status is that of the last command executed (an end-of-file will also cause the shell to exit.)

export [*name* ...]
: The given *names* are marked for automatic export to the *environment* of subsequently-executed commands. If no arguments are given, then a list of all names that are exported in this shell is printed.

newgrp [*arg* ...]
: Equivalent to **exec newgrp** *arg* ....

read [*name* ...]
: One line is read from the standard input and the first word is assigned to the first *name*, the second word to the second *name*, etc., with leftover words assigned to the last *name*. The return code is 0 unless an end-of-file is encountered.

readonly [*name* ...]
: The given *names* are marked *readonly* and the values of the these *names* may not be changed by subsequent assignment. If no arguments are given, then a list of all *readonly* names is printed.

set [**−−ekntuvx** [*arg* ...]]
: **−e** Exit immediately if a command exits with a non-zero exit status.
: **−k** All keyword arguments are placed in the environment for a command, not just those that precede the command name.
: **−n** Read commands but do not execute them.
: **−t** Exit after reading and executing one command.
: **−u** Treat unset variables as an error when substituting.
: **−v** Print shell input lines as they are read.
: **−x** Print commands and their arguments as they are executed.
: **−−** Do not change any of the flags; useful in setting $1 to -.

Using + rather than − causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in **$−**. The remaining arguments are positional parameters and are assigned, in order, to **$1**, **$2**, ... If no arguments are given then the values of all names are printed.

shift [*n*]
: The positional parameters from **$n+1** ... are renamed **$1** .... IF *n* is not given, it is assumed to be 1.

test
: Evaluate conditional expressions. See *test*(1) for usage and description.

times
: Print the accumulated user and system times for processes run from the shell.

trap [*arg*] [*n*] ...
: *arg* is a command to be read and executed when the shell receives signal(s) *n*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If *arg* is absent then all trap(s) *n* are reset to their original values. If *arg* is the null string then this signal is ignored by the shell and by the commands it invokes. If *n* is 0 then the command *arg* is executed on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

umask [*nnn*]
: The user file-creation mask is set to *nnn* (see *umask*(2)). If *nnn* is omitted, the current value of the mask is printed.

wait [*n*]
: Wait fo the specified process and report its termination status. If *n* is not given then all

currently active child processes are waited for and the return code is zero.

Invocation.
If the shell is invoked through *exec*(2) and the first character of argument zero is –, commands are initially read from **/etc/profile** and then from **$HOME/.profile**, if such files exist. Thereafter, commands are read as described below, which is also the case when the shell is invoked as **/bin/sh**. The flags below are interpreted by the shell on invocation only; Note that unless the **–c** or **–s** flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file:

–c *string*	If the **–c** flag is present then commands are read from *string*.
–s	If the **–s** flag is present or if no arguments remain then commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output is written to file descriptor 2.
–i	If the **–i** flag is present or if the shell input and output are attached to a terminal, then this shell is *interactive*. In this case TERMINATE is ignored (so that **kill 0** does not kill an interactive shell) and INTERRUPT is caught and ignored (so that **wait** is interruptible). In all cases, QUIT is ignored by the shell.
–r	If the **–r** flag is present the shell is a restricted shell (see *rsh*(1)).

The remaining flags and arguments are described under the **set** command above.

EXIT STATUS

Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively then execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see also the **exit** command above).

rsh ONLY

rsh is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of *rsh* are identical to those of *sh*, except that the following are disallowed:

cd
setting the value of **$PATH**
command names containing /
> and >>

The restrictions above are enforced after **.profile** is interpreted.

When a command to be executed is found to be a shell procedure, *rsh* invokes *sh* to execute it. Thus, it is possible to provide to the end user shell procedures that have access to the full power of the standard shell, while restricting him to a limited menu of commands; this scheme assumes that the end user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the **.profile** has complete control over user actions, by performing guaranteed setup actions, then leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands that can be safely invoked by *rsh*. Some systems also provide a restricted editor *red*.

FILES

/etc/profile
$HOME/.profile
/tmp/sh*
/dev/null

SEE ALSO

cd(1), env(1), login(1), newgrp(1), rsh(1), test(1), umask(1), dup(2), exec(2), fork(2), pipe(2), signal(2), umask(2), wait(2), a.out(5), profile(5), environ(7).

BUGS

The command **readonly** (without arguments) produces the same output as the command **export**.

If << is used to provide standard input to an asynchronous process invoked by &, the shell gets mixed up about naming the input document; a garbage file **/tmp/sh*** is created and the shell complains about not being able to find that file by another name.

NAME

size − size of an object file

SYNOPSIS

size [object ...]

DESCRIPTION

Size prints the (decimal) number of bytes required by the text, data, and bss portions, and their sum in octal and decimal, of each object-file argument. If no file is specified, **a.out** is used.

SEE ALSO

a.out(5)

NAME

sleep — suspend execution for an interval

SYNOPSIS

sleep time

DESCRIPTION

Sleep suspends execution for *time* seconds. It is used to execute a command after a certain amount of time as in:

```
(sleep 105; command)&
```

or to execute a command every so often, as in:

```
while true
do
        command
        sleep 37
done
```

SEE ALSO

alarm(2), sleep(3C)

BUGS

Time must be less than 65536 seconds.

NAME

sno − SNOBOL interpreter

SYNOPSIS

sno [file] ...

DESCRIPTION

Sno is a SNOBOL3 (with slight differences) compiler and interpreter. *Sno* obtains input from the concatenation of the named *files* and the standard input. All input through a statement containing the label **end** is considered program and is compiled. The rest is available to **syspit**.

Sno differs from SNOBOL3 in the following ways:

There are no unanchored searches. To get the same effect:

a ** b	unanchored search for *b*.
a *x* b = x c	unanchored assignment

There is no back referencing.

```
x = "abc"
a *x* x           is an unanchored search for abc.
```

Function declaration is done at compile time by the use of the (non-unique) label **define**. Execution of a function call begins at the statement following the **define**. Functions cannot be defined at run time, and the use of the name **define** is preempted. There is no provision for automatic variables other than parameters. Examples:

```
define f( )
define f(a, b, c)
```

All labels except **define** (even **end**) must have a non-empty statement.

Labels, functions and variables must all have distinct names. In particular, the non-empty statement on **end** cannot merely name a label.

If **start** is a label in the program, program execution will start there. If not, execution begins with the first executable statement; **define** is not an executable statement.

There are no builtin functions.

Parentheses for arithmetic are not needed. Normal precedence applies. Because of this, the arithmetic operators / and * must be set off by spaces.

The right side of assignments must be non-empty.

Either ´ or " may be used for literal quotes.

The pseudo-variable **sysppt** is not available.

SEE ALSO

awk(1)
"SNOBOL, a String Manipulation Language," by D. J. Farber, R. E. Griswold, and I. P. Polonsky, *JACM* **11** (1964), pp. 21-30.

NAME

sort − sort or merge files

SYNOPSIS

sort [**−mubdfinrt***x*] [**+***pos1* [**−***pos2*]] ... [**−o** name] [name] ...

DESCRIPTION

Sort sorts lines of all the named files together and writes the result on the standard output. The name − means the standard input. If no input files are named, the standard input is sorted.

The default sort key is an entire line. Default ordering is lexicographic by bytes in machine collating sequence. The ordering is affected globally by the following options, one or more of which may appear.

b	Ignore leading blanks (spaces and tabs) in field comparisons.
d	'Dictionary' order: only letters, digits and blanks are significant in comparisons.
f	Fold upper case letters onto lower case.
i	Ignore characters outside the ASCII range 040−0176 in nonnumeric comparisons.
n	An initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, is sorted by arithmetic value. Option **n** implies option **b.**
r	Reverse the sense of comparisons.
t*x*	'Tab character' separating fields is *x*.

The notation restricts a sort key to a field beginning at *pos1* and ending just before *pos2*. *Pos1* and *pos2* each have the form *m.n*, optionally followed by one or more of the flags **bdfinr**, where *m* tells a number of fields to skip from the beginning of the line and *n* tells a number of characters to skip further. If any flags are present they override all the global ordering options for this key. If the **b** option is in effect *n* is counted from the first nonblank in the field; **b** is attached independently to *pos2*. A missing .*n* means .0; a missing −*pos2* means the end of the line. Under the −**t***x* option, fields are strings separated by *x*; otherwise fields are nonempty nonblank strings separated by blanks.

When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant.

These option arguments are also understood:

c	Check that the input file is sorted according to the ordering rules; give no output unless the file is out of sort.
m	Merge only, the input files are already sorted.
o	The next argument is the name of an output file to use instead of the standard output. This file may be the same as one of the inputs.
u	Suppress all but one in each set of equal lines. Ignored bytes and bytes outside keys do not participate in this comparison.

