NAME

comments - display comments

SYNOPSIS

comments afile ...

DESCRIPTION

Comments will display all lines in the named files which contain all or part of a 'C' comment. The line number of the outputed lines is prepended to each displayed line.

FILES

BUGS

The program will not detect that a start of comment or end of comment pattern may be imbedded in quotes.

NAME

exl - examine lines

SYNOPSIS

exl [-n] afile linenum ... ...

DESCRIPTION

Exl copies selected lines from the named file, **afile** , to the standard output. The **linenum** arguments may be discrete line numbers or they may be a range of line numbers. The form for a range of numbers is m-n. Where m is less than n and there are no spaces in the argument.

Normally **exl** will prepend the line number to all lines displayed. This may be turned off by using the -n option.

The file name argument, afile, will accept one special case. If the file name is of the form -l<name>, then exl will build the path name /usr/lib/llib-l<name>. This is particularly useful to get lines referenced by lint diagnostics and has been provided explicitly for this purpose.

FILES

SEE ALSO

NAME

cpd - copy directory

SYNOPSIS

cpd [**o**=own] [**g**=grp] [**p**=mode] [**d**=[stime][,etime]] [**v**=on] [**u**=key] dir name1 [name2] ...

DESCRIPTION

Cpd will copy files, name1, name2, etc. to the target directory dir. The files, name1, name2, etc., are originating files and may be either a directory, special, or plain file. A nondirectory originating file is copied to dir. If an originating file is a directory, the contents of the directory and all subdirectories are copied to dir. Cpd will create subdirectories within dir as is required.

The keyword arguments available are:

o=own - all copied nondirectory files are given owner, own, which may be a numerical id or a login id.

g=grp - all copied nondirectory files are given group, grp, which may be a numerical id or a login id.

p=mode - all copied nondirectory files are given mode, mode, which is an octal used in a chmod system call.

d=stime,etime - copy only those nondirectory files whose modification dates fall between the specified start time, stime, and ending time, etime. Both stime and etime are of the format, mmddhhmmyy (month-day-hour-minute-year). Note that no seconds field is specified. Cpd assumes 00 for the seconds field. Either or both times may be omitted. If the starting time is omitted, then day 0 is assumed. The starting time is omitted as follows:

d=etime

If the ending time is omitted, than doomsday is assumed. If both are omitted, all files are processed. Directories are created regardless of their modification date. Silly dates give unpredictable results.

v=on - print out each file as it is copied. Without this option, only directory names are printed.

u=key - key may be either **orig** or **targ**. **u**=orig will unlink the original file after copying to the target file. Any key value starting with 'o' is acceptable for this option. **u**=targ will unlink the target file before copying to it. Any key value starting with 't' is acceptable for this option. Special files are always unlinked before copying them to their target. Normally no unlinking is performed. The **u**= option may be

specified twice, once as u=orig and a second time as u=targ. Directories are unaffected by this flag.

In absence of overriding keyword arguments, the mode, owner, and group of a copied file are preserved. The owner, mode, and group of a copied directory is not determined by the keyword arguments. Instead, a directory which did not exist within dir is created and the mode, owner, and group are preserved from the originating directory. A directory which already exists within dir is not disturbed with regard to the owner, mode, and group.

Cpd ignores interrupts while copying files and therfore several repetitive deletes may be required to kill it.

Cpd will attempt to create directories and special files as required although only the user with superuser capability will succeed.

Cpd does not know or care to know about links. If two files within an originating directory are linked, cpd will copy each one to a unique file. The link will be lost in the target directory.

Cpd will not allow a user to copy a file to itself.

FILES

/etc/passwd /etc/group

SEE ALSO

DIAGNOSTICS

"trouble doing mkdir <directory>" can be printed if the directory could not be made or if it could make it but the owner, group, or mode could not be set.

BUGS

Pathnames of greater than 120 characters cannot be correctly copied

NAME

cpp - copy pident

SYNOPSIS

cpp [**-d**] dir au-file ...

DESCRIPTION

The cpp command is used to copy each specified au-file (see au(V)) along with any files listed in the #PROGRAM UNITS or #DATA UNITS section of each au file. The files are copied to a directory whose path name is the concatenation of dir and the PR directory specified in the DOC: field of the au-file. If the **-d** flag is specified then the files are copied instead to the directory dir.

