NAME dc -- desk calculator

SYNOPSIS dc

DESCRIPTION dc is an arbitrary precision integer arithmetic package. The overall structure of dc is a stacking (reverse Polish) calculator. The following constructions are recognized by the calculator:

number
: The value of the number is pushed on the stack. If the number starts with a zero, it is taken to be octal, otherwise it is decimal.

+ - * / %
: The top two values on the stack are added (±), subtracted (—), multiplied (*), divided (*) or remaindered (%) The two entries are popped off of the stack, the result is pushed on the stack in their place.

sx
: The top of the stack is popped and stored into a register named x, where x may be any character.

lx
: The value in register x is pushed on the stack. The register x is not altered.

d
: The top value on the stack is pushed on the stack. Thus the top value is duplicated.

p
: The top value on the stack is printed in decimal. The top value remains unchanged.

f
: All values on the stack are popped off and printed in decimal.

r
: All values on the stack are popped.

q
: exit.

h
: print brief synopsis of commands to dc.

new—line
space
: ignored.

An example to calculate the monthly, weekly and hourly rates for a $10,000/year salary.

```
10000
    100* (now in cents)
    dsa (non—destructive store)
    12/        (pennies per month)
    1a52/      (pennies per week)
    d10*       (deci—pennies per week)
    375/       (pennies per hzur)
    f          (print all results)
      (3) 512
      (2)  19230
      (1)  83333
```

FILES

SEE ALSO

DIAGNOSTICS ? (x) for unrecognized character x.

BUGS % doesn’t work correctly.

OWNER ken

NAME df -- disk free

SYNOPSIS df [filesystem]

DESCRIPTION prints out the number of free blocks available on a file system. If the file system is unspecified, the free space on /dev/rf0 and /dev/rk0 is printed.

FILES /dev/rf0, /dev/rk0

SEE ALSO check

DIAGNOSTICS

BUGS

OWNER ken, dmr

NAME dsw -- delete interactively

SYNOPSIS dsw [directory]

DESCRIPTION For each file in the given directory ("." if not specified) dsw types its name. If y is typed, the file is deleted; if "x", dsw exits; if anything else, the file is not removed.

FILES

SEE ALSO rm

DIAGNOSTICS “?”

BUGS The name dsw is a carryover from the ancient past. Its etymology is amusing but the name is nonetheless ill—advised.

OWNER dmr, ken

NAME dtf -- DECtape format

SYNOPSIS /etc/dtf

DESCRIPTION dtf will write timing tracks, mark tracks and block numbers on a virgin DECtape. The format is DEC standard of 578 blocks of 256 words each. The end zones are a little longer than standard DEC.

Before use, the tape to be formatted should be mounted on drive 0. The ‘wall’ and ‘wtm’ switches should be enabled. After the tape is formatted, the switches should be disabled to prevent damage to subsequent tapes due to a controller logic error.

FILES

SEE ALSO sdate

DIAGNOSTICS "?" is typed for any error detected.

BUGS This program does physical I/O on drive 0. The processor priority is set very high due to very stringent real time requirements. This means that all time sharing activities are suspended during the formatting (about 1.5 minutes) The real time clock will also be slow.

OWNER ken

NAME du -- summarize disk usage

SYNOPSIS du [—s] [—a] [name...]

DESCRIPTION _du_ gives the number of blocks contained in all files and (recursively) directories within each specified directory or file name. If name is missing, "." is used.

The optional argument —_s_ causes only the grand total to be given. The optional argument —_a_ causes an entry to be generated for each file. Absence of either causes an entry to be generated for each directory only.

A file which has two links to it is only counted once.

FILES /

SEE ALSO

DIAGNOSTICS

BUGS Files at the top level (not under —a option) are not listed.

Removable file systems do not work correctly since i—numbers may be repeated while the corresponding files are distinct. Du should maintain an i—number list per root directory encountered.

OWNER dmr

NAME ed -- editor

SYNOPSIS ed [name]

DESCRIPTION

ed is the standard text editor. ed is based on QED [reference] but is fully if succinctly described here. Differences between ed and QED are also noted to simplify the transition to the less powerful editor.

If the optional argument is given, simulates an e command on the named file; that is to say, the file is read into ed's buffer so that it can be edited.

ed operates on a copy of any file it is editing; changes made in the copy have no effect on the file until an explicit write (w) command is given. The copy of the text being edited resides in a temporary file called the buffer. There is only one buffer.

Commands to ed have a simple and regular structure: zero or more addresses followed by a single character command, possibly followed by parameters to the command. These addresses specify one or more lines in the buffer. Every command which requires addresses has default addresses, so that the addresses can often be omitted.

