xlpprint: menu-driven	lp interface	lpprint(1M)
lp sched, lpshut, lpmove: start/stop the	LP print service and move requests	lp sched(1M)
lp, cancel: send/cancel requests to an	LP print service	lp(1)
lpadmin: configure the	LP print service	lpadmin(1M)
administer filters used with the	LP print service /lpfilter:	lpfilter(1M)
lpforms: administer forms used with the	LP print service	lpforms(1M)
information about the status of the	LP print service /lpstat: print	lpstat(1)
enable, disable: enable/disable	LP printers	enable(1)
	lpadmin: configure the LP print service	lpadmin(1M)
	lpc: line printer control program	lpc(1M)
	lpd: line printer spooler	lpd(1M)
the LP print service	lpfilter: administer filters used with	lpfilter(1M)
LP print service	lpforms: administer forms used with the	lpforms(1M)
and move requests /lp sched, lpshut,	lpmove: start/stop the LP print service	lp sched(1M)
interface	lpprint, xlpprint: menu-driven lp	lpprint(1M)
	lpq: examine the spool queue	lpq(1)
printer spooler	lpr: send print requests to a line	lpr(1)
spooling queue	lprm: remove jobs from the line printer	lprm(1)
LP print service and move requests	lp sched, lpshut, lpmove: start/stop the	lp sched(1M)
service and move requests /lp sched,	lpshut, lpmove: start/stop the LP print	lp sched(1M)
status of the LP print service	lpstat: print information about the	lpstat(1)
the print service	lpsystem: register remote systems with	lpsystem(1M)
with 40014A Terminal Server	lptermprinter: start printer session	lptermprinter(1)
	lpusers: set printing queue priorities	lpusers(1M)
rand48, seed48,/ drand48, erand48,	lrand48, nrand48, mrand48, jrand48,	drand48(3C)
	ls: list contents of directory	ls(1)
	lsd: load a system dump from tape	lsd(1M)
	lsearch, lfind: linear search and update	lsearch(3C)
position	lseek: change object pointer's current	lseek(2)
	lstat: get file status	lstat(2)
and long integers /l3tol,	ltol3: convert between 3-byte integers	l3tol(3C)
	m4: macro processor	m4(1)
provide truth value/ /machid: dghost,	m68k, m88k, i386, pdp11, u3b, u3b5, vax:	machid(1)
provide truth/ machid: dghost, m68k,	m88k, i386, pdp11, u3b, u3b5, vax:	machid(1)
u3b, u3b5, vax: provide truth value/	machid: dghost, m68k, m88k, i386, pdp11,	machid(1)
setuname: changes	machine information	setuname(1M)
values:	machine-dependent values	values(5)
sgetl: access long integer data in a	machine-independent fashion /sputl,	sputl(3X)
m4:	macro processor	m4(1)
invert, rpow, msqrt, mcomp, move,/ mp:	madd, msub, mult, mdiv, pow, gcd,	mp(3X)
rmt: start the remote	mag tape server	rmt(1M)
mt:	magnetic tape control	mt(1)
wmtd: start the WORM	magnetic tape device server	wmtd(1M)
rmt: character special	magnetic tape interface	rmt(7)
Once Read Multiple optical device) as	magtape interface /pseudo WORM (Write	wmt(7)
database admaliases: manage	mail alias information in the aliases	admaliases(1M)
mailaliases: translate	mail alias names	mailaliases(1)
mailcnfg: initialization information for	mail and rmail	mailcnfg(4M)
invoke recipient command for incoming	mail /mail_pipe:	mail_pipe(1M)
commands for routing and transport of	mail /mailsurr: surrogate	mailsurr(4M)
automatically respond to incoming	mail messages /vacation:	vacation(1)
notify user of the arrival of new	mail /notify:	notify(1)
mail, rmail: read	mail or send mail to users	mail(1)

	users	mail, rmail: read mail or send mail to	mail(1)
	mail, rmail: read mail or send	mail to users	mail(1)
	a binary file for transmission via	mail /uencode, udecode: encode/decode	uencode(1)
	mail and rmail	mailalias: translate mail alias names	mailalias(1)
	incoming mail	mailcnfg: initialization information for	mailcnfg(4M)
	and transport of mail	mail_pipe: invoke recipient command for	mail_pipe(1M)
	system	mailsurr: surrogate commands for routing	mailsurr(4M)
	main: enter a C	mailx: interactive message processing	mailx(1)
	mem:	main: enter a C main program	main(3C)
	ports /sttydefs:	main program	main(3C)
	of programs /make:	main system memory	mem(7)
	ar: archive and library	maintain line and hunt settings for TTY	sttydefs(1M)
Array/	gridman: menu interface for	maintain, update, and regenerate groups	make(1)
programs	intro: introduction to system	maintainer for portable archives	ar(1)
	intro: introduction to system	maintaining a High Availability Disk	gridman(1M)
	delta:	maintenance commands and application	intro(1M)
	via NFS /exportfs:	maintenance procedures	intro(8)
	mkdir:	make a delta (change) to an SCCS file	delta(1)
	elf_begin:	make a directory available for mounting	exportfs(2)
	elf_strptr:	make a directory	mkdir(1)
	mkstemp:	make a file descriptor	elf_begin(3E)
	mktemp:	make a string pointer	elf_strptr(3E)
database	helpadm:	make a unique file name	mkstemp(3C)
	logger:	make a unique file name	mktemp(3C)
	mkfifo:	make changes to the help facility	helpadm(1M)
	groups of programs	make entries in the system log	logger(1)
	banner:	make FIFO special file	mkfifo(1M)
/res_send, res_init, dn_comp, dn_expand:	script:	make: maintain, update, and regenerate	make(1)
	makekey: generate encryption key	make posters	banner(1)
	makekey: generate encryption key	make, send, and interpret packets to/	resolver(3C)
	makekey: generate encryption key	make typescript of a terminal session	script(1)
	makekey: generate encryption key	makekey: generate encryption key	makekey(1)
malloc, free, realloc, calloc, mallopt,	mallinfo: memory allocator	malloc, free, realloc, calloc, mallopt,	malloc(3X)
malloc, free, realloc, calloc, mallopt,	malloc, free, realloc, calloc, memalign,	malloc, free, realloc, calloc, memalign,	malloc(3C)
malloc, free, realloc, calloc,	malloc, mallinfo: memory allocator	malloc, mallinfo: memory allocator	malloc(3X)
reference manuals	man: locate and print entries from the	man: locate and print entries from the	man(1)
/admaccounting:	manage accounting system	manage accounting system	admaccounting(1M)
systems	admbackup: manage backup and recovery of file	manage backup and recovery of file	admbackup(1M)
tsearch, tfind, tdelete, twalk:	admpackage: manage DG/UX-style software packages	manage binary search trees	tsearch(3C)
nameservers database	admresolve: manage DNS resolver's domain name and	manage DG/UX-style software packages	admpackage(1M)
	/admdumpcycle: manage dump cycle tables	manage DNS resolver's domain name and	admresolve(1M)
	admether: manage ether database	manage dump cycle tables	admdumpcycle(1M)
	/admfilesystem: manage file systems	manage ether database	admether(1M)
database	admgroup: manage group information in the group	manage file systems	admfilesystem(1M)
hsearch, hcreate, hdestroy:	admhost: manage hosts database	manage group information in the group	admgroup(1M)
aliases database	admalias: manage mail alias information in the	manage hash search tables	hsearch(3C)
	admnetwork: manage network database	manage hosts database	admhost(1M)
	admclient: manage operating system clients	manage mail alias information in the	admalias(1M)
/t_optmgt:	t_optmgt: manage options for a transport endpoint	manage network database	admnetwork(1M)
/admportservice:	admportservice: manage port monitor services	manage operating system clients	admclient(1M)
/admportmonitor:	admportmonitor: manage port monitors	manage options for a transport endpoint	t_optmgt(3N)
	admprocess: manage processes	manage port monitor services	admportservice(1M)
	admroute: manage routing databases	manage port monitors	admportmonitor(1M)
and DNS databases	/admsvcorder: manage search order for /etc/hosts, NIS,	manage processes	admprocess(1M)
	admservice: manage service database	manage routing databases	admroute(1M)
	/admxtterminal: manage serving of X display terminals	manage search order for /etc/hosts, NIS,	admsvcorder(1M)
	admlock: manage simple process synchronization	manage service database	admservice(1M)
	admrelease: manage software release areas	manage serving of X display terminals	admxtterminal(1M)
	admswap: manage swap areas	manage simple process synchronization	admlock(1M)
reporting	admsar: manage system activity monitoring and	manage software release areas	admrelease(1M)
	installman: manage system installation	manage swap areas	admswap(1M)
	/admterminal: manage terminal ports	manage system activity monitoring and	admsar(1M)
	/admdumpdevice: manage the dump device table	manage system installation	installman(1M)
names /admshell:	admshell: manage the remote and restricted shell	manage terminal ports	admterminal(1M)
/admsnmpcommunity:	admsnmpcommunity: manage the SNMP community database	manage the dump device table	admdumpdevice(1M)
	/admsnmptrap: manage the SNMP traps database	manage the remote and restricted shell	admshell(1M)
	/admsnmpobject: manage the snmpd object database	manage the SNMP community database	admsnmpcommunity(1M)
	/admtcpiipparams: manage the TCP/IP host parameters	manage the SNMP traps database	admsnmptrap(1M)
		manage the snmpd object database	admsnmpobject(1M)
		manage the TCP/IP host parameters	admtcpiipparams(1M)

database /admipinterface:	manage the TCP/IP network interfaces	admipinterface(1M)
/admtcpipdaemon:	manage the TCP/IP servers	admtcpipdaemon(1M)
/admtrustedhost:	manage the trusted hosts database	admtrustedhost(1M)
database /admuser:	manage user information in the password	admuser(1M)
memcntl: memory	management control	memcntl(2)
alp: Algorithm Pool	management module	alp(7)
passmgmt: password files	management	passmgmt(1M)
plm: pseudo lock	manager device interface	plm(7)
dfm: DOS file	manager	dfm(4M)
hfm: high sierra file	manager	hfm(4)
shl: shell layer	manager	shl(1)
diskman: menu interface for	managing physical and logical disks	diskman(1M)
fwtmp, wtmpfix:	manipulate connect accounting records	fwtmp(1M)
elf_flagphdr, elf_flagscn, elf_flagshdr:	manipulate flags /elf_flagelf,	elf_flag(3E)
common/ ldlread, ldlinit, ldlitem:	manipulate line number entries of a	ldlread(3X)
/admnl:	manipulate national language variables	admnl(1M)
/overlay, overwrite, copywin: overlap and	manipulate overlapped curses windows	curs_overlay(3X)
/logb, modf, modif, nextafter, scalb:	manipulate parts of floating-point/	frexp(3C)
/endpwent, setpwfile, fgetpwent:	manipulate password file entry	getpwent(3C)
sigaddset, sigdelsset, sigismember:	manipulate sets of signals. /sigfillset,	sigsetops(3C)
/endspent, fgetspent, lckpwwdf, ulckpwwdf:	manipulate shadow password file entry	getspent(3C)
object file. mcs:	manipulate the comment section of an	mcs(1)
tapes admtape:	manipulate the default parameters for	admtape(1M)
time zone admdate:	manipulate the system date, time and	admdate(1M)
admkernel:	manipulate the system's kernel	admkernel(1M)
/swapcontext:	manipulate user contexts	swapcontext(3C)
bkgd, wbkgd: curses window background	manipulation routines /wbkgdset,	curs_bkgd(3X)
pair_content: curses color	manipulation routines /color_content,	curs_color(3X)
curses screen initialization and	manipulation routines /delscreen:	curs_initscr(3X)
inet_lnaof, inet_netof: Internet address	manipulation routines /inet_makeaddr,	inet(3N)
hide_panel, panel_hidden: panels deck	manipulation routines /show_panel,	panel_show(3X)
top_panel, bottom_panel: panels deck	manipulation routines /panel_top:	panel_top(3X)
str: strfind, strrspn, strtrns: string	manipulations	str(3G)
whereis: locate source, binary, and or	manual for program	whereis(1)
and print entries from the reference	manuals /man: locate	man(1)
ascii:	map of ASCII character set	ascii(5)
mmap:	map pages of memory	mmap(2)
memctl: set memory access for	mapping	memctl(2)
mprotect: set memory access for	mapping	mprotect(2)
ether_line: Ethernet address	mapping operations /ether_hostton,	ethers(3N)
kbdset: attach to att_kbd	mapping tables, set modes	kbdset(1)
cpz: compose-key	maps	cpz(4M)
set_menu_mark, menu_mark: menus	mark string routines /menu_mark:	menu_mark(3X)
umask: set file-creation mode	mask	umask(1)
umask: set and get file creation	mask	umask(2)
mkstr: create an error message file by	massaging C source	mkstr(1)
master: format of a	master file	master(4)
unlockpt: unlock a pseudo-terminal	master: format of a master file	master(4)
pty, pts, ptc: pseudo-terminal	master/slave pair	unlockpt(3C)
menu_pattern: set and get menus pattern	master/slave pseudo-device pair	pty(7)
device groups which contain devices that	match buffer /set_menu_pattern,	menu_pattern(3X)
advance: regular expression compile and	match criteria /getdgrp: lists	getdgrp(1M)
advance: regular expression compile and	match routines /regexp: compile, step,	regexp(5)
gmatch: shell global pattern	match routines /regexpr: compile, step,	regexpr(3G)
math:	matching	gmatch(3G)
intro: introduction to	math functions and constants	math(5)
printers postmd:	math libraries	intro(3M)
menus /menu_format: set and get	math: math functions and constants	math(5)
getrlimit, setrlimit: control	matherr: error-handling function	matherr(3M)
vlimit: control	matrix display program for PostScript	postmd(1)
character handling	maximum numbers of rows and columns in	menu_format(3X)
character conversion	maximum system resource consumption	getrlimit(2)
mbchar: mbtowc, wctomb, mblen: multibyte	maximum system resource consumption	vlimit(3C)
handling mbchar: mbtowc,	mbchar: mbtowc, mblen, wctomb: multibyte	mbchar(3C)
functions mbstring:	mbchar: mbtowc, wctomb, mblen: multibyte	mbchar(3W)
conversion mbstring:	mblen: multibyte character conversion	mbchar(3W)
string functions	mblen, wctomb: multibyte character	mbchar(3C)
string conversion	mbstowcs, wcstombs: multibyte string	mbstring(3C)
	mbstowcs, wctombs: multibyte string	mbstring(3W)
	mbstring: mbstowcs, wcstombs: multibyte	mbstring(3C)
	mbstring: mbstowcs, wctombs: multibyte	mbstring(3W)

character handling **mbchar**: **mbtowc**, **mblen**, **wctomb**: **multibyte** **mbchar(3C)**
 character conversion **mbchar**: **mbtowc**, **wctomb**, **mblen**: **multibyte** **mbchar(3W)**
 as: **MC88000 assembler** **as(1)**
 sifilter: **preprocess** **MC88100 assembly language** **sifilter(1)**
/mdiv, **pow**, **gcd**, **invert**, **rpow**, **msqrt**, **mcmp**, **move**, **min**, **omin**, **fmin**, **m_in**, **mout**,/
 an object file. **mp(3X)**
mcs: **manipulate the comment section of** **mcs(1)**
mdiv, **pow**, **gcd**, **invert**, **rpow**, **msqrt**, **mp(3X)**
mem: **main system memory** **mem(7)**
 malloc, **free**, **realloc**, **calloc**, **memalign**, **valloc**: **memory allocator** **malloc(3C)**
 elf_next: **sequential archive** **elf_next(3E)**
 elf_rand: **random archive** **elf_rand(3E)**
 /elf_getarhdr: **retrieve archive** **elf_getarhdr(3E)**
ldahread: **read the archive header of a** **ldahread(3X)**
 offsetof: **offset of structure** **offsetof(3C)**
 listdgrp: **lists** **listdgrp(1M)**
 groups: **show group** **groups(1)**
memmove, **memset**: **memory/ memory:** **memory(3C)**
memory operations /memory: memccpy, **memory(3C)**
operations /memory: memccpy, **memchr**, **memory(3C)**
 memory: memccpy, **memchr**, **memcmp**, **memory(3C)**
 memctl: **set memory access for mapping** **memctl(2)**
 memmove, **memset**: **memory operations** **memory(3C)**
 memory access faults **misalign(5)**
 memory access for mapping **memctl(2)**
 memory access for mapping **mprotect(2)**
 memory access **stkexec(2)**
 memory allocator /malloc, **free**, **malloc(3C)**
 memory allocator /malloc, **free**, **malloc(3X)**
 memory **bcmp(3C)**
 memory **bzero(3C)**
 memory control operations **shmctl(2)**
 memory **copylist(3G)**
 memory efficient way **vfork(2)**
 memory file system **mfs(4)**
 memory ID /ipcrm: **remove a** **ipcrm(1)**
 memory **kmem(7)**
 memory location /fetch_and_add: **fetch_and_add(2)**
 memory management control **memcntl(2)**
 memory **mem(7)**
 memory: memccpy, **memchr**, **memcmp**, **memcpy**, **memory(3C)**
 memory /mlock, **mlock(3C)**
 memory **mmap(2)**
 memory **munmap(2)**
 memory operation **shmsys(2)**
 memory operations /memory: memccpy, **memory(3C)**
 memory pages **dg_paging_info(2)**
 memory pages **mincore(2)**
 memory **plock(2)**
 memory resident file system information **sync(2)**
 memory segment **shmat(2)**
 memory segment **shmdt(2)**
 memory segment **shmget(2)**
 memory to a file **syacdump(1M)**
 memory with physical storage **msync(3C)**
 memset: **memory operations /memccpy**, **memory(3C)**
 menu interface for maintaining a High **gridman(1M)**
 menu interface for managing physical and **diskman(1M)**
 menu item /ckitem: **ckitem(1)**
 menu; prompt for and return a menu item **ckitem(1)**
 menu_attributes: set_menu_fore, **menu_attributes(3X)**
 menu_back, **set_menu_grey**, **menu_grey**,/
 menu_cursor: pos_menu_cursor: correctly **menu_cursor(3X)**
 menu-driven lp interface **lpprint(1M)**
 menu-driven system administration **sysadm(1M)**
 menu-driven system administration **osysadm(1M)**
 menu_driver: command processor for the **menu_driver(3X)**
 menu_fore, **set_menu_back**, **menu_back**,/
 menu_format: set and get maximum/ **menu_attributes(3X)**
 of rows/ /menu_format: set_menu_format, **menu_format(3X)**
 menu_format: set and get maximum numbers **menu_format(3X)**
 menu_grey, **set_menu_pad**, **menu_pad**:/ **menu_attributes(3X)**

set_item_term, item_term, /	menu_hook: set_item_init, item_init,	menu_hook(3X)
/set_item_term, item_term, set_menu_init,	menu_init, set_menu_term, menu_term: /	menu_hook(3X)
current_item, set_top_row, top_row, /	menu_item_current: set_current_item,	menu_item_current(3X)
item_description: get menus item name /	menu_item_name: item_name,	menu_item_name(3X)
create and destroy menus items	menu_item_new: new_item, free_item:	menu_item_new(3X)
item_opts_on, item_opts_off, item_opts: /	menu_item_opts: set_item_opts,	menu_item_opts(3X)
item_count: connect and disconnect /	menu_items: set_menu_items, menu_items,	menu_items(3X)
disconnect / menu_items: set_menu_items,	menu_items, item_count: connect and	menu_items(3X)
item_userptr: associate application /	menu_item_userptr: set_item_userptr,	menu_item_userptr(3X)
item_value: set and get menus item /	menu_item_value: set_item_value,	menu_item_value(3X)
if menus item is visible	menu_item_visible: item_visible: tell	menu_item_visible(3X)
menus mark string routines	menu_mark: set_menu_mark, menu_mark:	menu_mark(3X)
menu_mark: set_menu_mark,	menu_mark: menus mark string routines	menu_mark(3X)
and destroy menus	menu_new: new_menu, free_menu: create	menu_new(3X)
menu_opts_off, menu_opts: menus option /	menu_opts: set_menu_opts, menu_opts_on,	menu_opts(3X)
/menu_opts_on, menu_opts_off,	menu_opts: menus option routines	menu_opts(3X)
/menu_opts: set_menu_opts, menu_opts_on,	menu_opts_off, menu_opts: menus option /	menu_opts(3X)
menus option / /menu_opts: set_menu_opts,	menu_opts_on, menu_opts_off, menu_opts:	menu_opts(3X)
/set_menu_grey, menu_grey, set_menu_pad,	menu_pad: control menus display /	menu_attributes(3X)
menu_pattern: set and get menu pattern /	menu_pattern: set_menu_pattern,	menu_pattern(3X)
match / /menu_pattern: set_menu_pattern,	menu_pattern: set and get menu pattern	menu_pattern(3X)
write or erase menus from associated /	menu_post: post_menu, unpost_menu:	menu_post(3X)
	menus: character based menu package	menus(3X)
pos_menu_cursor: correctly position a	menu_cursor /menu_cursor:	menu_cursor(3X)
set_menu_pad, menu_pad: control	menus display attributes /menu_grey,	menu_attributes(3X)
/post_menu, unpost_menu: write or erase	menus from associated subwindows	menu_post(3X)
/item_visible: tell if	menu item is visible	menu_item_visible(3X)
/item_name, item_description: get	menu item name and description	menu_item_name(3X)
/item_opts_on, item_opts_off, item_opts:	menu item option routines	menu_item_opts(3X)
set_item_value, item_value: set and get	menu item values /menu_item_value:	menu_item_value(3X)
top_row, item_index: set and get current	menu items /current_item, set_top_row,	menu_item_current(3X)
new_item, free_item: create and destroy	menu items /menu_item_new:	menu_item_new(3X)
associate application data with	menu items /item_userptr:	menu_item_userptr(3X)
menu_mark: set_menu_mark, menu_mark:	menu mark string routines	menu_mark(3X)
maximum numbers of rows and columns in	menu /menu_format: set and get	menu_format(3X)
routines for automatic invocation by	menu /assign application-specific	menu_hook(3X)
connect and disconnect items to and from	menu /menu_items, item_count:	menu_items(3X)
new_menu, free_menu: create and destroy	menu /menu_new:	menu_new(3X)
associate application data with	menu /set_menu_userptr, menu_userptr:	menu_userptr(3X)
menu_opts_on, menu_opts_off, menu_opts:	menu option routines /set_menu_opts,	menu_opts(3X)
menus: character based	menu package	menus(3X)
/menu_pattern: set and get	menu pattern match buffer	menu_pattern(3X)
/set_menu_sub, menu_sub, scale_menu:	menu subsystem	menu_driver(3X)
/set_menu_win, menu_win, set_menu_sub,	menu window and subwindow association /	menu_win(3X)
/set_menu_init, menu_init, set_menu_term,	menu_sub, scale_menu: menu window and /	menu_win(3X)
menu_userptr: associate application /	menu_term: assign application-specific /	menu_hook(3X)
with / /menu_userptr: set_menu_userptr,	menu_userptr: set_menu_userptr,	menu_userptr(3X)
set_menu_sub, menu_sub, scale_menu: /	menu_userptr: associate application data	menu_userptr(3X)
scale_menu: / menu_win: set_menu_win,	menu_win: set_menu_win, menu_win,	menu_win(3X)
sort: sort and/or	menu_win, set_menu_sub, menu_sub,	menu_win(3X)
paste:	merge files	sort(1)
merge: three-way file	merge lines	paste(1)
acctmerge:	merge	merge(1)
rcsmmerge:	merge or add total accounting files	acctmerge(1M)
	merge RCS revisions	rcsmmerge(1)
catgets: print message from	merge: three-way file merge	merge(1)
catopen, catclose: open/close a	msg: permit or deny messages	msg(1)
gencat: generate a formatted	message catalog	catgets(1)
catgets: read a program	message catalogue	catopen(3C)
gettext: retrieve a text string from a	message	gencat(1)
of, or search for a text string in,	message data base	catgets(3C)
putmsg, putpmsg: pass a	message data bases /display contents	gettext(1)
mkstr: create an error	message down a stream	srchtxt(1)
mkmsgs: create	message file by massaging C source	putmsg(2)
recv: receive a	message files for use by gettext	mkstr(1)
recvfrom: receive a	message from a socket	mkmsgs(1)
recvmsg: receive a	message from a socket	recv(2)
send: send a	message from a socket	recvfrom(2)
sendmsg: send a	message from a socket	recvmsg(2)
	message from a socket	send(2)
	message from a socket	sendmsg(2)

sendto: send a	message from a socket	sendto(2)
getmsg, getpmsg: get a	message from a stream	getmsg(2)
catgets: print	message from message catalog	catgets(1)
msgrcv: receive a	message	msgrcv(2)
msgsnd: send a	message	msgsnd(2)
format and send listener service request	message /nlsrequest:	nlsrequest(3N)
fntmsg: display a	message on stderr or system console	fntmsg(1)
fntmsg: display a	message on stderr or system console	fntmsg(3C)
mailx: interactive	message processing system	mailx(1)
message queue msgctl: get or set	message queue attributes or destroy a	msgctl(2)
msgget: get	message queue identifier	msgget(2)
message queue attributes or destroy a	message queue /msgctl: get or set	msgctl(2)
msgsys: perform a	message queue operation	msgsys(2)
memory ID /ipcrm: remove a	message queue, semaphore set, or shared	ipcrm(1)
signal /dg_strsignal: get	message string describing the given	dg_strsignal(3C)
/extended_strerror: get extended error	message string	extended_strerror(3C)
strerror: get error	message string	strerror(3C)
t_error: produce error	message	t_error(3N)
/extended_perror: print an error	message to standard error	extended_perror(3C)
files /rlog: print log	messages and other information about RCS	rlog(1)
whether remote system can accept binary	messages /ckbinarsys: determine	ckbinarsys(1M)
msg: permit or deny	messages	msg(1)
perror: print system error	messages	perror(3C)
psignal, psiginfo: system signal	messages	psignal(3C)
strace: print STREAMS trace	messages	strace(1M)
syslogd: log systems	messages	syslogd(1M)
automatically respond to incoming mail	messages /vacation:	vacation(1)
/noecho, halfdelay, intrflush, keypad,	meta, nodelay, notimeout, raw, noraw,/	curl_inopts(3X)
	mfs: memory file system	mfs(4)
/msqrt, mcmp, move, min, omin, fmin,	m_in, mout, omout, fmout, m_out, sdiv,/	mp(3X)
/gcd, invert, rpow, msqrt, mcmp, move,	min, omin, fmin, m_in, mout, omout,/	mp(3X)
pages	mincore: determine residency of memory	mincore(2)
clone: open any	minor device on a STREAMS driver	clone(7)
access faults	misalign: handle misaligned memory	misalign(5)
misalign: handle	misaligned memory access faults	misalign(5)
/acctwtmp: overview of accounting and	miscellaneous accounting commands	acct(1M)
/putwin, getwin, delay_output, flushinp:	miscellaneous curses utility routines	curl_util(3X)
intro: introduction to	miscellany	intro(5)
	mkdir: create a directory file	mkdir(2)
	mkdir: make a directory	mkdir(1)
directories in a path	mkdirp, rmdirp: create, remove	mkdirp(3G)
	mkfifo: create a new FIFO	mkfifo(3C)
	mkfifo: make FIFO special file	mkfifo(1M)
	mkfs, newfs: create a file system	mkfs(1M)
gettxt	mkmsgs: create message files for use by	mkmsgs(1)
	mknod: build a special file	mknod(1M)
system	mknod: create a file entry in the file	mknod(2)
	mkstemp: make a unique file name	mkstemp(3C)
massaging C source	mkstr: create an error message file by	mkstr(1)
	mktemp: make a unique file name	mktemp(3C)
calendar time	mktime: converts a tm structure to a	mktime(3C)
in memory	mlock, munlock: lock (or unlock) pages	mlock(3C)
address space	mlockall, munlockall: lock or unlock	mlockall(3C)
	mmap: map pages of memory	mmap(2)
	mnttab: mounted file system table	mnttab(4)
chmod: change file	mode	chmod(1)
umask: set file-creation	mode mask	umask(1)
pckt: STREAMS Packet	Mode module	pckt(7)
chmod: change	mode of file	chmod(2)
fchmod: change	mode of file	fchmod(2)
attach to att_kbd mapping tables, set	modes /kbdset:	kbdset(1)
getty: set terminal type,	modes, speed, and line discipline	getty(1M)
manipulate parts of/ frexp, ldexp, logb,	modf, modff, nextafter, scalb:	frexp(3C)
parts of/ frexp, ldexp, logb, modf,	modf, nextafter, scalb: manipulate	frexp(3C)
touch: update access and	modification times of a file	touch(1)
utime: set file access and	modification times	utime(2)
utimes: set file access and	modification times	utimes(2)
/groupmod:	modify a group definition on the system	groupmod(1M)
system /usermod:	modify a user's login information on the	usermod(1M)
setlocale:	modify and query a program's locale	setlocale(3C)
dg_sysctl: display or	modify boot and dump parameters	dg_sysctl(1M)

alp: Algorithm Pool management	module	alp(7)
alpq: query the ALP STREAMS	module	alp(7)
att_kbd: generalized string translation	module	att_kbd(7)
kbdpipe: use the KBD	module in a pipeline	kbdpipe(1)
STREAMS terminal line discipline	module /ldterm: standard	ldterm(7)
pckt: STREAMS Packet Mode	module	pckt(7)
ptem: STREAMS Pseudo Terminal Emulation	module	ptem(7)
Transport Interface cooperating STREAMS	module /timod:	timod(7)
Interface read/write interface STREAMS	module /tirdwr: Transport	tirdwr(7)
V7, 4BSD and XENIX STREAMS compatibility	module /ttcompat:	ttcompat(7)
configure automatically pushed STREAMS	modules /autopush:	autopush(1M)
to EUC handling TTY drivers and	modules /eucioctl: generic interface	eucioctl(5)
chargefee, ckpacct, dodisk, lastlogin,	monacct, nulladm, prctmp, prdaily,/	acctsh(1M)
monthl: create	monetary database	monthl(1M)
pmadm: port	monitor administration	pmadm(1M)
ttyadm: format and output TTY port	monitor information	tyadm(1M)
	monitor: prepare execution profile	monitor(3C)
/admportservice: manage port	monitor services	admportservice(1M)
ttymon:	monitor terminal ports	tymon(1M)
admsar: manage system activity	monitoring and reporting	admsar(1M)
/admportmonitor: manage port	monitors	admportmonitor(1M)
	monthl: create monetary database	monthl(1M)
at a time	more, page: display file one screenful	more(1)
dg_mount:	mount a file system	dg_mount(2)
mount:	mount a file system	mount(2)
mount, umount:	mount and dismount filesystems	mount(1M)
	mount: mount a file system	mount(2)
setmnt: establish	mount table	setmnt(1M)
filesystems	mount, umount: mount and dismount	mount(1M)
fstatfs: get information about a	mounted file system	fstatfs(2)
statfs: get information about a	mounted file system	statfs(2)
mnttab:	mounted file system table	mnttab(4)
exportfs: make a directory available for	mounting via NFS	exportfs(2)
mouse:	mouse device	mouse(7)
	mouse: mouse device	mouse(7)
/mcmap, move, min, omin, fmin, m_in,	mount, omout, fmout, m_out, sdiv, itom:/	mp(3X)
/omin, fmin, m_in, mout, omout, fmout,	m_out, sdiv, itom: multiple precision/	mp(3X)
mvdire:	move a directory	mvdire(1M)
screen panel_move: move_panel:	move a panels window on the virtual	panel_move(3X)
curs_move: move, wmove:	move curses window cursor	curs_move(3X)
mv:	move files	mv(1)
/pow, gcd, invert, rpow, msqrt, mcmap,	move, min, omin, fmin, m_in, mout,/	mp(3X)
start/stop the LP print service and	move requests /lpsched, lpshut, lpmove:	lpsched(1M)
area large enough to hold string and	move string into it /strnsave: allocate	strsave(3C)
/curs_move:	move, wmove: move curses window cursor	curs_move(3X)
/form_fields, field_count,	move_field: connect fields to forms	form_field(3X)
virtual screen /panel_move:	move_panel: move a panels window on the	panel_move(3X)
invert, rpow, msqrt, mcmap, move, min,/	mp: madd, msub, mult, mdiv, pow, gcd,	mp(3X)
	mprotect: set memory access for mapping	mprotect(2)
drand48, erand48, lrand48, nrand48,	mrnd48, jrand48, srand48, seed48,/	drand48(3C)
attributes or destroy a message queue	msgctl: get or set message queue	msgctl(2)
	msgget: get message queue identifier	msgget(2)
	msgrcv: receive a message	msgrcv(2)
	msgsnd: send a message	msgsnd(2)
operation	msgsys: perform a message queue	msgsys(2)
/mult, mdiv, pow, gcd, invert, rpow,	msqrt, mcmap, move, min, omin, fmin,/	mp(3X)
rpow, msqrt, mcmap, move, min,/	msub, mult, mdiv, pow, gcd, invert,	mp(3X)
storage	msync: synchronize memory with physical	msync(3C)
msqrt, mcmap, move, min,/	mt: magnetic tape control	mt(1)
mp: madd, msub,	mult, mdiv, pow, gcd, invert, rpow,	mp(3X)
mbchar: mbtowc, wctomb, mblen:	multibyte character conversion	mbchar(3W)
mbchar: mbtowc, mblen, wctomb:	multibyte character handling	mbchar(3C)
widec:	multibyte character I/O routines	widec(3W)
mbstring: mbstowcs, wctombs:	multibyte string conversion	mbstring(3W)
mbstring: mbstowcs, wctombs:	multibyte string functions	mbstring(3C)
/dot3: IEEE 802.3 carrier sense	multiple access with collision detection	dot3(6P)
wmt: pseudo WORM (Write Once Read	Multiple optical device) as magtape/	wmt(7)
/mout, omout, fmout, m_out, sdiv, itom:	multiple precision integer arithmetic	mp(3X)
poll: input/output	multiplexing	poll(2)
memory mlock,	munlock: lock (or unlock) pages in	mlock(3C)
/mlockall,	munlockall: lock or unlock address space	mlockall(3C)

add a/ /curs_addch: addch, waddch,
 add/ /waddchstr, waddchnstr, mvaddchstr,
 add/ /waddchstr, waddchnstr, mvaddchstr,
 /addchnstr, waddchstr, waddchnstr,
 /addchnstr, waddchstr, waddchnstr,
 /addnstr, waddstr, waddnstr, mvaddstr,
 a/ /waddwstr, waddnwstr, mvaddwstr,
 /addstr, addnstr, waddstr, waddnstr,
 /curs_addwch: addwch, waddwch,
 waddwchstr, waddwchnstr, mvaddwchstr,
 /addwchnstr, waddwchstr, waddwchnstr,
 /addwstr, addnwstr, waddwstr, waddnwstr,
 /tparm, tputs, putp, vidputs, vidattr,
 under/ curs_delch: delch, wdelch,
 /newwin, delwin, mvwin, subwin, derwin,