FILES

SEE ALSO

au(5L)

DIAGNOSTICS

BUGS

NAME

cx - Make listing & cross ref of C program.

USAGE

cx [-mrhcsuix] file file ....

DESCRIPTION

Produces printed listing of one or more files. Each page is headed with the current date & time, the file name, the file date and page number. Each print line is prefixed with the line number and nesting level. Lines over 72 characters long are folded at the last blank, tab, comma, semicolon, closing bracket, brace or paren. Lines not containing any of the seven fold characters are folded after character 72. The resulting second line is prefixed with the leading blanks and tabs of the original line. If the second line exceeds 80 chars LPR will still truncate. A header line is inserted at each function definition. The cross reference list contains an entry for each global name, function call, function definition and local name. Each entry, in the cross reference list, includes the line number of each reference. Lines begining with two semicolons will cause a new page. Options:

-m Prints a start and a completion message, and causes cx to run nice, immune to hangups and interrupts.

-r Will suppress the line no. and nesting level on ROFF command lines i.e. lines begining with dot, and will begin a new page with (.;).

-h Hold (suppress) the cross reference list.

-hh Hold (suppress) all printed output. Intended to be used with the **x** option.

-c High light multiple line comments, with a series of **********.

-cc High light all comments.

-s Single cross ref list for a set of files, line numbers will be consecutive thru the set of files and local names are suppressed.

-u Prints built-in keyword list.

-uu Prints entire built-in table.

-i Ignore page slew on ;;.

-x Produce a function list on the file flist.

-xx Pass the file flist to callref for formatting then remove

flist.

SEE ALSO
callref(1L)

FILES
/tmp/CXnnnnn - Temp sort file
./flist - Function list

NAME

diskmap - generates maps of RP06 disk drives as defined in /dev

SYNOPSIS

diskmap [**-i**]

DESCRIPTION

Diskmap is a shell command file that executes an ls -l /dev or an ls -li /dev command and passes the output of that ls command to a C program called **diskmapper** for sorting and output. A file called **hpmap.c** defines the relationship between entries in the **/dev** directory and the physical areas on the RP06 disk drives. When **diskmapper.c** is compiled, an up-to-date copy of **hpmap.c** must reside in the same directory with **diskmapper.c**.

A new output page is started for each RP06 drive. The output includes the cylinder numbers and the number of blocks associated with each file system. If the **-i** option is included, the inode numbers will appear at the beginning of the output lines.

FILES

/usr/bin/diskmapper

BUGS

If **diskmap** is invoked with the **-i** option, more than 80 columns are required for the output and an appropriate output device should be used.

NAME

dsktime - measure CPU time and disk accesses of a command

SYNOPSIS

dsktime command

DESCRIPTION

dsktime works like **time** (I) except that the number of disk accesses is also printed.

FILES

SEE ALSO

time (1)

DIAGNOSTICS

BUGS

See time (I);

NAME

dumposr - dump a specified out-of-service record from the trump data base

SYNOPSIS

dumposr ofcid tgn tnn

DESCRIPTION

Dumposr writes a specified out-of-service record from the trump data base to the standard output. In order to specify a record to be dumped three pieces of information must be included. They are ofcid, the office id (its number), tgn, the trunk group number, and tnn, the trunk network number (in octal). Each field of the record is dumped in the form

"fieldname = value".

Where the field is part of a substructure in the record the form is

structure (fieldname1 = value1, fieldname2 = value2, ..., fieldnamen = valuen).

All values written are decimal numbers except for the tnn and the os misc which are dumped in octal and the os diag which is a string. If there is a comment record then it will also be dumped as a string.

FILES

/dev/trumpdev

NAME

exref - list external references of a C source file

SYNOPSIS

exref [**-aisu**] [**-D**[name]] file

DESCRIPTION

For each routine in a C source file, exref lists all references to other routines or to global variables. For routines that reference no one, itself is given as the routine it references. Static functions are indicated by (**S**), and undefined functions by (**UF**). The type of global variable is indicated as

(**D**) Initialized data

(**d**) Static initialized data

(**C**) Common (if the variable was previously defined, this is a redeclaration of the same variable; otherwise, the variable is defined as bss (initialized to zero))

(**c**) Static common

(**U**) Undefined (either an array with no dimension or an extern)

(**E**) Extern that is not referenced (only listed if the u option is used)

For global variables that are not referenced, ??? is given as the calling routine.