In general only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While ed is accepting text, it is said to be in input mode. In this mode, no commands are recognized; all input is merely collected. Input mode is left by typing a period (.) alone at the beginning of a line.

ed supports a limited form of regular expression notation. A regular expression is an expression which specifies a set of strings of characters. A member of this set of strings is said to be matched by the regular expression. The regular expressions allowed by are constructed as follows:

1 . An ordinary character (not one of those discussed below) is a regular expression and matches that character.

2. A circumflex (^) at the beginning of a regular expression matches the null character at the beginning of a line.

3. A currency symbol ($) at the end of a regular expression matches the null character at the end of a line.

4. A period (.) matches any character but a new–line character.

5. A regular expression followed by an asterisk (*) matches any number of adjacent occurrences (including zero) of the regular expression it follows.

6. A string of characters enclosed in square brackets ([]) matches any character in the string but no others. If, however, the first character of the string is a circumflex (^) the regular expression matches any character but new–line and the characters in the string.

7. The concatenation of regular expressions is a regular expression which matches the concatenation of the strings matched by the components of the regular expression.

8. The null regular expression standing alone is equivalent to the last regular expression encountered.

Regular expressions are used in addresses to specify lines and in one command (s, see below) to specify a portion of a line which is to be replaced.

If it is desired to use one of the regular expression metacharacters as an ordinary character, that character may be preceded by "\". This also applies to the character bounding the regular expression (often "/") and to \ itself.

Addresses are constructed as follows. To understand addressing in ed it is necessary to know that at any time there is a current line. Generally speaking, the current line is the last line affected by a command; however, the exact effect on the current line by each command is discussed under the description of the command.

1. The character "." addresses the current line.
2. The character "$" addresses the last line of the buffer.
3. A decimal number n addresses the nth line of the buffer.

4. A regular expression enclosed in slashes "/" addresses the first line found by searching toward the end of the buffer and stopping at the first line containing a string matching the regular expression. If necessary the search wraps around to the beginning of the buffer.

5. A regular expression enclosed in queries "?" addresses the first line found by searching toward the beginning of the buffer and stopping at the first line found containing a string matching the regular expression. If necessary the search wraps around to the end of the buffer.

6. An address followed by a plus sign "+" or a minus sign "–" followed by a decimal number specifies that address plus (resp. minus) the indicated number of lines. The plus sign may be omitted.

Commands may require zero, one, or two addresses. Commands which require no addresses regard the presence of an address as an error. Commands which require the presence of one address all assume a default address (often ".") but if given more than one address ignore any extras and use the last given. Commands which require two addresses have defaults in the case of zero or one address but use the last two if more than two are given.

Addresses are separated from each other typically by a comma (,). They may also be separated by a semicolon ";". In this case the current line is set to the the previous address before the next address is interpreted. This feature is used to control the starting line for forward and backward searches ("/", "?").

In the following list of ed commands, the default addresses are shown in parentheses. The parentheses are not part of the address, but are used to show that the given addresses are the default.

As mentioned, it is generally illegal for more than one command to appear on a line. However, any command may be suffixed by "p" (for "print). In that case, the current line is printed after the command is complete.

In any two–address command, it is illegal for the first address to lie after the second address.

```
(.)a
<text>
.
```

The append command reads the given text and appends it after the addressed line "." is left on the last line input, if there were any, otherwise at the addressed line. Address "0" is legal for this command; text is placed at the beginning of the buffer. (NOTE: the default address differs from that of QED.)

```
(.,.c)
(text>
.
```

The change command deletes the addressed lines, then accepts input text which replaces these lines. "." is left at the last line input; if there were none, it is left at the first line not changed.

(.,.)d

The delete command deletes the addressed lines from the buffer. "." is left at the first line not deleted.

e filename

The edit command causes the entire contents of the buffer to be deleted. and then the named file to be read in. "." is set to the last line of the buffer. The number of characters read is typed.

(1 ,s)g/regular expression/command

In the global command, the first step is to mark every line which matches the given regular expression. Then for every such line, the given command is executed with "." set to that line. The repeated command cannot be a, g, i, or c.

```
(.)i
<text>
.
```

This command inserts given text before the addressed line. "." is left at the last line input; if there were none, at the addressed line. This command differs from the <u>a</u> command only in the placement of the text.

(.,.)l

The list command prints the addressed lines in an unambiguous way. Non—printing characters are over—struck as follows:

```
char prints
bs   [overstruct - \]
tab   [overstruct - >]
ret   [overstruct - <]
SI    [overstruct I -]
SO    [overstruct O -]
```

All characters preceded by a prefix (ESC) character are printed over—struck with without the prefix. Long lines are folded with the sequence \newline.