EXAMPLES

Print in alphabetical order all the unique spellings in a list of words. Capitalized words differ from uncapitalized.

```
sort −u +0f +0 list
```

Print the password file (*passwd*(5)) sorted by user id number (the 3rd colon-separated field).

```
sort −t: +2n /etc/passwd
```

Print the first instance of each month in an already sorted file of (month day) entries. The options **−um** with just one input file make the choice of a unique representative from a set of equal lines predictable.

```
sort −um +0 −1 dates
```

FILES
/usr/tmp/stm???

SEE ALSO
uniq(1), comm(1), join(1)

DIAGNOSTICS
Comments and exits with nonzero status for various trouble conditions and for disorder discovered under option **−c**.

BUGS
Very long lines are silently truncated.

NAME

spell, spellin, spellout − find spelling errors

SYNOPSIS

spell [option] ... [file] ...

/usr/src/cmd/spell/spellin [list]

/usr/src/cmd/spell/spellout [**−d**] list

DESCRIPTION

Spell collects words from the named documents, and looks them up in a spelling list. Words that neither occur among nor are derivable (by applying certain inflections, prefixes, or suffixes) from words in the spelling list are printed on the standard output. If no files are named, words are collected from the standard input.

Spell ignores most *troff*(1), *tbl*(1), and *eqn*(1) constructions.

Under the **−v** option, all words not literally in the spelling list are printed, and plausible derivations from the words in the spelling list are indicated.

Under the **−b** option, British spelling is checked. Besides preferring *centre, colour, speciality, travelled,* etc., this option insists upon *-ise* in words like *standardise,* Fowler and the OED to the contrary notwithstanding.

Under the **−x** option, every plausible stem is printed with = for each word.

The spelling list is based on many sources, and while more haphazard than an ordinary dictionary, is also more effective in respect to proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine, and chemistry is light.

Pertinent auxiliary files may be specified by name arguments, indicated below with their default settings. Copies of all output are accumulated in the history file. The stop list filters out misspellings (e.g., thier=thy−y+ier) that would otherwise pass.

Two routines help maintain the hash lists used by *spell* (both expect a list of words, one per line, from the standard input): *spellin* adds the words on the standard input to the preexisting *list* and places a new list on the standard output. If no *list* is specified, the new list is created from scratch. *Spellout* looks up each word read from the standard input, and prints on the standard output those that are missing from (or, with the **−d** option, present in) the hash list.

FILES

D=/usr/dict/hlist[ab]	hashed spelling lists, American & British
S=/usr/dict/hstop	hashed stop list
H=/usr/dict/spellhist	history file
/tmp/spell.$$	temporary
/usr/lib/spellprog	program

SEE ALSO

deroff(1), eqn(1), sed(1), sort(1), tbl(1), tee(1), troff(1), typo(1).

BUGS

The spelling list's coverage is uneven; new installations will probably wish to monitor the output for several months to gather local additions.
British spelling was done by an American.

NAME

spline − interpolate smooth curve

SYNOPSIS

spline [option] ...

DESCRIPTION

Spline takes pairs of numbers from the standard input as abscissas and ordinates of a function. It produces a similar set, which is approximately equally spaced and includes the input set, on the standard output. The cubic spline output (R. W. Hamming, *Numerical Methods for Scientists and Engineers,* 2nd ed., 349ff) has two continuous derivatives, and sufficiently many points to look smooth when plotted, for example by *graph*(1G).

The following options are recognized, each as a separate argument.

−a Supply abscissas automatically (they are missing from the input); spacing is given by the next argument, or is assumed to be 1 if next argument is not a number.

−k The constant k used in the boundary value computation

$$y_0'' = ky_1'', \quad y_n'' = ky_{n-1}''$$

is set by the next argument. By default $k = 0$.

−n Space output points so that approximately n intervals occur between the lower and upper x limits. (Default $n = 100$.)

−p Make output periodic, i.e. match derivatives at ends. First and last input values should normally agree.

−x Next 1 (or 2) arguments are lower (and upper) x limits. Normally these limits are calculated from the data. Automatic abcissas start at lower limit (default 0).

SEE ALSO

graph(1G)

DIAGNOSTICS

When data is not strictly monotone in x, *spline* reproduces the input without interpolating extra points.

BUGS

A limit of 1000 input points is enforced silently.

NAME

split − split a file into pieces

SYNOPSIS

split [−*n*] [file [name]]

DESCRIPTION

Split reads *file* and writes it in *n*-line pieces (default 1000), as many as necessary, onto a set of output files. The name of the first output file is *name* with **aa** appended, and so on lexicographically. If no output name is given, **x** is default.

If no input file is given, or if − is given instead, then the standard input file is used.

NAME

spr − special print command

SYNOPSIS

spr [**−f** file] [**−u** dir] [**−h** header] [**−wn**] [**−ln**] [**−in**] [**−cn**] [**−d**] [**−t**] [**−b**] [**−v**] [**−s**] [+n] name ...

DESCRIPTION

Spr produces a listing of one or more files. Its special features include:

− incremental printing, i.e. printing only files which have changed since the last print.
− table of contents.
− control of heading and sub-heading.
− embedded control of listing format.

Normally, *spr* prints its *name* arguments or the files listed in the *file* argument. If neither of these arguments is present, *spr* prints its standard input.

The options have the following meanings:

−f *file* — *File* must be a list of of files to be printed. Internally, *file* should contain a list of files, one per line. Since *spr* looks up file names relative to the directory from which it is run; the file names in *file* must be correct relative to the current working directory (see *pwd*(1)). Complete path names are safest.

−u *dir* — *Dir* is the name of a file which will be used to record file status information. If you wish to do "incremental" listings, you must provide a *dir* argument. Assuming that a *dir* argument is given to *spr* and that file status information exists in *dir*, then only files which have been modified; or whose status has not been recorded in *dir* by a previous *spr* will be printed.

−h *header* — Use *header* instead of the name of the file being printed as the default main heading.

−w*n* — The output page width is taken to be *n* columns instead of the default (80).

−l*n* — The output page length is considered to be *n* lines instead of the default (66).

−i*n* — First line to be used for printing is line *n*. Default is line 0.

−c*n* — First column to be used is column *n*. Default is column 0.

−d — Causes table of contents to be printed.

−t — Turns off top headings. Default is on.

−b — Turns on bottom headings. Default is off.

−v — Turns on both top and bottom headings. Overrides both **−t** and **−b** options.

−s — Turns on subheadings. Default is off.

+*n* — First file to print is *n*; file status information is gathered on all files but nothing is printed until the *n*th file is reached. The page number of the table of contents is 999, thus +999 will cause only the table of contents to be printed. A word about page numbers; *spr* prints page numbers in the form:

nnn−mmm

where *nnn* is the "page" or "file" number and *mmm* is the page within the "file".

name — *Name*s of files to be printed. These files are printed in addition to (and before) the files printed by the **−f** option.

NOTE: The **−d**, **−t**, **−b**, **−v**, and **−s** options are merely on/off flags in *spr*. That means that successive occurences of the option will turn the flag to its opposite state. For instance:

```
spr −d −d −d
```

will cause the state of the table of contents flag to change from off to on to off to on.

EXAMPLE

```
spr −f fillst −u dir −s −d extra −s extra2
```

Causes *extra* to be printed with subheadings, *extra2* to be printed without subheadings and then prints all files listed in *fillst* without subheadings. After all files have been printed, a table of contents will be printed which includes *extra*, *extra2* and all files in *fillst*. File status information will be maintained on *dir* for all files printed.

EMBEDDED CONTROL SEQUENCES

Lines of the form:

```
/*.x'par1'par2'*/
```

are commands to *spr*. The character, ('), represents a delimiter which may be any character except **<nl>** or blank. These lines are not listed in prints produced by *spr*; if it is desired to see these lines, use *pr*(1) or *cat*(1). Note that the format of these lines is such that they will be ignored by the C compiler. The following are the *spr* commands (i.e. values which *x* may assume):

s Create a subheading of the form:

par1: *par2*

The **s** command also causes a page eject. *Par1* is a character string, ≤ 10 characters (usually the name of a subroutine) and *par2* is a comment whose length is ≤ 40 characters.

t Same as **s** except that the subheading is not changed and no page eject is done. The purpose of **t** is to make entries for the table of contents.

n Same effect as **s** except that no page eject is done. That is, the subheading on the page that the **n** command occurs will change, but it will not cause a page eject. The most common use of the **n** command was immediately following the **#** line in C files; however, with current versions of the C compiler that kluge is no longer necessary and so the value of the **n** command has diminished.

i The **i** command is intended to be used to make subentries in the table of contents under subroutine entries. *Par1* is a ≤ 10 character string which will appear in the table of contents indented 3 spaces after the **s**, **t**, or **n** entry which it follows. The first letter of *par2* appears in the description column of the table of contents. For instance:

```
/*.i'aardvark'G'*/
```

Might be used to document the appearance of the Global variable *aardvark*.

m The **m** command overrides the current main heading − which is normally taken to be the name of the file being printed (see **−h** option to *spr*). In this case, *par1* is a ≤ 40 character string.

e Causes an immediate page eject.

Except for the **m** and **e** commands, all commands cause entries to be made in the table of contents (assuming one is printed).