 back)/ /curs_getch: getch, wgetch,
 /getnstr, wgetstr, wgetnstr, mvgetstr,
 wchar_t/ /wgetstr, wgetwstr, mvgetwstr,
 /getstr, getnstr, wgetstr, wgetnstr,
 push/ /curs_getwch: getwch, wgetwch,
 /getwstr, getnwstr, wgetwstr, wgetnwstr,
 attributes/ /curs_inch: inch, winch,
 a/ /winchstr, winchnstr, mvinchstr,
 /inchstr, inchnstr, winchstr, winchnstr,
 /instr, innstr, winstr, winnstr, mvinstr,
 /innwstr, winwstr, winnwstr, mvinnwstr,
 before the/ curs_insch: insch, winsch,
 /insnstr, winsstr, winsnstr, mvinsstr,
 insert/ /winswstr, winsnwstr, mvinswstr,
 /insstr, insnstr, winsstr, winsnstr,
 get a/ /instr, innstr, winstr, winnstr,
 /curs_inswch: inswch, winswch,
 /inswstr, insnwstr, winswstr, winsnwstr,
 character/ curs_inwch: inwch, winwch,
 get/ /winwchstr, winwchnstr, mvwinwchstr,
 /inwchnstr, winwchstr, winwchnstr,
 /inwstr, innwstr, winwstr, winnwstr,
 /curs_printw: printw, wprintw,
 formatted/ curs_scanw: scanw, wscanw,
 curs_addch: addch, waddch, mvaddch,
 /mvaddchstr, mvaddchnstr, mvwaddchstr,
 /mvaddchstr, mvaddchnstr, mvwaddchstr,
 /waddchnstr, mvaddchstr, mvaddchnstr,
 /waddchnstr, mvaddchstr, mvaddchnstr,
 to a/ /mvaddstr, mvaddnstr, mvwaddstr,
 /mvaddwstr, mvaddnwstr, mvwaddwstr,
 /waddstr, waddnstr, mvaddstr, mvaddnstr,
 /curs_addwch: addwch, waddwch, mvaddwch,
 /mvaddwchstr, mvaddwchnstr, mvwaddwchstr,
 /waddwchnstr, mvaddwchstr, mvaddwchnstr,
 /waddnwstr, mvaddnwstr, mvwaddnwstr,
 in/ /curs_delch: delch, wdelch, mvdelch,
 curs_getch: getch, wgetch, mvgetch,
 curses/ /mvgetstr, mvgetnstr, mvwgetstr,
 /mvgetwstr, mvgetnwstr, mvwgetwstr,
 /wgetstr, wgetnstr, mvgetstr, mvgetnstr,
 /curs_getwch: getwch, wgetwch, mvgetwch,
 /wgetwstr, mvgetwstr, mvgetnwstr,
 wsyncup,/ /curs_window: newwin, delwin,
 curs_inch: inch, winch, mvinch,
 (and/ /mvinchstr, mvinchnstr, mvwinchstr,
 /winchnstr, mvinchstr, mvinchnstr,
 /winnstr, mvinstr, mvinnstr, mvwinstr,
 /mvinnwstr, mvinnwstr, mvwinwstr,
 /curs_insch: insch, winsch, mvinsch,
 /mvinsstr, mvinsnstr, mvwinsstr,
 /mvinswstr, mvinsnwstr, mvwinswstr,
 /winsstr, winsnstr, mvinsstr, mvinsnstr,

 munmap: unmap pages of memory munmap(2)
 mv: move files mv(1)
 mvaddch, mvwaddch, echochar, wechochar: curs_addch(3X)
 mvaddchnstr, mvwaddchstr, mvwaddchnstr: curs_addchstr(3X)
 mvaddchnstr, mvwaddchstr, mvwaddchnstr: curs_addchstr(3X)
 mvaddchstr, mvaddchnstr, mvwaddchstr,/ curs_addchstr(3X)
 mvaddchstr, mvaddchnstr, mvwaddchstr,/ curs_addchstr(3X)
 mvaddnstr, mvwaddnstr, mvwaddnstr: add a/ curs_addstr(3X)
 mvaddnwstr, mvwaddnwstr, mvwaddnwstr: add curs_addwstr(3X)
 mvaddstr, mvaddnstr, mvwaddstr,/ curs_addstr(3X)
 mvaddwch, mvwaddwch, echochar,/ curs_addwch(3X)
 mvaddwchnstr, mvwaddwchstr,/ /addwchnstr, curs_addwchstr(3X)
 mvaddwchstr, mvaddwchnstr, mvwaddwchstr,/ curs_addwchstr(3X)
 mvaddwstr, mvaddnwstr, mvwaddwstr,/ curs_addwstr(3X)
 mvcur, tigetflag, tigetnum, tigestr:/ curs_terminfo(3X)
 mvdelch, mvwdelch: delete character curs_delch(3X)
 mvderwin, dupwin, wsyncup, syncok,/ curs_window(3X)
 mvdir: move a directory mvdir(1M)
 mvgetch, mvwgetch, ungetch: get (or push curs_getch(3X)
 mvgetnstr, mvwgetnstr, mvwgetnstr: get/ curs_getstr(3X)
 mvgetnwstr, mvwgetwstr, mvwgetnwstr: get curs_getwstr(3X)
 mvgetstr, mvgetnstr, mvwgetstr,/ curs_getstr(3X)
 mvgetwch, mvwgetwch, ungetwch: get (or curs_getwch(3X)
 mvgetwstr, mvwgetwstr, mvwgetwstr,/ curs_getwstr(3X)
 mvinch, mvwinch: get a character and its curs_inch(3X)
 mvinchnstr, mvwinchnstr, mvwinchnstr: get curs_inchstr(3X)
 mvinchstr, mvwinchnstr, mvwinchnstr,/ curs_inchstr(3X)
 mvinnstr, mvinnwstr, mvwinnstr: get a/ curs_instr(3X)
 mvinnwstr, mvwinwstr, mvwinnwstr: get a/ curs_inwstr(3X)
 mvinsch, mvwinsch: insert a character curs_insch(3X)
 mvinsnstr, mvwinsnstr, mvwinsnstr: insert/ curs_insstr(3X)
 mvinswstr, mvinsnwstr, mvwinswstr: curs_inswstr(3X)
 mvinsstr, mvinsnstr, mvwinsstr,/ curs_insstr(3X)
 mvinstr, mvinnstr, mvwinstr, mvwinnstr: curs_instr(3X)
 mvinswch, mvwinswch: insert a wchar_t/ curs_inswch(3X)
 mvinswstr, mvinsnwstr, mvwinswstr,/ curs_inswstr(3X)
 mvinwch, mvwinwch: get a wchar_t curs_inwch(3X)
 mvinwchnstr, mvwinwchstr, mvwinwchnstr: curs_inwchstr(3X)
 mvinwchstr, mvinwchnstr, mvwinwchstr,/ curs_inwchstr(3X)
 mvinnwstr, mvinnwstr, mvwinwstr,/ curs_inwstr(3X)
 mvprintw, mvwprintw, vwprintw: print/ curs_printw(3X)
 mvscanw, mvwscanw, vwscanw: convert curs_scanw(3X)
 mvwaddch, echochar, wechochar: add a/ curs_addch(3X)
 mvwaddchnstr: add string of characters/ curs_addchstr(3X)
 mvwaddchnstr: add string of characters/ curs_addchstr(3X)
 mvwaddchstr, mvwaddchnstr: add string of/ curs_addchstr(3X)
 mvwaddchstr, mvwaddchnstr: add string of/ curs_addchstr(3X)
 mvwaddstr: add a string of characters curs_addstr(3X)
 mvwaddnwstr: add a string of wchar_t/ curs_addwstr(3X)
 mvwaddstr, mvwaddnstr: add a string of/ curs_addstr(3X)
 mvwaddwch, echochar, wechochar: add a/ curs_addwch(3X)
 mvwaddwchnstr: add string of wchar_t/ curs_addwchstr(3X)
 mvwaddwchstr, mvwaddwchnstr: add string/ curs_addwchstr(3X)
 mvwaddwstr, mvwaddnwstr: add a string of/ curs_addwstr(3X)
 mvwdelch: delete character under cursor curs_delch(3X)
 mvwgetch, ungetch: get (or push back)/ curs_getch(3X)
 mvwgetnstr: get character strings from curs_getstr(3X)
 mvwgetnwstr: get wchar_t character/ curs_getwstr(3X)
 mvwgetstr, mvwgetnstr: get character/ curs_getstr(3X)
 mvwgetwch, ungetwch: get (or push back)/ curs_getwch(3X)
 mvwgetwstr, mvwgetnwstr: get wchar_t/ curs_getwstr(3X)
 mvwin, subwin, derwin, mvderwin, dupwin, curs_window(3X)
 mvwinch: get a character and its/ curs_inch(3X)
 mvwinchnstr: get a string of characters curs_inchstr(3X)
 mvwinchstr, mvwinchnstr: get a string of/ curs_inchstr(3X)
 mvwinnstr: get a string of characters/ curs_instr(3X)
 mvwinnwstr: get a string of wchar_t/ curs_inwstr(3X)
 mvwinsch: insert a character before the/ curs_insch(3X)
 mvwinsnstr: insert string before/ curs_insstr(3X)
 mvwinswstr: insert wchar_t string/ curs_inswstr(3X)
 mvwinsstr, mvwinsnstr: insert string/ curs_insstr(3X)

/winstr, winnstr, mvinstr, mvinnstr,	mvwinstr, mvwinnstr: get a string of/	curs_instr(3X)
/curs_inswch: inswch, winswch, mvinswch,	mvwinswch: insert a wchar_t character/	curs_inswch(3X)
/winswstr, mvinswstr, mvinswstr,	mvwinswstr, mvwinswstr: insert wchar_t/	curs_inswstr(3X)
/curs_inwch: inwch, winwch, mvinwch,	mvwinwch: get a wchar_t character from a/	curs_inwch(3X)
/mvinwchstr, mvinwchnstr, mvwinwchstr,	mvwinwchnstr: get a string of wchar_t/	curs_inwchstr(3X)
of/ /winwchnstr, mvinwchstr, mvinwchnstr,	mvwinwchstr, mvwinwchnstr: get a string	curs_inwchstr(3X)
/winwstr, winnwstr, mvinwstr, mvinnwstr,	mvwinnwstr: get a string of/	curs_inwstr(3X)
/curs_printw: printw, wprintw, mvprintw,	mvwprintw, vwprintw: print formatted/	curs_printw(3X)
curs_scanw: scanw, wscanw, mvscanw,	mvwscanw, vwscanw: convert formatted/	curs_scanw(3X)
item_description: get menu item	name and description /item_name,	menu_item_name(3X)
id: print the user	name and ID, and group name and ID	id(1)
print the user name and ID, and group	name and ID /id:	id(1)
admresolve: manage DNS resolver's domain	name and nameservers database	admresolve(1M)
/get character login name or user	name associated with effective UID	cuserid(3S)
return the last element of a path	name /basename:	basename(3G)
devnm: device	name	devnm(1M)
the parent directory name of a file path	name /dirname: report	dirname(3G)
tmpnam, tmpnam: create a	name for a temporary file	tmpnam(3S)
/ldgetname: retrieve symbol	name for object file symbol table entry	ldgetname(3X)
ctermid: generate file	name for terminal	ctermid(3S)
descriptor fdetach: detach a	name from a STREAMS-based file	fdetach(3C)
getpw: get	name from UID	getpw(3C)
getenv: return value for environment	name	getenv(3C)
getlogin: get login	name	getlogin(3C)
getsockname: get socket	name	getsockname(2)
nlist: get entries from	name list	nlist(3C)
nm: print	name list of common object file	nm(1)
logname: get login	name	logname(1)
mkstemp: make a unique file	name	mkstemp(3C)
mktemp: make a unique file	name	mktemp(3C)
dirname: report the parent directory	name of a file path name	dirname(3G)
rename: change the	name of a file	rename(2)
ttyname, isatty: find	name of a terminal	ttyname(3C)
getpeername: get	name of connected peer	getpeername(2)
/getdomainname: get	name of current domain	getdomainname(2)
/setdomainname: set	name of current domain	setdomainname(2)
gethostname: get	name of current host	gethostname(2)
sethostname: set	name of current host	sethostname(2)
uname: print	name of current system	uname(1)
unname: get	name of current UNIX system	unname(2)
/ptsname: get	name of the slave pseudo-terminal device	ptsname(3C)
tty: get the	name of the terminal	tty(1)
/nlsprovider: get	name of transport provider	nlsprovider(3N)
logname: return login	name of user	logname(3X)
effective/ cuserid: get character login	name or user name associated with	cuserid(3S)
pwd: print working directory	name	pwd(1)
realpath: returns the real file	name	realpath(3C)
and interpret packets to Internet domain	name servers /dn_expand: make, send,	resolver(3C)
file descriptor to object in file system	name space /attach STREAMS-based	fattach(3C)
bind: bind a	name to a socket	bind(2)
admdefault: provide an interface to	named default sets	admdefault(1M)
pathfind: search for named file in	named directories	pathfind(3G)
pathfind: search for	named file in named directories	pathfind(3G)
manage the remote and restricted shell	names /admrshell:	admrshell(1M)
dirname: deliver portions of path	names /basename,	basename(1)
display a list of all valid group	names /dispgid:	dispgid(1)
display a list of all valid user	names /dispuid:	dispuid(1)
term: conventional	names for terminals	term(5)
ncheck: generate	names from i-numbers	ncheck(1M)
mailalias: translate mail alias	names	mailalias(1)
manage DNS resolver's domain name and	nameservers database /admresolve:	admresolve(1M)
/netdir_sperror: generic transport	name-to-address translation	netdir(3N)
/getsyx, setsyx, ripoffline, curs_set,	napms: low-level curses routines	curs_kernel(3X)
admnls: manipulate	national language variables	admnls(1M)
nl_types:	native language data types	nl_types(5)
processing language	nawk, awk: pattern scanning and	nawk(1)
of file systems processed by fsck and	ncheck /checklist: list	checklist(4)
subsystem	ncheck: generate names from i-numbers	ncheck(1M)
dbm_store, dbm_delete, dbm_firstkey,/	ncsc: AViiON family SCSI adapter	ncsc(7)
database	ndbm: dbm_open, dbm_close, dbm_fetch,	ndbm(3C)
	netconfig: network configuration	netconfig(4)

netdir_getbyaddr, netdir_free,/ netdir_getbyname, netdir_getbyaddr, netdir: netdir_getbyname, netdir_free, netdir_mergeaddr,/ netdir: /netdir_getbyaddr, netdir_free, transport/ /taddr2uaddr, uaddr2taddr, /taddr2uaddr, uaddr2taddr, netdir_perror, /get_myaddress, getnetname, /etc/netconfig entry corresponding to ntohs: convert values between host and /getnetconfig: get netconfi: admnetwork: manage getnetbyname, setnetent, endnetent: get setnetgrent, innetr: get sethostent, endhostent: get /yp_master, yperr_string, ypprot_err: /admpinterface: manage the TCP/IP intro: introduction to listen: /nlsadmin: services/ bcs_cat: type hosts, mkfifo: create a creat: create a groupadd: add (create) a newgrp: log in to a link: create a notify: notify user of the arrival of fork: create a efficient way vfork: spawn useradd: administer a free_field,: create and/ /form_field_new: set_fieldtype_arg,/ /form_fieldtype: file forms /form_new: mkfs, menus items /menu_item_new: menus /menu_new: pechochar, pechowchar: create/ /curs_pad: /form_new_page: set_new_page, panels /panel_new: news: print delscreen:/ /curs_initscr: initscr, mvderwin, dupwin, wsyncup,/ /curs_window: bgets: read stream up to frexp, ldexp, logb, modf, modff, dbminit, fetch, store, delete, firstkey, a directory available for mounting via nfssvc: start an specified socket ftw, priority manage search order for /etc/hosts, /setsrreg, wsetsrreg, scrollok, administration the listener provider service request message file intrflush,/ /curs_inopts: cbreak, dg_mknod: create a file system inode: file halfdelay, intrflush, keypad, meta,	netdir: netdir_getbyname, netdir(3N) netdir_free, netdir_mergeaddr,/ /netdir: netdir(3N) netdir_getbyaddr, netdir_free,/ netdir(3N) netdir_getbyname, netdir_getbyaddr, netdir(3N) netdir_mergeaddr, taddr2uaddr,/ netdir(3N) netdir_perror, netdir_sperror: generic netdir(3N) netdir_sperror: generic transport/ netdir(3N) netname2host, netname2user,/ rpc(3N) netname2user, pmap_getmaps,/ rpc(3N) NETPATH component /getnetpath: get getnetpath(3N) network byte order /htonl, htons, ntohl, byteorder(3N) network configuration database entry getnetconfig(3N) network configuration database netconfig(4) network database admnetwork(1M) network entry /getnetent, getnetbyaddr, getnetent(3N) network group entry /getnetgrent, getnetgrent(3N) network host entry /gethostbyname, gethostent(3N) Network Information Service client/ ypclnt(3N) network interfaces database admpinterface(1M) network library functions intro(3N) network listener server listen(1M) network listener service administration nlsadmin(1M) networks, passwd, protocols, group or new FIFO mkfifo(3C) new file or rewrite an existing one creat(2) new group definition on the system groupadd(1M) new group newgrp(1) new link to a file link(2) new mail notify(1) new process fork(2) new process in a virtual memory vfork(2) new user login on the system useradd(1M) new_field, dup_field, link_field, form_field_new(3X) new_fieldtype, free_fieldtype, form_fieldtype(3X) newform: change the format of a text newform(1) new_form, free_form: create and destroy form_new(3X) newfs: create a file system mkfs(1M) newgrp: log in to a new group newgrp(1) new_item, free_item: create and destroy menu_item_new(3X) new_menu, free_menu: create and destroy menu_new(3X) newpad, subpad, prefetch, pnoutrefresh, curs_pad(3X) new_page: forms pagination form_new_page(3X) new_panel, del_panel: create and destroy panel_new(3X) news items news(1) news: print news items news(1) newterm, endwin, isendwin, set_term, curs_initscr(3X) newwin, delwin, mvwin, subwin, derwin, curs_window(3X) next delimiter bgets(3G) nextafter, scalb: manipulate parts of/ frexp(3C) nextkey: data base subroutines dbm(3X) NFS /exportfs: make exportfs(2) NFS server on a specified socket nfssvc(2) nfssvc: start an NFS server on a nfssvc(2) nftw: walk a file tree ftw(3C) nice: change priority of a process nice(2) nice: run a command at a higher or lower nice(1) NIS, and DNS databases /admsvcorder: admsvcorder(1M) nl: line numbering filter nl(1) nl, nonl: curses terminal output option/ curs_outopts(3X) nlist: get entries from name list nlist(3C) nl_langinfo: language information nl_langinfo(3C) nlsadmin: network listener service nlsadmin(1M) nlsgetcall: get client's data passed via nlsgetcall(3N) nlsprovider: get name of transport nlsprovider(3N) nlsrequest: format and send listener nlsrequest(3N) nl_types: native language data types nl_types(5) nm: print name list of common object nm(1) nocbreak, echo, noecho, halfdelay, curs_inopts(3X) node dg_mknod(2) node structure inode(4) nodelay, notimeout, raw, noraw,/ /noecho, curs_inopts(3X)
--	--

/curs_inopts: cbreak, nocbreak, echo, and quits	noecho, halfdelay, intrflush, keypad,/	curs_inopts(3X)
object file strip: strip	nohup: run a command immune to hangups	nohup(1)
/setscrreg, wsetscrreg, scrollok, nl, setjmp, longjmp:	non-executable information from an	strip(1)
sigsetjmp, siglongjmp: a	nonl: curses terminal output option/	curs_inopts(3X)
used to distinguish prime and	non-local goto	setjmp(3C)
/meta, nodelay, notimeout, raw, noraw, /keypad, meta, nodelay, notimeout, raw, new mail	non-local goto with signal state	sigsetjmp(3C)
/notify:	non-prime days /accounting information	holidays(4)
/intrflush, keypad, meta, nodelay, seed48,/ drand48, erand48, lrand48, deroff: remove	noqiflush, qiflush, timeout, wtimeout,/	curs_inopts(3X)
host and network byte/ htonl, htons, network byte order /htonl, htons, ntohl, null: the	noraw, noqiflush, qiflush, timeout,/	curs_inopts(3X)
/ckpacct, dodisk, lastlogin, monacct, /linenum: line	notify: notify user of the arrival of	notify(1)
/ldlinit, ldlitem: manipulate line object/ /ldlseek, ldnlseek: seek to line	notify user of the arrival of new mail	notify(1)
factor: factor a	notimeout, raw, noraw, noqiflush,/	curs_inopts(3X)
getrpcport: get RPC port	nrand48, mrand48, jrand48, srand48,	drand48(3C)
determine type of floating-point	nroff/troff, tbl, and eqn constructs	deroff(1)
/df: report	ntohl, ntohs: convert values between	byteorder(3N)
can have /getdtablesize: return the	ntohs: convert values between host and	byteorder(3N)
convert string to double-precision	null file	null(7)
ecvt, fcvt, gcvt: convert floating-point	null: the null file	null(7)
nl: line	nulladm, prctmp, prdaily, prtacct,/	acctsh(1M)
/initstate, setstate: generate random	number entries in a common object file	linenum(4)
uniformly distributed pseudo-random	number entries of a common object file/	ldlread(3X)
manipulate parts of floating-point	number entries of a section of a common	ldlseek(3X)
introduction to system calls and error	number	factor(1)
/menu_format: set and get maximum	number	getrpcport(3R)
localeconv: get	number /finite, fpclass, unordered:	isnan(3C)
/uname,	number of free disk blocks and inodes	df(1M)
processing language	number of open files the current process	getdtablesize(2)
att_dump: dump parts of an object or	number /strtod, atof:	strtod(3C)
/close: close an	number to string	ecvt(3C)
dis:	numbering filter	nl(1)
/admsnmpobject: manage the snmpd	numbers better, or change the generator	random(3C)
dlclose: close a shared	numbers /seed48, lcong48: generate	drand48(3C)
dlopen: open a shared	numbers /modf, modff, nextafter, scalb:	frexp(3C)
get the address of a symbol in shared	numbers /intro:	intro(2)
elf:	numbers of rows and columns in menus	menu_format(3X)
cprs: compress a common	numeric formatting information	localeconv(3C)
elf_end: finish using an	nuname: get name of current UNIX system	uname(2)
/elf_getbase: get the base offset for an	oawk: old pattern scanning and	oawk(1)
dlopen, ldaopen: open an	object archive file	att_dump(1)
cof2elf: translate	object associated with a file descriptor	close(2)
line number entries of a common	object code disassembler	dis(1)
elf32_newehdr: retrieve class-dependent	object database	admsnmpobject(1M)
ldclose, ldaclose: close a common	object	dlclose(3X)
read the file header of a common	object	dlopen(3X)
number entries of a section of a common	object /dlsym:	dlsym(3X)
seek to the optional file header of an	object file access library	elf(3E)
entries of a section of a common	object file	cprs(1)
section header of a common	object file	elf_end(3E)
to an indexed/named section of a common	object file	elf_getbase(3E)
index of symbol table entry of an	object file for reading	ldopen(3X)
read an indexed symbol table entry of an	object file from COFF to ELF	cof2elf(1)
ldtbseek: seek to the symbol table of an	object file function /manipulate	ldlread(3X)
linenum: line number entries in a common	object file header /elf32_getehdr,	elf_getehdr(3E)
manipulate the comment section of an	object file	ldclose(3X)
nm: print name list of common	object file /ldfread:	ldfread(3X)
relocation information for a common	object file /ldnlseek: seek to line	ldlseek(3X)
strip non-executable information from an	object file /ldohseek:	ldohseek(3X)
ldgetname: retrieve symbol name for	object file /seek to relocation	ldrseek(3X)
syms: common	object file /read an indexed/named	ldshread(3X)
elf32_fsize: return the size of an	object file /ldsseek, ldnssseek: seek	ldsseek(3X)
	object file /ldtbindex: compute	ldtbindex(3X)
	object file /ldtbread:	ldtbread(3X)
	object file	ldtbseek(3X)
	object file	linenum(4)
	object file. /mcs:	mcs(1)
	object file	nm(1)
	object file /reloc:	reloc(4)
	object file /strip:	strip(1)
	object file symbol table entry	ldgetname(3X)
	object file symbol table format	syms(4)
	object file type /elf_fsize:	elf_fsize(3E)

filehdr: file header for common	object files	filehdr(4)
ld: link editor for common	object files	ld(1)
size: print section sizes of	object files	ld-coff(1)
/attach STREAMS-based file descriptor to	object files	size(1)
lorder: find ordering relation for an	object in file system name space	fattach(3C)
att_dump: dump parts of an	object library	lorder(1)
find the printable strings in an	object or object archive file	att_dump(1)
lseek: change	object or other binary file /strings:	strings(1)
read: read from an	object pointer's current position	lseek(2)
write: write to an	object	read(2)
index: search for the first	object	write(2)
rindex: search for the last	occurrence of a character in a string	index(3C)
od:	occurrence of a character in a string	rindex(3C)
	octal dump	od(1)
	od: octal dump	od(1)
/data_behind: tell if forms field has	off-screen data ahead or behind	form_data(3X)
/elf_getbase: get the base	offset for an object file	elf_getbase(3E)
offsetof:	offset of structure member	offsetof(3C)
	offsetof: offset of structure member	offsetof(3C)
language oawk:	old pattern scanning and processing	oawk(1)
/invert, rpow, msqrt, mcmp, move, min,	omout, fmout, m_out, sdiv, itom:/ /msqrt,	mp(3X)
mcmp, move, min, omin, fmin, m_in, mout,	Once Read Multiple optical device) as	mp(3X)
magtape/ /wmt: pseudo WORM (Write	one-line summary about a topic	wmt(7)
whatis: display a	onto a specific descriptor	whatis(1)
dup2: duplicate an open file descriptor	onto input stream	dup2(2)
ungetc: push character back	onto VSC synchronous controller	ungetc(3S)
/download board resident software	open a shared object	vsload(1M)
dlopen:	open a stream	dlopen(3X)
fopen, freopen, fdopen:	open an object file for reading	fopen(3S)
ldopen, ldaopen:	open any minor device on a STREAMS	ldopen(3X)
driver clone:	open, close pipes to and from a command	clone(7)
/p2open, p2close:	open DG/UX file /dg_flock:	p2open(3G)
apply or remove an advisory lock on an	open file descriptor	dg_flock(3C)
descriptor dup: duplicate an	open file descriptor onto a specific	dup(2)
descriptor dup2: duplicate an	open file for reading or writing	dup2(2)
open:	open files the current process can have	open(2)
/getdtablesize: return the number of	open: open file for reading or writing	getdtablesize(2)
	open/close a message catalogue	open(2)
catopen, catclose:	opendir, readdir, telldir, seekdir,	catopen(3C)
rewinddir, closedir:/ directory:	openlog, closelog, setlogmask: control	directory(3X)
system log /syslog,	operable /vscheck: verify	syslog(3C)
that the VSC synchronous controller is	operating system clients	vscheck(1M)
admclient: manage	operating system console pseudo-device	admclient(1M)
/syscon, console, systty: DG/UX	operating system profiler	syscon(7)
prf:	operating system profiler	prf(7)
prfld, prfstat, prfdc, prfsnap, prfpr:	operating system	profiler(1M)
reboot: restart the	operation	reboot(1M)
msgsys: perform a message queue	operation routines /tgetflag, tgetstr,	msgsys(2)
tgoto, tputs: terminal independent	operation	termcap(3X)
semsys: perform a semaphore	operation	semsys(2)
shmsys: perform a shared memory	operation	shmsys(2)
/wstok, wstokstr, strtows: wchar_t string	operations and type transformation	wstring(3W)
seekdir, rewinddir, closedir: directory	operations /opendir, readdir, telldir,	directory(3X)
dkctl: control special disk	operations	dkctl(1M)
ether_line: Ethernet address mapping	operations /ether_hostton,	ethers(3N)
memcmp, memcpy, memmove, memset: memory	operations /memory: memccpy, memchr,	memory(3C)
semctl: semaphore control	operations	semctl(2)
semop: semaphore	operations	semop(2)
shmctl: shared memory control	operations	shmctl(2)
strspn, strcspn, strtok, strtr: string	operations /strchr, strchr, strpbrk,	string(3C)
join: relational database	operator	join(1)
/pseudo WORM (Write Once Read Multiple	optical device) as magtape interface	wmt(7)
courses: CRT screen handling and	optimization package	courses(3X)
typeahead: curses terminal input	option control routines /wtimeout,	curl_inopts(3X)
nl, nonl: curses terminal output	option control routines /scrollk,	curl_outopts(3X)
getopt: get	option letter from argument vector	getopt(3C)
field_opts_off, field_opts: forms field	option routines /field_opts_on,	form_field_opts(3X)
form_opts_off, form_opts: forms	option routines /form_opts_on,	form_opts(3X)
item_opts_off, item_opts: menus item	option routines /item_opts_on,	menu_item_opts(3X)
menu_opts_off, menu_opts: menus	option routines /menu_opts_on,	menu_opts(3X)
/ldohseek: seek to the	optional file header of an object file	ldohseek(3X)

processor(s) reboot: reboot halts and optionally reboots the system reboot(2)
 fcntl: file control options fcntl(5)
 stty: set the options for a terminal stty(1)
 t_optmngmt: manage options for a transport endpoint t_optmngmt(3N)
 getopt: parse command options getopt(1)
 getopts, getoptcv: parse command options getopts(1)
 getsockopt: get options on a socket getsockopt(2)
 setsockopt: set options on sockets setsockopt(2)
 administrative shutdown and reboot options /uadmin: request uadmin(2)
 values between host and network byte order /htons, ntohl, ntohs: convert byteorder(3N)
 databases /admsvcorder: manage search order for /etc/hosts, NIS, and DNS admsvcorder(1M)
 postreverse: reverse the page order in a PostScript file postreverse(1)
 /rev: reverse order of characters in each line of file rev(1)
 /lorder: find ordering relation for an object library lorder(1)
 t_rcvrel: acknowledge receipt of an orderly release indication t_rcvrel(3N)
 t_sndrel: initiate an orderly release t_sndrel(3N)
 filesystem: file system organization filesystem(7)
 administration program osysadm: menu-driven system osysadm(1M)
 dial: establish an out-going terminal line connection dial(3C)
 a.out: assembler and link editor output a.out(4)
 concatenate and type files to standard output /cat: cat(1)
 fold: fold long lines for finite width output device fold(1)
 mvwprintw, vprintw: print formatted output in curses windows /mvprintw, curs_printw(3X)
 /vfprintf, vsprintf: print formatted output of a variable argument list vprintf(3S)
 /vfprintf, vsprintf: print formatted output of a variable argument list vprintf(3W)
 /scrollok, nl, nonl: curses terminal output option control routines curs_outopts(3X)
 printf: print formatted output printf(1)
 fprintf, sprintf: print formatted output /printf, printf(3S)
 fprintf, sprintf: print formatted output /printf, printf(3W)
 sysdef: output system definition sysdef(1M)
 ttyadm: format and output TTY port monitor information ttyadm(1M)
 windows /overlay, overwrite, copywin: overlap and manipulate overlapped curses curs_overlay(3X)
 copywin: overlap and manipulate overlapped curses windows /overwrite, curs_overlay(3X)
 manipulate overlapped/ /curs_overlay: overlay, overwrite, copywin: overlap and curs_overlay(3X)
 /acctdisk, acctdusg, accton, acctwtmp: overview of accounting and miscellaneous/ acct(1M)
 manipulate/ /curs_overlay: overlay, overwrite, copywin: overlap and curs_overlay(3X)
 chown: change file owner chown(1)
 chgrp: change the group ownership of a file chgrp(1)
 command /p2open, p2close: open, close pipes to and from a p2open(3G)
 and from a command p2open, p2close: open, close pipes to p2open(3G)
 files pack, pcat, unpack: compress and expand pack(1)
 pkginfo: package characteristics file pkginfo(4)
 pkgmap: package contents description file pkgmap(4)
 CRT screen handling and optimization package /curses: curses(3X)
 pkgtrans: translate package format pkgtrans(1)
 forms: character based forms package forms(3X)
 pkgrm: removes a package from the system pkgrm(1M)
 prototype: package information file prototype(4)
 pkginfo: display software package information pkginfo(1)
 menus: character based menus package menus(3X)
 panels: character based panels package panels(3X)
 pkgparam: displays package parameter values pkgparam(1)
 pkgmk: produce an installable package pkgmk(1)
 sa1, sa2, sadc: system activity report package /sar: sar(1M)
 stdio: standard buffered input/output package stdio(3S)
 standard interprocess communication package /stdipc: ftok: stdipc(3C)
 pkgadd: transfer software package to the system pkgadd(1M)
 admpackage: manage DG/UX-style software packages admpackage(1M)
 pckt: STREAMS Packet Mode module pckt(7)
 /dn_expand: make, send, and interpret packets to Internet domain name servers resolver(3C)
 pechowchar: create and display curses pads /prefresh, pnoutrefresh, pechowchar, curs_pad(3X)
 field_index: set forms current page and field /current_field, form_page(3X)
 time more, page: display file one screenful at a more(1)
 postreverse: reverse the page order in a PostScript file postreverse(1)
 getpagesize: get the system page size getpagesize(2)
 determine residency of memory pages /dg_paging_info: dg_paging_info(2)
 mlock, munlock: lock (or unlock) pages in memory mlock(3C)
 mincore: determine residency of memory pages mincore(2)
 mmap: map pages of memory mmap(2)
 munmap: unmap pages of memory munmap(2)
 set_new_page, new_page: forms pagination /form_new_page: form_new_page(3X)

specify additional devices for system
 swapon: add a swap device for demand
 socketpair: create a
 master/slave pseudo-device
 unlock a pseudo-terminal master/slave
 /can_change_color, color_content,
 associate application data with a panels
 or set the current window of a panels
 panels deck traversal primitives
 traversal primitives /panel_above:
 primitives /panel_above: panel_above,
 /panel_show: show_panel, hide_panel,
 window on the virtual screen
 and destroy panels

