csh-reference

COLLABORATORS

	TITLE : csh-reference	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		September 19, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

csh-reference

1.1 C-Shell documentation

```
                C-SHELL 5.20+ REFERENCE,  Copyright 1991-1993 by the authors.


                 Commands

                 Variables

                 Functions
```

1.2 abortline

```
   ABORTLINE


Usage : abortline
Example : echo a;abort;echo b
Results : a

Causes the rest of the line to be aborted. Intended for use in
conjunction with exception handling.
```

1.3 action

```
   ACTION


Usage : action [-a] actionname file [arguments]
Example : action extr csh515.lzh csh.doc

Sends an action to a file. See chapter XIV CLASSES

Options:
```

```
 -a (abort) returns 0 if failed and 1 if successful. Otherwise,
            normal error codes (10 or 11) are returned
```

1.4 addbuffers

```
  ADDBUFFERS


Usage : addbuffers drive
    addbuffers drive buffers [drive buffers ...]
Example : addbuffers df0: 24

Just like AmigaDOS addbuffers command, causes new buffers to be
allocated for disk I/O.  Each buffer costs 512 bytes of memory,
CHIP memory if a disk drive. Buffers may be negative (to remove
buffers from a drive).

To show actual amount of buffers use "addbuffers drive"
or "addbuffers drive 0".
```

1.5 addpart

```
  ADDPART (or TACKON)


Equivalent to TACKON.
```

1.6 alias

```
  ALIAS


Usage : alias [name [command string] ]
Example : alias vt "echo Starting VT100;run sys:tools/vt100"

Sets a name to be a string. You can alias a single name to a set
of commands if you enclose them in quotes as above. By simply
typing vt, the command line above would be executed.
Aliases may call each other, but direct recursion is prohibited,
so the following works: alias ls "ls -s"
To prevent alias replacement, enter: \ls

By typing "alias name", you will get the alias for that name, while
with "alias" you get a list of all alias.

ARGUMENT PASSING TO AN ALIAS:

Usage : alias name "%var[%var...] [ command_string ]"
    alias name "*var[%var...] [ command_string ]"
Example : alias xx "%q%w echo hi $q, you look $w
```

```
    xx Steve great today
Results : hi Steve, you look great today

The second form of the alias command allows passing of arguments
to any position within the command string via use of a variable
name. To pass arguments to the end of a command string this method
is actually not necessary. These variables are local, so they don't
destroy another variable with the same name.
If you specify multiple arguments, every argument will be assigned
one word, and the last argument will be assigned the rest of the
command line.

Using a '*' instead of the first '%' prevents wild card expansion:
  alias zoo "*a zoo $a
To expand the wild cards after you got them, use
  exec set a $a

IMPLICIT ALIASES:

Usage : {command;command}
    {%var command;command} arg arg
Example : {%tmp echo $tmp $tmp} hello    --> hello hello

Curly braces define temporary aliases. They can be redirected as
a whole, can have arguments and local variables. They count as
one argument, even if there are blanks inside (as with quotes), and
they may be nested. Complex alias definitions can use the braces
instead of quotes, although this will add some calling overhead.
The closing curly brace is optional if at the end of line.
Example:

alias assert {e "Are you sure? ";input -s sure
```

1.7 ascii

```
  ASCII


Usage : ascii [-ho]
    ascii [-ho] string

If called without arguments, ascii outputs a complete ASCII table.
Given a string, shows each character in ASCII.

Options:
 -h shows numbers in hexadecimal
 -o shows numbers in octal
```

1.8 assign

```
  ASSIGN
```

```
Usage : assign
    assign logical
    assign [-adlnp] logical1 physical1 [log2 phy2 ... ]

The first form shows all assigns.
The second form kills one assign.
The third form assigns logical1 to physical1 and so on.

Options:
 -a adds a new path to an existing assign
 -d creates a deferred (late-binding) assign
     (identical with old option -l)
 -p creates a path (non-binding) assign
     (identical with old option -n)

For definition of add/defer/path, refer to your AmigaDOS manual.
```

1.9 basename

```
  BASENAME


Usage : basename var path [path ...]
Example : basename x df0:c/Dir  # sets x to "Dir"

Sets var specified to basenames of paths.
```