TABLE OF CONTENTS

The table of contents consists of 2 parts. The first part contains 1 line per file printed giving the file number (see part 2 description below), name of the file and the date of the last modification of the file. Part 2 contains a listing of all **s**, **n** and **t** entries together with their subtending **i** entries sorted into alphabetic order. For each file printed, a default **s** entry is made where *par1* is the name of the file and *par2* is the full path name of the file (as given in the **−f** file or *name*). Each **s** entry line (or **n** or **t**) contains *par1*, the file number (see description of part 1), the page number on which the *spr* command occured and *par2*. Each **i** entry gives *par1* indented 3 spaces, the page number and the first character of *par2*.

DIAGNOSTICS

Presumably, all diagnostics are self−explanatory. However, the diagnostics:

par: **param error**

and

come again ?

are likely to be confusing. Both of these messages are an attempt to say that something was wrong with the parameters given when *spr* was invoked.

BUGS

The file names are limited to 40 characters.

Also, *spr* was originally written for use on output devices which have 80 columns per line and 66 lines per page. Some of its computations on the legality of the input values given to it assume that those limits are "absolute". For instance, if you change the first line to 5, you should change the page length to 61; otherwise, *spr* will add 5 to 66 and get 71 which is longer than it believes a page has any right to be. *Spr* should be fixed so that it does not assume anything about the output medium.

NAME

sps − detail process status

SYNOPSIS

sps [+pid ...] [=rep] [−opts] [width] [namelist]
or
sps ?

DESCRIPTION

Sps prints status information about one or more UNIX processes currently in existence.

The ? form causes a list of the options to be printed. (Note that the ? must be escaped from the shell.)

If one or more +*pid* options are given, only the processes with the given *pid*s are examined, overriding the *a* and *x* options discussed below. If the =*rep* option is used, the report will be repeated every *rep* seconds; otherwise, the report is given only once. If a *namelist* file is given, it will be used as the operating system namelist; otherwise **/unix** will be used.

The *opts* argument may be a list of characters from the following:

t	print the line number of the process
f	print the flag and status flags
i	print the priority (as opposed to the niceness)
n	print the niceness of the process
m	print the time in the current swapped-in/swapped-out state
p	print the process id
r	print the parent process id
g	print the process group
w	print the "wchan" (i.e. the reason for sleeping)
W	print the effective user id of the process
G	print the effective group id
R	print the real user id
Q	print the real group id
d	print the i-number of the current directory
T	print the text size in 64-byte segments
D	print the data size
S	print the stack size
E	print the separated I/D space flag
U	print the user CPU time in 1/60 sec.
K	print the system (kernel) time in 1/60 sec.
V	print the user time of the process's terminated children
L	print the system time of the children
B	print the number of block buffer requests
I	print the number of actual disk reads
O	print the number of actual disk writes
J	print the number of disk reads by children

Y	print the number of disk writes by children
Z	print the total number of disk accesses by process
F	print the "CPU factor" used for scheduling
z	print the swapable size of the process
X	print the octal value of the text address
A	print the process core or disk address
P	print the octal value of the process table entry
s	print the long signal word from the process table
c	print an approximation of the command line
-	print a summary of the count-like items for all processes
l	print the fields produced by the **−l** option in *ps*(1)
a	consider all processes with process groups
x	consider even processes without process groups
k	use the file **unixcore** instead of **/dev/mem**
!	print all available data about the process;
q	print the date and time after each report
C	use *width* for the number of columns on the output medium (79 default).

The default options are the same as *ps*(1). For the options *U*, *V*, *K*, *L*, *I*, *O*, *J*, *Y*, *Z*, and *B*, the difference between successive samples is reported on the second and subsequent iterations. Unlike *ps*(1), the line number column is printed for all processes; processes on the current line are marked with an *.

Sps is optimized to run much faster if the line number is not needed (note that the default includes the line number). It also runs somewhat faster if no data from the user block data maintained by the system is required; that is, if none of the options *t*, *W*, *G*, *R*, *Q*, *d*, *T*, *D*, *S*, *E*, *U*, *K*, *V*, *L*, *B*, *I*, *O*, *J*, *Y*, *Z*, *c*, or **l** are used (note that the default includes *t* and *c*.

FILES

/unix	for the system namelist
/dev/swapdev	for the swapped out user block
/dev/mem	for the process table and swapped in user block.
or **unixcore**	if the *k* option is used.

BUGS

When any of the options requiring the user table are used, an unavoidable race condition can occur which may result in occasional garbled output.

The number of options is cumbersome; it is recommended that users with commonly used sets of options create shell files to invoke *sps*.

NAME

Sprof − system profile

SYNOPSIS

sprof [**−rianpe**] [**−l** *lowpc* **−h** *highpc* [**−s** *intsize*]]
[**−o** *cutoff*] [**−t** *interval*] [**−c** *count*]

DESCRIPTION

Sprof initiates profiling of UNIX system activities. A summary of the results is given automatically. If the **−r** option is used (the default), then a counter is reserved for every system routine. Under this option, the system is profiled for *interval* seconds *count* times, and whenever the system clock routine interrupted a system routine, that system routine's counter is incremented by one. If the **−i** option is used, the system is profiled for *interval* seconds *count* times, with a separate counter for each *intsize* (default two) interval of bytes between byte addresses *lowpc* and *highpc*. *Lowpc*, *highpc*, and *intsize* should be even numbers, and may be entered either in decimal or octal (lead 0 implies octal).

After *interval* seconds (default 3600, or one hour), a report is printed containing the number of times the system clock routine found that it interrupted a system routine or interval of bytes. The report shows the number of hits for user, system, and idle time, and the percentage of the total for each. The routines' or intervals' percentages are percentages of system hits only, with the system idle routine excluded.

If the **−a** option is selected with **−r**, the system routines are sorted by start address before printing. If the **−n** option is used, the routines are sorted in alphabetical routine name order. If the **−p** option is used (the default), the routines are listed in order by decreasing percentage of number of hits.

If the *cutoff* specified with the **−o** option is greater than zero (the default), then no routines or intervals are reported that have a percentage of hits less than the *cutoff*. They are all lumped together at the end of the report under "Other". Under the **r** option (only), no routine is listed that has been hit zero times. To include those in the list, the **−e** (or *everything*) option must be specified. The **−e** option is the default for any **i** report.

Cautions:

> If *sprof* uses the system clock (1/60th second granularity), activities in sync with the clock will be missed. In particular, *sprof* would show zero hits for the system clock routines, including *sincupc*, which does the system profile counting. System profiling may be done using an independent clock (a DEC KW11-K or a Digital Pathways TCU100). The user should find out what profiling clock is being used on his machine.
>
> If the super-user initiates system profiling, the system will lock *sprof* in low main memory, which may cause shuffling of other proceses. If anyone else initiates profiling, the process will *not* be moved to low memory before being locked, and overall system performance may be degraded.
>
> If **/unix** does not have the namelist of the currently running operating system, the **−r** option may print garbage (but should not crash).

FILES

/unix, for system routine names and starting addresses

BUGS

Results can vary somewhat from run to run, even under similar system loads, due to variations in the clock interrupt times, correlated with system activities. This effect is greatly decreased if one of the independent profiling clocks is being used.

Sprof may not be run concurrently with other *sprof* processes.

Cutoff must be an integer.

NAME

sps − detail process status

SYNOPSIS

sps [+pid ...] [=rep] [−opts] [width] [namelist]
or
sps ?

DESCRIPTION

Sps prints status information about one or more UNIX processes currently in existence.

The ? form causes a list of the options to be printed. (Note that the ? must be escaped from the shell.)

If one or more +*pid* options are given, only the processes with the given *pid*s are examined, overriding the *a* and *x* options discussed below. If the =*rep* option is used, the report will be repeated every *rep* seconds; otherwise, the report is given only once. If a *namelist* file is given, it will be used as the operating system namelist; otherwise **/unix** will be used.

The *opts* argument may be a list of characters from the following:

t	print the line number of the process
f	print the flag and status flags
i	print the priority (as opposed to the niceness)
n	print the niceness of the process
m	print the time in the current swapped-in/swapped-out state
p	print the process id
r	print the parent process id
g	print the process group
w	print the "wchan" (i.e. the reason for sleeping)
W	print the effective user id of the process
G	print the effective group id
R	print the real user id
Q	print the real group id
d	print the i-number of the current directory
T	print the text size in 64-byte segments
D	print the data size
S	print the stack size
E	print the separated I/D space flag
U	print the user CPU time in 1/60 sec.
K	print the system (kernel) time in 1/60 sec.
V	print the user time of the process's terminated children
L	print the system time of the children
B	print the number of block buffer requests
I	print the number of actual disk reads
O	print the number of actual disk writes
J	print the number of disk reads by children

Y	print the number of disk writes by children
Z	print the total number of disk accesses by process
F	print the "CPU factor" used for scheduling
Z	print the swapable size of the process
X	print the octal value of the text address
A	print the process core or disk address
P	print the octal value of the process table entry
s	print the long signal word from the process table
c	print an approximation of the command line
~	print a summary of the count-like items for all processes
l	print the fields produced by the **−l** option in *ps*(1)
a	consider all processes with process groups
x	consider even processes without process groups
k	use the file **unixcore** instead of **/dev/mem**
!	print all available data about the process;
q	print the date and time after each report
C	use *width* for the number of columns on the output medium (79 default).