 /show_panel, hide_panel, panel_hidden:
 panel_top: top_panel, bottom_panel:
 /panel_above: panel_above, panel_below:
 panels: character based
 associate application data with a
 get or set the current window of a
 new_panel, del_panel: create and destroy
 /panel_update: update_panels:
 panel_move: move_panel: move a
 panel_hidden: panels deck manipulation/
 panels deck manipulation routines
 virtual screen refresh routine
 panel_userptr: associate application/
 data/ /panel_userptr: set_panel_userptr,
 replace_panel: get or set the current/
 the current window of a /panel_window:
 pkgparam: displays package
 /admtcpipparams: manage the TCP/IP host
 display or modify boot and dump
 admtape: manipulate the default
 tkey: set label and data translation
 name dirname: report the
 getpgid: get process, process group, and
 getopt: get
 getopt: getopt:
 getopts, getoptcv: getsubopt:
 clrtoeol, wclrtoeol: clear all or
 tail: deliver the last
 shutdown: shut down
 file att_dump: dump
 modff, nextafter, scalb: manipulate
 reference table from C, Fortran and
 putmsg, putpmsg:
 nlsgetcall: get client's data

 bcs_cat: type hosts, networks,
 /crypt:
 admuser: manage user information in the
 dialups: devices requiring a dial-up
 setpwfile, fgetpwent: manipulate
 lckpwdf, ulckpwdf: manipulate shadow
 putpwent: write
 putspent: write shadow
 passwd:
 vipw: edit the system
 passmgmt:
 getpass: read a
 pwck, grpck: check
 passwd: change login
 d_passwd: log-in programs and

 rmdirp: create, remove directories in a
 basename: return the last element of a

paging /swapon: swapon(1M)
 paging swapon(2)
 pair of connected sockets socketpair(2)
 pair /pty, pts, ptc: pseudo-terminal pty(7)
 pair /unlockpt: unlockpt(3C)
 pair_content: curses color manipulation/ curs_color(3X)
 panel /set_panel_userptr, panel_userptr: panel_userptr(3X)
 panel /panel_window, replace_panel: get panel_window(3X)
 panel_above: panel_above, panel_below: panel_above(3X)
 panel_above, panel_below: panels deck panel_above(3X)
 panel_below: panels deck traversal panel_above(3X)
 panel_hidden: panels deck manipulation/ panel_show(3X)
 panel_move: move_panel: move a panels panel_move(3X)
 panel_new: new_panel, del_panel: create panel_new(3X)
 panels: character based panels package panels(3X)
 panels deck manipulation routines panel_show(3X)
 panels deck manipulation routines panel_top(3X)
 panels deck traversal primitives panel_above(3X)
 panels package panels(3X)
 panels panel /panel_userptr: panel_userptr(3X)
 panels panel /replace_panel: panel_window(3X)
 panels /panel_new: panel_new(3X)
 panels virtual screen refresh routine panel_update(3X)
 panels window on the virtual screen panel_move(3X)
 panel_show: show_panel, hide_panel, panel_show(3X)
 panel_top: top_panel, bottom_panel: panel_top(3X)
 panel_update: update_panels: panels panel_update(3X)
 panel_userptr: set_panel_userptr, panel_userptr(3X)
 panel_userptr: associate application panel_userptr(3X)
 panel_window: panel_window, panel_window(3X)
 panel_window, replace_panel: get or set panel_window(3X)
 parameter values pkgparam(1)
 parameters admtcpipparams(1M)
 parameters /dg_sysctl: dg_sysctl(1M)
 parameters for tapes admtape(1M)
 parameters tkey(1)
 parent directory name of a file path dirname(3G)
 parent process IDs /getpgrp, getppid, getpid(2)
 parent process-id getppid(2)
 parse command options getopt(1)
 parse command options getopts(1)
 parse suboptions from a string getsubopt(3C)
 part of a curses window /wclrtoeol, curs_clear(3X)
 part of a file tail(1)
 part of a full-duplex connection shutdown(2)
 parts of an object or object archive att_dump(1)
 parts of floating-point numbers /modf, frexp(3C)
 Pascal sources /xref: generate cross xref(1)
 pass a message down a stream putmsg(2)
 passed via the listener nlsgetcall(3N)
 passmgmt: password files management passmgmt(1M)
 passwd: change login password passwd(1)
 passwd: password file passwd(4)
 passwd, protocols, group or services/ bcs_cat(1M)
 password and file encryption functions crypt(3X)
 password database admuser(1M)
 password. dialups(4)
 password file entry /setpwent, endpwent, getpwent(3C)
 password file entry /fgetspent, getspent(3C)
 password file entry putpwent(3C)
 password file entry putspent(3C)
 password file passwd(4)
 password file vipw(1M)
 password files management passmgmt(1M)
 password getpass(3C)
 password or group file pwck(1M)
 password passwd(1)
 passwords for dial-up devices d_passwd(4)
 paste: merge lines paste(1)
 path /mkdirp, mkdirp(3G)
 path name basename(3G)

the parent directory name of a file
 basename, dirname: deliver portions of
 pathname values
 directories
 display a prompt; verify and return a
 getwd: get current working directory
 getcwd: get
 pathconf, fpathconf: get configurable
 grep: search a file for a
 menu_pattern: set and get menus
 gmatch: shell global
 /nawk, awk:
 /oawk: old
 /egrep: search a file for a
 caught
 /pack,
 /popen,
 value/ machid: dghost, m68k, m88k, i386,
 /newpad, subpad, prefresh, pnoutrefresh,
 pads /prefresh, pnoutrefresh, pechochar,
 getpeername: get name of connected
 sigpending: examine
 lock database, start lock reclaim grace
 uucheck: check the uucp directories and
 mesg:
 acct:
 acctcms: command summary from
 screenful at a time
 diskman: menu interface for managing
 msync: synchronize memory with
 split: split a file into
 tee:
 popen, pclose: initiate
 kbdpipe: use the KBD module in a
 p2open, p2close: open, close
 unix_ipc:
 system
 script
 information
 file
 values
 interface
 memory
 postplot: PostScript translator for
 /getnetname, netname2host, netname2user,
 pmap_unset, /netname2user, pmap_getmaps,
 /pmap_getmaps, pmap_getport,
 /pmap_getport, pmap_rmtcall,
 /pmap_getport, pmap_rmtcall,
 pmap_set, pmap_rmtcall, pmap_set,
 pmap_unset, registerrpc, svc_destroy,/
 pnoutrefresh, pechochar, pechowchar:/
 curs_pad: newpad, subpad, prefresh,
 view the allocation limits for a control
 change the resource limits of a control
 elf_strptr: make a string
 fseek, rewind, ftell: reposition a file
 fsetpos, fgetpos: reposition a file
 lseek: change object
 alp: Algorithm
 process
 pmap:
 ttyadm: format and output TTY
 path name /dirname: report dirname(3G)
 path names basename(1)
 pathconf, fpathconf: get configurable pathconf(2)
 pathfind: search for named file in named pathfind(3G)
 pathname /ckpath: ckpath(1)
 pathname getwd(3C)
 pathname of current working directory getcwd(3C)
 pathname values pathconf(2)
 pattern grep(1)
 pattern match buffer /set_menu_pattern, menu_pattern(3X)
 pattern matching gmatch(3G)
 pattern scanning and processing language nawk(1)
 pattern scanning and processing language oawk(1)
 pattern using full regular expressions egrep(1)
 pause: suspend process until a signal is pause(2)
 pcat, unpack: compress and expand files pack(1)
 pckt: STREAMS Packet Mode module pckt(7)
 pclose: initiate pipe to/from a process popen(3S)
 pdp11, u3b, u3b5, vax: provide truth machid(1)
 pechochar, pechowchar: create and/ curs_pad(3X)
 pechowchar: create and display curses curs_pad(3X)
 peer getpeername(2)
 pending signals sigpending(2)
 period /dg_lock_reset: reset remote file dg_lock_reset(2)
 permissions file uucheck(1M)
 permit or deny messages mesg(1)
 per-process accounting file format acct(4)
 per-process accounting records acctcms(1M)
 perror: print system error messages perror(3C)
 pg: display file forward or backward one pg(1)
 physical and logical disks diskman(1M)
 physical storage msync(3C)
 pieces split(1)
 pipe: create an interprocess channel pipe(2)
 pipe fitting tee(1)
 pipe to/from a process popen(3S)
 pipeline kbdpipe(1)
 pipes to and from a command p2open(3G)
 piping communications within a host unix_ipc(6F)
 pkgadd: transfer software package to the pkgadd(1M)
 pkgask: stores answers to a request pkgask(1M)
 pkgchk: check accuracy of installation pkgchk(1M)
 pkginfo: display software package pkginfo(1)
 pkginfo: package characteristics file pkginfo(4)
 pkgmap: package contents description pkgmap(4)
 pkgmk: produce an installable package pkgmk(1)
 pkgparam: displays package parameter pkgparam(1)
 pkgproto: generate a prototype file pkgproto(1)
 pkgrm: removes a package from the system pkgrm(1M)
 pkgtrans: translate package format pkgtrans(1)
 plm: pseudo lock manager device plm(7)
 plock: lock data, text, or both into plock(2)
 plot(4) graphics files postplot(1)
 pmadm: port monitor administration pmadm(1M)
 pmap_getmaps, pmap_getport, / rpc(3N)
 pmap_getport, pmap_rmtcall, pmap_set, rpc(3N)
 pmap_rmtcall, pmap_set, pmap_unset, / rpc(3N)
 pmap_set, pmap_unset, registerrpc, / rpc(3N)
 pmap_unset, registerrpc, svc_destroy, / rpc(3N)
 pnoutrefresh, pechochar, pechowchar: / curs_pad(3X)
 point directory /cpd: change or cpd(1)
 point directory /dg_set_cpd_limits: dg_set_cpd_limits(2)
 pointer elf_strptr(3E)
 pointer in a stream fseek(3S)
 pointer in a stream fsetpos(3C)
 pointer's current position lseek(2)
 poll: input/output multiplexing poll(2)
 Pool management module alp(7)
 popen, pclose: initiate pipe to/from a popen(3S)
 port monitor administration pmadm(1M)
 port monitor information ttyadm(1M)

/admportservice: manage	port monitor services	admportservice(1M)
/admportmonitor: manage	port monitors	admportmonitor(1M)
getrpcport: get RPC	port number	getrpcport(3R)
ar: archive and library maintainer for	portable archives	ar(1)
bzero: zero a	portion of memory	bzero(3C)
basename, dirname: deliver	portions of path names	basename(1)
/admterminal: manage terminal	ports	admterminal(1M)
maintain line and hunt settings for TTY	ports /sttydefs:	sttydefs(1M)
ttymon: monitor terminal	ports	ttymon(1M)
cursor /form_cursor:	pos_form_cursor: position forms window	form_cursor(3X)
/menu_cursor: pos_menu_cursor: correctly	position a menu cursor	menu_cursor(3X)
/form_cursor: pos_form_cursor:	position forms window cursor	form_cursor(3X)
lseek: change object pointer's current	position	lseek(2)
tposn:	position tape to specified file	tposn(1)
menus cursor /menu_cursor:	pos_menu_cursor: correctly position a	menu_cursor(3X)
Diablo 630 files	postdaisy: PostScript translator for	postdaisy(1)
bitmap files	postdmd: PostScript translator for DMD	postdmd(1)
banner: make	posters	banner(1)
forms from associated/ /form_post:	post_form, unpost_form: write or erase	form_post(3X)
printers	postio: serial interface for PostScript	postio(1)
PostScript printers	postmd: matrix display program for	postmd(1)
menus from associated/ /menu_post:	post_menu, unpost_menu: write or erase	menu_post(3X)
plot(4) graphics files	postplot: PostScript translator for	postplot(1)
PostScript	postprint: translate text files into	postprint(1)
dpost: troff	postprocessor for PostScript printers	dpost(1)
PostScript file	postreverse: reverse the page order in a	postreverse(1)
postreverse: reverse the page order in a	PostScript file	postreverse(1)
download: download host resident	PostScript fonts	download(1)
postprint: translate text files into	PostScript	postprint(1)
dpost: troff postprocessor for	PostScript printers	dpost(1)
postio: serial interface for	PostScript printers	postio(1)
postmd: matrix display program for	PostScript printers	postmd(1)
files postdaisy:	PostScript translator for Diablo 630	postdaisy(1)
files postdmd:	PostScript translator for DMD bitmap	postdmd(1)
graphics files postplot:	PostScript translator for plot(4)	postplot(1)
files /posttek:	PostScript translator for tektronix 4014	posttek(1)
tektronix 4014 files	posttek: PostScript translator for	posttek(1)
move, min,/ mp: madd, msub, mult, mdiv,	pow, gcd, invert, rpow, msqrt, mcomp,	mp(3X)
/expf, cbprt, log, logf, log10, log10f,	pow, powf, sqrt, sqrtf: exponential,/	exp(3M)
sqrt, sqrtf: exponential, logarithm,	power, square root functions /pow, powf,	exp(3M)
cbprt, log, logf, log10, log10f, pow,	powf, sqrt, sqrtf: exponential,/ /expf,	exp(3M)
	pr: print files	pr(1)
/dodisk, lastlogin, monacct, nulladm,	prctmp, prdaily, prtacct, shutacct,/	acctsh(1M)
/lastlogin, monacct, nulladm, prctmp,	prdaily, prtacct, shutacct, startup,/	acctsh(1M)
fmout, m_out, sdiv, itom: multiple	precision integer arithmetic /omout,	mp(3X)
pechowchar:/ curs_pad: newpad, subpad,	prefresh, pnoutrefresh, pechochar,	curs_pad(3X)
monitor:	prepare execution profile	monitor(3C)
sifilter:	preprocess MC88100 assembly language	sifilter(1)
cpp: the C language	preprocessor	cpp(1)
signal: specify what to do upon	presentation of a signal	signal(2)
sigset: specify what to do upon	presentation of a signal	sigset(2)
sigvec: specify what to do upon	presentation of a signal	sigvec(2)
unset: undo a	previous get of an SCCS file	unset(1)
complete /dg_lock_wait: wait for	previously delayed lock requests to	dg_lock_wait(2)
	prf: operating system profiler	prf(7)
profiler /prfld, prfstat,	prfdc, prfsnap, prfpr: operating system	profiler(1M)
operating system profiler	prfld, prfstat, prfdc, prfsnap, prfpr:	profiler(1M)
prfld, prfstat, prfdc, prfsnap,	prfpr: operating system profiler	profiler(1M)
profiler prfld, prfstat, prfdc,	prfsnap, prfpr: operating system	profiler(1M)
operating system profiler prfld,	prfstat, prfdc, prfsnap, prfpr:	profiler(1M)
information used to distinguish	prime and non-prime days /accounting	holidays(4)
types:	primitive system data types	types(5)
panel_below: panels deck traversal	primitives /panel_above: panel_above,	panel_above(3X)
Server /termprinter:	print a file using the 40014A Terminal	termprinter(1)
/extended_perror:	print an error message to standard error	extended_perror(3C)
prs:	print an SCCS file	prs(1)
date:	print and set the date	date(1)
cal:	print calendar	cal(1)
/sum:	print checksum and block count of a file	sum(1)
development environment/ sde-target:	print commands to reset software	sde-target(1)
/sact:	print current SCCS file editing activity	sact(1)

/man: locate and	print entries from the reference manuals	man(1)
pr:	print files	pr(1)
/wprintw, mvprintw, mvwprintw, vwprintw:	print formatted output in curses windows	curs_printw(3X)
argument/ vprintf, vfprintf, vsprintf:	print formatted output of a variable	vprintf(3S)
argument/ vprintf, vfprintf, vsprintf:	print formatted output of a variable	vprintf(3W)
printf:	print formatted output	printf(1)
printf, fprintf, sprintf:	print formatted output	printf(3S)
printf, fprintf, sprintf:	print formatted output	printf(3W)
the LP print service lpstat:	print information about the status of	lpstat(1)
about RCS files /rlog:	print log messages and other information	rlog(1)
catgets:	print message from message catalog	catgets(1)
/nm:	print name list of common object file	nm(1)
uname:	print name of current system	uname(1)
news:	print news items	news(1)
infocmp: compare or	print out TERMINFO descriptions	infocmp(1M)
printenv:	print out the environment	printenv(1)
acctcom: search and	print process accounting file(s)	acctcom(1)
accept, reject: accept or reject	print requests	accept(1M)
/lpr: send	print requests to a line printer spooler	lpr(1)
size:	print section sizes of object files	size(1)
/lpshut, lpmove: start/stop the LP	print service and move requests	lpsched(1M)
cancel: send/cancel requests to an LP	print service /lp,	lp(1)
lpadmin: configure the LP	print service	lpadmin(1M)
administer filters used with the LP	print service /lpfilter:	lpfilter(1M)
administer forms used with the LP	print service /lpforms:	lpforms(1M)
information about the status of the LP	print service /lpstat: print	lpstat(1)
register remote systems with the	print service /lpssystem:	lpssystem(1M)
strace:	print STREAMS trace messages	strace(1M)
perror:	print system error messages	perror(3C)
name and ID /id:	print the user name and ID, and group	id(1)
pwd:	print working directory name	pwd(1)
binary file /strings: find the	printable strings in an object or other	strings(1)
printcap:	printer capability data base	printcap(5)
terminfo: terminal and	printenv: print out the environment	printenv(1)
lpc: line	printer capability database	terminfo(4)
Server /lptermprinter: start	printer control program	lpc(1M)
lp: DGC AViiON family line	printer session with 40014A Terminal	lptermprinter(1)
lpd: line	printer special files	lp(7)
lpr: send print requests to a line	printer spooler	lpd(1M)
lprm: remove jobs from the line	printer spooler	lpr(1)
troff postprocessor for PostScript	printer spooling queue	lprm(1)
enable, disable: enable/disable LP	printers /dpost:	dpost(1)
postio: serial interface for PostScript	printers	enable(1)
matrix display program for PostScript	printers	postio(1)
formatted output	printers /postmd:	postmd(1)
formatted output	printf, fprintf, sprintf: print	printf(3S)
lpusers: set	printf, fprintf, sprintf: print	printf(3W)
vwprintw: print formatted/ /curs_printw:	printf: print formatted output	printf(1)
lpusers: set printing queue	printing queue priorities	lpusers(1M)
getpriority: get process scheduling	printw, wprintw, mvprintw, mvwprintw,	curs_printw(3X)
nice: run a command at a higher or lower	priorities	lpusers(1M)
nice: change	priority	getpriority(2)
renice: alter	priority	nice(1)
setpriority: set process scheduling	priority of a process	nice(2)
probedev:	priority of running processes	renice(1)
library routines for remote	priority	setpriority(2)
shutacct, startup, turnacct: shell	probe system for devices	probedev(1M)
introduction to system maintenance	probedev: probe system for devices	probedev(1M)
filehandle dg_lcntl:	procedure calls /xpvt_unregister:	rpc(3N)
acct: enable or disable	procedures for accounting /prtacct,	acctsh(1M)
acctprc1, acctprc2:	procedures /intro:	intro(8)
acctcom: search and print	process a record lock request on a	dg_lcntl(2)
alarm: set a	process accounting	acct(2)
times: get	process accounting	acctprc(1M)
kill: terminate a	process accounting file(s)	acctcom(1)
the number of open files the current	process alarm clock	alarm(2)
the working directory of the calling	process and child process times	times(2)
	process by default	kill(1)
	process can have /getdtablesize: return	getdtablesize(2)
	process /chdir: change	chdir(2)

change the root directory of the calling process	/chroot:	chroot(2)
init, telinit: process control initialization	init(1M)
time: time a command; report process data and system activity	timex(1)
the extended errno for the current process	/dg_ext_errno: return	dg_ext_errno(2)
dg_kill: test for or terminate a process	dg_kill(1)
exit, _exit: terminate process	exit(2)
the working directory of the calling process	/fchdir: change	fchdir(2)
fork: create a new process	fork(2)
/getpgrp, getppid, getpgid: get process, process group, and parent process IDs	getpid(2)
getpgrp2: get process group	getpgrp2(2)
setpgid: set process group ID for job control	setpgid(2)
getpgrp: get process group ID	getpgrp(2)
setsid: create session and set process group ID	setsid(2)
tcsetpgrp: set terminal foreground process group id	tcsetpgrp(3C)
killpg: send signal to a process or a process group	killpg(2)
/get file usage information for process identified by process key	dg_file_info(2)
get process, process group, and parent process IDs	/getpgrp, getppid, getpgid:	getpid(2)
vfork: spawn new process in a virtual memory efficient	vfork(2)
information for process identified by process key	/get file usage	dg_file_info(2)
kill: send a signal to a process	kill(2)
ulimit: get or set process limits	ulimit(2)
nice: change priority of a process	nice(2)
sigsend, sigsendset: send a signal to a process or a group of processes	sigsend(2)
killpg: send signal to a process or a process group	killpg(2)
popen, pclose: initiate pipe to/from a process	popen(3S)
getpid, getpgrp, getppid, getpgid: get process, process group, and parent/	getpid(2)
set up execution time profiling for a process	/profil:	profil(2)
getpriority: get process scheduling priority	getpriority(2)
setpriority: set process scheduling priority	setpriority(2)
the effective group id of the current process	/setgid: set	setgid(2)
set the effective user id of the current process	/setuid:	setuid(2)
signal frame sigret: restore the process state to that contained in a	sigret(2)
ps: report process status	ps(1)
admlock: manage simple process synchronization	admlock(1M)
wait, waitpid: wait for process termination	wait(2)
times: get process and child process times	times(2)
waitid: wait for child process to change state	waitid(2)
wait3: wait for child process to stop or terminate	wait3(2)
wait4: wait for the specified child process to stop or terminate	wait4(2)
dg_xtrace: extended process trace	dg_xtrace(2)
ptrace: process trace	ptrace(2)
/set blocked signals and suspend process until a signal is caught	berk_sigpause(2)
pause: suspend process until a signal is caught	pause(2)
/clear a blocked signal and suspend the process until a signal is caught	sigpause(2)
wait: await completion of process	wait(1)
checklist: list of file systems processed by fsck and ncheck	checklist(4)
admprocess: manage processes	admprocess(1M)
/dg_allow_shared_descriptor_attach: let processes attach shared descriptor array	dg_allow_shared_descriptor_attach(2)
about the system's currently active processes	/get information	dg_process_info(2)
killall: kill all active processes	killall(1M)
renice: alter priority of running processes	renice(1)
send a signal to a process or a group of processes	/sigsend, sigsendset:	sigsend(2)
/fuser: identify processes using a file or file structure	fuser(1M)
setpgrp: set process-group-id	setpgrp(2)
setpgrp2: set process-group-id	setpgrp2(2)
getppid: get parent process-id	getppid(2)
nawk, awk: pattern scanning and processing language	nawk(1)
oawk: old pattern scanning and processing language	oawk(1)
mailx: interactive message processing system	mailx(1)
/form_driver: command processor for the forms subsystem	form_driver(3X)
/menu_driver: command processor for the menus subsystem	menu_driver(3X)
grfx: AViiON series workstation graphics processor	grfx(7)
halt: stop the system processor	halt(1M)
m4: macro processor	m4(1)
return the current contents of the processor status register	/getpsr:	getpsr(2)
setpsr: set the processor status register	setpsr(2)
vax: provide truth value about your processor type	/i386, pdp11, u3b, u3b5,	machid(1)
halts and optionally reboots the system processor(s)	/reboot: reboot	reboot(2)
sighold: add a signal to the calling process's set of blocked signals	sighold(2)
/remove a signal from the calling process's set of blocked signals	sigrelse(2)
/attach another process's shared descriptor array	dg_attach_to_shared_descriptors(2)

pkgmk:	produce an installable package	pkgmk(1)
t_error:	produce error message	t_error(3N)
	prof: display profile data	prof(1)
	prof: profile within a function	prof(5)
for a process	profil: set up execution time profiling	profil(2)
prof: display	profile data	prof(1)
monitor: prepare execution	profile	monitor(3C)
login time	profile: setting up an environment at	profile(4)
prof:	profile within a function	prof(5)
prf: operating system	profiler	prf(7)
prfdc, prfsnap, prfpr: operating system	profiler /prfld, prfstat,	profiler(1M)
profil: set up execution time	profiling for a process	profil(2)
assert: verify	program assertion	assert(3X)
cb: C	program beautifier	cb(1)
lint: a C	program checker	lint(1)
cxref: generate C	program cross-reference	cxref(1)
cscope: interactively examine a C	program	cscope(1)
end, etext, edata: last locations in	program	end(3C)
which: locate a	program file for csh(1) users	which(1)
postmd: matrix display	program for PostScript printers	postmd(1)
uucico: file transport	program for the uucp system	uucico(1M)
elf32_newphdr: retrieve class-dependent	program header table /elf32_getphdr,	elf_getphdr(3E)
lpc: line printer control	program	lpc(1M)
main: enter a C main	program	main(3C)
catgets: read a	program message	catgets(3C)
menu-driven system administration	program /osysadm:	osysadm(1M)
raise: send signal to	program	raise(3C)
sdiff: side-by-side difference	program	sdiff(1)
strclean: STREAMS error logger cleanup	program	strclean(1M)
syacdb: syac debugger utility	program	syacdb(1M)
atexit: add	program termination routine	atexit(3C)
ctrace: trace a C	program to debug it	ctrace(1)
units: conversion	program	units(1)
scheduler for the uucp file transport	program /uusched: the	uusched(1M)
locate source, binary, and or manual for	program /whereis:	whereis(1)
a standard/restricted command and	programming language /rksh: KornShell,	ksh(1)
sh, jsh, rsh, restsh: shell, the command	programming language	sh(1)
devices d_passwd: log-in	programs and passwords for dial-up	d_passwd(4)
lex: generate	programs for simple lexical tasks	lex(1)
introduction to commands and application	programs /intro:	intro(1)
introduction to commands and application	programs /intro:	intro(1)
maintenance commands and application	programs /intro: introduction to system	intro(1M)
setlocale: modify and query a	program's locale	setlocale(3C)
update, and regenerate groups of	programs /make: maintain,	make(1)
xstr: extract strings from C	programs to implement shared strings	xstr(1)
ckitem: build a menu;	prompt for and return a menu item	ckitem(1)
ckdate, errdate, helpdate, valdate:	prompt for and validate a date	ckdate(1)
ckgid, errgid, helpgid, valgid:	prompt for and validate a group id	ckgid(1)
ckkeywd:	prompt for and validate a keyword	ckkeywd(1)
ckuid:	prompt for and validate a user ID	ckuid(1)
ckrange:	prompt for and validate an integer	ckrange(1)
ckyorn:	prompt for and validate yes/no	ckyorn(1)
ckpath: display a	prompt; verify and return a pathname	ckpath(1)
answer ckstr: display a	prompt; verify and return a string	ckstr(1)
/cktime: display a	prompt; verify and return a time of day	cktime(1)
value ckint: display a	prompt; verify and return an integer	ckint(1)
setprotoent, endprotoent: get	protocol entry /getprotobyname,	getprotoent(3N)
snap: Subnetwork Access	Protocol	snap(6P)
/bcs_cat: type hosts, networks, passwd,	protocols, group or services information	bcs_cat(1M)
t_getinfo: get	protocol-specific service information	t_getinfo(3N)
pkgproto: generate a	prototype file	pkgproto(1)
	prototype: package information file	prototype(4)
sets admdefault:	provide an interface to named default	admdefault(1M)
/m68k, m88k, i386, pdp11, u3b, u3b5, vax:	provide truth value about your processor/	machid(1)
true, false:	provide truth values	true(1)
/nlsprovider: get name of transport	provider	nlsprovider(3N)
	prs: print an SCCS file	prs(1)
/monacct, nulladm, prctmp, prdaily,	prtacct, shutacct, startup, turnacct:/	acctsh(1M)
	ps: report process status	ps(1)
plm:	pseudo lock manager device interface	plm(7)
pitem: STREAMS	Pseudo Terminal Emulation module	pitem(7)

optical device) as magtape/ /wmt: pseudo WORM (Write Once Read Multiple . . . wmt(7)
 devtty: control terminal pseudo-device devtty(7)
 pts, ptc: pseudo-terminal master/slave pseudo-device pair /pty, pty(7)
 systty: DG/UX operating system console pseudo-device /syscon, console, syscon(7)
 lcong48: generate uniformly distributed pseudo-random numbers /srand48, seed48, drand48(3C)
 grantpt: grant access to the slave pseudo-terminal device grantpt(3C)
 ptsname: get name of the slave pseudo-terminal device ptsname(3C)
 unlockpt: unlock a pseudo-terminal master/slave pair unlockpt(3C)
 pseudo-device pair pty, pts, ptc: pseudo-terminal master/slave pty(7)
 psignal, psiginfo: system signal messages psignal(3C)
 messages psignal, psiginfo: system signal psignal(3C)
 pseudo-device pair pty, pts, ptc: pty(7)
 module ptem: STREAMS Pseudo Terminal Emulation ptem(7)
 pseudo-device pair /pty, ptrace: process trace ptrace(2)
 pseudo-terminal device pts, ptc: pseudo-terminal master/slave pty(7)
 master/slave pseudo-device pair ptsname: get name of the slave ptsname(3C)
 uuto, upick: pty, pts, ptc: pseudo-terminal pty(7)
 /mvgetch, mvwgetch, ungetch: get (or public UNIX-to-UNIX system file copy uuto(1)
 /mvgetwch, mvwgetwch, ungetwch: get (or push back) characters from curses/ curs_getch(3X)
 ungetc: push wchar_t characters from/ curs_getwch(3X)
 stream /ungetwc: push character back onto input stream ungetc(3S)
 autopush: configure automatically push wchar_t character back into input ungetwc(3W)
 puts, fputs: pushed STREAMS modules autopush(1M)
 putws, fputs: put a string on a stream puts(3S)
 putc, putchar, fputc, putw: put a wchar_t string on a stream putws(3W)
 putwc, putwchar, fputwc: put character or word on a stream putc(3S)
 character or word on a stream put wchar_t character on a stream putwc(3W)
 word on a stream /putc, putc, putchar, fputc, putw: put putc(3S)
 environment putchar, fputc, putw: put character or putc(3S)
 stream putdev: edit device table putdev(1M)
 /del_curterm, restartterm, tparm, tputs, putdgrp: edit device group table putdgrp(1M)
 putmsg, putpmsg: pass a message down a putenv: change or add value to putenv(3C)
 putp, vidputs, vidattr, mvcur,/ putmsg(2)
 putpmsg: pass a message down a stream putmsg(2)
 putpwent: write password file entry putpwent(3C)
 puts, fputs: put a string on a stream puts(3S)
 putspent: write shadow password file putspent(3C)
 /getut: getutent, getutid, getutline, pututline, setutent, endutent, utmpname:/ getut(3C)
 /putc, putchar, fputc, putw: put character or word on a stream putc(3S)
 character on a stream putwc, putwchar, fputwc: put wchar_t putwc(3W)
 on a stream /putwc, putwchar, fputwc: put wchar_t character putwc(3W)
 /unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushinp:/ curs_util(3X)
 stream putws, fputs: put a wchar_t string on a putws(3W)
 file pwck, grpck: check password or group pwck(1M)
 /notimeout, raw, noraw, noqiflush, pwd: print working directory name pwd(1)
 setlocale: modify and qiflush, timeout, wtimeout, typeahead:/ curs_inopts(3X)
 default-gcc: set or qsort: quicker sort qsort(3C)
 termattr, termname: curses environment query a program's locale setlocale(3C)
 strchg, strconf: change or query default version of GNU C default-gcc(1)
 tput: initialize a terminal or query routines /killchar, longname, curs_termattr(3X)
 alpq: query stream configuration strchg(1)
 queue msgctl: get or set message query terminfo database tput(1)
 msgget: get message queue attributes or destroy a message alpq(1)
 remque: insert/remove element from a queue identifier msgctl(2)
 lpq: examine the spool queue /insque, insque(3C)
 jobs from the line printer spooling queue lpq(1)
 queue attributes or destroy a message queue /lprm: remove lprm(1)
 msgsys: perform a message queue /msgctl: get or set message msgctl(2)
 lpusers: set printing queue operation msgsys(2)
 ID /ipcrm: remove a message queue priorities lpusers(1M)
 atq: display the jobs queue, semaphore set, or shared memory ipcrm(1)
 qsort: quicker sort atq(1)
 run a command immune to hangups and quits /nohup: nohup(1)
 div, ldiv: compute the quotient and remainder div(3C)
 generator raise: send signal to program raise(3C)
 elf_rand: rand, srand: simple random-number rand(3C)
 /srandom, initstate, setstate: generate random archive member access elf_rand(3E)
 random numbers better, or change the/ random(3C)

generate random numbers better, or/ rand, srand: simple fsplit: split f77 or	random, srand, initState, setstate:	random(3C)
	random-number generator	rand(3C)
	ratfor files	fsplit(1)
	ratfor: rational FORTRAN dialect	ratfor(1)
	ratfor: rational FORTRAN dialect	ratfor(1)
/keypad, meta, nodelay, notimeout, returning a stream to a remote command	raw, noraw, noqiflush, qiflush, timeout,/	curls_inopts(3X)
	rcmd, rresvport, ruserok: routines for	rcmd(3X)
	rcs: change RCS file attributes	rcs(1)
	RCS commands	rcsintro(1)
rcsintro: introduction to	RCS file attributes	rcs(1)
rcs: change	RCS file from SCCS file	scstorcs(1)
scstorcs: build	RCS file	rcsfile(4)
rcsfile: format of	RCS files /rlog: print	rlog(1)
log messages and other information about	RCS /rcsfreeze: freeze a configuration	rcsfreeze(1)
of sources checked in under	RCS revisions	ci(1)
ci: check in	RCS revisions	co(1)
co: check out	RCS revisions	rcsdiff(1)
rcsdiff: compare	RCS revisions	rcsmerge(1)
rcsmerge: merge	rcsclean: clean up working files	rcsclean(1)
	rcsdiff: compare RCS revisions	rcsdiff(1)
	rcsfile: format of RCS file	rcsfile(4)
	rcsfreeze: freeze a configuration of	rcsfreeze(1)
sources checked in under RCS	rcsintro: introduction to RCS commands	rcsintro(1)
	rcsmerge: merge RCS revisions	rcsmerge(1)
	rdsk: character special disk interface	rdsk(7)
	getpass: read a password	getpass(3C)
	catgets: read a program message	catgets(3C)
object file /ldtbread:	read an indexed symbol table entry of an	ldtbread(3X)
a common object/ /ldshread, ldshread:	read an indexed/named section header of	ldshread(3X)
	dump2label: read and write labels for dump tapes	dump2label(1M)
/dg_unbuffered_read: synchronously	read data from a file without system/	dg_unbuffered_read(2)
	tread: read file(s) from tape	tread(1)
	read: read from an object	read(2)
	readv: read from file	readv(2)
mail, rmail:	read mail or send mail to users	mail(1)
interface /wmt: pseudo WORM (Write Once	Read Multiple optical device) as magtape	wmt(7)
line:	read one line	line(1)
	read: read from an object	read(2)
	read stream up to next delimiter	bgets(3G)
COFF archive file /ldahread:	read the archive header of a member of a	ldahread(3X)
	readlink: read the contents of a symbolic link	readlink(2)
file /ldfhread:	read the file header of a common object	ldfhread(3X)
file /scr_restore, scr_init, scr_set:	read (write) a curses screen from (to) a	curls_scr_dump(3X)
closedir: directory/ directory: opendir,	readdir, telldir, seekdir, rewinddir,	directory(3X)
select: examine file descriptors for I/O	readiness	select(2)
/REELexchange: commands for	reading and writing IBM and ANSI tapes	reelexchange_intro(1)
ldopen, ldaopen: open an object file for	reading	ldopen(3X)
open: open file for	reading or writing	open(2)
symbolic link	readlink: read the contents of a	readlink(2)
	readv: read from file	readv(2)
tirdwr: Transport Interface	read/write interface STREAMS module	tirdwr(7)
/setgid: set the	real-, effective-, and saved-group-ids	setgid(2)
/setregid: set the	real-, effective-, and saved-group-ids	setregid(2)
setreuid: set the	real-, effective-, and saved-user-ids	setreuid(2)
setuid: set the	real-, effective-, and saved-user-ids	setuid(2)
realpath: returns the	real file name	realpath(3C)
getgid: get the	real-group-id	getgid(2)
memory allocator malloc, free,	realloc, calloc, mallopt, mallinfo:	malloc(3X)
memory allocator malloc, free,	realloc, calloc, memalign, valloc,:	malloc(3C)
	realpath: returns the real file name	realpath(3C)
	real-user-id	getuid(2)
system processor(s) /reboot:	reboot halts and optionally reboots the	reboot(2)
request administrative shutdown and	reboot options /uadmin:	uadmin(2)
reboots the system processor(s)	reboot: reboot halts and optionally	reboot(2)
	reboot: restart the operating system	reboot(1M)
reboot: reboot halts and optionally	reboots the system processor(s)	reboot(2)
/t_rcvrel: acknowledge	receipt of an orderly release indication	t_rcvrel(3N)
t_rcvudata:	receive a data unit	t_rcvudata(3N)
recv:	receive a message from a socket	recv(2)
recvfrom:	receive a message from a socket	recvfrom(2)
recvmsg:	receive a message from a socket	recvmsg(2)