Options are

-a Prefix each name with _

-i Include references in #include files - in this case the same information could also be obtained from nm(1)

-s Sort on second column instead of first

-u List unreferenced externs - useful with header files

-Dname Same as the **D** option for cc(1) to include conditionally compiled code

-D Produce separate listings for each possible combination of define names in #ifdef's, and warn about #if's (normally, #ifdef's are evaluated "as is")

The output of exref is compatible with the input required by tsort(1L), so that the calling structure of a file is obtained by

```
exref file | tsort -s | sort
```

FILES

/lib/cpp, /tmp/temp.<pid>.[01]

SEE ALSO

cc(1), nm(1), names(1L), tsort(1L)

BUGS

Not considering information in the #include files leads to some possible ambiguities: (1) in order to distinquish variables from names defined in includes, it is assumed that only such names and structure tag names are upper case, (2) references to routines that are only referenced indirectly and that are declared in an include will be missed, (3) macros defined in includes will be considered external routines.

NAME

fstat - print File STATus.

USAGE

fstat [**-a**] file file ....

DESCRIPTION

Prints file type, file name, date of last access and modify. Option **-a** will print all inode values.

SEE ALSO

fstat(1L), stat(2), fs(5)

AUTHOR

D. J. Jackowski

NAME

getcore-drop core of running process

SYNOPSIS

getcore pid outfile
getcore [-s] offset size outfile

DESCRIPTION

When called as **getcore offset size outfile** this program opens /dev/mem, seeks to offset number of cliques (64 bytes) and reads size number of cliques into its data space. Then it writes the data just read into outfile which it created. If the size is bigger than what fits into its data buffer, then it seats in a loop reading a buffer full at a time and writing a buffer full at a time. Since this process can not be as fast as reading a complete core at one buffer full an attempt is made to make the buffer as big as possible so the most cores can fit into a single buffer full. If the **-s** option is used the /dev/swap is opened instead, then a seek is made to offset block and size cliques are captured and written into outfile.

When called as **getcore pid outfile** the program will search the symbol table of /unix to find the address of the process table in the operating system. Then it will open /dev/mem and look in the process table of the running operating system for the occurrence of the given pid. If it finds it, it will use the size and offset found in the process table and will capture a core image of the process in memory or swapped according to the information in the process table that indicates where the process is.

BUGS

There is always a possibility although remote that a process moves from the time the information is obtained from the process table to the time that the core copy is made or also as the core copy is made the process can move around from under you. To minimize those possibilities this process is run at nice(-10) but it can still happen. Hence if your core makes absolutely no sense drop another one. If the second one doesn't make better sense I would suspect that you have a bug in your program which is causing the stack or something else to be overwritten. What I am saying is that the chances of it moving on you two times in a row are almost negligible. If you want to know if you got the core you wanted you can look at the end of the core (e.g. **tail -150c outfile ^bd**). This is the top of the user stack which contains the arguments to main unless your program has overwritten the stack. The first argv in main is by convention the name of the program.

FILES

/unix
/dev/mem
/dev/swap

NAME

getsrc - get latest source code

SYNOPSIS

getsrc [+v] <file_name> <pr_directory> <specific_make_name>

DESCRIPTION

This routine retrieves the latest (most up to date) source code for <file_name>. This is accomplished by using the <specific_make> argument to access a special data file containing control information. This special file contains the source pack and update pack names and the order in which they are to be searched.

EXAMPLE: getsrc +v gen_rng01.c pr-1p135 SC5I6

NOTE: The user must ensure that all valid update and source packs are mounted in order for this routine to run properly.

OPTION	EFFECT
+v	The user is informed if <file_name> exists on any other update pack. The output can be quite verbose.

Entering the command name "getsrc" alone, will result in an printout giving the command format and listing all acceptable make names.

FILES

/usr/bin/getsrc /usr/bin/sc11 /usr/bin/sc12

SEE ALSO

/usr/bin/putsrc

BUGS

NAME

intsort - sort routine for integer data files

SYNOPSIS

intsort [**-m**] [**-n**] [**-r**] [**-s**d] [+pos]... [**-o**name] [name]...

DESCRIPTION

Intsort sorts integer records in the named files and writes the result to the specified output file.