The default options are the same as *ps*(1). For the options *U*, *V*, *K*, *L*, *I*, *O*, *J*, *Y*, *Z*, and *B*, the difference between successive samples is reported on the second and subsequent iterations. Unlike *ps*(1), the line number column is printed for all processes; processes on the current line are marked with an *.

Sps is optimized to run much faster if the line number is not needed (note that the default includes the line number). It also runs somewhat faster if no data from the user block data maintained by the system is required; that is, if none of the options *t*, *W*, *G*, *R*, *Q*, *d*, *T*, *D*, *S*, *E*, *U*, *K*, *V*, *L*, *B*, *I*, *O*, *J*, *Y*, *Z*, *c*, or **l** are used (note that the default includes *t* and *c*.

FILES

/unix	for the system namelist
/dev/swapdev	for the swapped out user block
/dev/mem	for the process table and swapped in user block.
or **unixcore**	if the *k* option is used.

BUGS

When any of the options requiring the user table are used, an unavoidable race condition can occur which may result in occasional garbled output.

The number of options is cumbersome; it is recommended that users with commonly used sets of options create shell files to invoke *sps*.

NAME

stack − stack trace from crash file

SYNOPSIS

stack [**−IsSUL**] crashfile offset file [unixfile]

DESCRIPTION

Stack prints a stack trace of a user process from the crash dump in *crashfile*. More importantly, it can print a stack trace of the system stack belonging to a process so that the state of suspension of the process when the system crashed can be determined. The U and S options will force a trace of only the user process or system stack, respectively. If the S option is used the *file* argument should not be provided.

Stack assumes that the namelist for the system is in /unix unless the s option is used in which case an alternate *unixfile* can be supplied. It also assumes that the operating system is in I and D space format unless the I option is selected to indicate I space only (all 11/40's). The offset is the address of the process in memory blocks (which can be gleaned from the dead command). The algorithm used to do the stack trace of the user program is the one used by the C debugger and will give up easily if the user's stack frame has been destroyed. The algorithm for doing the trace within the system is slower but does not give up as easily.

For those who are still interested in octal type dumps, the l option will print the system stack and provide octal R5 backtraces at each subroutine level.

BUGS

The algorithm used for tracing the system stack can get lost depending on the code that the C compiler generates and the degree to which the system's stack is destroyed.

SEE ALSO

dead(1M), tcmp(1M)

NAME

stamp − version stamp utility

SYNOPSIS

stamp -t[b][c][sh][s][a] [-v] [-p] [-r] -y'ID string' file ...

DESCRIPTION

Stamp is a utility to manipulate information of two basic types which indicate the identity of a file, namely the SCCS ID string and the system environment version stamp.

First, the ID string identifier supplied as the argument after the flag **−y** may be inserted in the appropriate place in the file for subsequent identification with the *what*(1S) command. In order to do this, **stamp** must be told the type of the target file in the form of an argument **−tx**, where **x** is one of the following:

b	binary file (executable a.out format); the ID is placed at the end of the file.
c	C source file; the ID is placed as a comment on the first line of the file.
sh	shell command file; the ID is placed as a comment on the first line of the file.
s	assembler source file; the ID is placed as a comment on the first line of the file.
a	archive file; the ID is placed in an archive member named *stamp.out*.

The **−p** option may be used to print the resulting stamped file on the standard output.

Second, the **-v** option is used to add or change the version stamp of an a.out format file. This stamp determines the system environment presented to the process when the file is executed. The version may be specified either by an octal number or by the name of the desired system environment. Currently defined are the following version names:

default (= 0)
: The default environment for CB-UNIX Release 2.0 is the Release 1.0 environment. This allows existing modules to execute on either system with no change.

res7 (= 1)
: Research Version 7...environment not implemented.

cbu1 (= 11)
: CB UNIX Release 1.0. This stamp is automatically provided by *occ*(1).

cbu2 (= 12)
: There is no special enviroment for CB UNIX Release 2.0; it is provided as a vehicle to facilitate transition to the Version 7 file system in Release 3.0. Thus **cbu3** is the stamp which provides the normal system environment.

cbu3 (= 13)
: CB UNIX Release 3.0, and the normal environment in Release 2.0. This stamp is automatically provided by *cc*(1).

unixts (= 21)
: UNIX/TS Release 1.0...environment not implemented.

pwbunix (= 31)
: PWB UNIX...environment not implemented.

The current version stamp of an executable file may be read using the **−r** option.

FILES

stamp.out	temporary file

SEE ALSO

cc(1), occ(1), ld(1), a.out(5)

DIAGNOSTICS

"Not an object" -- an attempt is being made to stamp a file with the **-tb** option which does not have a valid a.out header.

BUGS

The archive stamp option, **-ta**, should not be used with object libraries, as *ld*(1) cannot search past the *stamp.out* file.

NAME

start − restore printing of queued line printer jobs

SYNOPSIS

start type item [item]

DESCRIPTION

Start restores printing of a set of line printer requests which were inhibited by a *hold* or *restrain* command. *Start* causes printing to begin at the beginning of the job. *Type* (either **user**, **printer**, or **job**) indicates the type of *items* which follow. At least one *item* is required.

In the case of *type***user,** the *items* are login ids of users with queued jobs. For each specified user, *start* begins printing of all the user's line printer jobs, regardless of the queue in which the request resides. If the *type* is **printer**, then each *item* is a printer name, and *start* allows the printers to service their respective queues. For *type***job,** each of the specified jobs is restored to a printable state.

SEE ALSO

abort(1), init(1), hold(1), lpr(1), release(1), restrain(1)

NAME
startrek − clobber klingons

SYNOPSIS
/usr/games/startrek [**−a**] [file]

DESCRIPTION

STAR
TREK
Preliminary Version
by
Eric Allman
University of California
Berkeley

INTRODUCTION

Well, the federation is once again at war with the Klingon empire.
It is up to you,
as captain of the U.S.S. Enterprise,
to wipe out the invasion fleet and save the Federation.

For the purposes of the game
the galaxy is divided into 64 quadrants
on an eight by eight grid,
with quadrant 0,0 in the upper left hand corner.
Each quadrant is divided into 100 sectors
on a ten by ten grid.
Each sector contains one object
(e.g., the Enterprise, a Klingon, or a star).

Navigation is handled in degrees,
with zero being straight up
and ninty being to the right.
Distances are measured in quadrants.
One tenth quadrant is one sector.

The galaxy contains starbases,
at which you can dock to refuel,
repair damages, etc.
The galaxy also contains stars.
Stars usually have a knack for getting in your way,
but they can be triggered into going nova
by shooting a photon torpedo at one,
thereby (hopefully) destroying any adjacent Klingons.
This is not a good practice however,
because you are penalized for destroying stars.
Also, a star will sometimes go supernova,
which obliterates an entire quadrant.
You must never stop in a supernova quadrant,
although you may "jump over" one.

Some starsystems
have inhabited planets.
Klingons can attack inhabited planets

and enslave the populace,
ıt to work building more Klingon battle cruisers.

.TING UP THE GAME
ɹquest the game, issue the command

/usr/games/startrek

from the shell.
If
file
is stated,
a log of the game is written
onto that file.
If omitted,
the file is not written.
If the
—a
flag is stated before
file ,
that file is appended to
rather than created.

ɐ game will ask you what length game
you would like.
Valid responses are
short ,
medium ,
and
long .
ɪdeally the length of the game does not
affect the difficulty,
but currently the shorter games
.end to be harder than the longer ones.

ʏou will then be prompted for the skill,
to which you must respond
novice ,
fair ,
good ,
or
expert .
You should start out as a novice
and work up,
but if you really want to see how fast
you can be slaughtered,
start out with an expert game.
ɔr those of you who want a real challenge try
impossible !

In general,
throughout theɔu forget what is appropriate type '?' and a list of valid responses will be typed.

To get a copy oɐs, execute the command

man 1X startrek

ISSUING COMMANDS

If the game expects you to enter a command, it will say '**Command:** ' and wait for your response. Most commands can be abbreviated.

At almost any time you can type more than one thing on a line. For example, to move straight up one quadrant, you can type:

```
move 0 1
```

or you could just type

```
move
```

and the game would prompt you with

```
Course:
```

to which you could type

```
0 1
```

The '**1**' is the distance, which could be put on still another line. Also, the '**move**' command could have been abbreviated '**mov**', '**mo**', or just '**m**'.

If you are partway through a command and you change your mind, an interrupt will cancel the command.

Klingons generally cannot hit you if you don't consume anything (e.g., time or energy), so some commands are considered "free". As soon as you consume anything though -- POW!