msgrcv:	receive a message	msgrcv(2)
t_rcvuderr:	receive a unit data error indication	t_rcvuderr(3N)
a connection /t_rcv:	receive data or expedited data sent over	t_rcv(3N)
request /t_rcvconnect:	receive the confirmation from a connect	t_rcvconnect(3N)
duart: Dual Asynchronous Receiver/Transmitter		duart(7)
mail_pipe: invoke recipient command for incoming mail		mail_pipe(1M)
remote file lock database, start lock expressions berk_regex, regex, dg_lcntl: process a lockf: record locking on files		dg_lock_reset(2)
tdisplay: display label and record translation settings		berk_regex(3C)
summary from per-process accounting records /acctcms: command		dg_lcntl(2)
wtmpfix: manipulate connect accounting records /fwtmp,		lockf(3C)
frec: recover files from a backup tape		tdisplay(1)
admbackup: manage backup and recovery of file systems		acctcms(1M)
socket		fwtmp(1M)
ed,		frec(1M)
/wrefresh, wnoutrefresh, douptdate, redrawwin, wredrawln: refresh curses/ REELexchange: commands for reading and re_exec: handle regular expressions		admbackup(1M)
berk_regex, regex, re_comp, re_exec: handle regular expressions		rcv(2)
man: locate and print entries from the reference manuals		rcvfrom(2)
Pascal sources xref: generate cross reference table from C, Fortran and refresh control routines /wtouchln, refresh curses windows and lines		rcvmsg(2)
is_linetouched, is_wintouched: curses refresh routine /panel_update:		ed(1)
/douptdate, redrawwin, wredrawln: refresh, wrefresh, wnoutrefresh, regcmp, regex: compile and execute		cursor_refresh(3X)
douptdate, redrawwin, / /curs_refresh: regular expression regcmp, regex: compile and execute		reelexchange_intro(1)
regular expression regcmp: regular expression compile regenerate groups of programs		berk_regex(3C)
make: maintain, update, and expression regcmp, expression regcmp, expressions /berk_regex, regex, re_comp, re_exec: handle regular regexpr: compile, step, advance: regular		man(1)
expression compile and match routines regexpr: compile, step, advance: regular		xref(1)
expression compile and match routines regexpr: compile, step, advance: regular		cursor_touch(3X)
current contents of the processor status register /getpsr: return the		cursor_refresh(3X)
service /lpsystem: register remote systems with the print register		panel_update(3X)
setpsr: set the processor status register		cursor_refresh(3X)
/pmap_rmtcall, pmap_set, pmap_unset, regexp: compile, step, advance: regular expression compile and match/		regcmp(3G)
regexpr: compile, step, advance: regular expression compile and match/		regcmp(3X)
regcmp: regular expression compile		regcmp(1)
regcmp, regex: compile and execute regular expression		make(1)
regcmp, regex: compile and execute regular expression		regcmp(3G)
regex, re_comp, re_exec: handle regular expressions /berk_regex,		regcmp(3X)
search a file for a pattern using full /accept, /comm: select or accept, reject: accept or reject print requests		berk_regex(3C)
lorder: find ordering relation for an object library		regexp(5)
join: relational database operator		regexpr(3G)
admrelease: manage software release areas		getpsr(2)
devfree: release devices from exclusive use		lpsystem(1M)
acknowledge receipt of an orderly t_sndrel: initiate an orderly common object file		setpsr(2)
common/ ldrseek, ldnrseek: seek to object file		rpc(3N)
reloc: relocation information for a relocation entries of a section of a relocation information for a common		regexp(5)
remainder, absolute value functions		regexpr(3G)
remainder		regcmp(1)
remainder		regcmp(3G)
remainder: floor, ceiling, remainder, /remexportent, endexportent, /		regcmp(3X)
exportent, endexportent, /		berk_regex(3C)
calendar: reminder service		egrep(1)
admshell: manage the remote and restricted shell names		accept(1M)
routines for returning a stream to a remote command /rresvport, ruserok:		comm(1)
uuxqt: execute remote command requests		accept(1M)
rexec: return stream to a remote command		lorder(1)
reclaim grace/ /dg_lock_reset: reset remote file lock database, start lock		join(1)
		admrelease(1M)
		devfree(1M)
		t_rcvrel(3N)
		t_sndrel(3N)
		reloc(4)
		ldrseek(3X)
		reloc(4)
		floor(3M)
		div(3C)
		drem(3M)
		floor(3M)
		exportent(3C)
		calendar(1)
		admshell(1M)
		rcmd(3X)
		uuxqt(1M)
		rexec(3X)
		dg_lock_reset(2)

/dg_lock_kill: remove locks held by rmt: start the xprt_unregister: library routines for /ckbinarsys: determine whether Uutry: try to contact lpsystem: register ct: spawn login to a rtime: get fingerd, in.fingerd: display information about local and rmdel: unlink: rmdir: removef: umount: or shared memory ID /ipcrm: process's set of blocked/ sigrelse: file /dg_flock: apply or rm, rmdir: mkdirp, rmdirp: create, remove: spooling queue lprm: atrm: /dg_lock_kill: constructs deroff: database pkgrm: queue insque, processes check file systems for consistency and uniq: report extract strings from source files, window of /panel_window: panel_window, clock: facilities status ipcs: inodes /df: tsniff: summary sar: sa1, sa2, sadc: system activity /timex: time a command; ps: uniq: file path name dirname: sar: system activity manage system activity monitoring and fseek, rewind, ftell: fsetpos, fgetpos: library routines for external data reboot options uadmin: format and send listener service dg_lcntl: process a record lock pkgask: stores answers to a t_accept: accept a connect t_listen: listen for a connect receive the confirmation from a connect t_snddis: send user-initiated disconnect accept, reject: accept or reject print start a BIOD server for asynchronous I/O start/stop the LP print service and move lpr: send print lp, cancel: send/cancel wait for previously delayed lock uuxqt: execute remote command space: disk space dialups: devices devreserv: lock reclaim grace/ /dg_lock_reset: sensible state target /sde-target: print commands to state reset:	remote lock clients dg_lock_kill(2) remote mag tape server rmt(1M) remote procedure calls /xprt_register, rpc(3N) remote system can accept binary messages ckbinarsys(1M) remote system with debugging on uutry(1M) remote systems with the print service lpsystem(1M) remote terminal ct(1) remote time rtime(3N) remote user information server fingerd(1M) remote users /finger: finger(1) remove a delta from an SCCS file rmdel(1) remove a directory entry unlink(2) remove a directory file rmdir(2) remove a file from software database removef(1M) remove a file system device umount(2) remove a message queue, semaphore set, ipcrm(1) remove a signal from the calling sigrelse(2) remove an advisory lock on an open DG/UX dg_flock(3C) rm(1), delete files or directories rm(1) remove directories in a path mkdirp(3G) remove file remove(3C) remove jobs from the line printer lprm(1) remove jobs spooled by at or batch atrm(1) remove locks held by remote lock clients dg_lock_kill(2) remove nroff/troff, tbl, and eqn deroff(1) remove: remove file remove(3C) removef: remove a file from software removef(1M) removes a package from the system pkgrm(1M) remque: insert/remove element from a insque(3C) rename: change the name of a file rename(2) renice: alter priority of running renice(1) repair them /fsck: fsck(1M) repeated lines in a file uniq(1) replace with catgets calls. /catexstr: catexstr(1) replace_panel: get or set the current panel_window(3X) report CPU time used clock(3C) report inter-process communication ipcs(1) report number of free disk blocks and df(1M) report of tape contents tsniff(1) report package sar(1M) report process data and system activity timex(1) report process status ps(1) report repeated lines in a file uniq(1) report the parent directory name of a dirname(3G) reporter sar(1) reporting /admsar: admsar(1M) reposition a file pointer in a stream fseek(3S) reposition a file pointer in a stream fsetpos(3C) representation /xdr_wrapstring: xdr(3N) request administrative shutdown and uadmin(2) request message /nlsrequest: nlsrequest(3N) request on a filehandle dg_lcntl(2) request script pkgask(1M) request t_accept(3N) request t_listen(3N) request /t_rcvconnect: t_rcvconnect(3N) request t_snddis(3N) requests accept(1M) requests /async_daemon: async_daemon(2) requests /lpsched, lpshut, lpmove: lpsched(1M) requests to a line printer spooler lpr(1) requests to an LP print service lp(1) requests to complete /dg_lock_wait: dg_lock_wait(2) requests uuxqt(1M) requirement file space(4) requiring a dial-up password. dialups(4) reserve devices for exclusive use devreserv(1M) reset remote file lock database, start dg_lock_reset(2) reset: reset the teletype bits to a reset(1) reset software development environment sde-target(1) reset the teletype bits to a sensible reset(1)
--	---

resetty,/ /def_prog_mode, def_shell_mode,	reset_prog_mode, reset_shell_mode,	curs_kernel(3X)
/def_shell_mode, reset_prog_mode,	reset_shell_mode, resetty, savetty,/	curs_kernel(3X)
/reset_prog_mode, reset_shell_mode,	resetty, savetty, getsyx, setsyx,/	curs_kernel(3X)
/dg_paging_info: determine	residency of memory pages	dg_paging_info(2)
mincore: determine	residency of memory pages	mincore(2)
sync: synchronize disk and memory	resident file system information	sync(2)
download: download host	resident PostScript fonts	download(1)
controller /vscloud: download board	resident software onto VSC synchronous	vscloud(1M)
send,/ resolver: res_mkquery, res_send,	res_init, dn_comp, dn_expand: make,	resolver(3C)
dn_comp, dn_expand: make,/ resolver:	res_mkquery, res_send, res_init,	resolver(3C)
res_init, dn_comp, dn_expand: make,/	resolver: res_mkquery, res_send,	resolver(3C)
database /admresolve: manage DNS	resolver's domain name and nameservers	admresolve(1M)
setrlimit: control maximum system	resource consumption /getrlimit,	getrlimit(2)
vlimit: control maximum system	resource consumption	vlimit(3C)
directory /dg_set_cpd_limits: change the	resource limits of a control point	dg_set_cpd_limits(2)
vtimes: get information about	resource usage	vtimes(3C)
getrusage: get information about	resource utilization	getrusage(2)
vacation: automatically	respond to incoming mail messages	vacation(1)
make, send, and/ /resolver: res_mkquery,	res_send, res_init, dn_comp, dn_expand:	resolver(3C)
reboot:	restart the operating system	reboot(1M)
/setterm, set_curterm, del_curterm,	restartterm, tparm, tputs, putp,/	curs_terminfo(3X)
restore: incrementally	restore a file system	restore(1M)
system	restore: incrementally restore a file	restore(1M)
contained in a signal frame sigret:	restore the process state to that	sigret(2)
admrshell: manage the remote and	restricted shell names	admrshell(1M)
language /sh, jsh, rsh,	restsh: shell, the command programming	sh(1)
examples /usage:	retrieve a command description and usage	usage(1)
data base gettxt:	retrieve a text string from a message	gettxt(1)
gettxt:	retrieve a text string	gettxt(3C)
/elf_getarhdr:	retrieve archive member header	elf_getarhdr(3E)
/elf_getarsym:	retrieve archive symbol table	elf_getarsym(3E)
header /elf32_getehdr, elf32_newehdr:	retrieve class-dependent object file	elf_getehdr(3E)
table /elf32_getphdr, elf32_newphdr:	retrieve class-dependent program header	elf_getphdr(3E)
/elf_getshdr: elf32_getshdr:	retrieve class-dependent section header	elf_getshdr(3E)
/elf_getident:	retrieve file identification data	elf_getident(3E)
t_rcvdis:	retrieve information from disconnect	t_rcvdis(3N)
symbol table entry ldgetname:	retrieve symbol name for object file	ldgetname(3X)
/elf_rawfile:	retrieve uninterpreted file contents	elf_rawfile(3E)
ckitem: build a menu; prompt for and	return a menu item	ckitem(1)
ckpath: display a prompt; verify and	return a pathname	ckpath(1)
ckstr: display a prompt; verify and	return a string answer	ckstr(1)
cktime: display a prompt; verify and	return a time of day	cktime(1)
ckint: display a prompt; verify and	return an integer value	ckint(1)
/fstatvfs:	return information about a file system	fstatvfs(2)
/statvfs:	return information about a file system	statvfs(2)
abs, labs:	return integer absolute value	abs(3C)
logname:	return login name of user	logname(3X)
rexec:	return stream to a remote command	rexec(3X)
processor status register getpsr:	return the current contents of the	getpsr(2)
current process /dg_ext_errno:	return the extended errno for the	dg_ext_errno(2)
entry containing filename getfh:	return the file handle of the export	getfh(2)
/basename:	return the last element of a path name	basename(3G)
current process can have /getdtablesize:	return the number of open files the	getdtablesize(2)
/elf_fsize: elf32_fsize:	return the size of an object file type	elf_fsize(3E)
getenv:	return value for environment name	getenv(3C)
call /dg_stat: data	returned by dg_stat and dg_fstat system	dg_stat(5)
stat: data	returned by stat system call	stat(5)
dg_mknod: data	returned by the dg_mknod system call	dg_mknod(5)
statfs: data	returned by the statfs system call	statfs(5)
ustat: data	returned by the ustat system call	ustat(5)
/rcmd, rresvport, ruserok: routines for	returning a stream to a remote command	rcmd(3X)
types sysfs:	returns information about file system	sysfs(2)
realpath:	returns the real file name	realpath(3C)
line of file	rev: reverse order of characters in each	rev(1)
col: filter	reverse line-feeds	col(1)
of file /rev:	reverse order of characters in each line	rev(1)
file /postreverse:	reverse the page order in a PostScript	postreverse(1)
ci: check in RCS	revisions	ci(1)
co: check out RCS	revisions	co(1)
rcsdiff: compare RCS	revisions	rcsdiff(1)
rcsmerge: merge RCS	revisions	rcsmerge(1)

in a stream /fseek,	rewind, ftell: reposition a file pointer	fseek(3S)
/opendir, readdir, telldir, seekdir,	rewinddir, closedir: directory/	directory(3X)
creat: create a new file or	rewrite an existing one	creat(2)
of a character in a string	rexec: return stream to a remote command	rexec(3X)
copysign, fmod, fmodf, fabs, fabsf,	rindex: search for the last occurrence	rindex(3C)
/resetty, savetty, getsyx, setsyx,	rint, remainder: floor, ceiling,/ /ceilf,	floor(3M)
command and programming language /ksh,	ripline, curs_set, napms: low-level/	curs_kernel(3X)
information about RCS files	rksh: KornShell, a standard/restricted	ksh(1)
directories	rlog: print log messages and other	rlog(1)
initialization information for mail and	rm, rmdir: remove, delete files or	rm(1)
/mail,	rmail /mailcfnf:	mailcfnf(4M)
	rmail: read mail or send mail to users	mail(1)
	rmdel: remove a delta from an SCCS file	rmdel(1)
	rmdir: remove a directory file	rmdir(2)
directories rm,	rmdir: remove, delete files or	rm(1)
path /mkdirp,	rmkdirp: create, remove directories in a	mkdirp(3G)
interface	rmt: character special magnetic tape	rmt(7)
	rmt: start the remote mag tape server	rmt(1M)
chroot: change	root directory for a command	chroot(1M)
chroot: change the	root directory of the calling process	chroot(2)
exponential, logarithm, power, square	root functions /pow, powf, sqrt, sqrtf:	exp(3M)
/dg_getrootkey: get	root's secret key	dg_getrootkey(2)
atexit: add program termination	routine	atexit(3C)
panels virtual screen refresh	routine /panel_update: update_panels:	panel_update(3X)
character and window attribute control	routines /standout, wstandout: curses	curs_attr(3X)
flash: curses bell and screen flash	routines /curs_beeper: beeper,	curs_beeper(3X)
curses window background manipulation	routines /wbkgdset, bkgd, wbkgd:	curs_bkgd(3X)
pair_content: curses color manipulation	routines /color_content,	curs_color(3X)
screen initialization and manipulation	routines /set_term, delscreen: curses	curs_initscr(3X)
curses terminal input option control	routines /timeout, wtimeout, typeahead:	curs_inopts(3X)
curs_set, napms: low-level curses	routines /getsyx, setsyx, ripoffline,	curs_kernel(3X)
curses terminal output option control	routines /scrollok, nl, nonl:	curs_outopts(3X)
slk_atroff: curses soft label	routines /slk_atron, slk_attrset,	curs_slk(3X)
termname: curses environment query	routines /killchar, longname, termattrs,	curs_termattrs(3X)
is_wintouched: curses refresh control	routines /wtouchln, is_linetouched,	curs_touch(3X)
flushinp: miscellaneous curses utility	routines /putwin, getwin, delay_output,	curs_util(3X)
/menu_term: assign application-specific	routines for automatic invocation by/	menu_hook(3X)
xdr_void, xdr_wrapstring: library	routines for external data/ /xdr_vector,	xdr(3N)
/field_term: assign application-specific	routines for invocation by forms	form_hook(3X)
/xprt_register, xprt_unregister: library	routines for remote procedure calls	rpc(3N)
remote/ rcmd, rresvport, ruserok:	routines for returning a stream to a	rcmd(3X)
field_opts: forms field option	routines /field_opts_on, field_opts_off,	form_field_opts(3X)
link_fieldtype: forms fieldtype	routines /set_fieldtype_choice,	form_fieldtype(3X)
form_opts_off, form_opts: forms option	routines /set_form_opts, form_opts_on,	form_opts(3X)
forms window and subwindow association	routines /form_sub, scale_form:	form_win(3X)
Internet address manipulation	routines /inet_lnaof, inet_netof:	inet(3N)
ldfcn: COFF executable file access	routines	ldfcn(4)
item_opts: menus item option	routines /item_opts_on, item_opts_off,	menu_item_opts(3X)
menu_mark: menus mark string	routines /menu_mark: set_menu_mark,	menu_mark(3X)
menu_opts_off, menu_opts: menus option	routines /set_menu_opts, menu_opts_on,	menu_opts(3X)
menus window and subwindow association	routines /menu_sub, scale_menu:	menu_win(3X)
panel_hidden: panels deck manipulation	routines /show_panel, hide_panel,	panel_show(3X)
bottom_panel: panels deck manipulation	routines /panel_top: top_panel,	panel_top(3X)
regular expression compile and match	routines /compile, step, advance:	regexp(5)
regular expression compile and match	routines /compile, step, advance:	regexpr(3G)
tputs: terminal independent operation	routines /tgetflag, tgetstr, tgoto,	termcap(3X)
widec: multibyte character I/O	routines	widec(3W)
mailsur: surrogate commands	routing and transport of mail	mailsur(4M)
admroute: manage	routing databases	admroute(1M)
/syac_routes: Change SYAC	routing information	syac_routes(1M)
set and get maximum numbers of	rows and columns in menus /menu_format:	menu_format(3X)
setrpercent, endrpercent: get	RPC entry /getrpcbyname, getrpcbynumber,	getrpercent(3N)
getrpcport: get	RPC port number	getrpcport(3R)
/msub, mult, mdiv, pow, gcd, invert,	rpow, msqrt, mcomp, move, min, omin,/	mp(3X)
returning a stream to a remote/ rcmd,	rresvport, ruserok: routines for	rcmd(3X)
programming language sh, jsh,	rsh, restsh: shell, the command	sh(1)
	rtime: get remote time	rtime(3N)
priority nice:	run a command at a higher or lower	nice(1)
quits nohup:	run a command immune to hangups and	nohup(1)
atq: display the jobs queued to	run at specified times	atq(1)
runacct:	run daily accounting	runacct(1M)

renice: alter priority of to a remote command /rcmd, rresvport, package /sar: package sar: sa1, administration activity	runacct: run daily accounting	runacct(1M)
	running processes	renice(1)
	ruserok: routines for returning a stream	rcmd(3X)
	sa1, sa2, sadc: system activity report	sar(1M)
	sa2, sadc: system activity report	sar(1M)
	sac: service access controller	sac(1M)
	sacadm: service access controller	sacadm(1M)
	sact: print current SCCS file editing	sact(1)
	sad: STREAMS Administrative Driver	sad(7)
	sadc: system activity report package	sar(1M)
	sar: sa1, sa2, sadc: system activity	sar(1M)
	sar: system activity reporter	sar(1)
	saved-group-ids	setgid(2)
	saved-group-ids	setregid(2)
	saved-user-ids	setreuid(2)
	saved-user-ids	setuid(2)
	savetty, getsyx, setsyx, ripoffline,	curl_kernel(3X)
	sbrk: change data segment space	sbrk(2)
	scalb: manipulate parts of/ /fexp,	fexp(3C)
	scale_form: forms window and subwindow/	form_win(3X)
	scale_menu: menus window and subwindow/	menu_win(3X)
	scan a directory	scandir(3C)
	scandir, alphasort: scan a directory	scandir(3C)
	scanf, fscanf, sscanf: convert formatted	scanf(3S)
	scanf, fscanf, sscanf: convert formatted	scanf(3W)
	scanner	bfs(1)
	scanning and processing language	nawk(1)
	scanning and processing language	oawk(1)
	scanw, wscanw, mvscanw, mvwscanw,	curl_scanw(3X)
	SCCS delta	cdc(1)
	SCCS deltas	comb(1)
	SCCS file	delta(1)
	SCCS file editing activity	sact(1)
	SCCS file	get(1)
	SCCS file	prs(1)
	SCCS file	rmidel(1)
	SCCS file	sccsdiff(1)
	SCCS file	sccsfile(4)
	SCCS file	sccstorcs(1)
	SCCS file	unget(1)
	SCCS file	val(1)
	SCCS files	admin(1)
	SCCS files	what(1)
	sccsdiff: compare two versions of an	sccsdiff(1)
	sccsfile: format of SCCS file	sccsfile(4)
	sccstorcs: build RCS file from SCCS file	sccstorcs(1)
	scheduler for the uucp file transport	uusched(1M)
	scheduling priority	getpriority(2)
	scheduling priority	setpriority(2)
	scr_dump: format of curses screen image	scr_dump(4)
	scr_dump, scr_restore, scr_init,	curl_scr_dump(3X)
	screen	clear(1)
	screen flash routines	curl_beep(3X)
	screen from (to) a file /scr_restore,	curl_scr_dump(3X)
	screen handling and optimization package	curses(3X)
	screen image file	scr_dump(4)
	screen initialization and manipulation/	curl_initscr(3X)
	screen /panel_move: move_panel:	panel_move(3X)
	screen refresh routine /panel_update:	panel_update(3X)
	screenful at a time	more(1)
	screenful at a time	pg(1)
	screen-oriented (visual) display editor	vi(1)
	scr_init, scr_set: read (write) a curses/ script	curl_scr_dump(3X)
	script	doconfig(3N)
	script for init	inittab(4)
	script: make typescript of a terminal	script(1)
	script	pkgask(1M)
	scroll a curses window	curl_scroll(3X)
	scroll, srcl, wscr: scroll a curses	curl_scroll(3X)
	scrollok, nl, nonl: curses terminal/	curl_outopts(3X)
	scr_restore, scr_init, scr_set: read	curl_scr_dump(3X)
setgid: set the real-, effective-, and		
setregid: set the real-, effective-, and		
setreuid: set the real-, effective-, and		
setuid: set the real-, effective-, and		
curl_set,/ /reset_shell_mode, resetty, allocation		
ldexp, logb, modf, modff, nextafter, /form_win, set_form_sub, form_sub, /menu_win, set_menu_sub, menu_sub, scandir, alphasort:		
input		
input		
bfs: big file		
nawk, awk: pattern		
oawk: old pattern		
vwscanw: convert formatted/ curl_scanw:		
cdc: change the delta commentary of an comb: combine		
delta: make a delta (change) to an sact: print current		
get: check out a version of an prs: print an		
rmidel: remove a delta from an		
sccsdiff: compare two versions of an sccsfile: format of		
sccstorcs: build RCS file from		
unget: undo a previous get of an val: validate		
admin: create and administer what: identify		
SCCS file		
program uusched: the		
getpriority: get process		
setpriority: set process		
file		
scr_set: read (write) a/ /curl_scr_dump:		
clear: clear terminal		
curl_beep: beep, flash: curses bell and		
scr_init, scr_set: read (write) a curses /curses: CRT		
scr_dump: format of curses		
/isendwin, set_term, delscreen: curses		
move a panels window on the virtual		
update_panels: panels virtual		
more, page: display file one		
pg: display file forward or backward one based on ex /vi, view, vedit:		
/curl_scr_dump: scr_dump, scr_restore, doconfig: execute a configuration		
inittab:		
session		
pkgask: stores answers to a request /curl_scroll: scroll, srcl, wscr:		
window /curl_scroll:		
/immedok, leaveok, setscreg, wsetscreg, (write) a/ /curl_scr_dump: scr_dump,		

from/ /scr_dump, scr_restore, scr_init,	scr_set: read (write) a curses screen	scr_set(3X)
cisc: AViiON family	SCSI adapter subsystem	cisc(7)
insc: AViiON family	SCSI adapter subsystem	insc(7)
ncsc: AViiON family	SCSI adapter subsystem	ncsc(7)
	sd: AViiON family disk subsystem	sd(7)
	sdb: symbolic debugger	sdb(1)
	sde: software development environment	sde(5)
environment-sensitive tool	sde-chooser: execute	sde-chooser(4)
data base	sdetab: software development environment	sdetab(4)
software development environment target	sde-target: print commands to reset	sde-target(1)
	sdiff: side-by-side difference program	sdiff(1)
/fmin, m_in, mout, omout, fmout, m_out,	sdiv, itom: multiple precision integer/	mp(3X)
fgrep:	search a file for a character string	fgrep(1)
grep:	search a file for a pattern	grep(1)
regular expressions /egrep:	search a file for a pattern using full	egrep(1)
bsearch: binary	search a sorted table	bsearch(3C)
file(s) acctcom:	search and print process accounting	acctcom(1)
lsearch, lfind: linear	search and update	lsearch(3C)
data/ srchtxt: display contents of, or	search for a text string in, message	srchtxt(1)
directories pathfind:	search for named file in named	pathfind(3G)
character in a string index:	search for the first occurrence of a	index(3C)
character in a string rindex:	search for the last occurrence of a	rindex(3C)
ttsrch: directory	search list for ttyname	ttsrch(4M)
DNS databases /admsvcorder: manage	search order for /etc/hosts, NIS, and	admsvcorder(1M)
hsearch, hcreate, hdestroy: manage hash	search tables	hsearch(3C)
tfind, tdelete, twalk: manage binary	search trees /tsearch,	tsearch(3C)
interface dgen:	second generation integrated Ethernet	dgen(7)
/dg_getrootkey: get root's	secret key	dg_getrootkey(2)
/dg_setsecretkey: store a client's	secret key in the keyserver	dg_setsecretkey(2)
elf_newdata, elf_rawdata: get	section data /elf_getdata,	elf_getdata(3E)
elf32_getshdr: retrieve class-dependent	section header /elf_getshdr:	elf_getshdr(3E)
/ldnshread: read an indexed/named	section header of a common object file	ldnshread(3X)
elf_ndxscn, elf_newscn, elf_nextscn: get	section information /elf_getscn,	elf_getscn(3E)
/seek to line number entries of a	section of a common object file	ldseek(3X)
/seek to relocation entries of a	section of a common object file	ldrseek(3X)
/ldnsseek: seek to an indexed/named	section of a common object file	ldsseek(3X)
mcs: manipulate the comment	section of an object file.	mcs(1)
size: print	section sizes of object files	size(1)
	sed: stream editor	sed(1)
/nrand48, mrand48, jrand48, srand48,	seed48, lcong48: generate uniformly/	drand48(3C)
dg_seek, dg_block_seek: extended	seek functions	dg_seek(3C)
common object file /ldsseek, ldnsseek:	seek to an indexed/named section of a	ldsseek(3X)
of a common object/ /ldlseek, ldnlseek:	seek to line number entries of a section	ldlseek(3X)
of a common object/ /ldrseek, ldnrseek:	seek to relocation entries of a section	ldrseek(3X)
object file /ldohseek:	seek to the optional file header of an	ldohseek(3X)
file ldtbseek:	seek to the symbol table of an object	ldtbseek(3X)
/directory: opendir, readdir, telldir,	seekdir, rewinddir, closedir: directory/	directory(3X)
shmat: attach a shared memory	segment	shmat(2)
shmdt: detach a shared memory	segment	shmdt(2)
shmget: get shared memory	segment	shmget(2)
brk: change data	segment space allocation	brk(2)
sbrk: change data	segment space allocation	sbrk(2)
readiness	select: examine file descriptors for I/O	select(2)
sorted files comm:	select or reject lines common to two	comm(1)
/cut: cut out	selected fields of each line of a file	cut(1)
semctl:	semaphore control operations	semctl(2)
semsys: perform a	semaphore operation	semsys(2)
semop:	semaphore operations	semop(2)
ipcrm: remove a message queue,	semaphore set, or shared memory ID	ipcrm(1)
semget: get a set of	semaphores	semget(2)
	semctl: semaphore control operations	semctl(2)
	semget: get a set of semaphores	semget(2)
	semop: semaphore operations	semop(2)
	semsys: perform a semaphore operation	semsys(2)
t_sndudata:	send a data unit	t_sndudata(3N)
send:	send a message from a socket	send(2)
sendmsg:	send a message from a socket	sendmsg(2)
sendto:	send a message from a socket	sendto(2)
msgsnd:	send a message	msgsnd(2)
kill:	send a signal to a process	kill(2)
processes /sigsend, sigsendset:	send a signal to a process or a group of	sigsend(2)