By default, the sort is in ascending numerical order over an entire record. If, however, the records are to be sorted only on certain selected integer elements in the record, then the sort keys are specified via the key specifications, **+pos**, as described below.

The valid arguments are:

-m Merge only, the input files are already sorted.

-n Sort on all integer elements except those specified via the **+pos** key specifications.

-r Sort records in descending numerical order rather than ascending numerical order.

-sd The record length size, in bytes, is set equal to the value of **"d"**, which should be a decimal number that is greater than zero and less than 513. The default record length is 100 bytes.

-o The next argument is the name of the output file. This file may have the same name as one of the input files, except under the merge flag, **-m.**

+pos Selected words or subwords of the integer data record may be used as sort keys. Pos has the form

<field> [,<left bit>,<right bit>]

where field is the integer element in the record that is to be used as the key. It may be equal to or greater than zero, but may not be equal to or greater than the maximum number of words in the record.

left bit is the left most bit (most significant), in the integer element, that is to be used as the key. If not specified, it is assumed to be 15. The permissible range of values for this item is 0 thru 15.

right bit is the right most bit, in the integer ele-

ment, that is to be used as the key. If not specified, it is assumed to be 0. The permissible range of values for this item is 0 thru 15; however, it must be less than or equal to the specified "left bit".

If more than one +pos specification is provided, successive specifications will be used only if preceding specifications compare equal. A maximum of ten key specifications may be provided.

FILES

/tmp/sort??<PID>

LIBRARY

DIAGNOSTICS

If this task is successful, a 0 is returned to the parent process. If the task is not successful, an appropriate error message is printed, via SCCERR, and one of the following error codes is returned to the parent process.

1 -- Can not create new temp file.
2 -- Can not create output file.
3 -- Can not open previously created file.
4 -- Improper argument for record size.
5 -- Not used.
6 -- Delete signal caught.
7 -- Invalid '-' option.
8 -- Invalid key specification.
9 -- Too many sort keys.
10 -- No valid input files.

BUGS

NAME

ipath

SYNOPSIS

ipath [**-i**][**-o**] inumber [filesystem]

DESCRIPTION

inumber: an inode number

filesystem: a special file on a device containing a file system (default is /dev/rootdev)

Searches all directories in the given file system for references to the given inumber and prints out the path name for these. If there are several links to the same inode, all will be printed out (e.g., for a directory, there will be itself, the . file and all .. files for directories in the directory).

-i ignore link count (keep looking in case the link count is wrong)

-o print only one pathname then stop

SEE ALSO

clri(1), fstat(1L), idump(1), ncheck(1)

NAME

lint - a C program verifier

SYNOPSIS

lint [**-abchnpuvx ll**] file ...

DESCRIPTION

Lint attempts to detect features of the C program files which are likely to be bugs, or non-portable, or wasteful. It also checks the type usage of the program more strictly than the compilers. Among the things which are currently found are unreachable statements, loops not entered at the top, automatic variables declared and not used, and logical expressions whose value is constant. Moreover, the usage of functions is checked to find functions which return values in some places and not in others, functions called with varying numbers of arguments, and functions whose values are not used.

By default, it is assumed that all the files are to be loaded together; they are checked for mutual compatibility. Function definitions for certain libraries are available to lint; these libraries are referred to by a conventional name, such as **-lm**, in the style of ld(1).

Any number of the options in the following list may be used. The **-D**, **-U**, and **-I** options of cc(1) are also recognized as separate arguments.

p Attempt to check portability to the IBM and GCOS dialects of C.

h Apply a number of heuristic tests to attempt to intuit bugs, improve style, and reduce waste.

b Report **break** statements that cannot be reached. (This is not the default because, unfortunately, most lex and many yacc outputs produce dozens of such comments.)

v Suppress complaints about unused arguments in functions.

x Report variables referred to by extern declarations, but never used.

a Report assignments of long values to int variables.

c Complain about casts which have questionable portability.

u Do not complain about functions and variables used and not defined, or defined and not used (this is suitable for running lint on a subset of files out of a larger program).

n Do not check compatibility against the standard library.

l1 Check compatibility against the local libraries /lib/lib1.a and /lib/lib3.a.

Exit(2) and other functions which do not return are not understood; this causes various lies.

Certain conventional comments in the C source will change the behavior of lint:

/*NOTREACHED*/
at appropriate points stops comments about unreachable code.