THE COMMANDS

```
*********************
* Short Range Scan *
*********************
```

Mnemonic: srscan
Shortest Abbreviation: s
Full Commands: srscan
srscan yes/no
Consumes: nothing

The short range scan gives you a picture of the quadrant you are in, and (if you say 'yes') a status report which tells you a whole bunch of interesting stuff. You can get a status report alone by using the **status** command. An example follows:

```
          0 1 2 3 4 5 6 7 8 9      S.R. sensor scan for quadrant 0,3
        0 . . . . . . . * . * 0       stardate                3702.16
        1 . . E . . . . . . . 1       condition               RED
        2 . . . . . . . . . * 2       position                0,3/1,2
        3 * . . . . # . . . . 3       warp factor             5.0
        4 . . . . . . . . . . 4       total energy            4376
        5 . . * . * . . . . . 5       torpedoes               9
        6 . . . @ . . . . . . 6       shields                 down, 78%
        7 . . . . . . . . . . 7       Klingons left           3
        8 . . . K . . . . . . 8       time left               6.43
9   . . . . . . * . . . 9       life support          damaged, reserves = 2.4
          0 1 2 3 4 5 6 7 8 9      Distressed Starsystem Marcus XII
```

The cast of characters is as follows:

E	the hero
K	the villain
#	the starbase
*	stars
@	inhabited starsystem
.	empty space

The name of the starsystem is listed underneath the short range scan. The word '**distressed**', if present, means that the starsystem is under attack.

Short range scans are absolutely free. They use no time, no energy, and they don't give the

Klingons another chance to hit you.

```
******************
* Status Report *
******************
```

Mnemonic: status
Shortest Abbreviation: st
Consumes: nothing

This command gives you information about the current status of the game and your ship, as follows:

Stardate -- The current stardate.

Condition -- as follows:
RED -- in battle
YELLOW -- low on energy
GREEN -- normal state
DOCKED -- docked at starbase
CLOAKED -- the cloaking device is activated

Position -- Your current quadrant and sector.

Warp Factor -- The speed you will move at when you move under warp power (with the move command).

Total Energy -- Your energy reserves. If they drop to zero, you die. Energy regenerates, but the higher the skill of the game, the slower it regenerates.

Torpedoes -- How many photon torpedoes you have left.

Shields -- Whether your shields are up or down, and how effective they are if up (what percentage of a hit they will absorb).

Klingons Left -- Guess.

Time Left -- How long the Federation can hold out if you sit on your fat ass and do nothing. If you kill Klingons quickly, this number goes up, otherwise, it goes down. If it hits zero, the Federation is conquered.

Life Support -- If '**active**', everything is fine. If '**damaged**', your reserves tell you how long you have to repair your life support or get to a starbase before you starve, suffocate, or something equally unpleasant.

Current Crew -- The number of crew members left. This figures does not include officers.

Brig Space -- The space left in your brig for Klingon captives.

Status information is absolutely free.

```
*********************
* Long Range Scan *
*********************
```

Mnemonic: lrscan
Shortest Abbreviation: l
Consumes: nothing

Long range scan gives you information about the eight quadrants that surround the quadrant you're in. A sample long range scan follows:

```
          -2-     -3-     -4-          Long range scan for quadrant 0,3
           *       *       *
     0    108      E6      19
     1      9     ///       8
```

The three digit numbers tell the number of objects in the quadrants. The units digit tells the number of stars, the tens digit the number of starbases, and the hundreds digit is the number of Klingons. '*' indicates the negative energy barrier at the edge of the galaxy, which you cannot enter. '///' means that that is a supernova quadrant and must not be entered.

```
*******************
* Damage Report *
*******************
```

Mnemonic: damages
Shortest Abbreviation: da
Consumes: nothing

A damage report tells you what devices are damaged and how long it will take to repair them. Repairs proceed faster when you are docked at a starbase.

```
********************
* Set Warp Factor *
********************
```

Mnemonic: warp
Shortest Abbreviation: w
Full Command: warp factor
Consumes: nothing

The warp factor tells the speed of your starship when you move under warp power (with the move command). The higher the warp factor, the faster you go, and the more energy you use.

The minimum warp factor is 1.0 and the maximum is 10.0. At speeds above warp 6 there is danger of the warp engines being damaged. The probability of this increases at higher warp speeds. Above warp 9.0 there is a chance of entering a time warp.

```
*****************************
* Move Under Warp Power *
*****************************
```

Mnemonic: move
Shortest Abbreviation: m
Full Command: move course distance
Consumes: time and energy

This is the usual way of moving. The course is in degrees and the distance is in quadrants. To move one sector specify a distance of 0.1.

Time is consumed proportionately to the inverse of the warp factor squared, and directly to the distance. Energy is consumed as the warp factor cubed, and directly to the distance. If you move with your shields up it doubles the amount of energy consumed.

When you move in a quadrant containing Klingons, they get a chance to attack you.

The computer detects navigation errors. If the computer is out, you run the risk of running into things.

The course is determined by the Space Inertial Navigation System [SINS]. As described in Star Fleet Technical Order TO:02:06:12, the SINS is calibrated, after which it becomes the base for navigation. If damaged, navigation becomes inaccurate. When it is fixed, Spock recalibrates it, however, it cannot be calibrated extremely accurately until you dock at starbase.

```
********************************
* Move Under Impulse Power *
********************************
```

Mnemonic: impulse
Shortest Abbreviation: i
Full Command: impulse course distance
Consumes: time and energy

The impulse engines give you a chance to maneuver when your warp engines are damaged; however, they are incredibly slow (0.095 quadrants/stardate). They require 20 units of energy to engage, and ten units per sector to move.

The same comments about the computer and the SINS apply as above.

There is no penalty to move under impulse power with shields up.

```
**********************
* Deflector Shields *
**********************
```

Mnemonic: shields
Shortest Abbreviation: sh
Full Command: shields up/down
Consumes: energy

Shields protect you from Klingon attack and nearby novas. As they protect you, they weaken. A shield which is 78% effective will absorb 78% of a hit and let 22% in to hurt you.

The Klingons have a chance to attack you every time you raise or lower shields. Shields do not rise and lower instantaneously, so the hit you receive will be computed with the shields at an intermediate effectiveness.

It takes energy to raise shields, but not to drop them.

```
********************
* Cloaking Device *
********************
```

Mnemonic: cloak
Shortest Abbreviation: cl
Full Command: cloak up/down
Consumes: energy

When you are cloaked, Klingons cannot see you, and hence they do not fire at you. They are useful for entering a quadrant and selecting a good position, however, weapons cannot be fired through the cloak due to the huge energy drain that it requires.

The cloak up command only starts the cloaking process; Klingons will continue to fire at you until you do something which consumes time.

```
****************
* Fire Phasers *
****************
```

Mnemonic: phasers
Shortest Abbreviation: p
Full Commands: phasers automatic amount
phasers manual amt1 amt2 ... amtn
Consumes: energy

Phasers are energy weapons; the energy comes from your ship's reserves ('*total energy*' on a **srscan**). It takes about 250 units of hits to kill a Klingon. Hits are cumulative as long as you stay in the quadrant.

Phasers become less effective the further from a Klingon you are. Adjacent Klingons receive about 90% of what you fire, at five sectors about 60%, and at ten sectors about 35%. They have no effect outside of the quadrant.

Phasers cannot be fired while shields are up; to do so would fry you. They have no effect on starbases or stars.

In automatic mode the computer decides how to divide up the energy among the Klingons present; in manual mode you do that yourself, specifying nearest Klingon first.

****
* Fire Photon Torpedoes *
****

Mnemonic: torpedo
Shortest Abbreviation: t
Full Command: torpedo course [yes/no] [burst angle]
Consumes: torpedoes

Torpedoes are projectile weapons -- there are no partial hits. You either hit your target or you don't. A hit on a Klingon destroys him. A hit on a starbase destroys that starbase (woops!). Hitting a star usually causes it to go nova, and occasionally supernova.

Photon torpedoes cannot be aimed precisely. They can be fired with shields up, but they get even more random as they pass through the shields.

Torpedoes may be fired in bursts of three. If this is desired, the burst angle is the angle between the three shots, which may vary from one to fifteen. The word '**no**' says that a burst is not wanted; the word '**yes**' (which may be omitted if stated on the same line as the burst angle) says that a burst is wanted.

Photon torpedoes have no effect outside the quadrant.

****
* Onboard Computer Request *
****

Mnemonic: computer
Shortest Abbreviation: c
Full Command: computer request; request;...
Consumes: nothing

The computer command gives you access to the facilities of the onboard computer, which allows you to do all sorts of fascinating stuff. Computer requests are:

score -- Shows your current score.

course [quad] [/] sect -- Computes the course and distance from wherever you are to the given location. The quadrant may be omitted if / is present; in this case the current quadrant is assumed.

record -- prints the computer record of the known galaxy, i.e., everything that you have seen with a long range scan. The format is the same as on a long range scan, except that '...' means that you don't yet know what is there, and '.#.' means that you know that a starbase exists, but you don't know anything else.

trajectory -- prints the course and distance to all the Klingons in the quadrant.

warpcost dist warp_factor -- computes the cost in time and energy to move 'dist' quadrants at warp 'warp_factor'.

impcost dist -- same as warpcost for impulse engines.

pheff range -- tells how effective your phasers are at a given range.

distresslist -- gives a list of currently distressed starbases and starsystems.