/res_init, dn_comp, dn_expand:	make, send, and interpret packets to Internet/	resolver(3C)
connection t_snd:	send data or expedited data over a	t_snd(3N)
nlsrequest: format and	send listener service request message	nlsrequest(3N)
mail, rmail: read mail or	send mail to users	mail(1)
spooler /lpr:	send print requests to a line printer	lpr(1)
	send: send a message from a socket	send(2)
group killpg:	send signal to a process or a process	killpg(2)
raise:	send signal to program	raise(3C)
/t_snddis:	send user-initiated disconnect request	t_snddis(3N)
service lp, cancel:	send/cancel requests to an LP print	lp(1)
	sendmsg: send a message from a socket	sendmsg(2)
	sendto: send a message from a socket	sendto(2)
detection dot3: IEEE 802.3 carrier	sense multiple access with collision	dot3(6P)
reset: reset the teletype bits to a	sensible state	reset(1)
elink: Environment variable	sensitive file link	elink(5)
t_rcv: receive data or expedited data	sent over a connection	t_rcv(3N)
elf_next:	sequential archive member access	elf_next(3E)
/postio:	serial interface for PostScript printers	postio(1)
grfx: AViiON	series workstation graphics processor	grfx(7)
kbd: AViiON	series workstation system keyboard	kbd(7)
in.fingerd: remote user information	server /fingerd,	fingerd(1M)
/async_daemon: start a BIOD	server for asynchronous I/O requests	async_daemon(2)
listen: network listener	server	listen(1M)
printer session with 40014A Terminal	Server /lptermprinter: start	lptermprinter(1)
nfssvc: start an NFS	server on a specified socket	nfssvc(2)
rmt: start the remote mag tape	server	rmt(1M)
strerr: STREAMS error logger	server	strerr(1M)
file for syslogd system log	server /syslog.conf: configuration	syslog.conf(5)
print a file using the 40014A Terminal	Server /termprinter:	termprinter(1)
start the WORM magnetic tape device	server /wmttd:	wmttd(1M)
/admtcpipdaemon: manage the TCP/IP	servers	admtcpipdaemon(1M)
biod: start block I/O	servers	biod(1M)
packets to Internet domain name	servers /make, send, and interpret	resolver(3C)
/sacadm:	service access controller administration	sacadm(1M)
sac:	service access controller	sac(1M)
nlsadmin: network listener	service administration	nlsadmin(1M)
lpshut, lpmove: start/stop the LP print	service and move requests /lpsched,	lpsched(1M)
calendar: reminder	service	calendar(1)
ypprot_err: Network Information	Service client interface /yperr_string,	ypclnt(3N)
admservice: manage	service database	admservice(1M)
setservent, endservent: get	service entry /getservbyname,	getservent(3N)
t_getinfo: get protocol-specific	service information	t_getinfo(3N)
send/cancel requests to an LP print	service /lp, cancel:	lp(1)
lpadmin: configure the LP print	service	lpadmin(1M)
filters used with the LP print	service /lpfilter: administer	lpfilter(1M)
administer forms used with the LP print	service /lpforms:	lpforms(1M)
about the status of the LP print	service /lpstat: print information	lpstat(1)
register remote systems with the print	service /lpsystem:	lpsystem(1M)
nlsrequest: format and send listener	service request message	nlsrequest(3N)
/admportservice: manage port monitor	services	admportservice(1M)
networks, passwd, protocols, group or	services information /type hosts,	bcs_cat(1M)
/admxterminal: manage	serving of X display terminals	admxterminal(1M)
setsid: create	session and set process group ID	setsid(2)
getsid: get	session ID	getsid(2)
script: make typescript of a terminal	session	script(1)
/lptermprinter: start printer	session with 40014A Terminal Server	lptermprinter(1)
ftruncate:	set a file to a specified length	ftruncate(3C)
alarm:	set a process alarm clock	alarm(2)
/stkprotect:	set access for future stack extensions	stkprotect(2)
/set_top_row, top_row, item_index:	set and get current menu items	menu_item_current(3X)
umask:	set and get file creation mask	umask(2)
/field_status, set_max_field:	set and get forms field attributes	form_field_buffer(3X)
columns/ /set_menu_format, menu_format:	set and get maximum numbers of rows and	menu_format(3X)
/set_item_value, item_value:	set and get menu item values	menu_item_value(3X)
/set_menu_pattern, menu_pattern:	set and get menu pattern match buffer	menu_pattern(3X)
sigstack:	set and/or get signal stack context	sigstack(2)
ascii: map of ASCII character	set	ascii(5)
ffs: find first	set bit	ffs(3C)
until a signal is caught /berk_sigpause:	set blocked signals and suspend process	berk_sigpause(2)
classify ASCII and supplementary code	set characters /isnumber, isspecial:	wctype(3W)
iconv: code	set conversion	iconv(1)

getcontext, setcontext: get and	set current user context	getcontext(2)
/settimeofday:	set date and time	settimeofday(2)
/timezone:	set default system time zone and locale	timezone(4)
/env:	set environment for command execution	env(1)
/utime:	set file access and modification times	utime(2)
/utimes:	set file access and modification times	utimes(2)
umask:	set file-creation mode mask	umask(1)
elf_fill:	set fill byte	elf_fill(3E)
/current_field, field_index:	set forms current page and field	form_page(3X)
parameters tkey:	set label and data translation	tkey(1)
memctl:	set memory access for mapping	memctl(2)
mprotect:	set memory access for mapping	mprotect(2)
a message queue /msgctl: get or	set message queue attributes or destroy	msgctl(2)
attach to att_kbd mapping tables,	set modes /kbdset:	kbdset(1)
/setdomainname:	set name of current domain	setdomainname(2)
sethostname:	set name of current host	sethostname(2)
sigblock: add to	set of blocked signals	sigblock(2)
add a signal to the calling process's	set of blocked signals /sighold:	sighold(2)
a signal from the calling process's	set of blocked signals /sigelse: remove	sigelse(2)
sigsetmask: specify	set of blocked signals	sigsetmask(2)
sigfillset: fill in the	set of implementation-defined signals	sigfillset(2)
semget: get a	set of semaphores	semget(2)
setsockopt:	set options on sockets	setsockopt(2)
eucset:	set or get EUC code set widths	eucset(1)
context sigaltstack:	set or get signal alternate stack	sigaltstack(2)
default-gcc:	set or query default version of GNU C	default-gcc(1)
ipcrm: remove a message queue, semaphore	set, or shared memory ID	ipcrm(1)
lpusers:	set printing queue priorities	lpusers(1M)
setpgid:	set process group ID for job control	setpgid(2)
setsid: create session and	set process group ID	setsid(2)
ulimit: get or	set process limits	ulimit(2)
setpriority:	set process scheduling priority	setpriority(2)
setpgrp:	set process-group-id	setpgrp(2)
setpgrp2:	set process-group-id	setpgrp2(2)
stkexec:	set stack memory access	stkexec(2)
/getgroups, setgroups: get or	set supplementary group access list IDs	getgroups(2)
sysinfo: get and	set system information strings	sysinfo(2)
tabs:	set tabs on a terminal	tabs(1)
/tcsetpgrp:	set terminal foreground process group id	tcsetpgrp(3C)
line discipline getty:	set terminal type, modes, speed, and	getty(1M)
/panel_window, replace_panel: get or	set the current window of a panels panel	panel_window(3X)
date: print and	set the date	date(1)
current process setegid:	set the effective group id of the	setegid(2)
process /setuid:	set the effective user id of the current	setuid(2)
stty:	set the options for a terminal	stty(1)
setpsr:	set the processor status register	setpsr(2)
saved-group-ids setgid:	set the real-, effective-, and	setgid(2)
saved-group-ids setregid:	set the real-, effective-, and	setregid(2)
saved-user-ids setreuid:	set the real-, effective-, and	setreuid(2)
saved-user-ids setuid:	set the real-, effective-, and	setuid(2)
'ignore' sigignore:	set the signal action of a signal to	sigignore(2)
stime:	set time	stime(2)
/syac_ttyaddr:	set tty specific internet addresses	syac_ttyaddr(1M)
sethostid:	set unique identifier of current host	sethostid(2)
process profil:	set up execution time profiling for a	profil(2)
gettimer, setitimer: get or	set value of interval timer	getitimer(2)
eucset: set or get EUC code	set widths	eucset(1)
stream	setbuf, setvbuf: assign buffering to a	setbuf(3S)
specified stream	setbuffer: assign a buffer to a	setbuffer(3C)
context getcontext,	setcontext: get and set current user	getcontext(2)
form_page: set_form_page, form_page,	set_current_field, current_field,/	form_page(3X)
set_top_row,/ /menu_item_current:	set_current_item, current_item,	menu_item_current(3X)
/curs_terminfo: setupterm, setterm,	set_curterm, del_curterm, restartterm,/	curs_terminfo(3X)
domain	setdomainname: set name of current	setdomainname(2)
the current process	setegid: set the effective group id of	setegid(2)
the current process	seteuid: set the effective user id of	seteuid(2)
remexportent,/ exportent, getexportent,	setexportent, addexportent,	exportent(3C)
/set_field_fore, field_fore,	set_field_back, field_back,/	form_field_attributes(3X)
set_field_status,/ /form_field_buffer:	set_field_buffer, field_buffer,	form_field_buffer(3X)
set_field_back,/ /form_field_attributes:	set_field_fore, field_fore,	form_field_attributes(3X)
/form_init, set_form_term, form_term,	set_field_init, field_init,/	form_hook(3X)

general appearance of /form_field_just: set_field_just, field_just: format the form_field_just(3X)
 field_opts_off,/ /form_field_opts: set_field_opts, field_opts_on, form_field_opts(3X)
 /field_fore, set_field_back, field_back, set_field_pad, field_pad: format the/ form_field_attributes(3X)
 /set_field_buffer, field_buffer, set_field_status, field_status,/ form_field_buffer(3X)
 /form_term, set_field_init, field_init, set_field_term, field_term: assign/ form_hook(3X)
 forms field data/ /form_field_validation: set_field_type, field_type, field_arg: form_field_validation(3X)
 /new_fieldtype, free_fieldtype, set_fieldtype_arg, set_fieldtype_choice,/ form_fieldtype(3X)
 /free_fieldtype, set_fieldtype_arg, set_fieldtype_choice, link_fieldtype:/ form_fieldtype(3X)
 associate/ /form_field_userptr: set_field_userptr, field_userptr: form_field_userptr(3X)
 field_count, move_field:/ form_field: set_form_fields, form_fields, form_field(3X)
 form_term, set_field_init,/ /form_hook: set_form_init, form_init, set_form_term, set_form_opts, form_opts_on, form_hook(3X)
 form_opts_off, form_opts:/ form_opts: set_form_opts, form_opts_on, form_opts(3X)
 set_current_field,/ form_page: set_form_page, form_page, form_page(3X)
 form_win: set_form_win, form_win, set_form_sub, form_sub, scale_form:/ form_win(3X)
 form_hook: set_form_init, form_init, set_form_term, form_term,/ form_hook(3X)
 associate application/ /form_userptr: set_form_userptr, form_userptr: form_userptr(3X)
 form_sub, scale_form: forms/ form_win: set_form_win, form_win, set_form_sub, setfsent, endfsent: get filesystem/ getfsent(3C)
 /getfsspec, getfsfile, getfstype, setgid: set the real-, effective-, and saved-group-ids setgid(2)
 file/ /getgrent, getgrgid, getgrnam, setgrent, endgrent, fgetgrent: get group getgrent(3C)
 group access list IDs getgroups, setgroups: get or set supplementary getgroups(2)
 entry /gethostbyaddr, gethostbyname, sethostent, endhostent: get network host gethostent(3N)
 current host sethostid: set unique identifier of sethostid(2)
 sethostname: set name of current host sethostname(2)
 item_term, set_menu_init,/ /menu_hook: set_item_init, item_init, set_item_term, menu_hook(3X)
 item_opts_off,/ /menu_item_opts: set_item_opts, item_opts_on, menu_item_opts(3X)
 /menu_hook: set_item_init, item_init, set_item_term, item_term, set_menu_init,/ menu_hook(3X)
 associate/ /menu_item_userptr: set_item_userptr, item_userptr: menu_item_userptr(3X)
 menus item values /menu_item_value: set_item_value, item_value: set and get menu_item_value(3X)
 timer /getitimer, setitimer: get or set value of interval getitimer(2)
 crypt, setjmp, longjmp: non-local goto setjmp(3C)
 specified stream setkey, encrypt: generate encryption crypt(3C)
 locale setlinebuf: assign line buffering for a setlinebuf(3C)
 syslog, openlog, closelog, setlocale: modify and query a program's setlocale(3C)
 /set_field_status, field_status, setlogmask: control system log syslog(3C)
 menu_grey,/ /set_menu_fore, menu_fore, set_max_field: set and get forms field/ form_field_buffer(3X)
 menu_back,/ /menu_attributes: set_menu_back, menu_back, set_menu_grey, menu_attributes(3X)
 menu_back,/ /menu_attributes: set_menu_fore, menu_fore, set_menu_back, menu_attributes(3X)
 get maximum numbers of/ /menu_format: set_menu_format, menu_format: set and menu_format(3X)
 /menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad,/ menu_attributes(3X)
 /item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term,/ menu_hook(3X)
 connect and disconnect/ /menu_items: set_menu_items, menu_items, item_count: menu_items(3X)
 string routines menu_mark: set_menu_mark, menu_mark: menu mark menu_mark(3X)
 menu_opts_off, menu_opts:/ menu_opts: set_menu_opts, menu_opts_on, menu_opts(3X)
 /menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad: control menus/ menu_attributes(3X)
 get menu pattern match/ /menu_pattern: set_menu_pattern, menu_pattern: set and menu_pattern(3X)
 menu_win: set_menu_win, menu_win, set_menu_sub, menu_sub, scale_menu:/ menu_win(3X)
 /item_term, set_menu_init, menu_init, set_menu_term, menu_term: assign/ menu_hook(3X)
 associate application/ /menu_userptr: set_menu_userptr, menu_userptr: menu_userptr(3X)
 menu_sub, scale_menu: menus/ menu_win: set_menu_win, menu_win, set_menu_sub, setmnt: establish mount table setmnt(1M)
 hasmntopt: get file system/ getmntent, setmntent, addmntent, endmntent, getmntent(3C)
 /getnetent, getnetbyaddr, getnetbyname, setnetent, endnetent: get network entry getnetent(3N)
 network group entry /getnetgrent, setnetgrent, endnetgrent, innnetgr: get getnetgrent(3N)
 /form_new_page: set_new_page, new_page: forms pagination form_new_page(3X)
 associate application/ /panel_userptr: set_panel_userptr, panel_userptr: panel_userptr(3X)
 control setpgid: set process group ID for job setpgid(2)
 priority setpgrp: set process-group-id setpgrp(2)
 entry /getprotobynumber, getprotobyname, setpgrp2: set process-group-id setpgrp2(2)
 register setpriority: set process scheduling setpriority(2)
 getpwent, getpwuid, getpwnam, setprotoent, endprotoent: get protocol getprotoent(3N)
 /getpwuid, getpwnam, setpwent, endpwent, setpsr: set the processor status setpsr(2)
 saved-group-ids setpwent, endpwent, setpwfile,/ getpwent(3C)
 saved-user-ids setpwfile, fgetpwent: manipulate/ getpwent(3C)
 resource consumption getrlimit, setregid: set the real-, effective-, and setregid(2)
 getrpcent, getrpcbyname, getrpcbynumber, setreuid: set the real-, effective-, and setreuid(2)
 provide an interface to named default setrlimit: control maximum system getrlimit(2)
 get information of supplementary code setrpcent, endrpcent: get RPC entry getrpcent(3N)
 sigdelset, sigismember: manipulate sets /admdefault: admdefault(1M)
 sets /getwidth: getwidth(3W)
 sets of signals. /sigfillset, sigaddset, sigsetops(3C)

/clearok, idlok, idcok immedok, leaveok, entry /getservbyport, getservbyname, group ID	setscreg, wsetscreg, scrollok, nl/ setservent, endservent: get service setsid: create session and set process setsockopt: set options on sockets setspent, endspent, fgetspent, lckpddf, setstate: generate random numbers setsyx, ripoffline, curs_set, napms set_term, delscreen: curses screen/ setterm, set_curterm, del_curterm, settimeofday: set date and time setting up an environment at login time settings for TTY ports settings information for ttymon settings /tdisplay: set_top_row, top_row, item_index: set setuid: set the real-, effective-, and setuname: changes machine information setupterm, setterm, set_curterm, setutent, endutent, utmpname: access/ setvbuf: assign buffering to a stream severity levels for application to be sgetl: access long integer data in a sh, jsh, rsh, restsh: shell, the command shadow password file entry /endspent, shadow password file entry shared descriptor array shared descriptor array shared memory control operations shared memory ID /ipcrm: remove shared memory operation shared memory segment shared memory segment shared memory segment shared object shared object shared object shared strings /xstr: extract shell (command interpreter) having a shell command shell global pattern matching shell layer manager shell names /admrshell: shell procedures for accounting shell, the command programming language shl: shell layer manager shmat: attach a shared memory segment shmctl: shared memory control operations shmdt: detach a shared memory segment shmget: get shared memory segment shmsys: perform a shared memory show group memberships show_panel, hide_panel, panel_hidden: shut down part of a full-duplex shut down system, change system state shutacct, startup, turnacct: shell/ shutdown and reboot options shutdown: shut down part of a shutdown: shut down system, change side-by-side difference program sierra file manager sifilter: preprocess MC88100 assembly sigaction: examine and change signal sigaddset, sigdelset, sigismember: sigaltstack: set or get signal alternate sigblock: add to set of blocked signals sigdelset, sigismember: manipulate sets sigemptyset, sigfillset, sigaddset, sigfillset: fill in the set of sigfillset, sigaddset, sigdelset, sighold: add a signal to the calling sigignore: set the signal action of a	curs_outopts(3X) getservent(3N) setsid(2) setsockopt(2) getspent(3C) random(3C) curs_kernel(3X) curs_initscr(3X) curs_terminfo(3X) settimeofday(2) profile(4) sttydefs(1M) ttydefs(4M) tdisplay(1) menu_item_current(3X) setuid(2) setuname(1M) curs_terminfo(3X) getut(3C) setbuf(3S) addseverity(3C) sputl(3X) sh(1) getspent(3C) putspent(3C) dg_allow_shared_descriptor_attach(dg_attach_to_shared_descriptors(2) shmctl(2) ipcrm(1) shmsys(2) shmat(2) shmdt(2) shmget(2) dlclose(3X) dlopen(3X) dlsym(3X) xstr(1) csh(1) system(3S) gmatch(3G) shl(1) admrshell(1M) acctsh(1M) sh(1) shl(1) shmat(2) shmctl(2) shmdt(2) shmget(2) shmsys(2) groups(1) panel_show(3X) shutdown(2) shutdown(1M) acctsh(1M) uadmin(2) shutdown(2) shutdown(1M) sdiff(1) hfm(4) sifilter(1) sigaction(2) sigsetops(3C) sigaltstack(2) sigblock(2) sigsetops(3C) sigsetops(3C) sigfillset(2) sigsetops(3C) sighold(2) sigignore(2)
ulckpddf: /getspent, getspnam, better, or/ random, srandom, initstate, low-level/ /resetty, savetty, getsyx, /initscr, newterm, endwin, isendwin, restartterm,/ /curs_terminfo: setupterm, /profile: sttydefs: maintain line and hunt ttydefs: terminal line display label and record translation and get/ /set_current_item, current_item, saved-user-ids del_curterm,/ /curs_terminfo: /getutent, getutid, getutline, pututline, setbuf, used with/ /addseverity: build list of machine-independent fashion sputl, programming language fgetspent, lckpddf, ulckpddf: manipulate putspent: write /let processes attach /attach another process's shmctl: a message queue, semaphore set, or shmsys: perform a shmat: attach a shmdt: detach a shmget: get dlclose: close a dlopen: open a dlsym: get the address of a symbol in strings from C programs to implement C-like syntax csh: invoke a system: issue a gmatch: shl: manage the remote and restricted /prtacct, shutacct, startup, turnacct: /sh, jsh, rsh, restsh:		

	siginfo: signal generation information	siginfo(5)
	sigismember: manipulate sets of signals.	sigsetops(3C)
	siglongjmp: a non-local goto with signal	sigsetjmp(3C)
	sign on	login(1)
abort: generate an abnormal termination	signal	abort(3C)
	signal action of a signal to 'ignore'	sigignore(2)
	signal action	sigaction(2)
	signal alternate stack context	sigaltstack(2)
	signal and suspend the process until a	sigpause(2)
	signal: base signals	signal(5)
get message string describing the given	signal /dg_strsignal:	dg_strsignal(3C)
/berk_signal, signal: simplified software	signal facilities	berk_signal(3C)
the process state to that contained in a	signal frame /sigret: restore	sigret(2)
blocked signals /sigrelse: remove a	signal from the calling process's set of	sigrelse(2)
	signal generation information	siginfo(5)
signals and suspend process until a	signal is caught /set blocked	berk_sigpause(2)
pause: suspend process until a	signal is caught	pause(2)
signal and suspend the process until a	signal is caught /clear a blocked	sigpause(2)
	signal messages	psignal(3C)
psignal, psiginfo: system	signal /signal: specify	signal(2)
what to do upon presentation of a	signal /sigset: specify	sigset(2)
what to do upon presentation of a	signal	sigsuspend(2)
sigsuspend: wait for a	signal /sigvec: specify	sigvec(2)
what to do upon presentation of a	signal: simplified software signal	berk_signal(3C)
facilities /berk_signal,	signal: specify what to do upon	signal(2)
presentation of a signal	signal stack context	sigstack(2)
sigstack: set and/or get	signal state /sigsetjmp,	sigsetjmp(3C)
siglongjmp: a non-local goto with	signal to a process	kill(2)
kill: send a	signal to a process or a group of	sigsend(2)
processes sigsend, sigsendset: send a	signal to a process or a process group	killpg(2)
	signal to 'ignore'	sigignore(2)
/killpg: send	signal to program	raise(3C)
sigignore: set the signal action of a	signal to the calling process's set of	sighold(2)
	signals and suspend process until a	berk_sigpause(2)
raise: send	signals	sigblock(2)
blocked signals /sighold: add a	signals /sigfillset: fill	sigfillset(2)
signal is/ /berk_sigpause: set blocked	signals /sighold: add a signal	sighold(2)
sigblock: add to set of blocked	signals	signal(5)
in the set of implementation-defined	signals	sigpending(2)
to the calling process's set of blocked	signals	sigprocmask(2)
	signals /sigrelse: remove a signal from	sigrelse(2)
signal: base	signals	sigsetmask(2)
sigpending: examine pending	signals	sigsetops(3C)
signals	signals	signal(3C)
sigprocmask: examine and change blocked	sigpause: clear a blocked signal and	sigpause(2)
the calling process's set of blocked	sigpending: examine pending signals	sigpending(2)
sigsetmask: specify set of blocked	sigprocmask: examine and change blocked	sigprocmask(2)
sigismember: manipulate sets of	sigrelse: remove a signal from the	sigrelse(2)
ssignal, gsignal: software	sigret: restore the process state to	sigret(2)
suspend the process until a signal is/	sigsend, sigsendset: send a signal to a	sigsend(2)
	sigsendset: send a signal to a process	sigsend(2)
signals	sigset: specify what to do upon	sigset(2)
calling process's set of blocked/	sigsetjmp, siglongjmp: a non-local goto	sigsetjmp(3C)
that contained in a signal frame	sigsetmask: specify set of blocked	sigsetmask(2)
process or a group of processes	sigstack: set and/or get signal stack	sigstack(2)
or a group of processes /sigsend,	sigsuspend: wait for a signal	sigsuspend(2)
presentation of a signal	sigvec: specify what to do upon	sigvec(2)
with signal state	simple lexical tasks	lex(1)
signals	simple process synchronization	admlock(1M)
context	simple random-number generator	rand(3C)
	simple text formatter	fmt(1)
presentation of a signal	simplified software signal facilities	berk_signal(3C)
lex: generate programs for	sin, sinf, cos, cosf, tan, tanf, asin,	trig(3M)
admlock: manage	sinf, cos, cosf, tan, tanf, asin, asinf,	trig(3M)
rand, srand:	sinh, sinh, cosh, coshf, tanh, tanhf,	sinh(3M)
fmt:	sinh, cosh, coshf, tanh, tanhf, asinh,	sinh(3M)
/berk_signal, signal:	size	deblock(1)
asinf, acos, acosf, atan, atanf, /trig:	size	getpagesize(2)
acos, acosf, atan, atanf, /trig: sin,	size of an object file type	elf_fsize(3E)
asinh, acosh, atanh: hyperbolic/	size: print section sizes of object	size(1)
acosh, atanh: hyperbolic/ /sinh,	sizes	fez(1)
deblock: change blocking		
getpagesize: get the system page		
elf_fsize: elf32_fsize: return the		
files		
fez: display file element		

size: print section	sizes of object files	size(1)
grantpt: grant access to the	slave pseudo-terminal device	grantpt(3C)
ptsname: get name of the	slave pseudo-terminal device	ptsname(3C)
	sleep: suspend execution for an interval	sleep(1)
	sleep: suspend execution for interval	sleep(3C)
/slk_touch, slk_atron, slk_attrset,	slk_attroff: curses soft label routines	curl_slk(3X)
/slk_clear, slk_restore, slk_touch,	slk_atron, slk_attrset, slk_attroff:/	curl_slk(3X)
/slk_restore, slk_touch, slk_atron,	slk_attrset, slk_attroff: curses soft/	curl_slk(3X)
/slk_refresh, slk_noutrefresh, slk_label,	slk_clear, slk_restore, slk_touch,/	curl_slk(3X)
slk_noutrefresh, slk_label,/	slk_init, slk_set, slk_refresh,	curl_slk(3X)
curl_slk:	slk_label, slk_clear, slk_restore,/	curl_slk(3X)
/slk_set, slk_refresh, slk_noutrefresh,	slk_noutrefresh, slk_label, slk_clear,/	curl_slk(3X)
/slk_init, slk_set, slk_refresh,	slk_refresh, slk_noutrefresh, slk_label,/	curl_slk(3X)
/curl_slk: slk_init, slk_set,	slk_restore, slk_touch, slk_atron,/	curl_slk(3X)
/slk_noutrefresh, slk_label, slk_clear,	slk_set, slk_refresh, slk_noutrefresh,	curl_slk(3X)
slk_label,/	slk_touch, slk_atron, slk_attrset,/	curl_slk(3X)
/curl_slk: slk_init,	slot in the utmp file of the current	ttyslot(3C)
/slk_label, slk_clear, slk_restore,	smooth curve	spline(1G)
user	snop: Subnetwork Access Protocol	snop(6P)
ttyslot: find the	SNMP community database	admsnmpcommunity(1M)
spline: interpolate	SNMP traps database	admsnmptrap(1M)
	snmpd object database	admsnmpobject(1M)
/admsnmpcommunity: manage the	sno: SNOBOL interpreter and compiler	sno(1)
/admsnmptrap: manage the	SNOBOL interpreter and compiler	sno(1)
/admsnmpobject: manage the	socket	accept(2)
	socket	bind(2)
sno:	socket: create an endpoint for	connect(2)
accept: accept a connection on a	socket	socket(2)
bind: bind a name to a	socket	getsockopt(2)
connect: initiate a connection on a	socket	listen(2)
communication	socket name	getsockname(2)
getsockopt: get options on a	socket /nfssvc:	nfssvc(2)
listen: listen for connections on a	socket	recv(2)
getsockname: get	socket	recvfrom(2)
start an NFS server on a specified	socket	recvmsg(2)
recv: receive a message from a	socket	send(2)
recvfrom: receive a message from a	socket	sendmsg(2)
recvmsg: receive a message from a	socket	sendto(2)
send: send a message from a	socketpair: create a pair of connected	socketpair(2)
sendmsg: send a message from a	sockets	setsockopt(2)
sendto: send a message from a	sockets	socketpair(2)
sockets	soft label routines /slk_atron,	curl_slk(3X)
setsockopt: set options on	software database	removef(1M)
socketpair: create a pair of connected	software dependencies files	depend(4)
slk_attrset, slk_attroff: curses	base sdetab: software development environment data	sdetab(4)
removef: remove a file from	sde: software development environment	sde(5)
depend:	software development environment target	sde-target(1)
base sdetab:	software installation database	installf(1M)
sde:	software onto VSC synchronous controller	vsclload(1M)
/sde-target: print commands to reset	software package information	pkginfo(1)
installf: add a file to the	software package to the system	pkgadd(1M)
/vsclload: download board resident	software packages	admpackage(1M)
pkginfo: display	software release areas	admrelease(1M)
pkgadd: transfer	software signal facilities	berk_signal(3C)
admpackage: manage DG/UX-style	software signals	ssignal(3C)
admrelease: manage	sort and/or merge files	sort(1)
/berk_signal, signal: simplified	sort	qsort(3C)
ssignal, gsignal:	sort: sort and/or merge files	sort(1)
sort:	sorted files /comm:	comm(1)
qsort: quicker	sorted table	bsearch(3C)
	source, binary, and or manual for	whereis(1)
tsort: topological	source files	exstr(1)
select or reject lines common to two	source files, replace with catgets	catexstr(1)
bsearch: binary search a	source level debugger	dbx(1)
program	source /mkstr: create	mkstr(1)
whereis: locate	source of zeroes	zero(7)
exstr: extract strings from	sources checked in under RCS	rcsfreeze(1)
calls.	sources /xref: generate cross reference	xref(1)
catexstr: extract strings from	space allocation	brk(2)
dbx:	space allocation	sbrk(2)
an error message file by massaging C		
zero:		
rcsfreeze: freeze a configuration of		
table from C, Fortran and Pascal		
brk: change data segment		
sbrk: change data segment		

descriptor to object in file system name	space: disk space requirement file	space(4)
munlockall: lock or unlock address	space /attach STREAMS-based file	fattach(3C)
space: disk	space /mlockall,	mlockall(3C)
ct:	space requirement file	space(4)
efficient way	spawn login to a remote terminal	ct(1)
vfork:	spawn new process in a virtual memory	vfork(2)
dsk: block	special disk interface	dsk(7)
rdsd: character	special disk interface	rdsd(7)
dkctl: control	special disk operations	dkctl(1M)
mkfifo: make FIFO	special file	mkfifo(1M)
mknod: build a	special file	mknod(1M)
intro: introduction to DG/UX System	special files	intro(7)
lp: DGC AViiON family line printer	special files	lp(7)
rmt: character	special magnetic tape interface	rmt(7)
duplicate an open file descriptor onto a	specific descriptor /dup2:	dup2(2)
/syac_ttyaddr: set tty	specific internet addresses	syac_ttyaddr(1M)
strftime: language	specific strings	strftime(4)
fspec: format	specification in text files	fspec(4)
terminate wait4: wait for the	specified child process to stop or	wait4(2)
tposn: position tape to	specified file	tposn(1)
ftruncate: set a file to a	specified length	ftruncate(3C)
truncate: truncate a file to a	specified length	truncate(2)
nfssvc: start an NFS server on a	specified socket	nfssvc(2)
setbuffer: assign a buffer to a	specified stream	setbuffer(3C)
setlinebuf: assign line buffering for a	specified stream	setlinebuf(3C)
atq: display the jobs queued to run at	specified times	atq(1)
swapon:	specify additional devices for system	swapon(1M)
sigsetmask:	specify set of blocked signals	sigsetmask(2)
a signal /signal:	specify what to do upon presentation of	signal(2)
a signal /sigset:	specify what to do upon presentation of	sigset(2)
a signal /sigvec:	specify what to do upon presentation of	sigvec(2)
getty: set terminal type, modes,	speed, and line discipline	getty(1M)
find spelling errors	spell, hashmake, spellin, hashcheck:	spell(1)
/spell, hashmake,	spellin, hashcheck: find spelling errors	spell(1)
hashmake, spellin, hashcheck: find	spelling errors /spell,	spell(1)
split:	spline: interpolate smooth curve	spline(1G)
bufsplit:	split a file into pieces	split(1)
csplit: context	split buffer into fields	bufsplit(3G)
fsplit:	split	csplit(1)
uucleanup: uucp	split f77 or ratfor files	fsplit(1)
lpq: examine the	split: split a file into pieces	split(1)
atrm: remove jobs	spool directory clean-up	uucleanup(1M)
lpd: line printer	spool queue	lpq(1)
send print requests to a line printer	spooled by at or batch	atrm(1)
lprm: remove jobs from the line printer	spooler	lpd(1M)
printf, fprintf,	spooler /lpr:	lpr(1)
printf, fprintf,	spooling queue	lprm(1)
in a machine-independent fashion	sprintf: print formatted output	printf(3S)
/log, logf, log10, log10f, pow, powf,	sprintf: print formatted output	printf(3W)
/logf, log10, log10f, pow, powf, sqrt,	sputl, sgetl: access long integer data	sputl(3X)
sqrtf: exponential, logarithm, power,	sqrt, sqrtf: exponential, logarithm,/	exp(3M)
rand,	sqrtf: exponential, logarithm, power,/	exp(3M)
/lrand48, nrand48, mrand48, jrand48,	square root functions /pow, powf, sqrt,	exp(3M)
random numbers better, or /random,	srand: simple random-number generator	rand(3C)
for a text string in, message data/	srand48, seed48, lcong48: generate/	drand48(3C)
/curs_scroll: scroll,	random, initstate, setstate: generate	random(3C)
scanf, fscanf,	srchtxt: display contents of, or search	srchtxt(1)
scanf, fscanf,	srcl, wscr: scroll a curses window	curs_scroll(3X)
Driver	sscanf: convert formatted input	scanf(3S)
sigaltstack: set or get signal alternate	sscanf: convert formatted input	scanf(3W)
sigstack: set and/or get signal	ssid: Streams Synchronous Interface	ssid(7)
stkprotect: set access for future	ssignal, gsignal: software signals	ssignal(3C)
stkexec: set	st: AViiON family tape subsystem	st(7)
/stdio:	stack context	sigaltstack(2)
print an error message to	stack context	sigstack(2)
package stdipc: ftok:	stack extensions	stkprotect(2)
cat: concatenate and type files to	stack memory access	stkexec(2)
	standard buffered input/output package	stdio(3S)
	standard error /extended_perror:	extended_perror(3C)
	standard interprocess communication	stdipc(3C)
	standard output	cat(1)

discipline module	ldterm:	standard STREAMS terminal line	ldterm(7)
programming/ ksh, rksh: KornShell, a	standard/restricted command and	ksh(1)
/attron, wattron, attrset, wattrset,	standend, wstandend, standout,/	curl_attr(3X)
/attrset, wattrset, standend, wstandend,	standout, wstandout: curses character/	curl_attr(3X)
requests /async_daemon:	start a BIOD server for asynchronous I/O	async_daemon(2)
socket nfssvc:	start an NFS server on a specified	nfssvc(2)
biod:	start block I/O servers		biod(1M)
/reset remote file lock database,	start lock reclaim grace period		dg_lock_reset(2)
Terminal Server /lptermprinter:	start printer session with 40014A		lptermprinter(1)
rmt:	start the remote mag tape server		rmt(1M)
server wmtd:	start the WORM magnetic tape device		wmtd(1M)
has_colors,/ curl_color:	start_color, init_pair, init_color,		curl_color(3X)
starter:	information for beginning users		starter(1)
requests /lpsched, lpshut, lpmove:	start/stop the LP print service and move		lpsched(1M)
/prctmp, prdaily, prtacct, shutacct,	startup, turnacct: shell procedures for/	acctsh(1M)
stat: data returned by	stat: data returned by stat system call		stat(5)
get information about current IPCs	stat: get file status		stat(2)
reset the teletype bits to a sensible	stat system call		stat(5)
shut down system, change system	state /dg_ipc_info:		dg_ipc_info(2)
siglongjmp: a non-local goto with signal	state /reset:		reset(1)
t_getstate: get the current	state /shutdown:		shutdown(1M)
frame sigret: restore the process	state /sigsetjmp,		sigsetjmp(3C)
waitid: wait for child process to change	state		t_getstate(3N)
fsync: synchronize a file's in-core	state to that contained in a signal		sigret(2)
system call	state		waitid(2)
file system	state with that on disk		fsync(2)
stats: data returned by the	stats: data returned by the stats		stats(5)
fstab:	stats: get information about a mounted		stats(2)
ustat: get file system device	stats system call		stats(5)
dg_mstat: get file	static information about file systems		fstab(4)
fstat: get file	statistics		ustat(2)
dg_fstat: get extended file	status		dg_mstat(2)
dg_stat: get extended file	status		fstat(2)
error, feof, clearerr, fileno: stream	status information		dg_fstat(2)
uustat: uucp	status information		dg_stat(2)
inter-process communication facilities	status inquiries		error(3S)
lstat: get file	status inquiry and job control		uustat(1)
lpstat: print information about the	status /ipcs: report		ipcs(1)
ps: report process	status		lstat(2)
the current contents of the processor	status of the LP print service		lpstat(1)
setpsr: set the processor	status		ps(1)
stat: get file	status register /getpsr: return		getpsr(2)
wstat: wait	status register		setpsr(2)
system	status		stat(2)
statvfs: return information about a file	status		wstat(5)
stdarg: handle variable argument list	statvfs: return information about a file		statvfs(2)
stderr or system console	stdarg: handle variable argument list		stdarg(5)
package	stderr or system console		fmtmsg(1)
communication package	stderr or system console		fmtmsg(3C)
compile and match/ regexp: compile,	stdio: standard buffered input/output		stdio(3S)
compile and match/ regexpr: compile,	stdipc: ftok: standard interprocess		stdipc(3C)
extensions	step, advance: regular expression		regexp(5)
wait3: wait for child process to	step, advance: regular expression		regexpr(3G)
wait for the specified child process to	stime: set time		stime(2)
halt:	stkexec: set stack memory access		stkexec(2)
msync: synchronize memory with physical	stkprotect: set access for future stack		stkprotect(2)
keyserver /dg_setsecretkey:	stop or terminate		wait3(2)
base subroutines /dbminit, fetch,	stop or terminate /wait4:		wait4(2)
and swap	stop the system processor		halt(1M)
pkgask:	storage		msync(3C)
manipulations	store a client's secret key in the		dg_setsecretkey(2)
compressing or/ strccpy: streadd,	store, delete, firstkey, nextkey: data		dbm(3X)
strncmp, strcpy, strncpy,/ string:	store_conditional: indivisible compare		store_conditional(2)
strings, compressing or expanding/	stores answers to a request script		pkgask(1M)
configuration	str: strfind, strspn, strtrns: string		str(3G)
/strncmp, strcpy, strncpy, strlen,	strace: print STREAMS trace messages		strace(1M)
	strcadd, strecpy: copy strings,		strccpy(3G)
	strcat, strdup, strncat, strcmp,		string(3C)
	strccpy: streadd, strcadd, strecpy: copy		strccpy(3G)
	strchg, strconf: change or query stream		strchg(1)
	strchr, strrchr, strbrk, strspn,/		string(3C)