(.,.p)

The print command prints the addressed lines. . is left at the last line printed

q

The quit command causes ed to exit. No automatic write of a file is done.

($)r filename

The read command reads in the given file after the addressed line. If no file name is given, the file last mentioned in e, r, or w commands is read. Address "0" is legal for r and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed."." is left at the last line of the file.

(.,.)s/regular expression/replacement/

The substitute command searches each addressed line for an occurence of the specified regular expression. On each line in which a match is found, the first (and only first, compare QED) matched string is replaced by the replacement specified. It is an error for the substitution to fail on all addressed lines. Any character other than space or new—line may be used instead of "/" to delimit the regular expression and the replacement. "." is left at the last line substituted.

The ampersand "&" appearing in the replacement is replaced by the regular expression that was matched. The special meaning of "&" in this context may be suppressed by preceding it by "\".

(1,$)w filename

The write command writes the addressed lines onto the given file. If no file name is given, the file last named in e, r, or w commands is written. "." is unchanged. If the command is successful, the number of characters written is typed. The line number of the addressed line is typed. . is unchanged by this command.

($)=
: The line number of the addressed line is typed. "." is unchanged by this command.

!UNIX command
: The remainder of the line after the "!" is sent to UNIX to be interpreted as a command. "."is unchanged.

(newline>
: A blank line alone is equivalent to ".+1p"; it is useful for stepping through text.

_Ed_ can edit at most 1500 lines and the maximum size of a line is 256 characters, The differences between _ed_ and QED are:

1. There is no \f character; input mode is left by typing . alone on a line.

2. There is only one buffer and hence no stream directive.

3. The commands are limited to:

   a c d e g i l p q r s w = !

   where _e_ is new.

4. The only special characters in regular expressions are:

   * ^ $ [.

   which have the usual meanings. However, "^" and "$" are only effective if they are the first or last character respectively of the regular expression. Otherwise suppression of special meaning is done by preceding the character by "\ , which is not otherwise special.

5. In the substitute command, only the leftmost occurrence of the matched regular expression is substituted.

7. The _a_ command has a different default address.

FILES
: /tmp/etma, etmb, ... temporary
/etc/msh is used to implement the "!" command.

SEE ALSO

DIAGNOSTICS ? for any error

BUGS ed is used as the shell for the editing system. It has the editing system UID built in and if invoked under this UID will give slightly different responses. This is a little kludgy.

OWNER ken

NAME find -- find file with given name

SYNOPSIS find name or number

DESCRIPTION find searches the entire file system hierarchy and gives the path names of all files with the specified names or (decimal) i—numbers.

FILES

SEE ALSO

DIAGNOSTICS

BUGS

OWNER dmr

NAME for -- fortran

SYNOPSIS <u>for</u> file

DESCRIPTION <u>for</u> is a nearly complete fortran compiler. file is the name of a fortran source program to be compiled. The following is a list of differences between <u>for</u> and ANSI standard fortran:

1. arbitrary combination of types are allowed in expressions. Not all combinations are expected to be supported in runtime. All of the normal conversions involving integer, real and double precision are allowed.

FILES

f.tmpl, 2 3	temporary
/etc/f1, 2 3 4	passes
/etc/xx	runtime

SEE ALSO

DIAGNOSTICS Diagnostics are given by number. If the source code is available, it is printed with an underline at the current character pointer. A listing of error numbers is available.

BUGS The following is a list of those features not yet implemented: functions, arithmetic statement functions. data statements, complex constants, hollerith constants, continuation cards

OWNER dmr, ken

NAME form -- form letter generator

SYNOPSIS form proto arg1 ...

DESCRIPTION form generates a form letter from a prototype letter, an associative memory, arguments and in a special case, the current date.

If form is invoked with the argument x, the following files come into play:

x.f prototype input
x.r form letter output
x.am associative memory
form.am associative memory if x.am not found.

Basically, form is a copy process from the file x.f to the file x.r. If an element of the −form \n (where n is a digit from 1 to 9) is encountered, The nth argument is inserted in its place, and that argument is then rescanned. If \0 is encountered, the current date is inserted. If the desired argument has not been given, a message of the form "\n:" is typed. The response typed in then is used for that argument.

If an element of the form [name] is encountered, the name is looked up in the associative memory. If it is found, the contents of the memory under this name replaces the original element (again rescanned.) If the name is not found, a message of the form "name: " is typed. The response typed in is used for that element. If the associative memory is writable, the response is entered in the memory under the name. Thus the next search for that name will succeed without interaction.