/*VARARGS n */
suppresses the usual checking for variable numbers of arguments in the following function declaration. The data types of the first n arguments are checked. A missing n is taken to be 0.

/*NOSTRICT*/
shuts off strict type checking in the next expression.

/*ARGSUSED*/
turns on the -v option for the next function.

/*LINTLIBRARY*/
at the beginning of a file shuts off complaints about unused functions in this file.

FILES

/usr/lib/lint[12]	programs
/usr/lib/llib-lc	declarations for standard functions
/usr/lib/llib-port	declarations for portable functions
/usr/lib/llib-l1	declarations for local functions in lib1 and lib3
/usr/tmp/lint.*	temporaries

SEE ALSO
cc(1).

NAME

lprstat - gets status of the line printer queues.

SYNOPSIS

lprstat [<keyword>[<data>]]

DESCRIPTION

Lprstat prints out the status of the line printer queues. The output from lprstat would be an ordered list of jobs to be printed by the line printer(s). The list contains the following information about each job. User id, jobnumber, date, time, size of job, rank, and the queue associated with that job.

Example:

USER	JOB	DATE	TIME	SIZE	RANK	QUEUE
TIM	2333	Sept10	10:05	4000	1	2

The following keywords are allowable:

usr - gets jobs of user specified by data argument, which is a user name. If data is not provided, then it defaults to the current user.

q - gets jobs on line printer queue specified by data argument, which is a queue name.

job - gets job specified by data argument, which is a jobnumber.

BUGS

The way lprstat list jobs does not mean they will always be printed in that order. The order in which jobs are printed is a function of the age and size of the jobs. Consequently, a new job may procede an older, but larger job.

NAME

ldedit -- user-friendly linedata editor

SYNOPSIS

ldedit

DESCRIPTION

Ldedit is an interactive program that allows the user to view and change the contents of the linedata file. It is intended to be much easier to use than ADB. Ldedit attempts to open the linedata file, and, if successful, reads and displays the first record. The user can then display any record by entering the MLN number of the desired record, the next sequential record by typing an 'n', or the previous record by typing a 'p'. If the currently displayed record is in some way incorrect, the user can enter the change mode by typing a 'c' in response to the prompt, and change some or all of the fields within the record, by moving the highlighted area to the desired field and entering a 'c'. If at any time you are unsure what commands to enter, help messages are always available by typing an 'h' or '?'. These give more complete details about the commands and capabilities of the program.

FILES

/gensrc1/src/pr-1p134/linedata05.h
/usr/include/sys/ioctl.h

DIAGNOSTICS

Any error messages should be self-explanatory. Any condition that causes a warning to be issued has not in anyway been corrected. If the file has no contents, the program will abort. The linedata file must be writable, even if no writing will be done (this generally means the user must be super-user).

BUGS

Line discipline types greater than 5 are considered to be unknown. Strings with embedded blanks will be displayed correctly, but will be truncated after the first blank when written to the disk.

NAME

mrt - message retrieval.

SYNOPIS

mrt -h

mrt [<ofc_type> [<ofc_gen>]]

DESCRIPTION

MRT retrieves messages from the message data base (MDB). The output will be a file of filtered messages from the MDB that satisfy the user's request. All messages retrieved are messages from the field.

The routine allows '?' and '^' for help and backup respectively for most of the prompts.

The '-h' option provides the user with a brief description of the routine before the routine starts.

MRT prompts for the office type (e.g. ess1, ess2b, ...) and the generic of that office type. The generic may be specified with "all" and mrt will then search all generics of that office type.

The messages are selected based on the pattern that is supplied to the routine. The pattern may be created during the routine, beforehand, or a predefine SCC pattern. The SCC patterns are limited to the ones currently available on the machine that mrt resides on. These can be listed by typing '?' to the SCC pattern prompt.

The "maximum number of messages" prompt provides the user the ability to specify the maximum number of a particular message. Note the MDB does not contain a vast number of any one message type, but it may have several of some sort codes.

The routine will provide a listing of which sort codes for a particular office and generic that currently reside in the MDB. To view, type 'y' to the prompt "Care to see which sort codes on the support list are in the MDB?". This listing does not provide all the sort codes that reside in the MDB, but rather shows which sort codes on the support list that reside in the MDB.