program	strclean: STREAMS error logger cleanup	strclean(1M)
string: strcat, strdup, strncat,	strcmp, strncmp, strcpy, strncpy,/	string(3C)
configuration strchg,	strcoll: string collation	strcoll(3C)
/strdup, strncat, strncmp, strncmp,	strconf: change or query stream	strchg(1)
strchr, strchr, strpbrk, strspn,	strcpy, strncpy, strlen, strchr,/	string(3C)
strcpy, strncpy,/ string: strcat,	strcspn, strtok, strstr: string/ /strlen,	string(3C)
compressing or expanding/ /strccpy:	strdup, strncat, strncmp, strncmp,	string(3C)
strchg, strconf: change or query	streadd, strcadd, strecpy: copy strings,	strccpy(3G)
connld: line discipline for unique	stream configuration	strchg(1)
sed:	stream connections	connld(7)
fclose, fflush: close or flush a	stream editor	sed(1)
fopen, freopen, fdopen: open a	stream	fclose(3S)
ftell: reposition a file pointer in a	stream	fopen(3S)
fgetpos: reposition a file pointer in a	stream /fseek, rewind,	fseek(3S)
getw: get character or word from a	stream /fsetpos,	fsetpos(3C)
getmsg, getpmsg: get a message from a	stream /getc, getchar, fgetc,	getc(3S)
gets, fgets: get a string from a	stream	getmsg(2)
fgetwc: get wchar_t character from a	stream	gets(3S)
fgetws: get a wchar_t string from a	stream /getwc, getwchar,	getwc(3W)
fputc, putw: put character or word on a	stream /getws,	getws(3W)
putmsg, putpmsg: pass a message down a	stream /putc, putchar,	putc(3S)
puts, fputs: put a string on a	stream	putmsg(2)
fputwc: put wchar_t character on a	stream	puts(3S)
putws, fputws: put a wchar_t string on a	stream /putwc, putwchar,	putwc(3W)
setbuf, setvbuf: assign buffering to a	stream	putws(3W)
assign a buffer to a specified	stream	setbuf(3S)
assign line buffering for a specified	stream /setbuffer:	setbuffer(3C)
error, feof, clearerr, fileno:	stream /setlinebuf:	setlinebuf(3C)
ruserok: routines for returning a	stream status inquiries	ferror(3S)
rexec: return	stream to a remote command /rresvport,	rcmd(3X)
ungetc: push character back onto input	stream to a remote command	rexec(3X)
push wchar_t character back into input	stream	ungetc(3S)
bgets: read	stream /ungetc:	ungetwc(3W)
	stream up to next delimiter	bgets(3G)
	streamio: STREAMS ioctl commands	streamio(7)
	STREAMS Administrative Driver	sad(7)
	STREAMS compatibility module	ttcompat(7)
	STREAMS driver	clone(7)
	STREAMS error logger cleanup program	strclean(1M)
	STREAMS error logger server	strerr(1M)
	STREAMS error logging and event tracing	log(7)
	STREAMS ioctl commands	streamio(7)
	STREAMS module	alp(1)
	STREAMS module	timod(7)
	STREAMS module /tirdwr:	tirdwr(7)
	STREAMS modules	autopush(1M)
	STREAMS Packet Mode module	pckt(7)
	STREAMS Pseudo Terminal Emulation module	ptem(7)
	Streams Synchronous Interface Driver	ssid(7)
	STREAMS terminal line discipline module	ldterm(7)
	STREAMS trace messages	strace(1M)
	STREAMS-based file descriptor	fdetach(3C)
	STREAMS-based file descriptor to object	fattach(3C)
	strecpy: copy strings, compressing or	strccpy(3G)
	strerr: STREAMS error logger server	strerr(1M)
	strerror: get error message string	strerror(3C)
	strfind, strrspn, strtrns: string	str(3G)
	strftime, cftime, ascftime: convert date	strftime(3C)
	strftime: language specific strings	strftime(4)
	string /a64l, l64a: convert	a64l(3C)
	string and move string into it	strsave(3C)
	string answer /ckstr:	ckstr(1)
	string before character under the cursor/	cursor(3X)
	string before character under the cursor/	cursor(3X)
	string collation	strcoll(3C)
	string conversion	mbstring(3W)
	string /ctime, localtime, gmtime,	ctime(3C)
	string describing the given signal	dg_strsignal(3C)
	string /ecvt, fcvt,	ecvt(3C)
	string /extended_strerror:	extended_strerror(3C)
	string	fgrep(1)
manipulations str:		
and time to string		
between long integer and base-64 ASCII		
/allocate area large enough to hold		
display a prompt; verify and return a		
/mvinsnstr, mvwinstr, mvwinstr: insert		
/mvwinstr, mvwinstr: insert wchar_t		
strcoll:		
mbstring: mbstowcs, wctombs: multibyte		
asctime, tzset: convert date and time to		
/dg_strsignal: get message		
gcvt: convert floating-point number to		
get extended error message		
fgrep: search a file for a character		

gettxt: retrieve a text	string from a message data base	gettxt(1)
gets, fgets: get a	string from a stream	gets(3S)
getws, fgets: get a wchar_t	string from a stream	getws(3W)
mbstring: mbstowcs, wcstombs: multibyte	string functions	mbstring(3C)
getsubopt: parse suboptions from a	string	getsubopt(3C)
gettxt: retrieve a text	string	gettxt(3C)
contents of, or search for a text	string in, message data bases /display	srchtxt(1)
the first occurrence of a character in a	string /index: search for	index(3C)
large enough to hold string and move	string into it /strnsave: allocate area	strsave(3C)
convert an integer to an ASCII character	string /itoa:	itoa(3C)
str: strfind, strrspn, strtrns:	string manipulations	str(3G)
from a/ /mvwinchstr, mvwinchnstr: get a	string of characters (and attributes)	curl_inchstr(3X)
a curses/ /mvwaddchstr, mvwaddchnstr: add	string of characters (and attributes) to	curl_addchstr(3X)
a curses/ /mvwaddchstr, mvwaddchnstr: add	string of characters (and attributes) to	curl_addchstr(3X)
/mvinnstr, mvwinstr, mvinnstr: get a	string of characters from a curses/	curl_instr(3X)
/mvaddnstr, mvwaddstr, mvwaddnstr: add a	string of characters to a curses window/	curl_addstr(3X)
curses/ /mvwinwchstr, mvwinwchnstr: get a	string of wchar_t characters from a	curl_inwchstr(3X)
/mvinnwstr, mvwinwstr, mvinnwstr: get a	string of wchar_t characters from a/	curl_inwstr(3X)
window /mvwaddwchstr, mvwaddwchnstr: add	string of wchar_t characters to a curses	curl_addwchstr(3X)
window /mvwaddwstr, mvwaddwnstr: add a	string of wchar_t characters to a curses	curl_addwstr(3X)
puts, fputs: put a	string on a stream	puts(3S)
putws, fputws: put a wchar_t	string on a stream	putws(3W)
wscspn, wstok, wstokr, strtows: wchar_t	string operations and type/ /wssp,	wstring(3W)
strspn, strcspn, strtok, strstr:	string operations /strchr, strpbrk,	string(3C)
elf_strptr: make a	string pointer	elf_strptr(3E)
the last occurrence of a character in a	string /rindex: search for	rindex(3C)
set_menu_mark, menu_mark: menu mark	string routines /menu_mark:	menu_mark(3X)
strncmp, strcpy, strncpy, strlen,/	string: strcat, strdup, strncat, strcmp,	string(3C)
strerror: get error message	string	strerror(3C)
asctime: convert date and time to	string /strftime, cftime,	strftime(3C)
strtod, atof,: convert	string to double-precision number	strtod(3C)
strtol, strtoul, atol, atoi: convert	string to integer	strtol(3C)
strxfrm:	string transformation	strxfrm(3C)
att_kbd: generalized	string translation module	att_kbd(7)
/strncpy: streadd, strcadd, strcpy: copy	strings, compressing or expanding escape/	strccpy(3G)
an object or other binary file	strings: find the printable strings in	strings(1)
shared strings xstr: extract	strings from C programs to implement	xstr(1)
/mvwgetstr, mvwgetnstr: get character	strings from curses terminal keyboard	curl_getstr(3X)
/mvwgetnwstr: get wchar_t character	strings from curses terminal keyboard	curl_getwstr(3X)
exstr: extract	strings from source files	exstr(1)
catgets calls. /catexstr: extract	strings from source files, replace with	catexstr(1)
file strings: find the printable	strings in an object or other binary	strings(1)
strftime: language specific	strings	strftime(4)
sysinfo: get and set system information	strings	sysinfo(2)
from C programs to implement shared	strings /xstr: extract strings	xstr(1)
object file /strip:	strip non-executable information from an	strip(1)
from an object file	strip: strip non-executable information	strip(1)
/strcmp, strncmp, strcpy, strncpy,	strlen, strchr, strrchr, strpbrk,/	string(3C)
strncpy,/ string: strcat, strdup,	strncat, strcmp, strncmp, strcpy,	string(3C)
string: strcat, strdup, strncat, strcmp,	strncmp, strcpy, strncpy, strlen,/	string(3C)
/strncat, strcmp, strncmp, strcpy,	strncpy, strlen, strchr, strrchr,/	string(3C)
hold string and move string/ /strsave,	strnsave: allocate area large enough to	strsave(3C)
/strncpy, strlen, strchr, strrchr,	strpbrk, strspn, strcspn, strtok,/	string(3C)
/strcpy, strncpy, strlen, strchr,	strchr, strpbrk, strspn, strcspn,/	string(3C)
/str: strfind,	strrspn, strtrns: string manipulations	str(3G)
enough to hold string and move string/	strsave, strnsave: allocate area large	strsave(3C)
/strlen, strchr, strrchr, strpbrk,	strspn, strcspn, strtok, strstr: string/	string(3C)
strpbrk, strspn, strcspn, strtok,	strstr: string operations /strchr,	string(3C)
double-precision number	strtod, atof,: convert string to	strtod(3C)
/strchr, strpbrk, strspn, strcspn,	strtok, strstr: string operations	string(3C)
string to integer	strtol, strtoul, atol, atoi: convert	strtol(3C)
integer /strtol,	strtoul, atol, atoi: convert string to	strtol(3C)
/wspbrk, wssp, wscspn, wstok, wstokr,	strtows: wchar_t string operations and/	wstring(3W)
str: strfind, strrspn,	strtrns: string manipulations	str(3G)
identify processes using a file or file	structure /fuser:	fuser(1M)
inode: file node	structure	inode(4)
offsetof: offset of	structure member	offsetof(3C)
t_alloc: allocate a library	structure	t_alloc(3N)
t_free: free a library	structure	t_free(3N)
mktime: converts a tm	structure to a calendar time	mktime(3C)
	strxfrm: string transformation	strxfrm(3C)

settings for TTY ports	stty: set the options for a terminal	stty(1)
	sttydefs: maintain line and hunt	sttydefs(1M)
	su: become super-user or another user	su(1)
	snap: Subnetwork Access Protocol	snap(6P)
getsubopt: parse	suboptions from a string	getsubopt(3C)
pechochar,/ curs_pad: newpad,	subpad, prefetch, pnoutrefresh,	curs_pad(3X)
intro: introduction to	subroutines and libraries	intro(3)
delete, firstkey, nextkey: data base	subroutines /dbminit, fetch, store,	dbm(3X)
dbm_error, dbm_clearerr: data base	subroutines /dbm_firstkey, dbm_nextkey,	ndbm(3C)
cied: AViiON family disk	subsystem	cied(7)
cimd: AViiON family disk	subsystem	cimd(7)
cird: AViiON family disk	subsystem	cird(7)
cisc: AViiON family SCSI adapter	subsystem	cisc(7)
da: AViiON family disk array	subsystem	da(7)
command processor for the forms	subsystem /form_driver:	form_driver(3X)
a High Availability Disk Array	subsystem /interface for maintaining	gridman(1M)
High Availability Disk Array adapter	subsystem /hada: AViiON family	hada(7)
insc: AViiON family SCSI adapter	subsystem	insc(7)
command processor for the menus	subsystem /menu_driver:	menu_driver(3X)
ncsc: AViiON family SCSI adapter	subsystem	ncsc(7)
sd: AViiON family disk	subsystem	sd(7)
st: AViiON family tape	subsystem	st(7)
/curs_window: newwin, delwin, mvwin,	subwin, derwin, mvderwin, dupwin,/	curs_window(3X)
/form_sub, scale_form: forms window and	subwindow association routines	form_win(3X)
/menu_sub, scale_menu: menus window and	subwindow association routines	menu_win(3X)
write or erase forms from associated	subwindows /post_form, unpost_form:	form_post(3X)
write or erase menus from associated	subwindows /post_menu, unpost_menu:	menu_post(3X)
file	sum: print checksum and block count of a	sum(1)
du:	summarize disk usage	du(1)
whatis: display a one-line	summary about a topic	whatis(1)
records acctcms: command	summary from per-process accounting	acctcms(1M)
/job:	summary of DG/UX job control facilities	jobs(3C)
tsniff:	summary report of tape contents	tsniff(1)
sync: update the	super-block	sync(1M)
su: become	super-user or another user	su(1)
getwidth: get information of	supplementary code sets	getwidth(3W)
getgroups, setgroups: get or set	supplementary group access list IDs	getgroups(2)
initgroups: initialize the	supplementary group access list	initgroups(3C)
/isnumber, isspecial: classify ASCII and	supplementary code set characters	wctype(3W)
transport of mail mailsurr:	surrogate commands for routing and	mailsurr(4M)
sleep:	suspend execution for an interval	sleep(1)
sleep:	suspend execution for interval	sleep(3C)
/berk_sigpause: set blocked signals and	suspend process until a signal is caught	berk_sigpause(2)
/pause:	suspend process until a signal is caught	pause(2)
sigpause: clear a blocked signal and	suspend the process until a signal is/	sigpause(2)
/pmap_set, pmap_unset, registerrpc,	svc_destroy, svc_freeargs, svc_getargs,/	rpc(3N)
/svc_run, svc_sendreply, svc_unregister,	svcerr_auth, svcerr_decode,/	rpc(3N)
/svc_unregister, svcerr_auth,	svcerr_decode, svcerr_noproc,/	rpc(3N)
/svcerr_auth, svcerr_decode,	svcerr_noproc, svcerr_noprog,/	rpc(3N)
/svcerr_decode, svcerr_noproc,	svcerr_noprog, svcerr_progvers,/	rpc(3N)
/svcerr_noproc, svcerr_noprog,	svcerr_progvers, svcerr_systemerr,/	rpc(3N)
/svcerr_noprog, svcerr_progvers,	svcerr_systemerr, svcerr_weakauth,/	rpc(3N)
/svcerr_progvers, svcerr_systemerr,	svcerr_weakauth, svcraw_create,/	rpc(3N)
/svcraw_create, svctcp_create,	svcf_create, svcudp_create,/	rpc(3N)
pmap_unset, registerrpc, svc_destroy,	svc_freeargs, svc_getargs,/ /pmap_set,	rpc(3N)
registerrpc, svc_destroy, svc_freeargs,	svc_getargs, svc_getcaller,/ /pmap_unset,	rpc(3N)
/svc_destroy, svc_freeargs, svc_getargs,	svc_getcaller, svc_getreqset,/	rpc(3N)
/svc_getcaller, svc_getreqset,	svc_getreq, svc_register, svc_run,/	rpc(3N)
svc_run,/ /svc_getargs, svc_getcaller,	svc_getreqset, svc_getreq, svc_register,	rpc(3N)
/svcerr_systemerr, svcerr_weakauth,	svcraw_create, svctcp_create,/	rpc(3N)
/svc_getreqset, svc_getreq,	svc_register, svc_run, svc_sendreply,/	rpc(3N)
/svc_getreqset, svc_getreq, svc_register,	svc_run, svc_sendreply, svc_unregister,/	rpc(3N)
/svc_getreq, svc_register, svc_run,	svc_sendreply, svc_unregister,/	rpc(3N)
/svcerr_weakauth, svcraw_create,	svctcp_create, svcf_create,/	rpc(3N)
/svctcp_create, svcf_create,	svcudp_create, user2netname,/	rpc(3N)
/svc_register, svc_run, svc_sendreply,	svc_unregister, svcerr_auth,/	rpc(3N)
admswap: manage	swab: swap bytes	swab(3C)
swab:	swap areas	admswap(1M)
swapon: add a	swap bytes	swab(3C)
indivisible compare and	swap device for demand paging	swapon(2)
	swap /store_conditional:	store_conditional(2)

	swapcontext: manipulate user contexts	swapcontext(3C)
	paging	swapon: add a swap device for demand
	system paging	swapon(2)
	asynchronous controller	swapon: specify additional devices for
	syacdb: syac debugger utility program	swapon(1M)
	syacdump: dump syac memory to a file	syac(7)
	/syac_routes: Change SYAC routing information	syacdb(1M)
	vtc.addr: SYAC VTC configuration file	syacdump(1M)
	syacdb: syac debugger utility program	syac_routes(1M)
	syacdump: dump syac memory to a file	vtc.addr(4M)
	syac_routes: Change SYAC routing	syacdb(1M)
information	syac_routes: set tty specific internet	syacdump(1M)
addresses	symbol in shared object	syac_routes(1M)
dlsym: get the address of a	symbol table entry /ldgetname:	syac_ttyaddr: set tty specific internet
entry /ldgetname: retrieve	symbol name for object file symbol table	symbol table
/elf_getarsym: retrieve archive	symbol table	symbol table entry /ldgetname:
retrieve symbol name for object file	symbol table entry of an object file	symbol table entry of an object file
ldtbindex: compute index of	symbol table entry of an object file	symbol table format
ldtbread: read an indexed	symbol table of an object file	symbolic debugger
syms: common object file	symbolic link	symbolic link file
ldtbseek: seek to the	symbolic link	symbols /glossary:
sdb: symbolic debugger	symlink: create a symbolic link file	symlink: create a symbolic link file
symlink: create a	syms: common object file symbol table	sync: synchronize disk and memory
readlink: read the contents of a	sync: update the super-block	synchronization
definitions of common terms and	synchronization of the system clock	synchronization of the system clock
format	synchronize a file's in-core state with	synchronize disk and memory resident
resident file system information	synchronize hardware caches for execute	synchronize memory with physical storage
admlock: manage simple process	synchronize transport library	Synchronous Chip Driver
adjtime: correct the time to allow	Synchronous controller is operable	synchronous controller /download
that on disk /fsync:	synchronous controller /download	Synchronous Interface Driver
file system information sync:	synchronously read data from a file	synchronously write data to a file
access /csync:	syncok, wcursyncup, wsyncdown : create/	syncok, wcursyncup, wsyncdown : create/
/msync:	syntax /csh: invoke a shell	sysadm, asysadm, xsysadm: menu-driven
t_sync:	sysadm, asysadm, xsystty: DG/UX operating	syscon, console, systty: DG/UX operating
iscd: Integrated	sysconf: get configurable system values	sysconf: get configurable system values
vsccheck: verify that the VSC	sysdef: output system definition	sysdef: output system definition
board resident software onto VSC	sysfs: returns information about file	sysfs: returns information about file
ssid: Streams	sysinfo: get and set system information	sysinfo: get and set system information
without system/ /dg_unbuffered_read:	sys_local: invoke an extended system	sys_local: invoke an extended system
without system/ /dg_unbuffered_write:	syslog, openlog, closelog, setlogmask:	syslog, openlog, closelog, setlogmask:
/derwin, mvdwderwin, dupwin, wsyncup,	syslog.conf: configuration file for	syslog.conf: configuration file for
(command interpreter) having a C-like	syslogd: log systems messages	syslogd: log systems messages
system administration interface	syslogd system log server	syslogd system log server
system console pseudo-device	system activity monitoring and reporting	system activity monitoring and reporting
system types	system activity report package	system activity report package
strings	system activity reporter	system activity reporter
call	system activity /timex:	system activity /timex:
control system log	system	system
syslogd system log server	system administration interface	system administration interface
syslog.conf: configuration file for	system administration program	system administration program
/admsar: manage	system backup	system backup
sar: sa1, sa2, sadc:	system backup	system backup /synchronously
sar:	system buffering /synchronously	system buffering /synchronously
time a command; report process data and	system call	system call
/admaccounting: manage accounting	system call /dg_stat:	system call /dg_stat:
sysadm, asysadm, xsysadm: menu-driven	system call	system call
osysadm: menu-driven	system call	system call
dump2: incremental file	system call	system call
filesave, tapesave: daily/weekly file	system call	system call
read data from a file without	system call	system call
write data to a file without	system call	system call
dg_mknod: data returned by the dg_mknod	system call	system call
data returned by dg_stat and dg_fstat	system call	system call
stat: data returned by stat	system call	system call
statfs: data returned by the statfs	system call	system call
sys_local: invoke an extended	system call	system call
ustat: data returned by the ustat	system call	system call

intro: introduction to	system calls and error numbers	intro(2)
link, unlink: exercise link and unlink	system calls	link(1M)
ckbinarsys: determine whether remote	system can accept binary messages	ckbinarsys(1M)
shutdown: shut down	system, change system state	shutdown(1M)
admclient: manage operating	system clients	admclient(1M)
the time to allow synchronization of the	system clock /adjtime: correct	adjtime(2)
uux: UNIX-to-UNIX	system command execution	uux(1)
config: configure a	system	config(1M)
functions dg_sysctl: perform	system configuration and control	dg_sysctl(2)
fmtmsg: display a message on stderr or	system console	fmtmsg(1)
fmtmsg: display a message on stderr or	system console	fmtmsg(3C)
syscon, console, systty: DG/UX operating	system console pseudo-device	syscon(7)
uucp, uulog, uuname: UNIX-to-UNIX	system copy	uucp(1)
crash: what to do when the DG/UX	system crashes	crash(8)
cu: call another UNIX	system	cu(1)
types: primitive	system data types	types(5)
admdate: manipulate the	system date, time and time zone	admdate(1M)
dg_fsdb: file	system debugger	dg_fsdb(1M)
fsdb: file	system debugger	fsdb(1M)
sysdef: output	system definition	sysdef(1M)
endmntent, hasmntopt: get file	system descriptor file entry /addmntent,	getmntent(3C)
ustat: get file	system device statistics	ustat(2)
umount: remove a file	system device	umount(2)
dg_mount: mount a file	system	dg_mount(2)
dump: incremental file	system dump	dump(1M)
lsd: load a	system dump from tape	lsd(1M)
perror: print	system error messages	perror(3C)
uuto, uupick: public UNIX-to-UNIX	system file copy	uuto(1)
probedev: probe	system for devices	probedev(1M)
fs: file	system format	fs(4)
file	system: format of a kernel description	system(4)
get information about a mounted file	system /fstatfs:	fstatfs(2)
return information about a file	system /fstatvfs:	fstatvfs(2)
(create) a new group definition on the	system /groupadd: add	groupadd(1M)
delete a group definition from the	system /groupdel:	groupdel(1M)
modify a group definition on the	system /groupmod:	groupmod(1M)
hier: DG/UX file	system hierarchy	hier(5)
systemid: display the unique	system identifier	systemid(1M)
crash: examine	system images	crash(1M)
dirent: file	system independent directory entry	dirent(4)
dg_sys_info: get	system information	dg_sys_info(2)
dumpfs: dump file	system information	dumpfs(1M)
getexportopt: get exported file	system information /endexportent,	exportent(3C)
sysinfo: get and set	system information strings	sysinfo(2)
disk and memory resident file	system information /sync: synchronize	sync(2)
installman: manage	system installation	installman(1M)
kbd: AViiON series workstation	system: issue a shell command	system(3S)
logger: make entries in the	system keyboard	kbd(7)
configuration file for syslogd	system log	logger(1)
openlog, closelog, setlogmask: control	system log server /syslog.conf:	syslog.conf(5)
logins: list user and	system log /syslog,	syslog(3C)
mailx: interactive message processing	system login information	logins(1M)
application/ intro: introduction to	system	mailx(1)
intro: introduction to	system maintenance commands and	intro(1M)
mem: main	system maintenance procedures	intro(8)
mfs: memory file	system memory	mem(7)
mkfs, newfs: create a file	system	mfs(4)
mknod: create a file entry in the file	system	mkfs(1M)
mount: mount a file	system	mknod(2)
file descriptor to object in file	system	mount(2)
dg_mknod: create a file	system name space /attach STREAMS-based	fattach(3C)
filesystem: file	system node	dg_mknod(2)
getpagesize: get the	system organization	filesystem(7)
swapon: specify additional devices for	system page size	getpagesize(2)
vipw: edit the	system paging	swapon(1M)
pkgadd: transfer software package to the	system password file	vipw(1M)
pkgrm: removes a package from the	system	pkgadd(1M)
halt: stop the	system	pkgrm(1M)
reboot halts and optionally reboots the	system processor	halt(1M)
prf: operating	system processor(s) /reboot:	reboot(2)
	system profiler	prf(7)

prfdc, prfsnap, prfpr: operating	system profiler /prfld, prfstat,	profiler(1M)
reboot: restart the operating	system	reboot(1M)
getrlimit, setrlimit: control maximum	system resource consumption	getrlimit(2)
vlimit: control maximum	system resource consumption	vlimit(3C)
restore: incrementally restore a file	system	restore(1M)
psignal, psiginfo:	system signal messages	psignal(3C)
intro: introduction to DG/UX	System special files	intro(7)
shutdown: shut down system, change	system state	shutdown(1M)
get information about a mounted file	system /statfs:	statfs(2)
statvfs: return information about a file	system	statvfs(2)
mnttab: mounted file	system table	mnttab(4)
time: get	system time	time(2)
timezone: set default	system time zone and locale	timezone(4)
tunefs: tune an existing file	system	tunefs(1M)
sysfs: returns information about file	system types	sysfs(2)
uname: print name of current	system	uname(1)
uname, nuname: get name of current UNIX	system	uname(2)
administer a new user login on the	system /useradd:	useradd(1M)
userdel: delete a user's login from the	system	userdel(1M)
modify a user's login information on the	system /usermod:	usermod(1M)
file transport program for the uucp	system /uucico:	uucico(1M)
sysconf: get configurable	system values	sysconf(2)
who: who is on the	system	who(1)
Uutry: try to contact remote	system with debugging on	uutry(1M)
identifier	systemid: display the unique system	systemid(1M)
manage backup and recovery of file	systems /admbackup:	admbackup(1M)
/admfilesystem: manage file	systems	admfilesystem(1M)
/get information about the	system's currently active processes	dg_process_info(2)
/fsck: check file	systems for consistency and repair them	fsck(1M)
fstab: static information about file	systems	fstab(4)
admkernel: manipulate the	system's kernel	admkernel(1M)
syslogd: log	systems messages	syslogd(1M)
checklist: list of file	systems processed by fsck and ncheck	checklist(4)
volcopy, labelit: copy file	systems with label checking	volcopy(1M)
lpssystem: register remote	systems with the print service	lpssystem(1M)
pseudo-device /syscon, console,	systty: DG/UX operating system console	syscon(7)
/admdumpdevice: manage the dump device	table	admdumpdevice(1M)
bsearch: binary search a sorted	table	bsearch(3C)
/elf_getarsym: retrieve archive symbol	table	elf_getarsym(3E)
retrieve class-dependent program header	table /elf32_getphdr, elf32_newphdr:	elf_getphdr(3E)
symbol name for object file symbol	table entry /ldgetname: retrieve	ldgetname(3X)
ldtbindex: compute index of symbol	table entry of an object file	ldtbindex(3X)
ldtbread: read an indexed symbol	table entry of an object file	ldtbread(3X)
dumptab: tape	table file for dump2	dumptab(4)
syms: common object file symbol	table format	syms(4)
/xref: generate cross reference	table from C, Fortran and Pascal sources	xref(1)
mnttab: mounted file system	table	mnttab(4)
ldtbseek: seek to the symbol	table of an object file	ldtbseek(3X)
putdev: edit device	table	putdev(1M)
putdgrp: edit device group	table	putdgrp(1M)
setmnt: establish mount	table	setmnt(1M)
/admdumpcycle: manage dump cycle	tables	admdumpcycle(1M)
character classification and conversion	tables /chrtbl: generate	chrtbl(1M)
hcreate, hdestroy: manage hash search	tables /hsearch,	hsearch(3C)
kbdcomp: compile att_kbd	tables	kbdcomp(1M)
kbdload: load or link att_kbd	tables	kbdload(1M)
kbdset: attach to att_kbd mapping	tables, set modes	kbdset(1)
character classification and conversion	tables /wchrtbl: generate	wchrtbl(1M)
tabs: set	tabs on a terminal	tabs(1)
tabs: set tabs on a terminal	tabs: set tabs on a terminal	tabs(1)
t_accept: accept a connect request	t_accept: accept a connect request	t_accept(3N)
taccess: initiate access to labeled tape	taccess: initiate access to labeled tape	taccess(1)
/netdir_free, netdir_mergeaddr,	taddr2uaddr, uaddr2taddr, netdir_perror,/	netdir(3N)
ctags: create a	tags file	ctags(1)
tail: deliver the last part of a file	tail: deliver the last part of a file	tail(1)
t_alloc: allocate a library structure	t_alloc: allocate a library structure	t_alloc(3N)
atan, / trig: sin, sinf, cos, cosf,	tan, tanf, asin, asinf, acos, acosf,	trig(3M)
atanf, / trig: sin, sinf, cos, cosf, tan,	tanf, asin, asinf, acos, acosf, atan,	trig(3M)
hyperbolic/ sinh, sinhf, cosh, coshf,	tanh, tanhf, asinh, acosh, atanh:	sinh(3M)
/sinh, sinhf, cosh, coshf, tanh,	tanhf, asinh, acosh, atanh: hyperbolic/	sinh(3M)
tar:	tape archive file format	tar(5)

tsniff: summary report of	tape contents	tsniff(1)
mt: magnetic	tape control	mt(1)
wmtd: start the WORM magnetic	tape device server	wmtd(1M)
tar:	tape file archiver	tar(1)
frec: recover files from a backup	tape	frec(1M)
rmt: character special magnetic	tape interface	rmt(7)
lsd: load a system dump from	tape	lsd(1M)
rmt: start the remote mag	tape server	rmt(1M)
st: AViiON family	tape subsystem	st(7)
dumptab:	tape table file for dump2	dumptab(4)
taccess: initiate access to labeled	tape	taccess(1)
tposn: position	tape to specified file	tposn(1)
tread: read file(s) from	tape	tread(1)
trelease: terminate access to a	tape	trelease(1)
twrite: writes a file to	tape	twrite(1)
label: initialize a	tape with a volume label	label(1)
manipulate the default parameters for	tapes /admtape:	admtape(1M)
read and write labels for dump	tapes /dump2label:	dump2label(1M)
for reading and writing IBM and ANSI	tapes /REELexchange: commands	reelexchange_intro(1)
backup filesave,	tapesave: daily/weekly file system	filesave(1M)
	tar: tape archive file format	tar(5)
	tar: tape file archiver	tar(1)
reset software development environment	target /sde-target: print commands to	sde-target(1)
generate programs for simple lexical	tasks /lex:	lex(1)
endpoint	t_bind: bind an address to a transport	t_bind(3N)
deroff: remove nroff/troff,	tbl, and eqn constructs	deroff(1)
/tcsetattr, tcsetattr, tcsendbreak,	tcdrain, tcflush, tcflow, cfgetospeed,/	termios(3C)
/tcsendbreak, tcdrain, tcflush,	tcfow, cfgetospeed, cfgetispeed,/	termios(3C)
/tcsetattr, tcsendbreak, tcdrain,	tcflush, tcflow, cfgetospeed,/	termios(3C)
tcdrain, tcflush, tcflow,/	tcsetattr, tcsetattr, tcsendbreak,	termios(3C)
termios:	tcsetpgrp, tcsetpgrp, tcgetsid: general/	termios(3C)
/cfgetispeed, cfsetispeed, cfsetospeed,	tcgetsid: general terminal interface	termios(3C)
/cfsetospeed, tcgetpgrp, tcsetpgrp,	tcload: load terminal controller devices	tcload(1M)
	t_close: close a transport endpoint	t_close(3N)
another transport user	t_connect: establish a connection with	t_connect(3N)
/admtcpiparams: manage the	TCP/IP host parameters	admtcpiparams(1M)
/admipinterface: manage the	TCP/IP network interfaces database	admipinterface(1M)
/admtcpidaemon: manage the	TCP/IP servers	admtcpidaemon(1M)
/termios: tcsetattr, tcsetattr,	tcsendbreak, tcdrain, tcflush, tcflow,/	termios(3C)
tcflush, tcflow,/	tcsetattr, tcsendbreak, tcdrain,	termios(3C)
process group id	tcsetpgrp: set terminal foreground	tcsetpgrp(3C)
/cfsetispeed, cfsetospeed, tcgetpgrp,	tcsetpgrp, tcgetsid: general terminal/	termios(3C)
trees tsearch, tfind,	tdelete, twalk: manage binary search	tsearch(3C)
translation settings	tdisplay: display label and record	tdisplay(1)
legend: Debugging information	technology	legend(5)
	tee: pipe fitting	tee(1)
posttek: PostScript translator for	tektronix 4014 files	posttek(1)
reset: reset the	teletype bits to a sensible state	reset(1)
/init,	telinit: process control initialization	init(1M)
/form_data: data_ahead, data_behind:	tell if forms field has off-screen data/	form_data(3X)
/menu_item_visible: item_visible:	tell if menu item is visible	menu_item_visible(3X)
directory/ /directory: opendir, readdir,	telldir, seekdir, rewinddir, closedir:	directory(3X)
file /tmpnam,	tmpnam: create a name for a temporary	tmpnam(3S)
tmpfile: create a	temporary file	tmpfile(3S)
tmpnam, tmpnam: create a name for a	temporary file	tmpnam(3S)
chgtinfo: create a	temporary version of a TERMINFO entry	chgtinfo(1)
	term: conventional names for terminals	term(5)
/has_ic, has_il, killchar, longname,	termattr, termname: curses environment/	curs_termattr(3X)
captoinfo: convert a	TERMCAP entry into a TERMINFO entry	captoinfo(1M)
curses interfaces (emulated) to the	termcap library /tgetstr, tgoto, tputs:	curs_termcap(3X)
	termcap: terminal capability data base	termcap(5)
tgetstr, tgoto, tputs: terminal/	termcap: tgetent, tgetnum, tgetflag,	termcap(3X)
/terminfo:	terminal and printer capability database	terminfo(4)
termcap:	terminal capability data base	termcap(5)
tcload: load	terminal controller devices	tcload(1M)
ct: spawn login to a remote	terminal	ct(1)
ctermid: generate file name for	terminal	ctermid(3S)
pitem: STREAMS Pseudo	Terminal Emulation module	pitem(7)
tcsetpgrp: set	terminal foreground process group id	tcsetpgrp(3C)
/tgetflag, tgetstr, tgoto, tputs:	terminal independent operation routines	termcap(3X)
/timeout, wtimeout, typeahead: curses	terminal input option control routines	curs_inopts(3X)