1.10 cat

```
  CAT (or TYPE)


Usage : cat [-n] [file file....]
Example : cat foo.txt

Type the specified files onto the screen.  If no file is specified,
STDIN in used (note: ^\ is EOF).  CAT is meant to output text files
only.

Options:
  -n  output numbered lines.
```

1.11 cd

```
  CD


Usage : cd [path]
    cd -g device1 [device2 [device3 ...]]
```

```
Options:
 -g generate a list of all directories on the given devices.

Change your current working directory.  You may specify '..' to go
back one directory (this is a CD specific feature, and does not
work with normal path specifications).

In most cases, you won't have to use the CD command. Just type
the desired directory at the prompt (very handy in conjunction with
file name completion). Typing a ~ alone on a command line cd's
to previous current directory.

There are two situations left when you still need it:

Entering 'cd *tem' will cd to the first name matched.

The second form generates a list (an ASCII file) of all direc-
tories on the given devices. It will be stored in the file given
in $_qcd (default: 'csh:csh-qcd'). Note that this ASCII file will
not be merged but overwritten. Once you have generated this file,
you can cd to any directory on your harddisk(s) even if it's not
in the current directory.
If you have two directories of the same name and you use one of
them more, move the more important one to the beginning of the
qcd file. You might also sort the file.
It is legal to type just an abbreviation of the directory name
you want to cd to. No asterisk '*' necessary. If you end up in
the wrong directory, cd to the same directory again (best done
by Cursor-Up + RETURN). You will cycle through all directories
that matched the given abbreviation. The other possibility is to
specify the full name of the parent directory: cd devs/keym
You may also add devices and assigns, so if 'PageStream:' is one
line in the qcd-file, a cd to 'page' is successful.

CD without any arguments displays the path of the directory you
are currently in.
```

1.12 chgrp

```
  CHGRP


Usage : chgrp group file1 ... filen
Example : chgrp 42 myfile

Set group-id (0-65535) or group-name of specified files.
(currently, name-to-id mapping is not implemented)
```

1.13 chmod

```
  CHMOD


Usage : chmod [u|g|o|a][+|-|=][flags] file1 ... filen
Example : chmod u+rwe myfile

Set AmigaDOS file protection flags for the file specified.
Valid flags are h, s, p, a, r, w, e, d.  (x is the same as e)

Ownership:
  u  Set specified bits for User (aka Owner)
  g  Set specified bits for Group
  o  Set specified bits for Other (not User, not Group)
  a  all, alias for "ugo" (User/Group/Other)
Specifying no ownership is equal to 'u'.

Modes:
  +  Set specified bits, leave all others
  -  Clear specified bits, leave all others
  =  Set specified bits, clear all others
Specifying no mode is equal to '='.

Archive bit cleared by default!

Note: This command is equivalent to "protect" except that the
      arguments for filename(s) and flag(s) are reversed.
```

1.14 chown

```
  CHOWN


Usage : chown owner file1 ... filen
Example : chown 42 myfile

Set owner-id (0-65535) or owner-name of specified files.
(currently, name-to-id mapping is not implemented)
```

1.15 class

```
  CLASS


Usage : [-n] name {type=param} [ "actions" {action=command} ]
Example : class zoo offs=20,dca7c4fd ext=.zoo actions view="zoo l"

Defines a new class of files and the actions to be taken on them
in various cases, or shows old definitions if given no arguments.
See section XIV: OBJECTS

Options:
```

```
  -n (new)  forgets old definitions
```

1.16 close

```
  CLOSE


Usage : close [filenumber]

Close the specified file opened by open. Without filenumber, closes
all open files. See open and flist for more info.
```

1.17 cls

```
  CLS


Usage : cls

This is an alias. It only clears the screen, but also works on a
terminal (echo ^L doesn't).
```