The routine prompts for where to place the message file after the search. This file will have the messages appended to it, if so desired.

"Incorrect generic type"
The user must choose one of the generics listed after this error message.

"Incorrect office type"
The user must choose one of the offices listed after this error message.

EXAMPLES

This is a sample run of mrt (the user's input is denoted by < >).

```
$ <mrt>
Enter office type: <ess1>
Type in one of the following
cc1e5
cc1e6
sp1e5
sp1e6
Enter generic: <cc1e5>
Care to see which sort code on the support list are in the
MDB? <n>
Will you be using an SCC predefine pattern ('s')
                     Your own predefine pattern('p')
               or need to create a pattern('c')
Type 's', 'p', or 'c' : <s>
Enter the SCC define pattern (e.i. tnØ8 for tnØ8.p): <ccis>
Enter the maximum number of this scrt code: <2Ø>
 IP

5 MESSAGES MATCHED OUT OF 4652

Enter the full path for a file for the message(s):
</usr/lts/wan/msgfile>
Do you want to search for other message types? ('y' or 'n'):
<n>
$
```

DIAGNOSTICS

"Error in number of arguments"
The command line had more than two arguments.

"Sorry no previous prompt available."
The user tried to backup past the first prompt.

"No report is currently available."
The report for the specified office and generic does not exist.

"Incorrect response"
The user typed an invalid response.

"File can not be moved, try again or hit the delete key."
The routine was unable to move your pattern to its location. Check the permissions on your file. Also the path may have been entered incorrectly.

"Error in call to rc:pat"
RC:PAT returned in error to MRT. The routine will try again.

NAME

page - page through a file

SYNOPSIS

page <filename>

DESCRIPTION

Page writes a file onto the standard output a screenful (approximately 20 lines) at a time based on input from the terminal. The command,

page <filename>

clears the screen, prints the present page number, the total number of pages and the first screenful of the file <filename>, and awaits input from the terminal. This input may be any one of the following forms:

<carriage return>
clears the screen and produces the next 20 lines of print. After the last page has been printed, another carrage return will terminate the program. A **delete** or **cntrl -d** will also terminate the program.

+**num**<carrage return>
where num is the number of pages to be skipped forward in the file.

+<carriage return>
performs the same function as +1<carriage return>.

-**num**<carriage return>
where **num** is the number of pages to be skipped backward in the file.

-<carriage return>
performs the same function as -1<carriage return>.

num<carriage return>
produces the page numbered **num** on the screen.

DIAGNOSTICS

Error messages will be printed if the index numbers for skipping pages are out of range, if the page numbers themselves are out of range, and if incorrect input is entered.

An error/usage message will be printed if the file to be paged does not exist.

In order for the page program to clear the screen on a CRT, the terminal type must be correctly set. If this has not been done (terminal type= "none") before page is called, page will prompt

the user for the correct terminal type before beginning to page through a file. If a question mark is typed after the user has been prompted for the terminal type, a listing of the available terminal types is given.

Escape sequences will be printed accurately , however, the length of the page may be sacrificed because the page program interprets the invisible escape sequences as if they were legitimate visible characters.

If the Tektronix 4014 (tex) graphics terminal is used, the user will be prompted as to whether he wants to see normal output of page (18 lines) or the extended output of page (58 lines) which the tex terminal can handle. Also, due to the operation of the tex terminal, page will sleep for two seconds after clearing the screen before the paged output appears.

If the user enters a question mark (?) when page prompts him for a terminal type, the currently available terminal types will be printed, along with a repeat type prompt.

NAME

plistp - print pident listing

SYNOPSIS

plistp [**-opq**] [**PRINT**] file ...

DESCRIPTION

The plistp command will print the files listed in the #PROGRAM UNITS section of any au files given as arguments (file). The command will also print the au file. In addition the file arguments may be pr-index files and if the PRINT argument is specified then the files listed in the #PRINT section of a prindex file are printed. This is useful for printing only changed pidents for a letter issue of a generic.

The flag, **-o**, will offset the output by one tab space.

The flag **-q** will cause a form-feed character (octal 14) to be printed at the end of each page. This should be used when printing listings for the IBM printer. Each page thus contains a page control character to allow realignment of the page in case it should become out of alignment, e.g., running out of paper or the system goes down.