termio: general	terminal interface	termio(7)
tcgetpgrp, tcsetpgrp, tcgetsid: general	terminal interface /cfsetospeed,	termios(3C)
termiox: extended general	terminal interface	termiox(7)
(or push back) characters from curses	terminal keyboard /ungetch: get	curl_getch(3X)
get character strings from curses	terminal keyboard /mvwgetnstr:	curl_getstr(3X)
back) wchar_t characters from curses	terminal keyboard /get (or push	curl_getwch(3X)
wchar_t character strings from curses	terminal keyboard /mvwgetnwstr: get	curl_getwstr(3X)
dial: establish an out-going	terminal line connection	dial(3C)
ldterm: standard STREAMS	terminal line discipline module	ldterm(7)
ttymon /ttydefs:	terminal line settings information for	ttydefs(4M)
last: indicate last user or	terminal logins	last(1)
tput: initialize a	terminal or query terminfo database	tput(1)
/wsetsrreg, scrollok, nl, nonl: curses	terminal output option control routines	curl_outopts(3X)
/admterminal: manage	terminal ports	admterminal(1M)
ttymon: monitor	terminal ports	ttymon(1M)
devtty: control	terminal pseudo-device	devtty(7)
clear: clear	terminal screen	clear(1)
start printer session with 40014A	Terminal Server /lptermprinter:	lptermprinter(1)
print a file using the 40014A	Terminal Server /termprinter:	termprinter(1)
script: make typescript of a	terminal session	script(1)
stty: set the options for a	terminal	stty(1)
tabs: set tabs on a	terminal	tabs(1)
tty: get the name of the	terminal	tty(1)
ttyname, isatty: find name of a	terminal	ttyname(3C)
discipline getty: set	terminal type, modes, speed, and line	getty(1M)
virtually hang up the current control	terminal /vhangup:	vhangup(2)
manage serving of X display	terminals /admxtterminal:	admxtterminal(1M)
term: conventional names for	terminals	term(5)
kill:	terminate a process by default	kill(1)
dg_kill: test for or	terminate a process	dg_kill(1)
trelease:	terminate access to a tape	trelease(1)
exit, _exit:	terminate process	exit(2)
wait3: wait for child process to stop or	terminate	wait3(2)
the specified child process to stop or	terminate /wait4: wait for	wait4(2)
atexit: add program	termination routine	atexit(3C)
abort: generate an abnormal	termination signal	abort(3C)
wait, waitpid: wait for process	termination	wait(2)
tic:	TERMINFO compiler	tic(1M)
tigetnum, tigetstr: curses interfaces to	terminfo database /mvcurl, tigetflag,	curl_terminfo(3X)
tput: initialize a terminal or query	terminfo database	tput(1)
infocmp: compare or print out	TERMINFO descriptions	infocmp(1M)
convert a TERMcap entry into a	TERMINFO entry /captainfo:	captainfo(1M)
create a temporary version of a	TERMINFO entry /chginfo:	chginfo(1)
capability database	terminfo: terminal and printer	terminfo(4)
termio: general terminal interface	termio: general terminal interface	termio(7)
termios: tcgetattr, tcsetattr,	termios: tcgetattr, tcsetattr,	termios(3C)
termiox: extended general terminal	termiox: extended general terminal	termiox(7)
termname: curses environment query/	termname: curses environment query/	curl_termattrs(3X)
termprinter: print a file using the	termprinter: print a file using the	termprinter(1)
terms and symbols	terms and symbols	glossary(1)
t_error: produce error message	t_error: produce error message	t_error(3N)
isastream:	test a file descriptor	isastream(3C)
test: condition evaluation command	test: condition evaluation command	test(1)
dg_kill:	test for or terminate a process	dg_kill(1)
testlocale:	test locale definition	testlocale(1M)
testlocale: test locale definition	testlocale: test locale definition	testlocale(1M)
ed, red:	text editor	ed(1)
ex:	text editor	ex(1)
users) edit:	text editor (variant of ex for casual	edit(1)
newform: change the format of a	text file	newform(1)
fspec: format specification in	text files	fspec(4)
postprint: translate	text files into PostScript	postprint(1)
fmt: simple	text formatter	fmt(1)
plock: lock data,	text, or both into memory	plock(2)
gettxt: retrieve a	text string from a message data base	gettxt(1)
gettxt: retrieve a	text string	gettxt(3C)
/display contents of, or search for a	text string in, message data bases	srchtxt(1)
search trees tsearch,	tfind, tdelete, twalk: manage binary	tsearch(3C)
tgoto, tputs: curses/ /curl_termcap:	t_free: free a library structure	t_free(3N)
tgoto, tputs: terminal/ termcap:	tgetent, tgetflag, tgetnum, tgetstr,	curl_termcap(3X)
	tgetent, tgetnum, tgetflag, tgetstr,	termcap(3X)

display a one-line summary about a	<code>t_open</code> : establish a transport endpoint	<code>t_open(3N)</code>
manipulation routines <code>panel_top</code> :	<code>topic /whatis</code> :	<code>whatis(1)</code>
menus items <code>/current_item</code> , <code>set_top_row</code> ,	<code>topological sort</code>	<code>tsort(1)</code>
transport endpoint	<code>top_panel</code> , <code>bottom_panel</code> : panels deck	<code>panel_top(3X)</code>
<code>acctmerg</code> : merge or add	<code>top_row</code> , <code>item_index</code> : set and get current	<code>menu_item_current(3X)</code>
times of a file	<code>t_optmgmt</code> : manage options for a	<code>t_optmgmt(3N)</code>
<code>is_linetouched</code> , <code>/ curs_touch</code> : <code>touchwin</code> ,	<code>total accounting files</code>	<code>acctmerg(1M)</code>
<code>wtouchln</code> , <code>is_linetouched</code> , <code>/ curs_touch</code> :	<code>touch</code> : update access and modification	<code>touch(1)</code>
characters <code>/conv</code> : <code>toupper</code> , <code>tolower</code> ,	<code>touchline</code> , <code>untouchwin</code> , <code>wtouchln</code> ,	<code>curs_touch(3X)</code>
<code>toascii</code> : translate characters <code>conv</code> :	<code>touchwin</code> , <code>touchline</code> , <code>untouchwin</code> ,	<code>curs_touch(3X)</code>
<code>wconv</code> : <code>towupper</code> ,	<code>_toupper</code> , <code>_tolower</code> , <code>toascii</code> : translate	<code>conv(3C)</code>
<code>/wconv</code> :	<code>toupper</code> , <code>tolower</code> , <code>_toupper</code> , <code>_tolower</code> ,	<code>conv(3C)</code>
<code>/set_curterm</code> , <code>del_curterm</code> , <code>restartterm</code> ,	<code>towlower</code> : translate characters	<code>wconv(3W)</code>
terminfo database	<code>towupper</code> , <code>towlower</code> : translate characters	<code>wconv(3W)</code>
the/ <code>/tgetflag</code> , <code>tgetnum</code> , <code>tgetstr</code> , <code>tgoto</code> ,	<code>tparam</code> , <code>tputs</code> , <code>putp</code> , <code>vidputs</code> , <code>vidattr</code> ,/	<code>curs_terminfo(3X)</code>
<code>/del_curterm</code> , <code>restartterm</code> , <code>tparam</code> ,	<code>tposn</code> : position tape to specified file	<code>tposn(1)</code>
<code>/tgetnum</code> , <code>tgetflag</code> , <code>tgetstr</code> , <code>tgoto</code> ,	<code>tput</code> : initialize a terminal or query	<code>tput(1)</code>
<code>ctrace</code> :	<code>tputs</code> : curses interfaces (emulated) to	<code>curs_termcap(3X)</code>
<code>dg_xtrace</code> : extended process	<code>tputs</code> , <code>putp</code> , <code>vidputs</code> , <code>vidattr</code> , <code>mvcur</code> ,/	<code>curs_terminfo(3X)</code>
<code>strace</code> : print STREAMS	<code>tputs</code> : terminal independent operation/	<code>termcap(3X)</code>
<code>ptrace</code> : process	<code>tr</code> : translate characters	<code>tr(1)</code>
to STREAMS error logging and event	<code>trace</code> a C program to debug it	<code>ctrace(1)</code>
<code>/pkgadd</code> :	<code>trace</code>	<code>dg_xtrace(2)</code>
<code>strxfrm</code> : string	<code>trace messages</code>	<code>strace(1M)</code>
<code>wchar_t</code> string operations and type	<code>trace</code>	<code>ptrace(2)</code>
<code>tolower</code> , <code>_toupper</code> , <code>_tolower</code> , <code>toascii</code> :	<code>tracing /log</code> : interface	<code>log(7)</code>
<code>tr</code> :	<code>transfer software package to the system</code>	<code>pkgadd(1M)</code>
<code>wconv</code> : <code>towupper</code> , <code>towlower</code> :	<code>transformation</code>	<code>strxfrm(3C)</code>
<code>mailalias</code> :	<code>transformation /wstok</code> , <code>wstostr</code> , <code>strtows</code> :	<code>wstring(3W)</code>
<code>/cof2elf</code> :	<code>translate characters /conv</code> : <code>toupper</code> ,	<code>conv(3C)</code>
<code>pkgtrans</code> :	<code>tr</code> : translate characters	<code>tr(1)</code>
<code>postprint</code> :	<code>translate characters</code>	<code>wconv(3W)</code>
<code>elf32_xlatetom</code> : class-dependent data	<code>translate mail alias names</code>	<code>mailalias(1)</code>
<code>att_kbd</code> : generalized string	<code>translate object file from COFF to ELF</code>	<code>cof2elf(1)</code>
generic transport name-to-address	<code>translate package format</code>	<code>pkgtrans(1)</code>
<code>tkey</code> : set label and data	<code>translate text files into PostScript</code>	<code>postprint(1)</code>
<code>tdisplay</code> : display label and record	<code>translation /elf_xlate</code> : <code>elf32_xlatetof</code> ,	<code>elf_xlate(3E)</code>
<code>ctl</code> : COFF-to-legend	<code>translation module</code>	<code>att_kbd(7)</code>
<code>postdaisy</code> : PostScript	<code>translation /netdir_sperror</code> :	<code>netdir(3N)</code>
<code>postdmd</code> : PostScript	<code>translation parameters</code>	<code>tkey(1)</code>
<code>postplot</code> : PostScript	<code>translation settings</code>	<code>tdisplay(1)</code>
<code>posttek</code> : PostScript	<code>translator</code>	<code>ctl(1)</code>
encode/decode a binary file for	<code>translator for Diablo 630 files</code>	<code>postdaisy(1)</code>
<code>t_bind</code> : bind an address to a	<code>translator for DMD bitmap files</code>	<code>postdmd(1)</code>
<code>t_close</code> : close a	<code>translator for plot(4) graphics files</code>	<code>postplot(1)</code>
<code>t_look</code> : look at the current event on a	<code>translator for tektronix 4014 files</code>	<code>posttek(1)</code>
<code>t_open</code> : establish a	<code>transmission via mail /uudecode</code> :	<code>uuencode(1)</code>
<code>t_optmgmt</code> : manage options for a	<code>transport endpoint</code>	<code>t_bind(3N)</code>
<code>t_unbind</code> : disable a	<code>transport endpoint</code>	<code>t_close(3N)</code>
module <code>/timod</code> :	<code>transport endpoint</code>	<code>t_look(3N)</code>
STREAMS module <code>/tirdwr</code> :	<code>transport endpoint</code>	<code>t_open(3N)</code>
<code>t_sync</code> : synchronize	<code>transport endpoint</code>	<code>t_optmgmt(3N)</code>
<code>/netdir_perror</code> , <code>netdir_sperror</code> : generic	<code>Transport Interface cooperating STREAMS</code>	<code>t_unbind(3N)</code>
surrogate commands for routing and	<code>Transport Interface read/write interface</code>	<code>timod(7)</code>
<code>uucico</code> : file	<code>transport library</code>	<code>tirdwr(7)</code>
<code>uusched</code> : the scheduler for the uucp file	<code>transport name-to-address translation</code>	<code>t_sync(3N)</code>
<code>/nlsprovider</code> : get name of	<code>transport of mail /mailsur</code> :	<code>netdir(3N)</code>
establish a connection with another	<code>transport program for the uucp system</code>	<code>mailsur(4M)</code>
<code>/admsnmpttrap</code> : manage the SNMP	<code>transport program</code>	<code>uucico(1M)</code>
<code>panel_above</code> , <code>panel_below</code> : panels deck	<code>transport provider</code>	<code>uusched(1M)</code>
sent over a connection	<code>transport user /t_connect</code> :	<code>nlsprovider(3N)</code>
from a connect request	<code>traps database</code>	<code>t_connect(3N)</code>
disconnect	<code>traversal primitives /panel_above</code> :	<code>admsnmpttrap(1M)</code>
orderly release indication	<code>t_rcv</code> : receive data or expedited data	<code>panel_above(3X)</code>
indication	<code>t_rcvconnect</code> : receive the confirmation	<code>t_rcv(3N)</code>
	<code>t_rcvdis</code> : retrieve information from	<code>t_rcvconnect(3N)</code>
	<code>t_rcvrel</code> : acknowledge receipt of an	<code>t_rcvdis(3N)</code>
	<code>t_rcvudata</code> : receive a data unit	<code>t_rcvrel(3N)</code>
	<code>t_rcvuderr</code> : receive a unit data error	<code>t_rcvudata(3N)</code>
	<code>tread</code> : read file(s) from tape	<code>t_rcvuderr(3N)</code>
		<code>tread(1)</code>

ftw, nftw: walk a file	tree	ftw(3C)
tdelete, twalk: manage binary search	trees /tsearch, tfind,	tsearch(3C)
asin, asinf, acos, acosf, atan, atanf, /	release: terminate access to a tape	release(1)
acos, acosf, atan, atanf, atan2, atan2f:	trig: sin, sinf, cos, cosf, tan, tanf,	trig(3M)
printers dpost:	trigonometric functions /asin, asinf,	trig(3M)
	troff postprocessor for PostScript	dpost(1)
	true, false: provide truth values	true(1)
truncate:	truncate a file to a specified length	truncate(2)
length	truncate: truncate a file to a specified	truncate(2)
/admtrustedhost: manage the	trusted hosts database	admtrustedhost(1M)
/i386, pdp11, u3b, u3b5, vax: provide	truth value about your processor type	machid(1)
true, false: provide	truth values	true(1)
debugging on Uutry:	try to contact remote system with	uutry(1M)
binary search trees	tsearch, tfind, tdelete, twalk: manage	tsearch(3C)
a connection	t_snd: send data or expedited data over	t_snd(3N)
request	t_snddis: send user-initiated disconnect	t_snddis(3N)
	t_sndrel: initiate an orderly release	t_sndrel(3N)
	t_sndudata: send a data unit	t_sndudata(3N)
	tsniff: summary report of tape contents	tsniff(1)
	tsort: topological sort	tsort(1)
	t_sync: synchronize transport library	t_sync(3N)
compatibility module	ttcompat: V7, 4BSD and XENIX STREAMS	ttcompat(7)
generic interface to EUC handling	TTY drivers and modules /eucioctl:	eucioctl(5)
	tty: get the name of the terminal	tty(1)
ttyadm: format and output	TTY port monitor information	ttyadm(1M)
maintain line and hunt settings for	TTY ports /sttydefs:	sttydefs(1M)
/syac_ttyaddr: set	tty specific internet addresses	syac_ttyaddr(1M)
monitor information	ttyadm: format and output TTY port	ttyadm(1M)
information for ttymon	ttydefs: terminal line settings	ttydefs(4M)
terminal line settings information for	ttymon: monitor terminal ports	ttymon(1M)
	ttymon /ttydefs:	ttydefs(4M)
	ttyname, isatty: find name of a terminal	ttyname(3C)
	ttyname	ttysrch(4M)
ttysrch: directory search list for	ttyslot: find the slot in the utmp file	ttyslot(3C)
of the current user	ttysrch: directory search list for	ttysrch(4M)
ttyname	t_unbind: disable a transport endpoint	t_unbind(3N)
	tunefs: tune an existing file system	tunefs(1M)
	tunefs: tune an existing file system	tunefs(1M)
prdaily, prtacct, shutacct, startup,	turnacct: shell procedures for /prctmp,	acctsh(1M)
tsearch, tfind, tdelete,	twalk: manage binary search trees	tsearch(3C)
bcmp: compare	two areas of memory	bcmp(3C)
computes the difference between	two calendar times /difftime:	difftime(3C)
dircmp: compare	two directories	dircmp(1)
cmp: compare	two files	cmp(1)
comm: select or reject lines common to	two sorted files	comm(1)
scscdiff: compare	two versions of an SCCS file	scscdiff(1)
	twrite: writes a file to tape	twrite(1)
return the size of an object file	type /elf_fsize: elf32_fsize:	elf_fsize(3E)
elf_kind: determine file	type	elf_kind(3E)
file: determine file	type	file(1)
cat: concatenate and	type files to standard output	cat(1)
group or services information /bcs_cat:	type hosts, networks, passwd, protocols,	bcs_cat(1M)
provide truth value about your processor	type /m88k, i386, pdp11, u3b, u3b5, vax:	machid(1)
/getty: set terminal	type, modes, speed, and line discipline	getty(1M)
finite, fpclass, unordered: determine	type of floating-point number /isnanf,	isnanf(3C)
strtows: wchar_t string operations and	type transformation /wstok, wstokr,	wstring(3W)
field_type, field_arg: forms field data	type validation /set_field_type,	form_field_validation(3X)
/noqiflush, qiflush, timeout, wtimeout,	typeahead: curses terminal input option/	curs_inopts(3X)
nl_types: native language data	types	nl_types(5)
	types: primitive system data types	types(5)
returns information about file system	types /sysfs:	sysfs(2)
types: primitive system data	types	types(5)
script: make	typescript of a terminal session	script(1)
/ctime, localtime, gmtime, asctime,	tzset: convert date and time to string	ctime(3C)
machid: dghost, m68k, m88k, i386, pdp11,	u3b, u3b5, vax: provide truth value/	machid(1)
/dghost, m68k, m88k, i386, pdp11, u3b,	u3b5, vax: provide truth value about/	machid(1)
/netdir_mergeaddr, taddr2uaddr,	uaddr2taddr, netdir_perror, /	netdir(3N)
and reboot options	uadmin: request administrative shutdown	uadmin(2)
or user name associated with effective	ucontext: user context	ucontext(5)
getpw: get name from	UID /cuserid: get character login name	cuserid(3S)
	UID	getpw(3C)

/setspent, endspent, fgetspent, lckpwwdf,	ul: do underlining	ul(1)
	ulckpwwdf: manipulate shadow password/	getspent(3C)
	ulimit: get or set process limits	ulimit(2)
	umask: set and get file creation mask	umask(2)
	umask: set file-creation mode mask	umask(1)
/mount,	umount: mount and dismount filesystems	mount(1M)
	umount: remove a file system device	umount(2)
system	uname, nuname: get name of current UNIX	uname(2)
	uname: print name of current system	uname(1)
display expanded files compress,	uncompress, zcat: compress, expand or	compress(1)
putwin, getwin,/ curs_util:	unctrl, keyname, filter, use_env,	curs_util(3X)
/mvdelch, mvwdelch: delete character	under cursor in a curses window	curs_delch(3X)
a configuration of sources checked in	under RCS /rcsfreeze: freeze	rcsfreeze(1)
/insert a character before the character	under the cursor in a curses window	curs_insch(3X)
/insert string before character	under the cursor in a curses window	curs_insstr(3X)
/a wchar_t character before the character	under the cursor in a curses window	curs_inswch(3X)
/insert wchar_t string before character	under the cursor in a curses window	curs_inswstr(3X)
	ul: do underlining	ul(1)
	unget: undo a previous get of an SCCS file	unget(1)
	file unget: undo a previous get of an SCCS	unget(1)
	stream ungetc: push character back onto input	ungetc(3S)
from/ /getch, wgetch, mvwgetch, mvwgetch,	ungetch: get (or push back) characters	curs_getch(3X)
into input stream	ungetwc: push wchar_t character back	ungetwc(3W)
/getwch, wgetwch, mvwgetwch, mvwgetwch,	ungetwch: get (or push back) wchar_t/	curs_getwch(3X)
/srand48, seed48, lcong48: generate	uniformly distributed pseudo-random/	drand48(3C)
/elf_rawfile: retrieve	uninterpreted file contents	elf_rawfile(3E)
	uniq: report repeated lines in a file	uniq(1)
	unique file name	mkstemp(3C)
mkstemp: make a	unique file name	mktemp(3C)
mktemp: make a	unique identifier of current host	gethostid(2)
gethostid: get	unique identifier of current host	sethostid(2)
sethostid: set	unique stream connections	connld(7)
connld: line discipline for	unique system identifier	systemid(1M)
systemid: display the	unit data error indication	t_rcvuderr(3N)
t_rcvuderr: receive a	unit	t_rcvudata(3N)
t_rcvudata: receive a data	unit	t_sndudata(3N)
t_sndudata: send a data	units: conversion program	units(1)
	UNIX system	cu(1)
cu: call another	UNIX system	uname(2)
uname, nuname: get name of current	unix_ipc: piping communications within a	unix_ipc(6F)
host	UNIX-to-UNIX system command execution	uux(1)
/uux:	UNIX-to-UNIX system copy	uucp(1)
uucp, uulog, uuname:	UNIX-to-UNIX system file copy	uuto(1)
uuto, uupick: public	unlink: exercise link and unlink system	link(1M)
calls /link,	unlink: remove a directory entry	unlink(2)
	unlink system calls	link(1M)
link, unlink: exercise link and	unlock a pseudo-terminal master/slave	unlockpt(3C)
pair unlockpt:	unlock address space	mlockall(3C)
mlockall, munlockall: lock or	unlock pages in memory	mlock(3C)
mlock, munlock: lock (or	unlockpt: unlock a pseudo-terminal	unlockpt(3C)
master/slave pair	unmap pages of memory	munmap(2)
munmap:	unordered: determine type of/	isnan(3C)
isnan, isnand, isnanf, finite, fpclass,	unpack: compress and expand files	pack(1)
pack, pcat,	unpost_form: write or erase forms from	form_post(3X)
associated/ /form_post: post_form,	unpost_menu: write or erase menus from	menu_post(3X)
associated/ /menu_post: post_menu,	untouchwin, wtouchln, is_linetouched,/	curs_touch(3X)
curs_touch: touchwin, touchline,	up an environment at login time	profile(4)
profile: setting	up execution time profiling for a	profil(2)
process profil: set	up the current control terminal	vhangup(2)
vhangup: virtually hang	up to next delimiter	bgets(3G)
bgets: read stream	up working files	rcsclean(1)
rcsclean: clean	update access and modification times of	touch(1)
a file /touch:	update an ELF descriptor	elf_update(3E)
elf_update:	update, and regenerate groups of	make(1)
programs make: maintain,	update	lsearch(3C)
lsearch, lfind: linear search and	update the super-block	sync(1M)
sync:	update_panels: panels virtual screen	panel_update(3X)
refresh routine /panel_update:	upon presentation of a signal	signal(2)
signal: specify what to do	upon presentation of a signal	sigset(2)
sigset: specify what to do	upon presentation of a signal	sigvec(2)
sigvec: specify what to do	usage	du(1)
du: summarize disk		

retrieve a command description and by process key	usage examples /usage:	usage(1)
/dg_file_info: get file and usage examples	usage information for process identified	dg_file_info(2)
vtimes: get information about resource	usage: retrieve a command description	usage(1)
mkmsgs: create message files for	usage	vtimes(3C)
devfree: release devices from exclusive	use by gettxt	mkmsgs(1)
devreserv: reserve devices for exclusive	use	devfree(1M)
kbdpipe:	use	devreserv(1M)
/idi_log, idi_warning: tools for	use the KBD module in a pipeline	kbdpipe(1)
clock: report CPU time	use with the interface description/	idi_tools(1)
days /holidays: accounting information	used	clock(3C)
of severity levels for application to be	used to distinguish prime and non-prime	holidays(4)
lpfilter: administer filters	used with fmtmsg /build list	addseverity(3C)
lpforms: administer forms	used with the LP print service	lpfilter(1M)
/curs_util: unctrl, keyname, filter,	used with the LP print service	lpforms(1M)
logins: list	use_env, putwin, getwin, delay_output,/	curs_util(3X)
setcontext: get and set current	user and system login information	logins(1M)
ucontext:	user context /getcontext,	getcontext(2)
/swapcontext: manipulate	user context	ucontext(5)
crontab:	user contexts	swapcontext(3C)
environ:	user crontab file	crontab(1)
getdate, getdate_err: convert	user environment	environ(5)
chown, lchown: change	user format date and time	getdate(3C)
fchown: change	user id and group id of a file	chown(2)
ckuid: prompt for and validate a	user id and group id of a file	fchown(2)
generate disk accounting data by	user ID	ckuid(1)
seteuid: set the effective	user id /diskusg:	diskusg(1M)
database admuser: manage	user id of the current process	seteuid(2)
fingerd, in.fingerd: remote	user information in the password	admuser(1M)
listusers: list	user information server	fingerd(1M)
useradd: administer a new	user login information	listusers(1)
logname: return login name of	user login on the system	useradd(1M)
/id: print the	user	logname(3X)
/cuserid: get character login name or	user name and ID, and group name and ID	id(1)
dispuid: display a list of all valid	user name associated with effective UID	cuserid(3S)
notify: notify	user names	dispuid(1)
last: indicate last	user of the arrival of new mail	notify(1)
su: become super-user or another	user or terminal logins	last(1)
a connection with another transport	user	su(1)
the slot in the utmp file of the current	user /t_connect: establish	t_connect(3N)
write: write to another	user /ttslot: find	ttslot(3C)
/svcfd_create, svcudp_create,	user	write(1)
the system	user2netname, xdr_accepted_reply,/	rpc(3N)
system	useradd: administer a new user login on	useradd(1M)
t_snddis: send	userdel: delete a user's login from the	userdel(1M)
information on the system	user-initiated disconnect request	t_snddis(3N)
text editor (variant of ex for casual	usermod: modify a user's login	usermod(1M)
information about local and remote	users) /edit:	edit(1)
userdel: delete a	users /finger: display	finger(1)
/usermod: modify a	user's login from the system	userdel(1M)
mail, rmail: read mail or send mail to	user's login information on the system	usermod(1M)
starter: information for beginning	users	mail(1)
wall: write to all	users	starter(1)
which: locate a program file for csh(1)	users	wall(1M)
call	users	which(1)
ustat: data returned by the	ustat: data returned by the ustat system	ustat(5)
ustat: get file system device statistics	ustat: get file system device statistics	ustat(2)
ustat system call	ustat system call	ustat(5)
utility program	utility program	syacdb(1M)
flushinp: miscellaneous curses	utility routines /getwin, delay_output,	curs_util(3X)
get information about resource	utilization /getrusage:	getrusage(2)
times	utime: set file access and modification	utime(2)
times	utimes: set file access and modification	utimes(2)
utmp, wtmp:	utmp and wtmp entry formats	utmp(4)
setutent, endutent, utmpname: access	utmp file entry /getutline, pututline,	getut(3C)
ttslot: find the slot in the	utmp file of the current user	ttslot(3C)
/pututline, setutent, endutent,	utmp, wtmp: utmp and wtmp entry formats	utmp(4)
permissions file	utmpname: access utmp file entry	getut(3C)
uucp system	uucp directories and permissions file	uucpcheck(1M)
uucp system	uucico: file transport program for the	uucico(1M)
uucp system	uucleanup: uucp spool directory clean-up	uucleanup(1M)
uucp system	uucp directories and permissions file	uucpcheck(1M)

uusched: the scheduler for the	uucp file transport program	uusched(1M)
uucleanup:	uucp spool directory clean-up	uucleanup(1M)
uustat:	uucp status inquiry and job control	uustat(1)
uucico: file transport program for the	uucp system	uucico(1M)
copy	uucp, uulog, uuname: UNIX-to-UNIX system	uucp(1)
for transmission via mail uencode,	uudecode: encode/decode a binary file	uencode(1)
binary file for transmission via mail	uencode, uudecode: encode/decode a	uencode(1)
/uucp,	uulog, uuname: UNIX-to-UNIX system copy	uucp(1)
uucp, uulog,	uuname: UNIX-to-UNIX system copy	uucp(1)
copy /uuto,	uupick: public UNIX-to-UNIX system file	uuto(1)
transport program	uusched: the scheduler for the uucp file	uusched(1M)
control	uustat: uucp status inquiry and job	uustat(1)
file copy	uuto, uupick: public UNIX-to-UNIX system	uuto(1)
debugging on	Uutry: try to contact remote system with	uutry(1M)
execution	uux: UNIX-to-UNIX system command	uux(1)
	uuxqt: execute remote command requests	uuxqt(1M)
module /ttcompat:	V7, 4BSD and XENIX STREAMS compatibility	ttcompat(7)
incoming mail messages	vacation: automatically respond to	vacation(1)
	val: validate SCCS file	val(1)
/ckdate, errdate, helpdate,	valdate: prompt for and validate a date	ckdate(1)
id /ckgid, errgid, helpgid,	valgid: prompt for and validate a group	ckgid(1)
dispgid: display a list of all	valid group names	dispgid(1)
dispuid: display a list of all	valid user names	dispuid(1)
helpdate, valdate: prompt for and	validate a date /ckdate, errdate,	ckdate(1)
errgid, helpgid, valgid: prompt for and	validate a group id /ckgid,	ckgid(1)
ckkeywd: prompt for and	validate a keyword	ckkeywd(1)
ckuid: prompt for and	validate a user ID	ckuid(1)
ckrange: prompt for and	validate an integer	ckrange(1)
val:	validate SCCS file	val(1)
ckyorn: prompt for and	validate yes/no	ckyorn(1)
field_arg: forms field data type	validation /set_field_type, field_type,	form_field_validation(3X)
valtools: introduction to	validation tools	valtools(1)
malloc, free, realloc, calloc, memalign,	valloc.: memory allocator	malloc(3C)
tools	valtools: introduction to validation	valtools(1)
pdp11, u3b, u3b5, vax: provide truth	value about your processor type /i386,	machid(1)
abs, labs: return integer absolute	value	abs(3C)
a prompt; verify and return an integer	value /ckint: display	ckint(1)
elf_hash: compute hash	value	elf_hash(3E)
getenv: return	value for environment name	getenv(3C)
floor, ceiling, remainder, absolute	value functions /fabsf, rint, remainder:	floor(3M)
getitimer, setitimer: get or set	value of interval timer	getitimer(2)
putenv: change or add	value to environment	putenv(3C)
htonl, htons, ntohl, ntohs: convert	values between host and network byte/	byteorder(3N)
	values: machine-dependent values	values(5)
item_value: set and get menus item	values /set_item_value,	menu_item_value(3X)
fpathconf: get configurable pathname	values /pathconf,	pathconf(2)
pkgparam: displays package parameter	values	pkgparam(1)
sysconf: get configurable system	values	sysconf(2)
true, false: provide truth	values	true(1)
values: machine-dependent	values	values(5)
vscanf: convert formatted input using	varargs argument list /vscanf, vfscanf,	vscanf(3S)
	varargs: handle variable argument list	varargs(5)
stdarg: handle	variable argument list	stdarg(5)
varargs: handle	variable argument list	varargs(5)
vsprintf: print formatted output of a	variable argument list /vfprintf,	vsprintf(3S)
vsprintf: print formatted output of a	variable argument list /vfprintf,	vsprintf(3W)
elink: Environment	variable sensitive file link	elink(5)
admnl: manipulate national language	variables	admnl(1M)
edit: text editor	(variant of ex for casual users)	edit(1)
/m68k, m88k, i386, pdp11, u3b, u3b5,	vax: provide truth value about your/	machid(1)
getopt: get option letter from argument	vc: version control	vc(1)
editor based on ex /vi, view,	vector	getopt(3C)
ckpath: display a prompt;	vedit: screen-oriented (visual) display	vi(1)
ckstr: display a prompt;	verify and return a pathname	ckpath(1)
cktime: display a prompt;	verify and return a string answer	ckstr(1)
ckint: display a prompt;	verify and return a time of day	cktime(1)
assert:	verify and return an integer value	ckint(1)
controller is operable vsccheck:	verify program assertion	assert(3X)
vc:	verify that the VSC synchronous	vsccheck(1M)
chgtnfo: create a temporary	version control	vc(1)
	version of a TERMINFO entry	chgtnfo(1)

get: check out a	version of an SCCS file	get(1)
default-gcc: set or query default	version of GNU C	default-gcc(1)
coordinate library and application	versions /elf_version:	elf_version(3E)
compver: compatible	versions file	compver(4)
sccsdiff: compare two	versions of an SCCS file	sccsdiff(1)
create curses borders, horizontal and	vertical lines /box, whline, wvline:	curs_border(3X)
memory efficient way	vfork: spawn new process in a virtual	vfork(2)
output of a variable argument/ vprintf,	vfprintf, vsprintf: print formatted	vprintf(3S)
output of a variable argument/ vprintf,	vfprintf, vsprintf: print formatted	vprintf(3W)
input using varargs argument/ vscanf,	vfscanf, vsscanf: convert formatted	vscanf(3S)
control terminal	vhangup: virtually hang up the current	vhangup(2)
(visual) display editor based on ex	vi, view, vedit: screen-oriented	vi(1)
a binary file for transmission	via mail /uudecode: encode/decode	uencode(1)
make a directory available for mounting	via NFS /exportfs:	exportfs(2)
nlsgtcall: get client's data passed	via the listener	nlsgtcall(3N)
tigetstr:/ tparm, tputs, putp, vidputs,	vidattr, mvcur, tigetflag, tigetnum,	curs_terminfo(3X)
/restartterm, tparm, tputs, putp,	vidputs, vidattr, mvcur, tigetflag,/	curs_terminfo(3X)
point directory /cpd: change or	view the allocation limits for a control	cpd(1)
display editor based on ex /vi,	view, vedit: screen-oriented (visual)	vi(1)
vitr:	Vilya TokenRing Controller interface	vitr(7)
vfork: spawn new process in a	vipw: edit the system password file	vipw(1M)
move_panel: move a panels window on the	virtual memory efficient way	vfork(2)
/panel_update: update_panels: panels	virtual screen /panel_move:	panel_move(3X)
terminal vhangup:	virtual screen refresh routine	panel_update(3X)
item_visible: tell if menus item is	virtually hang up the current control	vhangup(2)
vi, view, vedit: screen-oriented	visible /menu_item_visible:	menu_item_visible(3X)
interface	(visual) display editor based on ex	vi(1)
consumption	vitr: Vilya TokenRing Controller	vitr(7)
label checking	vlimit: control maximum system resource	vlimit(3C)
tlabel: initialize a tape with a	volcopy, labelit: copy file systems with	volcopy(1M)
formatted output of a variable argument/	volume label	tlabel(1)
formatted output of a variable argument/	vprintf, vfprintf, vsprintf: print	vprintf(3S)
/vsccheck: verify that the	vprintf, vfprintf, vsprintf: print	vprintf(3W)
download board resident software onto	VSC synchronous controller is operable	vsccheck(1M)
formatted input using varargs argument/	VSC synchronous controller /vsclod:	vsclod(1M)
synchronous controller is operable	vscanf, vsscanf, vsscanf: convert	vscanf(3S)
software onto VSC synchronous/	vsccheck: verify that the VSC	vsccheck(1M)
variable argument/ vprintf, vfprintf,	vsclod: download board resident	vsclod(1M)
variable argument/ vprintf, vfprintf,	vsprintf: print formatted output of a	vprintf(3S)
varargs argument list /vscanf, vfscanf,	vsprintf: print formatted output of a	vprintf(3W)
vtc.addr: SYAC	vsscanf: convert formatted input using	vscanf(3S)
usage	VTC configuration file	vtc.addr(4M)
/printw, wprintw, mvprintw, mvwprintw,	vtc.addr: SYAC VTC configuration file	vtc.addr(4M)
/scanw, wscanw, mvscanw, mvwscanw,	vtimes: get information about resource	vtimes(3C)
wechochar: add a/ curs_addch: addch,	vwprintw: print formatted output in/	curs_printw(3X)
/addchstr, addchnstr, waddchstr,	vwscanw: convert formatted input from a/	curs_scanw(3X)
/addchstr, addchnstr, waddchstr,	waddch, mvaddch, mvwaddch, echochar,	curs_addch(3X)
/curs_addchstr: addchstr, addchnstr,	waddchnstr, mvaddchstr, mvaddchnstr,/	curs_addchst(3X)
/curs_addchstr: addchstr, addchnstr,	waddchnstr, mvaddchstr, mvaddchnstr,/	curs_addchstr(3X)
/curs_addstr: addstr, addnstr, waddstr,	waddchstr, waddchnstr, mvaddchstr,/	curs_addchst(3X)
/addwstr, addnwstr, waddwstr,	waddchstr, waddchnstr, mvaddchstr,/	curs_addchstr(3X)
/curs_addstr: addstr, addnstr,	waddnstr, mvaddstr, mvaddnstr,/	curs_addstr(3X)
wchowchar: add a/ /curs_addwch: addwch,	waddnwstr, mvaddwstr, mvaddnwstr,/	curs_addwstr(3X)
/addwchstr, addwchnstr, waddwchstr,	waddstr, waddnstr, mvaddstr, mvaddnstr,/	curs_addstr(3X)
/curs_addwchstr: addwchstr, addwchnstr,	waddwch, mvaddwch, mvwaddwch, echochar,	curs_addwch(3X)
/curs_addwstr: addwstr, addnwstr,	waddwchnstr, mvaddwchstr, mvaddwchnstr,/	curs_addwchstr(3X)
	waddwchstr, waddwchnstr, mvaddwchstr,/	curs_addwchstr(3X)
	waddwstr, waddnwstr, mvaddwstr,/	curs_addwstr(3X)
	wait: await completion of process	wait(1)
sigsuspend:	wait for a signal	sigsuspend(2)
/waitid:	wait for child process to change state	waitid(2)
terminate wait3:	wait for child process to stop or	wait3(2)
requests to complete /dg_lock_wait:	wait for previously delayed lock	dg_lock_wait(2)
wait, waitpid:	wait for process termination	wait(2)
stop or terminate /wait4:	wait for the specified child process to	wait4(2)
wstat:	wait status	wstat(5)
termination	wait, waitpid: wait for process	wait(2)
terminate	wait3: wait for child process to stop or	wait3(2)
process to stop or terminate	wait4: wait for the specified child	wait4(2)
state	waitid: wait for child process to change	waitid(2)
wait,	waitpid: wait for process termination	wait(2)