1.18 copy

```
  COPY (or CP)


Usage : copy [-udfpmo]  file file
or  : copy [-udfpmo]  file1 file2...fileN dir
or  : copy [-rudfpo] dir1...dirN file1...fileN dir

Options :
 -r recursive, copy all subdirectories as well.
 -u update, if newer version exists on dest, don't copy
 -f freshen, if file doesn't exist on dest or newer, don't copy
 -q suppresses 'not newer' and 'not there' messages in -u and -f
 -d don't set destination file date to that of source.
 -p don't set destination protection bits to those of source.
 -m erases the original. does not work with -r
 -o overwrites write/delete-protected, reads read-protected
 -a don't clear archive bit

Example : copy -r df0: df1:

Copy files or directories. When copying directories, the -r option
must be specified to copy subdirectories as well.  Otherwise, only
top level files in the source directory are copied.

All files will be displayed as they are copied and directory's
```

```
displayed as they are created. This output can be suppressed by
redirecting to nil: eg. copy -r >nil: df0: df1:

Copy will abort after current file on Control-C.

Copy by default sets the date of the destination file to that of
the source file. To overide this feature use the -d switch.

Similarly, it sets protection bits (flags) to those of source and
any file comment will be copied. To avoid this use -p. The archive
bit is always cleared by default (use option -a to leave it
untouched).

Another useful option is the -u (update) mode where copy will not
copy any files which exists already in the destination directory
if the destination file is newer or equal to the source file.
This is useful when developing code say in ram: eg. 'copy *.c ram:'
when done you can copy -u ram: df1: and only those modules you have
modified will be copied back.

Copy command will now create the destination directory if it does
not exist when specified as 'copy [-r] dir dir'. If you specify
copy file file file dir, then 'dir' must already exist.
```

1.19 cp

```
  CP (or COPY)


Equivalent to COPY.
```

1.20 date

```
  DATE


Usage : date [-bsr] [new date and/or time]
Example : date Wednesday  # this refers to NEXT wed, of course

Options:
  -b print date/time from the battery clock, if existent
  -s stores the current time internally
  -r shows time relative to last stored in secs and hundredths

Used to read or set system date and/or time. All standard options
may be used (yesterday, tomorrow, monday, etc.).
Leading zero's are not necessary.
Without parameters shows Dddddd DD-MMM-YY HH:MM:SS.
```

1.21 dec

```
  DEC


Usage : dec varname [value]
Example : dec abc

Decrement the numerical equivalent of the variable with specified
value (default: 1) and place the ASCII-string result back into
that variable.
```

1.22 delete

```
  DELETE (or RM)


Usage : delete [-fpqrv] file file file...
Example : delete foo.txt test.c

Remove (delete) the specified files.  Remove always returns
errorcode 0.  You can remove empty directories.

Options:
 -r recursively remove non-empty directories.
 -p (or f); remove delete-protected files.
 -v toggle verbose output.  Useful if 'delete' is aliased.
 -q (quit), delete aborts if the file to be removed didn't exist
     or couldn't be deleted. This does _not_ affect non-matching
     wildcards.

If you specify any wildcard deletes the files will be listed as
they are deleted. This can be suppressed by redirecting to nil:
```