The flag, **-p**, will cause the page size of every pident which is printed to also be printed. The output is to file descriptor 2 and the format of the output is as follows:

<pident> <pr document> <page size>

The format of the outputted files is acceptable as Bell System Standards PR listings. Output is to the standard output (file descriptor one). If the au file specified has in the #IDENTIFICATION section a field,

PLIST:OFF

then the pident and associated source files will not be printed. In fact, anything starting with a 'O' after the colon character will turn printing off. plistp will print an empty page (with footer and header only) if the number of pages of the pident is odd.

SEE ALSO

prindex file(5L), au file(5L)

NAME

plt - plot and provide a graphical dislay of data in a file

SYNOPSIS

plt [xlim] [ylim] [xlbl] [gttl] [mltg] [linf] [bgrf] [sldf] [idxf] [crtf] [lprf] <filename>

DESCRIPTION

Plt produces a graphical dislay of the data contained in the data file '**filename**'. By default , plt will normalize the graph. In other words, plt will make the largest and smallest x- and y-coordinates the upper and lower boundries for the graph. The ascii numbers in the data file may be separated by any combination of tabs, spaces, and/or newlines. Integers or decimals may be placed in this data file (i.e. **3, -3, 3.5, 3.05, .05, 3.0005,** etc.). However, only the first two digits to the right of the decimal point will be used, regardless of the number of digits to the right of the said point. A colon ':' is used to separate the points to be used for one plot from those to be used for another plot. An error message will be printed and the program will exit if the data contains any characters other than numbers.

OPTION

The following options may also be used:

xlim iXXhXX

allows the user to determine the minimum and maximum abscisa points. The low limit must be given before the high limit and there must be no spaces between the limits themselves. The numerical limits may be given in either integer or decimal form (i.e. **3, 03, 3.5, 3.05, 3.0005, .04,** etc.). If given in decimal form, any number of digits to the right of the decimal point may be entered, but only the first two will be used. The delimiters and the limits must be entered immediately after the keyword '**xlim**'. When the x- axis limits are given in this fashion, normalization of the points on the x-axis is not performed. Error messages will be printed and the program will terminate if the delimiters and limits are not entered in their proper sequence, if characters are interspersed in the limits, or if the low limit is greater than the high limit. Negative numbers may be given also.

ylim iXXhXX

allows the user to determine

the minimum and maximum ordinate points. The usage is the same as the '**xlim**' argument. The delimiters and limits must be entered immediately after the keyword '**ylim**'. When the y-axis limits are given in this fashion, normalization of the points on the y-axis is not performed.

xlbl HH HHHH HHH

allows the user to determine the label for the x-axis. All of

the characters within the double quotes which must follow the **'xlbl'** keyword will be printed as the label for the x-axis. An error message will be printed if the number of cahcharacters in the label are longer than the space allowed for it in the particular format (see the arguments **'crtf'** and **'lprf'** for format explanation), and the program will exit.

ylbl HH HHHH HHH
allows the user to determine the label for the y-axis. The usage for the y-axis label argument is the same as that for the x-axis label argument.

gttl HH HHHH HHH
allows the user to determine the title for the graph. The usage for the graph title argument is the same as that for the x-axis label argument.

mltg pXX
allows the user to have more than one set of points plotted on a graph, where **'XX'** is the number of graphs to be plotted. If the number of plots requested is more than the maximum number allowed in the program code will exit. If the number of plots requested is less than the number of sets of points represented by the data file, only the plots for which there is data are plotted (i.e. if the user requests three plots and data exists for only two plots, only two plots will appear on the graph). If **'XX'** is zero, no points will be plotted and the x-axis and y-axis numerical labels will all be **'0.00'**. However, if limits are given, the x- and y-axis labels will be correctly printed as normal but there will be no points on the graph. If this argument is not used, plt will default and plot only one set of points on a graph, irrespective of the number of sets of points represented by data in the data file.

linf
allows the user to create a line graph format. A vertical line will be drawn from the point to the x-axis using the same character as the point itself. This format cannot be used when multiple plots are being made.

bgrf
allows the user to create a bar graph format. The area vertically below and horizontally over to one space behind the next point will be filled in with the same character as the point itself. In other words, the bars will be separated by a column of blank space. In order for a bar graph to be made therefore, there must be at least one column of blank space between two points. If this margin does not exist, some of the points on the graph may not appear. This format cannot be used when multiple plots are being made.