In both of the above cases, the response is typed in by entering arbitrary text terminated by two new lines. Only the first of the two new lines is passed with the text. The process is instantly terminated if an end of file is encountered anywhere except in the associative memory.

FILES

x.f	input file
x.r	output file
x.am	associative memory
form.am	associative memory

SEE ALSO type

DIAGNOSTICS "settup error" when the appropriate files cannot be located or created.

BUGS "settup" is misspelled.

OWNER rhm, ken

NAME hup -- hang up typewriter

SYNOPSIS _hup_

DESCRIPTION _hup_ hangs up the phone on the typewriter which uses it.

FILES

SEE ALSO

DIAGNOSTICS

BUGS should not be used; sometimes causes the typewriter channel to be lost.

OWNER dmr, ken

NAME lbppt -- load binary paper tapes

SYNOPSIS _lbppt_ output [input]

DESCRIPTION _lbppt_ loads a paper tape in standard UNIX binary paper tape format. It is used to bring files to a UNIX installation. Currently there is a GECOS program to prepare a GECOS file in binary paper tape format.

If the input file is specified, the character stream from that input is expected to be in UNIX binary paper tape format. If it is not present, /dev/ppt is assumed. The input stream is interpreted, checksummed, and copied to the output file.

FILES /dev/ppt

SEE ALSO dbppt, bppt format

DIAGNOSTICS "checksum"; "usage:"; "read error".

BUGS

OWNER ken

NAME ld -- link editor

SYNOPSIS ld [—usaol] name1]

DESCRIPTION ld combines several object programs into one; resolves external references; and searches libraries. In the simplest case the names of several object programs are given, and ld combines them, producing an object module which can be either executed or become the input for a further ld run.

The argument routines are concatenated in the order specified. The entry point of the output is the beginning of the first routine.

If any argument is a library, it is searched, and only those routines defining an unresolved external reference are loaded. If any routine loaded from a library refers to an undefined symbol which does not become defined by the end of the library, the library is searched again. Thus the order of libraries primarily affects the efficiency of loading, not what routines get loaded.

ld understands several flag arguments which are written preceded by a "—"

—s "squash" the output, that is, remove the symbol table and relocation bits to save space (but impair the usefulness of the debugger). This information can also be removed by strip.

—u take the following argument as a symbol and enter it as undefined in the symbol table. This is useful for loading wholly from a library, since initially the symbol table is empty and an unresolved reference is needed to force the loading of the first routine.

—o set the origin of the load to the octal number which is given as the next argument. This option affects only the definition of relocatable external symbols. See DMR before using.

—l This option is an abbreviation for a library name. "-l" alone stands for "/etc/liba.a", which is the standard system library for assembly language programs. "-lx" stands for /etc/libx.a where x is any character. There are libraries for Fortran (x=″f″) and B (x=″b″).

—a means absolute" (load at origin absolute 0) but it doesn't work.

The output of ld is left on a.out. This file is executable only if no errors occurred during the load.

FILES

/etc/libx.a, for various x;
/etc/ltma, ltmb, ... (temporary)
a.out (output file)

SEE ALSO

as, strip, ar (maintains libraries)

DIAGNOSTICS

"can't create temp file"—— unwritable directory or someone else is using ld in the same directory.

"can't open temp file" -- maybe someone has deleted it out from under you.

"file not found" -- bad argument

"bad format" -- bad argument

"relocation error" -- bad argument (relocation bits corrupted)

"bad relocation" -- user error: a relocatable reference to an external symbol that turns out to be absolute.

"multiply defined" -- same symbol defined twice in same load

"un" -- stands for undefined symbol

"symbol not found" —— loader bug

BUGS

Option "-a" doesn't work at all; option "-o" doesn't work right.

OWNER

dmr

NAME ln -- make a link

SYNOPSIS ln name1[name2]

DESCRIPTION ln creates a link to an existing file name1. If name2 is given, the link has that name; otherwise it is placed in the current directory and its name is the last component of name1.

It is forbidden to link to a directory or to link across file systems.

FILES

SEE ALSO rm, to unlink

DIAGNOSTICS "?"

BUGS There is nothing particularly wrong with ln, but links don't work right with respect to the backup system: one copy is backed up for each link, and (more serious) in case of a file system reload both copies are restored and the information that a link was involved is lost.

OWNER ken, dmr

NAME ls -- list contents of directory

SYNOPSIS ls [-ltasd] name1 ...