1.23 dir

```
  DIR (or LS)


Usage : dir [-abcdfhiklnoqstuv] [-z [lformat]] [path path ... ]
Example : dir -ts downloads:
    dir -lz "%7s %-.16n %m" *.c
Options:
 -d list directories only
 -f list files only
 -h list only files which not start with a dot, end with '.info' or
    have the h-flag set. Adds an 'i' bit to the flags which tells
    if an according .info file exists.
 -s short multi(4) column display.
 -c don't change colors for directories
 -q quiet display. does not show length in blocks
 -o display file nOtes
 -n display names only
```

```
 -p display full path names and suppress directory titles
 -a shows the age of all files in the format  days hours:minutes
 -i identifies every file, shows the type instead of the date.
    See chapter XIV CLASSES
 -v (viewdir) recursively sums up lengths of the files in a dir
 -l sorts the files by their length, longest first.
 -t sorts the files by their time, most recent first.
 -k sorts the files by their class (klass)
 -b sorts the files backwards.
 -g prints directories at the beGinning
 -e prints directories at the End
 -u must be given exactly two directories. Shows files only in
    the first directory, files in both and files in the second.
 -z custom format
    (must be followed by an argument which holds the format string)

Displays a directory of specified files. Default output shows
date, protection, block size, byte size and total space used.
Protections flags include new 1.2/1.3 flags (see under protect),
plus a 'c' flag which indicates that this file has a comment.
Files are alphabetically sorted, without case sensitivity, and
directories are in red pen (unless you use -c). Dir takes in
account the width of your window.

To recursively show the contents of a directory and all its
sub-directories use the special wildcard pattern ".../*", see
section "WILDCARDS" (man wildcards).

The lformat string (option -z) is used to create your own directory
format. Instead of the "-z lformat" command line argument you can
set the variable "_dirformat" (which holds "lformat", but you must
set option -z anyhow).
Your custom format may contain the following codes:
 %a age               %l LF if comment      %t time
 %b size in blocks    %m multi column       %u size in K
 %c flag c (comment)  %n name               %v dir size in eng.
 %d date              %o filenote (comment) %w dir size in K
 %e flag i (.info)    %p name w/ path       %x translated date
 %f flags "hsparwed"  %q name w/ slash      %+ flag i as '+' or ' '
 %i flag d (dIr)      %r size in eng.
 %k class             %s size

 %I link information (S: softlink, H: hardlink, P: pipe, -: else)
 %L name of original file if link (empty otherwise)
 %N name + original name (of link)
 %F protection bits (flags "rwed") for group/other
 %U user-id
 %G group-id

Between the '%' and the identifying letter, there may be an optional
field width. If the number is preceded by a '-', the field contents
will be left adjusted. If by a dot, the contents will be cut down
to match the field width if they are longer.
If the format string contains a %m, cshell will try to print more
than one entry on one line. The column width is the field width of
the %m entry. If omitted, it's assumed to be the one of the first
file. If a file is longer, it will use two columns.
```

```
If you prefer the old output-style of this command (5.19 and before)
add the following line to your .cshrc file:
 set _dirformat "   %-24n %c%f %7s %4b %d %t"
```

1.24 diskchange

```
  DISKCHANGE


Usage : diskchange drive...drive

Like AmigaDOS diskchange.  Multiple drive names are allowed.
```

1.25 echo

```
  ECHO


Usage : echo [-en] string
Example : echo hi there
Results : hi there

Options:
 -n don't append newline.
 -e echo to stderr.

Echo the given string.
```

1.26 else

```
  ELSE


Usage : else ; command
Usage : if -f foo.c ; else ; echo "Not there" ; endif

Else clause, must follow an IF statement.
```

1.27 endif

```
  ENDIF


Usage : endif

The end of an if statement.
```

```
Note: if you return from a script file with unterminated IF's
and the last IF was false, prompt will be changed to an
underscore ('_') and no commands will be executed until
'endif' is typed.
```

1.28 error

```
  ERROR


Usage : error n

Generates return code n.
```

1.29 exec

```
  EXEC


Usage : exec [-i] command [args]
Example : set cmdline "dir ram:"
    exec $cmdline   # would not work without exec

Options:
 -i  return code 0.

Execute the command specified; exec command is equivalent to
command, only you can use variables to specify command name.
Note that the command line is parsed TWICE! Examples:
 set a dir ram:; exec $a              # right
 set a mkdir; exec $a "My directory"  # wrong! creates 2 directories

Exec returns the return code of the command executed unless
option -i (ignore) is set, in which case always 0 is returned.
```

1.30 fault

```
  FAULT


Usage : fault error1 .. errorN
Example : fault 205 212

Like AmigaDOS fault, prints specified error messages.
```

1.31 filenote

```
  FILENOTE


Usage : filenote file1 .. fileN  note
    filenote -s file1...fileN

Options:
 -s (second form) ; displays the file notes of the given files.

The first form sets AmigaDOS comment of the specified file.
```

1.32 flist

```
  FLIST


Usage : flist

Lists the filenumbers of files opened by open.
See open and close for more info.
```

1.33 fltlower

```
  FLTLOWER


Usage : fltlower
Example : dir | fltlower
Or  : fltlower <readme

This is a filter command, i.e. it reads from stdin and writes to
stdout. The more natural way to use it is a pipe, or it can be
redirected.
Its purpose is to convert all alphabetic to lower case.
```