sldf
allows the user to create a solid graph format. The area

vertically below and horizontally over to the next point to the right will be filled in with the same character as the point itself. This format cannot be used when multiple plots are being made.

idxf
produces a display specially formatted for office indices. The data is read from an octal file created by another program.

crtf
scales the graph down to a format that will fit on a crt screen. If neither this argument nor the '**lprf**' (see below) argument is used, plt defaults to a format which will fit on an 8 1/2 by 11 inch page.

lprf
scales the graph to a format that will fit on a 14 by 11 inch page. If neither this argument nor the '**crtf**' (see above) argument is used, plt defaults to a format which will fit on an 8 1/2 by 11 inch page.

DIAGNOSTICS
An error message will be printed if the data file does not exist.

BUGS
plt may not yield an accurate graph if the range between limits is less than six, or if the upper limit (on either of the axes) is zero.

NAME

ppcfpat - pattern package C formatter for standard format patterns.

SYNOPSIS

ppcfpat patfilename outfile structname

DESCRIPTION

Ppcfat creates a C formatted file, outfile, of the header, variable information, and pattern parts of a standard format pattern file.

The argument patfilename is the name of a standard format pattern file.

The argument outfile is the name of the file where the C formatted source code is to be written.

The argument structname is the name given to the structure PPCFPAT in the C source code.

FILES

/usr/include/ppsubs.h common pattern package header file

SEE ALSO

ppfopenpat(3L),ppatell(3L),ppatsiz(3L),fseek(3),fread(3S), ppmkpat(1L),pattern(5L)

DIAGNOSTICS

The diagnostics produced by ppcfpat are intended to be self explanatory. For more information, an external variable, pperrno, is set by the subroutines used on error and the return codes are defined in the common pattern package header file.

NAME

ppdpat - pattern package verifier diagnostic output utility

SYNOPSIS

ppdpat patfilename

DESCRIPTION

Ppdpat provides a basic output dump of the pattern file. The **patfilename** is the name of the pattern file (which must include the format dependant ending, for example "temp.p").

For the pattern file **/compat/sctab.p** the following is a sample output from **ppdpat:**

```
PATTERN FILE NAME:  /compat/sctab.p
VARIABLE INFORMATION SIZE:  0
PATTERN PART SIZE:  34
PATTERN SOURCE SIZE:  308
PATTERN HEADER (SHORT-SIZE, OCTAL-NUMBER):

      7      42       0       0       0       0       0       0

ADDRESS MNEMONIC                RAW_DEC_WORD      RAW_OCTAL_BYTE
    0: break    "\212"             22    138     26   0 212   0
    4: alt      20                 10     12     12   0  14   0
    8: tab      4                   2      4      2   0   4   0
   12: arbn     3                   6      3      6   0   3   0
   16: jmp      32                  8     12     10   0  14   0
   20: tab      1                   2      1      2   0   1   0
   24: string   "Z"                 4     90      4   0 132   0
   28: tab      1                   2      1      2   0   1   0
   32: patsucc                         60         |  74   0

PATTERN DEFINITION :

/*
 *        COMMON APPLICATION PATTERN
 *
 *  OFFICE TYPE:  ALL
 *
 *  FUNCTION:  This pattern successfully matches everything up to
 *       the message Sort Code found in SPCS output messages. This
 *       pattern will fail if no log header is found in the
 *       message.
 */

break("\212") (
       tab(4) arb(3)
     + tab "Z" tab
     )
```

SEE ALSO

ppvpat(1L), pattern(5L)

NAME

ppmap - pattern package text formatter

SYNOPSIS

ppmap <patfilename> <filename>

DESCRIPTION

Ppmap outputs the text area of a reversed compiled pattern in a formatted form.

The argument <patfilename> is the name of a pattern file to be used by the pattern matcher, **ppmatch**(), to match on.

The argument <filename> is the name of a file containing the text to be formatted.

FILES

/usr/include/ppsubs.h	pattern package header file
/usr/include/sys/types.h	types header file
/usr/include/sys/stat.h	stat structure header file
/usr/include/errfct.h	error function header file

SEE ALSO

ppmkpat(1L), ppvpat(1L), ppmatch(3L), pattern(5L)

DIAGNOSTICS

The diagnostics produced are intended to be self-explanatory.