More than one request may be stated on a line by seperating them with semicolons.

```
********************
* Dock at Starbase *
********************
```

Mnemonic: dock
Shortest Abbreviation: do
Consumes: nothing

You may dock at a starbase when you are in one of the eight adjacent sectors. Your shields must be down to dock.

When you dock you are resupplied with energy, photon torpedoes, and life support reserves. Repairs are also done faster at starbase.

Starbases have their own deflector shields, so you are safe from attack while docked.

```
**************************
* Undock from Starbase *
**************************
```

Mnemonic: undock
Shortest Abbreviation: u
Consumes: nothing

This just allows you to leave starbase so that you may proceed on your way.

```
********
* Rest *
********
```

Mnemonic: rest
Shortest Abbreviation: r
Full Command: rest time
Consumes: time

This command allows you to rest to repair damages. It is not advisable to rest while under attack.

```
***************************
* Call Starbase For Help *
***************************
```

Mnemonic: help
Shortest Abbreviation: help
Consumes: nothing

You may call starbase for help via your subspace radio. Starbase has long range transporter beams to get you. Problem is, they can't always rematerialize you.

You should avoid using this command unless absolutely necessary, for the above reason and because it counts heavily against you in the scoring.

```
****************
* Visual Scan *
****************
```

Mnemonic: visual
Shortest Abbreviation: v
Full Command: visual course
Consumes: time

When your short range scanners are out, you can still see what is out 'there' by doing a visual scan. Unfortunately, you can only see three sectors at one time, and it takes 0.005 stardates to perform.

The three sectors in the general direction of the course specified are examined and displayed.

```
******************
* Abandon Ship *
******************
```

Mnemonic: abandon
Shortest Abbreviation: abandon
Consumes: nothing

The officers escape the Enterprise in the shuttlecraft. If the transporter is working and there is an inhabitable starsystem in the area, the crew beams down, otherwise you leave them to die. You are given an old but still usable ship, the Faire Queene.

```
************************
* Terminate the Game *
************************
```

Mnemonic: quit
Shortest Abbreviation: quit
Full Command: quit

Cancels the current game. No score is computed. If you answer 'yes', a new game will be started, otherwise trek exits.

```
*****************
* Call the Shell *
*****************
```

Mnemonic: !
Shortest Abbreviation: !

Temporarily escapes to the shell. When you log out of the shell you will return to the game.

SCORING

The scoring algorithm is rather complicated. Basically, you get points for each Klingon you kill, for your Klingon per stardate kill rate, and a bonus if you win the game. You lose points for the number of Klingons left in the galaxy at the end of the game, for getting killed, for each star, starbase, or inhabited starsystem you destroy, for calling for help, and for each casualty you incur.

You will be promoted if you play very well. You will never get a promotion if you call for help, abandon the Enterprise, get killed, destroy a starbase or inhabited starsystem, or destroy too many stars.

REFERENCES

Command	Uses	Consumes
ABANDON	shuttlecraft, transporter	-
CLoak Up/Down	cloaking device	energy
Computer request; request;...	computer	-
DAmages	-	-
DESTRUCT	computer	-
DOck	-	-
HELP	subspace radio	-
Impulse course distance	impulse engines, computer, SINS	time, energy
Lrscan	L.R. sensors	-
Move course distance	warp engines, computer, SINS	time, energy
Phasers Automatic amount	phasers, computer	energy
Phasers Manual amt1 amt2... amtn	phasers	energy
Torpedo course [Yes] angle/No	torpedo tubes	torpedoes
Rest time	-	time
SHELL	-	-
SHields Up/Down	shields	energy
Srscan [Yes/No]	S.R. sensors	-
STatus	-	-
QUIT Yes/No	-	-
Undock	-	-
Visual course	-	time
Warp warp_factor	-	-

NAME

strip − remove symbols and relocation bits

SYNOPSIS

strip name ...

DESCRIPTION

Strip removes the symbol table and relocation bits ordinarily attached to the output of the assembler and loader. This is useful to save space after a program has been debugged.

The effect of *strip* is the same as use of the −s option of *ld.*

If *name* is an archive file, *strip* will remove the local symbols from any a.out format members it finds in the archive.

FILES

/tmp/stm*	temporary file

SEE ALSO

ld(1)

NAME

stty – set teletype options

SYNOPSIS

stty option ...

DESCRIPTION

Stty will set certain I/O options on the current output teletype. The option strings are selected from the following set:

even	allow even parity
–even	disallow even parity
odd	allow odd parity
–odd	disallow odd parity
raw	raw mode input (no erase, kill, interrupt, quit, EOT; parity bit passed back)
–raw	negate raw mode
–nl	allow carriage return for new-line, and output CR-LF for carriage return or new-line
nl	accept only new-line to end lines
echo	echo back every character typed
–echo	do not echo characters
lcase	map upper case to lower case
–lcase	do not map case
–tabs	replace tabs by spaces in output
tabs	preserve tabs
delay	calculate cr, tab, and form-feed delays
–delay	no cr/tab/ff delays
tdelay	calculate tab delays
–tdelay	no tab delays
hdplx	half duplex (disable reception during transmission)
–hdplx	full duplex
tstread	test read (don't wait for carriage return to terminate read)
–tstread	normal read termination
hangup	dial-up line (hangup after last close)
–hangup	non dial-up
lfdelay	line-feed delay
–lfdelay	no line-feed delay
sccmod	enable sccmode (special editing, break and del characters)
–sccmod	disable sccmode
xclude	exclude future opens of this teletype
–xclude	Allow future opens of this teletype
tandemi	respond to xoff and xon when transmitted by the terminal by stopping and starting output.

-tandemi	treat xoff and xon as normal input characters

In addition to the above modes it is also possible to specify the terminal type. If this is done it must be the first option. The legal terminal types depend on the configuration of the system, but the character strings accepted by *stty* include "**none**", "**tec**", "**delta**", "**dta**", "**tektronix**", "**tex**", "**vt61**", "**ds40-1**", "**ds40-2b**", and "**hp45**". When a terminal type other than "**none**" is in use, several more flags become valid:

anl	automatic new-line when character printed in column 80
−anl	no automatic new-line
snl	special new-line function (only implemented for "**ds40-1**")
−snl	no special new-line function
lcf	special action on last column of last line
−lcf	no special action on last column of display
default	set terminal flags to default for this terminal type

Certain special options look at the following argument and interpret it as a numeric value. These options are "**speed**", "**vrow**", "**stop**", and "**data**". "**Speed**" allows the user to change the speed of a line and "**vrow**" allows the user to set the variable row on crts above which the screen will be frozen. Note that **vrow** is meaningful only on crts. "**stop**" and "**data**" allow the user to specify the number of stop and data bits to be transmitted.

SEE ALSO

stty(2), gtty(1)

NAME

su − become super-user or another user

SYNOPSIS

su [−] [name [arg ...]]

DESCRIPTION

Su allows one to become another user without logging off. The default user *name* is **root** (i.e., super-user).

To use *su*, the appropriate password must be supplied (unless one is already super-user). If the password is correct, *su* will execute the Shell with the user-id set to that of the specified user. To restore normal user-id privileges, type an end-of-file to the super-user Shell.

Any additional arguments are passed to the Shell, permitting the super-user to run Shell procedures with restricted privileges (an *arg* of the form **−c** *string* executes *string* via the Shell).

An initial − flag causes the environment to be changed to the one that would be expected if the user actually logged in again. This is done by invoking the shell with an *arg0* of **−su** causing the **.profile** in the home directory of the new user-id to be executed. Otherwise, the environment is passed along. Note that the **.profile** can check *arg0* for **−sh** or **−su** to determine how it was invoked.

FILES

/etc/passwd	system's password file
$HOME/.profile	user's profile

SEE ALSO

env(1), login(1), sh(1), environ(7).

NAME
sum − sum and count blocks in a file

SYNOPSIS
sum file

DESCRIPTION
Sum calculates and prints a 16-bit checksum for the named file, and also prints the number of blocks in the file. It is typically used to look for bad spots, or to validate a file communicated over some transmission line.

SEE ALSO
wc(1)

DIAGNOSTICS
'Read error' is indistinguishable from end of file on most devices; check the block count.

NAME

sync − update the super block

SYNOPSIS

sync

DESCRIPTION

Sync executes the *sync* system primitive. If the system is to be stopped, *sync* must be called to insure file system integrity. See *sync*(2) for details.

SEE ALSO

sync(2), update(1)

NAME

tail − deliver the last part of a file

SYNOPSIS

tail [±[number][**lbc**] [**f**]] [file]

DESCRIPTION

Tail copies the named file to the standard output beginning at a designated place. If no file is named, the standard input is used.