1.34 fltupper

```
  FLTUPPER


The same of fltlower, only this converts to upper case.
```

1.35 foreach

```
  FOREACH


Usage : foreach [-v] varname ( strings ) command
Example : foreach i ( a b c d ) "echo -n $i;echo \" ha\""
Result  : a ha
    b ha
    c ha
    d ha

Options:
 -v display arguments every time command is executed.

'strings' is broken up into arguments.  Each argument is placed in
the local variable 'varname' in turn and 'command' executed. Put
the command(s) in quotes.

Foreach is especially useful when interpreting passed arguments in
an alias.

eg.
  foreach i ( *.pic ) viewilbm $i
assuming a.pic and b.pic in current directory the following commands
will occur:
  viewilbm a.pic
  viewilbm b.pic

All 'for...' commands can be interrupted using CTRL-D or CTRL-E.
```

1.36 forever

```
  FOREVER


Usage : forever command
or  : forever "command;command;command..."

The specified commands are executed over and over again forever.

Execution stops if you hit ^C or ^D, or if the commands return with
an error code.
```

1.37 forline

```
  FORLINE


Usage : forline var filename command
or  : forline var filename "command;command..."
Example : forline i RAM:temp "echo line $_linenum=$i"
```

```
For each ASCII line of file specified commands are executed and
var points to line content. You can check system variable _linenum
to find the number of the line currently read.
If STDIN (case sensitive) is specified as input file, the lines are
read from standard input.
```

1.38 fornum

```
  FORNUM


Usage : fornum [-v] var n1 n2 command
or  : fornum [-v] -s var n1 n2 step command

Example : fornum -v x 1 10 echo $x
or  : fornum -s x 10 1 -1 echo $x # counts backwards

Executes  command(s)  for  all numerical values of x between n1 and
n2.   If  more  than  one  command  is  specified,  or  command  is
redirected, include command(s) in quotes.


Options:
 -v (verbose) causes printing of progressive numbers.
 -s specify a step; if negative, the count will be backwards.
```

1.39 getenv

```
  GETENV


Usage : getenv [shellvar] envvar

Gets the value of an ENV: variable and stores it in the shell
variable 'shellvar'. If shellvar is omitted, the value of the
ENV: variable is printed to stdout.
This command is obsolete since ENV: variables can be retrieved
by writing $envvar anywhere on the command line.
```

1.40 goto

```
  GOTO


Usage : goto label
Example :
    label start
    echo "At start"
    dir ram:
    goto start
```

```
Goto  the specified label name.  You can only use this command from
a  source  file.  Labels  may  be  forward or reverse from current
position. It is legal to jump out of if's.
```

1.41 head

```
  HEAD


Usage : head [filename] [num]
Example : head readme 20

Display first "num" lines of "filename".  If num is not specified,
10 is assumed. If filename is not specified, standard input (stdin)
is taken instead.
```

1.42 help

```
  HELP


Usage : help [-f]
Example : help
Options :
 -f list functions also

Simply  displays  all  the  available  commands.  The commands are
displayed in search-order.  That is, if you give a partial name the
first  command  that  matches  that  name  in  this list is the one
executed.  Generally,  you  should  specify enough of a command so
that it is completely unique.
```

1.43 history

```
  HISTORY


Usage : history [-nr] [partial_string]
Example : history

Options :
 -n omits line numbering
 -r reads history from stdin

Displays  the  enumerated  history  list.  The size of the list is
controlled  by  the  _history  variable.  If you specify a partial
string, only those entries matching that string are displayed.
```

1.44 howmany

```
  HOWMANY


Usage : howmany

This  command  tells you how many instances of Shell are running in
your system.
```

1.45 htype

```
  HTYPE


Usage : htype [-r] [file1..fileN]

Options:
 -r display all files in a directory.

Displays the specified files in hex and ASCII, just like the system
command 'Type file opt h'.  Especially suitable for binary files.

If there are no filenames specified, standard input is used,
so you can use htype as the destination for a pipe.
```