DESCRIPTION ls lists the contents of one or more directories under control of several options:

1 list in long format, giving i—number, mode, owner, size in bytes, and time of last modification for each file. (see stat for format of the mode)

t sort by time modified (latest first) instead of by name, as is normal

a list all entries; usually those beginning with "." are suppressed

s give size in blocks for each entry

d if argument is a directory, list only its name, not its contents (mostly used with —l to get status on directory)

If no argument is given, "." is listed. If an argument is not a directory, its name is given.

FILES /etc/uids to get user ID's for ls —l

SEE ALSO stat

DIAGNOSTICS "name nonexistent"; "name unreadable"; "name unstatable".

BUGS In ls -l, when a user cannot be found in /etc/uids, the user number printed instead of a name is incorrect. It is correct in stat.

OWNER dmr, ken

NAME mail -- send mail to another user

SYNOPSIS mail [letter person ...]

DESCRIPTION mail without an argument searches for a file called mailbox, prints it if present, asks if it should be saved. If the answer is y , the mail is renamed mail, otherwise it is deleted. The answer to the above question may be supplied in the letter argument.

When followed by the names of a letter and one or more people, the letter is appended to each person's mailbox. Each letter is preceded by the sender's name and a postmark.

A person is either the name of an entry in the directory /usr, in which case the mail is sent to /usr/person/mailbox, or the path name of a directory, in which case mailbox in that directory is used.

When a user logs in he is informed of the presence of mail.

FILES /etc/uids to map the sender's numerical user ID to name; mail and mailbox in various directories.

SEE ALSO init

DIAGNOSTICS "Who are you?" if the user cannot be identified for some reason (a bug). "Cannot send to user" if mailbox cannot be opened.

BUGS

OWNER ken

NAME mesg -- permit or deny messages

SYNOPSIS _mesg_ [n][y]

DESCRIPTION _mesg_ _n_ forbids messages via write by revoking non—user write permission on the user's typewriter. _mesg_ y reinstates permission. _mesg_ with no argument reverses the current permission. In all cases the previous state is reported.

FILES /dev/ttyn

SEE ALSO write

DIAGNOSTICS "?" if the standard input file is not a typewriter

BUGS

OWNER dmr, ken

NAME mkdir -- make a directory

SYNOPSIS mkdir dirname

DESCRIPTION mkdir creates directory _dirname_. The standard entries "."and ".." are made automatically.

FILES

SEE ALSO mkdir to remove directories

DIAGNOSTICS "?"

BUGS No permissions are checked. The system's user ID, not that of the creator of the directory, becomes the owner of the directory.

OWNER ken, dmr

NAME mkfs -- make file system

SYNOPSIS /etc/mkfs t
/etc/mkfs r

DESCRIPTION mkfs initializes either a DECtape (argument “t”) or an RK03 disk pack (argument "r”) so that it contains an empty file system. mkfs or its equivalent must be used before a tape or pack can be mounted as a file system.

In both cases the super—block, i—list, and free list are initialized and a root directory containing entries for "." and ".." are created. For RK03’s the number of available blocks is 4872, for tapes 578.

This program is kept in /etc to avoid inadvertant use and consequent destruction of information.

DIAGNOSTICS "Arg count", "Unknown argument", "Open error".

SEE ALSO

DIAGNOSTICS "Arg count", "Unknown argument", "Open error".

BUGS

OWNER ken, dmr

NAME mount -- mount file system

SYNOPSIS mount special dir

DESCRIPTION *mount* announces to the system that a removable file system has been mounted on the device corresponding to special file *special*. Directory *dir* (which must exist already) becomes the name of the root of the newly mounted file system.

FILES

SEE ALSO umount

DIAGNOSTICS "?", if the special file is already in use, cannot be read, or if *dir* does not exist.

BUGS Should be usable only by the super—user.

OWNER ken, dmr

NAME mv -- move or rename a file

SYNOPSIS my name1 name2 ...

DESCRIPTION _mv_ changes the name of name by linking to it under the name name2 and then unlinking name1. Several pairs of arguments may be given. If the new name is a directory, the file is moved to that directory under its old name. Directories may only be moved within the same parent directory (just renamed).

FILES

SEE ALSO

DIAGNOSTICS
"?a? -- incorrect argument count
"d" -- attempt to move a directory
"s" -- moving file to itself
"l"-- link error; old file doesn't exist or can't write new directory
"u" -- can't unlink old name

BUGS If mv succeeds in removing the target file, but then in unable to link back to the old file, the result is ?l and the removal of the target file. This is common with demountable file systems and should be circumvented. Also in such cases, mv should copy if it can.

OWNER ken, dmr