Copying begins at distance +*number* from the beginning, or −*number* from the end of the input (if *number* is null, the value 10 is assumed). *Number* is counted in units of lines, blocks, or characters, according to the appended option **l**, **b**, or **c**. When no units are specified, counting is by lines.

With the **f** ("follow") option, if the input file is not a pipe, the program will not terminate after the line of the input file has been copied, but will enter an endless loop, wherein it sleeps for a second and then attempts to read and copy further records from the input file. Thus it may be used to monitor the growth of a file that is being written by some other, asynchronous process. For example, the command:

```
tail −f fred
```

will print the last ten lines of the file *fred*, followed by any lines that are appended to *fred* between the time *tail* is initiated and killed.

SEE ALSO

dd(1).

BUGS

Tails relative to the end of the file are treasured up in a buffer, and thus are limited in length. Various kinds of anomalous behavior may happen with character special files.

NAME

talk − allow user to listen and talk to one or more other users simultaneously

SYNOPSIS

talk [**−n** alias] [user user ...]

DESCRIPTION

Talk allows a group of users to simultaneously talk and listen to each other. If you run *talk* without specifying any users with whom you would like to speak, you immediately join the first conversation currently in progress which is not restricted in such a way that you cannot join it. All other participants in the conversation are alerted to the fact that you have arrived by "*The Chair*". "*The Chair*" is the name of the central distributor process, which oversees the conversation and distributes lines of text typed in by each participant to all the other users. If there is no conversation in progress when you type *talk*, a new central distributor process is created with you as the first member of the conversation.

If you run *talk* and specify a list of users with whom you would like to speak, *talk* attempts to put you into a single conversation with these users. If some or all of the users specified are already running *talk* and are in more than one conversation, *talk* will ask you to pick the conversation you wish to join. If those users who are running talk and with whom you wish to speak are all in the same conversation, *talk* attempts to join you to that conversation. All users you listed who are not running *talk* at the time are alerted to the fact that you wish to speak to them if you successfully join the conversation.

When you first join a conversation, "*The Chair*" tells you the names and aliases of all the other people in the conversation, and which tty line they are logged in on. Any time a user leaves the conversation all the remaining members are apprised on the fact. If the owner of the conversation leaves, ownership is passed to the first person that *the Chair* finds who is still in the conversation and they are alerted to the fact that they now own the conversation. If there is no one left in the conversation, it becomes available again. When the last user leaves *talk*, the central distributor process terminates.

Talk is structured in such a way that people are not interrupted if more than one person is talking at the same time. If there is a terminal type specified (see *stty*(1) and *ioctl*(2)), *talk* divides the terminal screen into two sections, reserving the top two lines for your typing. Every partially typed line appears at the top. When it is complete it is sent to all the participants of the convervation.

If there is no terminal type, then *talk* saves up any output from other users while you are typing in a line. When you either finish your line or erase it, *talk* then sends you all the queued output. In this way you are not interrupted while typing.

Normally you are identified by your login name. This name can be changed to an *alias* at the time *talk* is invoked with the **−n** flag. It can also be changed during the conversation. See **˜name** below. This is useful if more than one person with the same login id is involved in a conversation.

Talk can be asked to *alert* other users that you would like to talk to them. This is the only time that *talk* sends anything to a user's terminal uninvited. If you specify a list of other user ids at the time *talk* is invoked, it will send out the following message to each of those users who it finds logged in:

"*your_login_id*" wants to "talk" to you.

Talk will report any user listed, who it doesn't find logged in. Also any user whose terminal is set so that *talk* can't send a message will be reported.

Talk has a number commands that can be typed once you have started *talk*. Any line beginning with a ˜ (tilda) is assumed to be a command to *talk*. The commands follow. Abbreviations appear in ()'s.

˜!*cmd*

Escape to the shell and perform *cmd*. All messages from other users will continue to arrive, but nothing typed by you will be transmitted to the other participants while you are escaped.

˜**help** (˜**h**) [cmd cmd ...]

Type a help message. If one or more of the *talk* commands is specified, then type a help message for the specific command. There are help messages for **!**, **help**(**h**), **alert**(**a**), **name**(**n**), and **users**(**u**).

˜**alert** (˜**a**) user [user ...]

Send message to the specified users, if they are logged in, saying that you would like to *talk* to them. This results in the same action as specifying users when originally starting *talk*.

˜**name** (˜**n**) [alias]

If no argument is specified, your current name and alias, if any, is printed out. If a new alias is specified, then the alias is changed to the new one and all other participants in the conversation are alerted to the fact that you have changed your alias.

˜**users** (˜**u**)

List all the current participants in the conversation, their aliases, and what terminal they are logged in on. You are marked with an ->. The owner of a conversation is denoted by an *. If the conversation is restricted and if you are the owner, the various restrictions to the conversation are listed. If the conversation is restricted, but you are not the owner, only the word *Restricted* appears after the conversation number.

˜**join** (˜**j**) cnum

˜**knock** (˜**k**) cnum

Attach to the conversation numbered *cnum*. The conversation must already be in existence. If you can join immediately, you are detached from your current conversation and attached to the new conversation, with appropriate warnings to the members of the old and new conversations. If you must get permission to join the conversation, *talk* puts you into a wait loop and requests permission from the owner of the conversation. If you get tired, a <*del*> will exit you from the waiting state.

˜**create** (˜**c**) [−**l**] [[+**u**|−**u** |+**g**|−**g**] name [...]] ...

Create a new conversation and make you the owner of it. If there is room for another conversation this command will succeed. At the time of creation it is possible to specify restrictions on who may join the new conversation. See ˜*lock* for the details on the restriction switches.

˜**lock** (˜**l**) [−**l**] [[+**u**|−**u** |+**g**|−**g**] name [...]] ...

The owner of a conversation may restrict entry to the conversation. If no arguments are given to the lock command or when the conversation is created with the ˜**create** command only the −**l** switch is specified, the conversation will be totally locked and each new participant will be required to get specific permission from the owner of the conversation to join.

Specific restrictions can be made with the **u** and **g** switches. +**u** followed by a list of user ids means that these users are allowed to enter the conversation without having to ask permission. −**u** means that these users must ask permission. It is not permissible to mix +**u** and −**u** or +**g** and −**g** switches. If you specify that some people can enter without knocking, then all other users will have to ask specific permission. If you specify that some users are to ask permission, then all other users will be allowed to enter without asking. The +g and −g switches work the same way for groups.

~unlock (~unl)

If you are the owner of the conversation, the conversation is unlocked and all restrictions against joining the conversation are removed.

~admit (~**ad**) name [name ...]

Admit any person having one of the listed names to this conversation if they are waiting for permission. Only the owner of a conversation can admit people. If no one is waiting with the names listed, a message from *the Chair* will be sent to you.

~deny (~**d**) name [name ...] [−**r** reason]

Deny any person having one of the listed names permission to enter your conversation. Only the owner of a conversation can deny people entry. If no one is waiting with the names listed, a message from *the Chair* will be sent to you. A reason for denying permission can be included with the −**r** switch. Everything after the −**r** switch to the end of the line is taken as a reason and is appended to the end of the "Permission denied." message that is sent back to a user who is being denied entry.

If at any time the divider of "*******"s one the second line of the screen gets scribbled, as might happen during a shell breakout if something is run that alters the variable scrolling, typing a control L (^*L*) will cause the divider on line two to be redrawn.

SEE ALSO

write(1)

BUGS

A maximum of 16 people can use *talk* at one time.

Echo can be slow since *talk* runs in **RAW NOECHO** mode. When running on a terminal without a terminal type set, the first character of a line causes the typing of a newer header, if required, as well as echoing of the character. It can also be slower than any other character of the line since before it is echoed, the local process informs the central distributing process not to send any more messages and waits for an acknowledgement. When a line is complete, the local process tells the central process that it is okay to send lines again.

NAME

tar − tape archiver

SYNOPSIS

tar [key] [name ...]

DESCRIPTION

Tar saves and restores files on magtape. Its actions are controlled by the *key* argument. The *key* is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to the command are file or directory names specifying which files are to be dumped or restored. The names may be patterns using the name generating syntax of the shell. In this case, the pattern meta-characters will match / characters. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

The function portion of the key is specified by one of the following letters:

r The named files are written on the end of the tape. The **c** function implies this.

x The named files are extracted from the tape. If the named file matches a directory whose contents had been written onto the tape, this directory is (recursively) extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, the entire content of the tape is extracted. Note that if multiple entries specifying the same file are on the tape, the last one overwrites all earlier.

t The names of the specified files are listed each time they occur on the tape. If no file argument is given, all of the names on the tape are listed.

u The named files are added to the tape if either they are not already there or have been modified since last put on the tape.

c Create a new tape; writing begins on the beginning of the tape instead of after the last file. This command implies **r**.

The following characters may be used in addition to the letter which selects the function desired.