ftw, nftw:	walk a file tree	ftw(3C)
	wall: write to all users	wall(1M)
	wattroff, attron, wattron, attrset,	curs_attr(3X)
curs_attr: attron, wattroff, attron,	wattron, attrset, wattrset, standend,/	curs_attr(3X)
/wattroff, attron, wattron, attrset,	wattrset, standend, wstandend, standout,/	curs_attr(3X)
process in a virtual memory efficient	way /vfork: spawn new	vfork(2)
curs_bkgd: bkgdset, wbkgdset, bkgd,	wbkgd: curses window background/	curs_bkgd(3X)
background/ curs_bkgd: bkgdset,	wbkgdset, bkgd, wbkgd: curses window	curs_bkgd(3X)
curses borders,/ /curs_border: border,	wborder, box, whline, wvline: create	curs_border(3X)
	wc: word count	wc(1)
	wchar_t character back into input stream	ungetwc(3W)
/winswch, mvinswch, mvwinswch: insert a	wchar_t character before the character/	curs_inswch(3X)
/inwch, winwch, mvinwch, mvwinwch: get a	wchar_t character from a curses window	curs_inwch(3X)
getwc, getwchar, fgetwc: get	wchar_t character from a stream	getwc(3W)
putwc, putwchar, fputwc: put	wchar_t character on a stream	putwc(3W)
/mvgetwstr, mvwgetwstr, mvwgetwstr: get	wchar_t character strings from curses/	curs_getwstr(3X)
/mvwaddwch, echowchar, wechowchar: add a	wchar_t character to a curses window	curs_addwch(3X)
/mvwinnchnstr: get a string of	wchar_t characters from a curses window	curs_inwchstr(3X)
/mvwinnwstr, mvwinnwstr: get a string of	wchar_t characters from a curses window	curs_inwstr(3X)
/mvwgetwch, ungetwch: get (or push back)	wchar_t characters from curses terminal/	curs_getwch(3X)
/mvwaddwchnstr: add string of	wchar_t characters to a curses window	curs_addwchstr(3X)
/mvwaddwstr, mvwaddwstr: add a string of	wchar_t characters to a curses window	curs_addwstr(3X)
the/ /mvwinswstr, mvwinswstr: insert	wchar_t string before character under	curs_inswstr(3X)
getws, fgetws: get a	wchar_t string from a stream	getws(3W)
putws, fputws: put a	wchar_t string on a stream	putws(3W)
/wssp, wscspn, wstok, wstocr, strtows:	wchar_t string operations and type/	wstring(3W)
classification and conversion tables	wchrtbl: generate character	wchrtbl(1M)
/curs_clear: erase, werase, clear,	wclear, clrtoeb, wclrtoeb, clrtoeol,/	curs_clear(3X)
/erase, werase, clear, wclear, clrtoeb,	wclrtoeb, clrtoeol, wclrtoeol: clear/	curs_clear(3X)
/wclear, clrtoeb, wclrtoeb, clrtoeol,	wclrtoeol: clear all or part of a curses/	curs_clear(3X)
characters	wconv: towupper, tolower: translate	wconv(3W)
mbstring: mbstowcs,	wctombs: multibyte string functions	wctombs(3C)
conversion mbchar: mbtowc,	wctomb, mblen: multibyte character	mbchar(3W)
mbchar: mbtowc, mblen,	wctomb: multibyte character handling	mbchar(3C)
mbstring: mbstowcs,	wctombs: multibyte string conversion	mbstring(3W)
iswdigit, iswxdigit, iswalnum,/	wctype: iswalnum, iswupper, iswlower,	wctype(3W)
/mvderwin, dupwin, wsyncup, syncok,	wcursyncup, wsyncdown: create curses/	curs_window(3X)
character under/ curs_delch: delch,	wdelch, mvdelch, mvwdelch: delete	curs_delch(3X)
insertln,/ /curs_deleteln: deleteln,	wdeleteln, insdelln, winsdelln,	curs_deleteln(3X)
waddch, mvaddch, mvwaddch, echowchar,	wechochar: add a character (with/ /addch,	curs_addch(3X)
/waddwch, mvaddwch, mvwaddwch, echowchar,	wchowchar: add a wchar_t character to a/	curs_addwch(3X)
wclrtoeb,/ curs_clear: erase,	werase, clear, wclear, clrtoeb,	curs_clear(3X)
(or push back)/ /curs_getch: getch,	wgetch, mvgetch, mvwgetch, ungetch: get	curs_getch(3X)
/curs_getstr: getstr, getnstr, wgetstr,	wgetnstr, mvgetstr, mvgetnstr,/	curs_getstr(3X)
/getwstr, getnwstr, wgetwstr,	wgetnwstr, mvgetwstr, mvgetnwstr,/	curs_getwstr(3X)
/curs_getstr: getstr, getnstr,	wgetstr, wgetnstr, mvgetstr, mvgetnstr,/	curs_getstr(3X)
get (or push/ /curs_getwch: getwch,	wgetwch, mvgetwch, mvwgetwch, ungetwch:	curs_getwch(3X)
/curs_getwstr: getwstr, getnwstr,	wgetwstr, wgetnwstr, mvgetwstr,/	curs_getwstr(3X)
	what: identify SCCS files	what(1)
/signal: specify	what to do upon presentation of a signal	signal(2)
/sigset: specify	what to do upon presentation of a signal	sigset(2)
/sigvec: specify	what to do upon presentation of a signal	sigvec(2)
/crash:	what to do when the DG/UX system crashes	crash(8)
whodo: who is doing	what	whodo(1M)
a topic	whatis: display a one-line summary about	whatis(1)
crash: what to do	when the DG/UX system crashes	crash(8)
manual for program	whereis: locate source, binary, and or	whereis(1)
/isencrypt: determine	whether a character buffer is encrypted	isencrypt(3G)
messages /ckbinarsys: determine	whether remote system can accept binary	ckbinarsys(1M)
criteria getdgrp: lists device groups	which contain devices that match	getdgrp(1M)
users	which: locate a program file for csh(1)	which(1)
/curs_border: border, wborder, box,	whline, wvline: create curses borders,/	curs_border(3X)
whodo: who is doing what	who is doing what	whodo(1M)
who:	who is on the system	who(1)
	who: who is on the system	who(1)
	whodo: who is doing what	whodo(1M)
	widec: multibyte character I/O routines	widec(3W)
convert formatted input from a curses	widow /mvscanw, mvwscanw, vwscanw:	curs_scanw(3X)
fold: fold long lines for finite	width output device	fold(1)
eucset: set or get EUC code set	widths	eucset(1)
and its attributes/ /curs_inch: inch,	winch, mvinch, mvwinch: get a character	curs_inch(3X)

/inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvinchnstr, / curs_inchstr(3X)
 /curs_inchstr: inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvinchnstr, / curs_inchstr(3X)
 add a string of characters to a curses routine /form_sub, scale_form: forms window and subwindow association form_win(3X)
 routines /menu_sub, scale_menu: menus window and subwindow association menu_win(3X)
 /wstandout: curses character and window attribute control routines curs_attr(3X)
 /bkgdset, wbkgdset, bkgd, wbkgd: curses window background manipulation routines curs_bkgd(3X)
 getmaxyx: get curses cursor and window coordinates /getparyx, getbegyx, curs_getyx(3X)
 character (with attributes) to a curses window /echochar, wechochar: add a curs_addch(3X)
 characters (and attributes) to a curses window /mvwaddchnstr: add string of curs_addchstr(3X)
 characters (and attributes) to a curses window /mvwaddchnstr: add string of curs_addchstr(3X)
 add a wchar_t character to a curses window /echochar, wechochar: curs_addwch(3X)
 string of wchar_t characters to a curses window /mvwaddwchstr, mvwaddwchnstr: add curs_addwchstr(3X)
 string of wchar_t characters to a curses window /mvwaddwstr, mvwaddnwstr: add a curs_addwstr(3X)
 character wclrtoeol: clear all or part of a curses window /clrtoeb, wclrtoeb, clrtoeol, curs_clear(3X)
 character under the cursor in a curses window. /mvdelch, mvwdelch: delete curs_mvdelch(3X)
 delete and insert lines in a curses window /winsdelln, insertln, winsertln: curs_deleteln(3X)
 and its attributes from a curses window /mvinch, mvwinch: get a character curs_inch(3X)
 (and attributes) from a curses window /get a string of characters curs_inchstr(3X)
 character under the cursor in a curses window /insert a character before the curs_insch(3X)
 character under the cursor in a curses window /mvwinsnstr: insert string before curs_insstr(3X)
 get a string of characters from a curses window /mvinnstr, mvwinstr, mvwinnstr: curs_instr(3X)
 character under the cursor in a curses window /a wchar_t character before the curs_inswch(3X)
 character under the cursor in a curses window /insert wchar_t string before curs_inswstr(3X)
 get a wchar_t character from a curses window /winwch, mvwinwch, mvwinwch: curs_inwch(3X)
 of wchar_t characters from a curses window /mvwinwchnstr: get a string curs_inwchstr(3X)
 of wchar_t characters from a curses window /mvwinnwstr: get a string curs_inwstr(3X)
 curs_move: move, wmove: move curses window cursor curs_move(3X)
 pos_form_cursor: position forms window cursor /form_cursor: form_cursor(3X)
 scroll, srcl, wscl: scroll a curses window /curs_scroll: curs_scroll(3X)
 replace_panel: get or set the current window of a panels panel /panel_window, panel_window(3X)
 panel_move: move_panel: move a panels window on the virtual screen panel_move(3X)
 redrawwin, wredrawln: refresh curses windows and lines /doupdate, curs_refresh(3X)
 overlap and manipulate overlapped curses windows /overlay, overwrite, copywin: curs_overlay(3X)
 print formatted output in curses windows /mvprintw, mvwprintw, vwprintw: curs_printw(3X)
 wcursyncup, wsyncdown : create curses windows /dupwin, wsyncup, syncok, curs_window(3X)
 curs_instr: instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, / curs_instr(3X)
 /curs_inwstr: inwstr, innwstr, winwstr, winnwstr, mvinnwstr, / curs_inwstr(3X)
 character before/ curs_insch: insch, wish, mvinsch, mvwinsch: insert a curs_insch(3X)
 and/ /deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln: delete curs_deleteln(3X)
 curses/ /insdelln, winsdelln, insertln, winsertln: delete and insert lines in a curs_deleteln(3X)
 /curs_instr: insstr, insnstr, winsstr, winsnstr, mvinsnstr, / curs_insstr(3X)
 /inswstr, insnwstr, winswstr, winsnwstr, mvinsnwstr, / curs_inswstr(3X)
 /curs_instr: insstr, insnstr, winsstr, winsnstr, mvinsnstr, / curs_insstr(3X)
 mvwinstr, / curs_instr: instr, innstr, winstr, winnstr, mvinstr, mvinnstr, curs_instr(3X)
 wchar_t character/ /curs_inswch: inswch, insnwch, mvinsnwch: insert a curs_inswch(3X)
 /curs_instr: inswstr, insnwstr, winwstr, winnwstr, mvinnwstr, / curs_inswstr(3X)
 character from a / /curs_inwch: inwch, innwch, mvinnwch: get a wchar_t curs_inwch(3X)
 /inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchnstr, / curs_inwchstr(3X)
 /curs_inwchstr: inwchstr, inwchnstr, winwchstr, winwchnstr, mvinnwchnstr, / curs_inwchstr(3X)
 /curs_inwstr: inwstr, innwstr, winwstr, winnwstr, mvinnwstr, / curs_inwstr(3X)
 /echochar, wechochar: add a character (with attributes) to a curses window curs_addch(3X)
 prof: profile within a function prof(5)
 unix_ipc: piping communications within a host unix_ipc(6F)
 /synchronously read data from a file without system buffering dg_unbuffered_read(2)
 /synchronously write data to a file without system buffering dg_unbuffered_write(2)
 curs_move: move, wmove: move curses window cursor curs_move(3X)
 Multiple optical device) as magtape/ wmt: pseudo WORM (Write Once Read wmt(7)
 device server wmt: start the WORM magnetic tape wmt(1M)
 /curs_refresh: refresh, wrefresh, wnoutrefresh, doupdate, redrawwin, / curs_refresh(3X)
 wc: word count wc(1)
 getchar, fgetc, getw: get character or word from a stream /getc,getc(3S)
 putchar, fputc, putw: put character or word on a stream /putc,putc(3S)
 cd: change working directory cd(1)
 getcwd: get pathname of current working directorygetcwd(3C)
 pwd: print working directory namepwd(1)
 /chdir: change the working directory of the calling processchdir(2)
 /fchdir: change the working directory of the calling processfchdir(2)
 getwd: get current working directory pathnamegetwd(3C)
 rcsclean: clean up working filesrcsclean(1)
 grfx: AViiON series workstation graphics processorgrfx(7)

kbd: AViiON series workstation system keyboard	kbd(7)
wmtd: start the WORM magnetic tape device server	wmtd(1M)
device) as magtape/ /wmt: pseudo WORM (Write Once Read Multiple optical	wmt(7)
print formatted/ /curs_printw: printw, wprintw, mvprintw, mvwprintw, vwprintw:	curs_printw(3X)
/wnoutrefresh, douppdate, redrawwin, wredrawln: refresh curses windows and/	curs_refresh(3X)
redrawwin,/ /curs_refresh: refresh, wrefresh, wnoutrefresh, douppdate,	curs_refresh(3X)
/scr_restore, scr_init, scr_set: read (write) a curses screen from (to) a file	curs_scr_dump(3X)
/dg_unbuffered_write: synchronously write data to a file without system/	dg_unbuffered_write(2)
dump2label: read and write labels for dump tapes	dump2label(1M)
writev: write on a file	writev(2)
device) as magtape/ wmt: pseudo WORM (Write Once Read Multiple optical	wmt(7)
form_post: post_form, unpost_form: write or erase forms from associated/	form_post(3X)
menu_post: post_menu, unpost_menu: write or erase menus from associated/	menu_post(3X)
putpwent: write password file entry	putpwent(3C)
putspent: write shadow password file entry	putspent(3C)
wall: write to all users	wall(1M)
write: write to an object	write(2)
write: write to another user	write(1)
write: write to an object	write(2)
write: write to another user	write(1)
twrite: writes a file to tape	twrite(1)
writev: write on a file	writev(2)
writing IBM and ANSI tapes	reelexchange_intro(1)
writing	open(2)
/REELexchange: commands for reading and open: open file for reading or	curs_scanw(3X)
convert formatted/ curs_scanw: scanw, wscanw, mvscanw, mvwscanw, vwscanw:	wstring(3W)
wsncpy, wslen, wschr, wsrchr,/ wstring: wscat, wscat, wscmp, wscmp, wscopy, wscopy,	wstring(3W)
/wscmp, wscmp, wscopy, wscopy, wslen, wschr, wsrchr,/ wstring: wscat, wscat, wscmp,	wstring(3W)
wscmp, wscmp, wscopy, wscopy, wslen, wscopy, wscopy, wslen, wschr, wsrchr,/	wstring(3W)
wstring: wscat, wscat, wscmp, wscmp, wscmp, wscmp, wscmp, wscmp,	curs_scroll(3X)
/curs_scroll: scroll, srcl, wscrl: scroll a curses window	wstring(3W)
/wslen, wschr, wsrchr, wspbrk, wsspn, wscspn, wstok, wstostr, strtows: wchar_t/	wstring_outopts(3X)
/idcok immedok, leaveok, setscreg, wssetscreg, scrollok, nl, nonl: curses/	wstring(3W)
/wscat, wscmp, wscmp, wscopy, wscopy, wslen, wschr, wsrchr,/ wstring: wscat,	wstring(3W)
wscat, wscmp, wscmp, wscopy, wscopy, wslen, wschr, wsrchr,/ wstring: wscat,	wstring(3W)
wscat, wscat, wscmp, wscmp, wscopy, wscopy, wslen, wschr, wsrchr,/	wstring(3W)
/wscopy, wscopy, wslen, wschr, wsrchr, wsrchr, wsrchr, wsrchr, wsrchr,	wstring(3W)
/wscmp, wscopy, wscopy, wslen, wschr, wsrchr, wsrchr, wsrchr, wsrchr,	wstring(3W)
/wscopy, wslen, wschr, wsrchr, wsrchr, wsrchr, wsrchr, wsrchr, wsrchr,	wstring(3W)
/wattron, attrset, wattrset, standend, wstandend, standend, standend,	curs_attr(3X)
/wattrset, standend, wstandend, standend, wstat: wait status	curs_attr(3X)
wstat: wait status	wstat(5)
wstok, wstostr, strtows: wchar_t string/	wstring(3W)
wstostr, strtows: wchar_t string/ /wschr, wstring: wscat, wscat, wscmp, wscmp,	wstring(3W)
wstring: wscat, wscat, wscmp, wscmp, wscopy, wscopy,	wstring(3W)
wsyncdown : create curses windows	curs_window(3X)
wsyncup, syncok, wcursyncup, wsyncdown :/	curs_window(3X)
wtimetype, typeahead: curses terminal/	curs_inopts(3X)
wtmp entry formats	utmp(4)
wtmp: utmp and wtmp entry formats	utmp(4)
wtmpfix: manipulate connect accounting	fwtmp(1M)
wtouchln, is_linetouched, is_wintouched:	curs_touch(3X)
wvline: create curses borders,/	curs_border(3X)
X display terminals	admterminal(1M)
xargs: construct argument list(s) and	xargs(1)
xdr_accepted_reply, xdr_authunix_parms,/	rpc(3N)
xdr_array, xdr_bool, xdr_bytes,	xdr(3N)
xdr_authunix_parms, xdr_callhdr,/	rpc(3N)
xdr_bool, xdr_bytes, xdr_char,	xdr(3N)
xdr_bytes, xdr_char, xdr_destroy,	xdr(3N)
xdr_callhdr, xdr_callmsg,/ /user2netname,	rpc(3N)
xdr_callmsg, xdr_opaque_auth, xdr_pmap,/	rpc(3N)
xdr_char, xdr_destroy, xdr_double,/	xdr(3N)
xdr_destroy, xdr_double, xdr_int,/	xdr(3N)
xdr_double, xdr_int, xdr_long,/	xdr(3N)
xdr_int, xdr_long, xdrmem_create,/	xdr(3N)
xdr_long, xdrmem_create, xdr_opaque,/	xdr(3N)
xdrmem_create, xdr_opaque, xdr_pointer,/	xdr(3N)
xdr_opaque, xdr_pointer, xdrrec_create,/	xdr(3N)
xdr_opaque_auth, xdr_pmap, xdr_pmaplist,/	rpc(3N)
xdr_pmap, xdr_pmaplist,/ /xdr_callhdr,	rpc(3N)

/xdr_callmsg, xdr_opaque_auth, xdr_pmap,	xdr_pmaplist, xdr_rejected_reply,/	rpc(3N)
xdr_long, xdrmem_create, xdr_opaque,	xdr_pointer, xdrrec_create,/ /xdr_int,	xdr(3N)
/xdrmem_create, xdr_opaque, xdr_pointer,	xdrrec_create, xdrrec_endofrecord,/	xdr(3N)
/xdr_opaque, xdr_pointer, xdrrec_create,	xdrrec_endofrecord, xdrrec_eof,/	xdr(3N)
/xdrrec_create, xdrrec_endofrecord,	xdrrec_eof, xdrrec_skiprecord,/	xdr(3N)
/xdrrec_endofrecord, xdrrec_eof,	xdrrec_skiprecord, xdr_reference,/	xdr(3N)
/xdrrec_eof, xdrrec_skiprecord,	xdr_reference, xdr_setpos, xdr_short,/	xdr(3N)
/xdr_opaque_auth, xdr_pmap, xdr_pmaplist,	xdr_rejected_reply, xdr_replymsg,/	rpc(3N)
xdr_pmaplist, xdr_rejected_reply,	xdr_replymsg, xpvt_register,/ /xdr_pmap,	rpc(3N)
/xdrrec_skiprecord, xdr_reference,	xdr_setpos, xdr_short, xdrstdio_create,/	xdr(3N)
xdr_u_char,/ /xdr_reference, xdr_setpos,	xdr_short, xdrstdio_create, xdr_string,	xdr(3N)
/xdr_reference, xdr_setpos, xdr_short,	xdrstdio_create, xdr_string, xdr_u_char,/	xdr(3N)
/xdr_setpos, xdr_short, xdrstdio_create,	xdr_string, xdr_u_char, xdr_u_int,/	xdr(3N)
/xdr_short, xdrstdio_create, xdr_string,	xdr_u_char, xdr_u_int, xdr_u_long,/	xdr(3N)
/xdrstdio_create, xdr_string, xdr_u_char,	xdr_u_int, xdr_u_long, xdr_u_short,/	xdr(3N)
/xdr_string, xdr_u_char, xdr_u_int,	xdr_u_long, xdr_u_short, xdr_union,/	xdr(3N)
/xdr_u_int, xdr_u_long, xdr_u_short,	xdr_union, xdr_vector, xdr_void,/	xdr(3N)
/xdr_u_char, xdr_u_int, xdr_u_long,	xdr_u_short, xdr_union, xdr_vector,/	xdr(3N)
/xdr_u_long, xdr_u_short, xdr_union,	xdr_vector, xdr_void, xdr_wrapstring:/	xdr(3N)
/xdr_u_short, xdr_union, xdr_vector,	xdr_void, xdr_wrapstring: library/	xdr(3N)
/xdr_union, xdr_vector, xdr_void,	xdr_wrapstring: library routines for/	xdr(3N)
ttcompat: V7, 4BSD and	XENIX STREAMS compatibility module	ttcompat(7)
lpprint,	xlpprint: menu-driven lp interface	lpprint(1M)
/xdr_rejected_reply, xdr_replymsg,	xpvt_register, xpvt_unregister: library/	rpc(3N)
remote/ /xdr_replymsg, xpvt_register,	xpvt_unregister: library routines for	rpc(3N)
from C, Fortran and Pascal sources	xref: generate cross reference table	xref(1)
implement shared strings	xstr: extract strings from C programs to	xstr(1)
administration/ sysadm, asysadm,	xsysadm: menu-driven system	sysadm(1M)
bessel: j0, j1, jn,	y0, y1, yn: Bessel functions	bessel(3M)
bessel: j0, j1, jn, y0,	y1, yn: Bessel functions	bessel(3M)
ckyor: prompt for and validate	yacc: yet another compiler-compiler	yacc(1)
bessel: j0, j1, jn, y0, y1,	yes/no	ckyor(1)
yp_unbind, yp_match, yp_first, yp_next,	yn: Bessel functions	bessel(3M)
yp_next,/ /ypclnt, yp_get_default_domain,	yp_all, yp_order, yp_master,/ /yp_bind,	ypclnt(3N)
yp_unbind, yp_match, yp_first, yp_next,/	yp_bind, yp_unbind, yp_match, yp_first,	ypclnt(3N)
/yp_next, yp_all, yp_order, yp_master,	ypclnt, yp_get_default_domain, yp_bind,	ypclnt(3N)
/yp_bind, yp_unbind, yp_match,	yperr_string, ypprot_err: Network/	ypclnt(3N)
yp_unbind, yp_match, yp_first,/ ypclnt,	yp_first, yp_next, yp_all, yp_order,/	ypclnt(3N)
/yp_first, yp_next, yp_all, yp_order,	yp_get_default_domain, yp_bind,	ypclnt(3N)
/yp_bind, yp_unbind,	yp_master, yperr_string, ypprot_err:/	ypclnt(3N)
/yp_bind, yp_unbind,	yp_match, yp_first, yp_next, yp_all,	ypclnt(3N)
/yp_bind, yp_unbind, yp_match, yp_first,	yp_next, yp_all, yp_order, yp_master,/	ypclnt(3N)
/yp_match, yp_first, yp_next, yp_all,	yp_order, yp_master, yperr_string,/	ypclnt(3N)
/yp_order, yp_master, yperr_string,	ypprot_err: Network Information Service/	ypclnt(3N)
/ypclnt, yp_get_default_domain, yp_bind,	yp_unbind, yp_match, yp_first, yp_next,/	ypclnt(3N)
expanded files compress, uncompress,	zcat: compress, expand or display	compress(1)
bzero:	zdump: time zone dumper	zdump(1M)
zero: source of zeroes	zero a portion of memory	bzero(3C)
zero: source of	zero: source of zeroes	zero(7)
the system date, time and time	zeroes	zero(7)
timezone: set default system time	zic: time zone compiler	zic(1M)
zic: time	zone /admdate: manipulate	admdate(1M)
zdump: time	zone and locale	timezone(4)
	zone compiler	zic(1M)
	zone dumper	zdump(1M)

End of Chapter

Index

Note: Boldfaced page numbers (e.g., 1-5) indicate definitions of terms or other key information.

A

a.out(4) **4-3**
acct(4) **4-9**
aliases(4) **4-11**
ar(4) **4-14**
ascii(5) **5-3**

B

bootparams(4) **4-17**

C

CC environment variable **4-154**
CFTIME environment variable **5-13**
checklist(4) **4-18**
CHRCLASS environment variable **5-13**
compver(4) **4-19**
copyright(4) **4-20**
core(4) **4-21**
cpio(4) **4-22**

D

d_passwd(4) **4-23**
depend(4) **4-24**
dg_mknod(5) **5-4**
dg_stat(5) **5-6**
dialups(4) **4-26**
dirent(4) **4-27**
Documentation
 AViiON and DG/UX, Guide to **RD-1**
 related **RD-1**
dot3(6P) **6-5**
dumptab(4) **4-28**

E

elink(5) **5-9**
environ(5) **5-11**
Environment variable, *see* CC; CFTIME;
 CHRCLASS; HOME; LANG;
 LANGUAGE; LC_COLLATE;
 LC_CTYPE; LC_MESSAGES;
 LC_MONETARY; LC_NUMERIC;
 LC_TIME; LEGENDS; MAIL;
 MSGVERB; NETPATH; NLSPATH;
 PATH; PRINTER; SEV_LEVEL;
 TARGET_BINARY_INTERFACE;
 TERM; TERMCAP; TERMINFO; TZ
ethers(4) **4-29**
eucioctl(5) **5-17**
Executable file **4-3**
exports(4) **4-30**

F

fcntl(5) **5-19**
filehdr(4) **4-32**
fs(4) **4-33**
fspec(4) **4-39**
fstab(4) **4-40**

G

group(4) **4-43**

H

hfm(4) **4-45**
hier(5) **5-20**
holidays(4) **4-47**
HOME environment variable **4-103**
hostname(5) **5-25**
hosts(4) **4-48**

I

idl(4) **4-49**

inet(6F) 6-6
 inittab(4) 4-68
 inode(4) 4-71
 intro(4) 4-2
 intro(5) 5-2
 intro(6) 6-2
 ip(6P) 6-7
 issue(4) 4-76

L

LANG environment variable 4-176, 5-11
 langinfo(5) 5-26
 LANGUAGE environment variable 5-14
 LC_COLLATE environment variable 5-11
 LC_CTYPE environment variable 5-11
 LC_MESSAGES environment variable 5-12
 LC_MONETARY environment variable 5-12
 LC_NUMERIC environment variable 5-12
 LC_TIME environment variable 5-12
 ldfcn(4) 4-77
 legend(5) 5-28
 LEGENDS environment variable 5-28
 limits(4) 4-79
 linenum(4) 4-81
 Link editor output 4-3

M

MAIL environment variable 4-103
 master(4) 4-82
 math(5) 5-29
 mfs(4) 4-85
 misalign(5) 5-30
 mnttab(4) 4-87
 MSGVERB environment variable 5-12

N

netconfig(4) 4-89
 netgroup(4) 4-92
 NETPATH environment variable 4-89, 5-12
 networks(4) 4-93
 nfs(6P) 6-8
 nL_types(5) 5-33
 NLSPATH environment variable 4-176, 5-12

O

Object file 4-125

P

passwd(4) 4-94
 PATH environment variable 4-103, 5-13
 pkginfo(4) 4-97
 pkgmap(4) 4-100
 printcap(5) 5-34
 PRINTER environment variable 5-34
 prof(5) 5-36
 profile(4) 4-103
 protocols(4) 4-104
 prototype(4) 4-105
 publickey(4) 4-108

R

rcsfile(4) 4-109
 regexp(5) 5-37
 Related documents RD-1
 reloc(4) 4-112
 rpc(4) 4-113

S

sccsfile(4) 4-114
 scr_dump(4) 4-117
 sde(5) 5-41
 sde-chooser(4) 4-118
 sdetab(4) 4-119
 services(4) 4-120
 SEV_LEVEL environment variable 5-12
 siginfo(5) 5-43
 signal(5) 5-46
 snap(6P) 6-9
 space(4) 4-121
 stat(5) 5-47
 statd(4) 4-122
 statfs(5) 5-49
 stdarg(5) 5-51
 strftime(4) 4-123
 svcorder(4) 4-124
 Symbol table 4-125
 syms(4) 4-125
 syslog.conf(5) 5-53
 system(4) 4-128

T

tar(5) 5-55
 TARGET_BINARY_INTERFACE environment variable 5-10

tcp(6P) 6-10
TERM environment variable 4-129, 5-13
TERMCAP environment variable 5-62
termcap(5) 5-58
TERMINFO environment variable 4-140
terminfo(4) 4-129
timezone(4) 4-176
types(5) 5-72
TZ environment variable 4-176, 5-14

U

ucontext(5) 5-73
udp(6P) 6-12
unix_ipc(6F) 6-13
updaters(4) 4-179
ustat(5) 5-74
utmp(4) 4-180

V

values(5) 5-75
varargs(5) 5-76

W

wstat(5) 5-78

Y

ypfiles(4) 4-182

Related Documents

The following list of related manuals gives titles of Data General manuals followed by nine-digit numbers used for ordering. You can order any of these manuals via mail or telephone (see the TIPS Order Form in the back of this manual).

For a complete list of AViiON® and DG/UX™ manuals, see the *Guide to AViiON® and DG/UX™ Documentation* (069-701085). The on-line version of this manual found in `/usr/release/doc_guide` contains the most current list.

Data General Software Manuals

User's Manuals

User's Reference for the DG/UX™ System

Contains an alphabetical listing of DG/UX, TCP/IP, and ONC/NFS manual pages for commands relating to general system operation. Ordering Number — 093-701054

Using the DG/UX™ Editors

Describes the text editors `vi` and `ed`, the batch editor `sed`, and the command line editor `editread`. Ordering Number — 069-701036

Using the DG/UX™ System

Describes the DG/UX system and its major features, including the C and Bourne shells, typical user commands, the file system, and communications facilities such as `mailx`. Ordering Number — 069-701035

Installation and Administration Manuals

Managing ONC™/NFS® and Its Facilities on the DG/UX™ System

Explains how to manage and use the DG/UX ONC™/NFS® product. Contains information on the Network File System (NFS), the Network Information Service (NIS), Remote Procedure Calls (RPC), and External Data Representation (XDR). Ordering Number — 093-701049

System Manager's Reference for the DG/UX™ System

Contains an alphabetical listing of DG/UX, TCP/IP, and ONC/NFS manual pages for commands relating to system administration or operation. Ordering Number — 093-701050

Programming Manuals

Porting and Developing Applications on the DG/UX™ System

A compendium of useful information for experienced programmers developing or porting applications to the DG/UX™ system. It includes information on how to: set up your environment, use the software development tools, compile and link programs, port to the windowing environment, and build BCS applications. It also describes available debuggers and the various industry standards the DG/UX system supports. Ordering Number — 069-701059

Programmer's Guide: ANSI C and Programming Support Tools (UNIX System V Release 4)

Describes the standard tools of the UNIX program development environment including compiling, linking, debugging, and analysis and revision control. An accompanying supplement, *Supplement for Programmer's Guide: ANSI C and Programming Support Tools* (086-000180) describes the DG/UX system enhancements and differences. Ordering Number — 093-701104

Programmer's Guide: Systems Services and Application Packaging Tools (UNIX System V Release 4)

Describes standard programming procedures and interfaces available to the C application developer in the UNIX environment. Topics include interprocess communications, memory management, file and record locking and application packaging. **Note:** Chapters 5 and 9 of this Prentice Hall manual discuss topics that do not apply to the DG/UX system. Ordering Number — 093-701103

Programmer's Reference for the DG/UX™ System, (Volume 1)

Alphabetical listing of manual pages for DG/UX programming commands and system calls. This is part of a three-volume set. Ordering Number — 093-701055

Programmer's Reference for the DG/UX™ System, (Volume 2)

Alphabetical listing of manual pages for DG/UX and ONC/NFS subroutines and libraries. This is part of a three-volume set. Ordering Number — 093-701056

Programming with TCP/IP on the DG/UX™ System

Describes how to use the socket system calls and Transport Layer Interface (TLI library routines to access TCP, UDP, and IP protocol software. Ordering Number — 093-701024

End of Related Documents

DATA GENERAL CORPORATION TECHNICAL INFORMATION AND PUBLICATIONS SERVICE TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION

Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES

Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS

Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY

DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY

EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY

A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL

A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)

Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

**Programmer's
Reference for the
DG/UX™ System
(Volume 3)**

093-701102-01

Cut here and insert in binder spine pocket