0,...,7 This modifier selects the drive on which the tape is mounted. The default is **1**.

v Normally *tar* does its work silently. The **v** (verbose) option causes it to type the name of each file it treats preceded by the function letter. With the **t** function, **v** gives more information about the tape entries than just the name.

w causes *tar* to print the action to be taken followed by file name, then wait for user confirmation. If a word beginning with 'y' is given, the action is performed. Any other input means don't do it.

f causes *tar* to use the next argument as the name of the archive instead of /dev/mt?. If the name of the file is '−', tar writes to standard output or reads from standard input, whichever is appropriate. Thus, *tar* can be used as the head or tail of a filter chain *Tar* can also be used to move hierarchies with the command

```
cd fromdir; tar cf − . | (cd todir; tar xf −)
```

b causes *tar* to use the next argument as the blocking factor for tape records. The default is 1, the maximum is 20. This option should only be used with raw magnetic tape archives (See **f** above). The block size is determined automatically when reading tapes (key letters 'x' and 't').

l tells *tar* to complain if it cannot resolve all of the links to the files dumped. If this is not specified, no error messages are printed.

m tells *tar* to not restore the modification times. The mod time will be the time of extraction.

FILES

/dev/mt?
/tmp/tar*

DIAGNOSTICS

Complaints about bad key characters and tape read/write errors.
Complaints if enough memory is not available to hold the link tables.

BUGS

There is no way to ask for the *n*-th occurrence of a file.
Tape errors are handled ungracefully.
The **u** option can be slow.
The **b** option should not be used with archives that are going to be updated. The current magtape driver cannot backspace raw magtape. If the archive is on a disk file the **b** option should not be used at all, as updating an archive stored in this manner can destroy it.
The current limit on file name length is 100 characters.

NAME

tbl − format tables for nroff or troff

SYNOPSIS

tbl [**−TX**] [files] ...

DESCRIPTION

Tbl is a preprocessor for formatting tables for *nroff*(1) or *troff*(1). The input files are copied to the standard output, except for lines between .TS and .TE command lines, which are assumed to describe tables and are re-formatted by *tbl*. (The .TS and .TE command lines are not altered by *tbl*). Details are given in the reference manual cited below. As an example, letting → represent a tab (which should be typed as a genuine tab), the input:

```
.TS
center ;
cI s s
c c s
c c c
l n n .
Household Population
Town→Households
→Number→Size
=
Bedminster→789→3.26
Bernards Twp.→3087→3.74
Bernardsville→2018→3.30
Bound Brook→3425→3.04
Bridgewater→7897→3.81
Far Hills→240→3.19
.TE
```

yields:

Household Population

Town	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Bridgewater	7897	3.81
Far Hills	240	3.19

If no arguments are given, *tbl* reads the standard input, so it may be used as a filter. When it is used with *eqn*(1) or *neqn*(1), *tbl* should come first to minimize the volume of data passed through pipes.

The −TX option forces *tbl* to use only full vertical line motions, making the output more suitable for devices that cannot generate partial vertical line motions (e.g., line printers).

SEE ALSO

TBL − A Program to Format Tables by M. E. Lesk
eqn(1), mm(1), mmt(1), nroff(1), mm(7), mv(7).

NAME

tc − phototypesetter simulator

SYNOPSIS

tc [**−t**] [**−s***N*] [**−p***L*] [*file*]

DESCRIPTION

Tc interprets its input (standard input default) as device codes for a Wang-Graphic Systems, Inc. phototypesetter (C/A/T). The standard output of *tc* is intended for a Tektronix 4014 terminal with ASCII and APL character sets. The sixteen typesetter sizes are mapped into the 4014's four sizes; the entire TROFF character set is drawn using the 4014's character generator, using overstruck combinations where necessary. Typical usage is:

```
troff −t file | tc
```

At the end of each page, *tc* waits for a new-line (empty line) from the keyboard before continuing on to the next page. In this wait state, the command **e** will *suppress* the screen erase before the next page; **s***N* will cause the next *N* pages to be skipped; and **!***line* will send *line* to the shell.

The command line options are:

−t Don't wait between pages (for directing output into a file).

−s*N* Skip the first *N* pages.

−p*L* Set page length to *L*; *L* may include the scale factors **p** (points), **i** (inches), **c** (centimeters), and **P** (picas); default is picas.

SEE ALSO

plot(1G), troff(1).

BUGS

Font distinctions are lost.

NAME

tcmp − text comparison for crash dump

SYNOPSIS

tcmp [**−Iusvtabcdho**] crashfile [offset][.][**m**] [file]

DESCRIPTION

When a unix system crashes with no console message, it is sometimes fruitful to determine whether any parts of the operating system have been destroyed. *Tcmp* is a program for comparing the text (instructions) of a program or the operating system with the text that appears in a crash dump. The differences can be printed in summary form or the contents of the modified and original locations can be printed. There are a number of options which allow the differences to be printed in ascii, octal, etc. and options which allow the address of the differences to be printed in absolute, virtual or symbolic terms. If the values of the differences are not explicitly requested by using one of the format options, only a summary of the differences is printed.

There are several options which select whether a user program or the operating system is to be compared. The default is to compare the text in the file **/unix** with the *crashfile* and to assume that the system is in an I and D space format. The options are,

I indicates that an I space only operating system (all 11/40's) is to be compared.

s This option allows the specification of a file other than **/unix** for the comparison. No *offset* argument needed when comparing the operating system text with the *crashfile.*

u indicates that a user program is to be compared. The address (*offset*) of the user program must be specified and the file name of the user program must be given (see below).

The following options select the type of address to be printed. The default is to print the absolute address. The address options are,

v indicates that the virtual address of the start address is to be printed.

t indicates that a symbolic address is to be constructed from the namelist for the start address.

If only the above options are used, a summary of the differences is printed. The summary consists of the absolute address of the first difference followed by a byte count of the number of locations starting at that address which differ. If the **v** and/or **t** options are used, then the start address is repeated and interpreted in virtual and/or symbolic terms. In all cases, when a summary is requested, the absolute memory address and the byte count are printed.

If any of the following options are used, the contents of the modified locations are printed. The differences are printed as two line pairs. The first line prints what the location should contain and the second prints the contents of the location from the crashfile. (An asterisk is printed at the beginning of each line containing modified data.) The meaning of the format argument options are:

a interprets words as PDP-11 instructions and dis-assembles the operation code. Unknown operation codes print as ???.

b interprets bytes in octal.

c interprets bytes in ascii. Unknown ascii characters are printed as \?.

d interprets words in decimal.

h interprets words in hex.

o interprets words in octal.

The *crashfile* argument specifies a file which contains a system crash dump.

The *offset* argument is used when comparing a user program with the crash file and specifies the offset in the crash file where the text is to be found. This offset can be found by using the **−m** option of the *dead*(1) command which produces a sorted memory map for the crash file. For nonreentrant programs, the offset is simply the address given by the *dead*(1) command. For reentrant programs, the **−m** option of the *dead*(1) command indicates where the text is located by appending a T as a subscript to the memory address.

The *offset* is normally interpreted as octal bytes. If '**.**' is appended, the offset is interpreted in decimal. If an **m** is appended, the offset is interpreted in memory blocks (64 bytes). The address printed by the dead command is in memory blocks.

The *file* argument specifies where the object for the operating system resides when a file other than "**/unix**" contains the object (**s** option) or the pathname of a file containing the object for a user program when a user program's text is being compared (**u** option). *Tcmp* complains if the file specified does not have a namelist and the **t** option has been selected.

As part of every invokation, *tcmp* gives the absolute and virtual address range for the program being compared.

The command

```
tcmp /dev/mem
```

should produce no differences while the system is running.

SEE ALSO

od(1), dead(1M)

BUGS

The '**−t**' option is slow.
Crashfiles produced on I space only systems have locations 04-034 modified by the system dump routine.

NAME

tee − pipe fitting

SYNOPSIS

tee [**−i**] [**−a**] [file] ...

DESCRIPTION

Tee transcribes the standard input to the standard output and makes copies in the *files.* The **−i** option ignores interrupts; the **−a** option causes the output to be appended to the *files* rather than overwriting them.

NAME

tek, vplot, t300, t300s, t450 − graphics filters

SYNOPSIS

tek
vplot [name]
t300
t300s
t450
vt0

DESCRIPTION

These commands read plotting instructions (see *plot*(5)) from the standard input, and produce device-dependent plotting instructions on the standard output.

Tek produces a plot for a Tektronix 4014 terminal on the standard output.

Vplot produces a plot on the Versatec matrix printer-plotter, and leaves a scan-converted copy of the file on **/sys/tmp/picture**. *Vplot* interprets an argument as the name of a previously scan-converted file, and plots that file on the Versatec.

T300 produces a plot for a GSI 300 terminal on the standard output.

T300s produces a plot for a GSI 300s terminal on the standard output.

T450 produces a plot for a DASI 450 terminal on the standard output.

Vt0 produces a plot for a storage scope on an auxiliary computer.

FILES

/sys/tmp/picture: scan-conversion buffer for *vplot.*

SEE ALSO

plot(1)