1.46 if

```
  IF


Usage : if [-n] argument conditional argument [then]
or  : if [-n] argument
or  : if [-n] -f file    or   -e file
or  : if [-n] -d file/dir
or  : if [-n] -m
or  : if [-n] -t file file1 .. fileN
or  : if [-n] -r rpnexpression
or  : if [-n] -v varname
or  : if [-n] -o char arg ... arg

Options:
 -n (NOT) reverses the result.
 -d tests the type of the object specified: if it is a
    directory, then TRUE; if it is a file (or it doesn't exist)
    then FALSE.
 -f (or -e) checks for existance of the specified file.
 -m test if FAST memory is present.
 -o tests for option 'char' in the rest of the arguments.
 -r evaluates a given RPN expression (see under RPN for more info).
    If value on top of stack is 0, then FALSE, else TRUE.
 -t compare the date and time of the first file with all the others;
```

```
    if the first is younger than ALL the others, then FALSE,
    else TRUE. If a file doesn't exists, it is considered as
    being older.
 -v test if a given variable is defined.


Makes the following instructions up to the next endif conditinal.
The 'then' is optional. The if clause must be followed by a semi-
colon if instructions follow on the same line.

If a single argument is something to another argument.  Conditional
clauses allowed:

<, >, =, ! and combinations.  Thus != is not-equal, >= larger or
equal, etc...

If arguments are not numeric, they are compared as strings.

Usually the argument is either a constant or a variable ($varname).

The second form of IF is conditional on the existence of the
argument.  If the argument is a "" string, then FALSE, else TRUE.

The third form of IF used by -f switch checks for existance of
the specified file. -e is the same as -f

Option -m is used to test if FAST memory is present.
Example (to be included in a login.sh file):
if -m; resident -d lc1 lc2 blink; endif

Using -t form compares the date and time of the first file with
all the others; if the first is younger than ALL the others, then
FALSE, else TRUE. If a file doesn't exists, it is considered as
being older.
This feature is especially useful for building makefiles without
using any MAKE utility.

Example:
if -t test.o test.asm test.i ; asm -o test.o test.asm ; endif


Option -o tests for option 'char' in the rest of the arguments.

Example: if -o r -rp ram:comm1.c     will yield TRUE.

When using 'IF' command interactively if you are entering commands
following an 'IF' that was false, the prompt will be set to a
underscore '_ ' to indicate all commands will be ignored until
an 'ELSE' or 'ENDIF' command is seen.
```

1.47 inc

```
  INC
```

```
Usage : inc varname [value]
Example : inc abc 5

Increment the numerical equivalent of the variable with specified
value (default: 1) and place the ASCII-string result back into
that variable.
```

1.48 info

```
  INFO


Usage : info [-pt] [path1 path2 ... pathN]

Options :
 -p  only display drives with readable (present) disks
 -t  print disk/fs type and bytes used instead of block sizes

If called without arguments, info gives you the drive information
on all devices. If one or more paths are specified, only infor-
mation on those drives will be displayed.

Note: Cshell does (correct) rounding for all displayed values,
      Commodore's Info command does not. So values may slightly
      change.
```

1.49 input

```
  INPUT


Usage : input [-sr] var var ... var
Example : input abc

Options :
 -s the whole line is read in as one word, including spaces.
 -r puts the console to single character mode before reading,
    ie. does not wait for RETURN to be pressed). Use with care.

Input from STDIN (or a redirection, or a pipe) to a variable.  The
next input line is broken up in words (unless quoted) and placed in
the variable.
```

1.50 join

```
  JOIN


Usage : join [-r] file1..fileN destfile
Example : join part1 part2 part3 total
```

```
Options :
 -r overwrite any existent destfile.

Joins (concatenates) the specified files to get destfile. Join
will refuse to overwrite an existing destfile, unless the
'r' option is used.
```

1.51 keymap

```
  KEYMAP


Usage : keymap [number {key=function}]
Example : keymap 0 1030=4 1032=12

Defines one keymap for the csh command line editing. See chapter XV.
```

1.52 label

```
  LABEL


Usage : label name

Create a program label right here. Used in source files, you can
then GOTO a label.
```

1.53 linecnt

```
  LINECNT


Another filter. Counts the number of lines of its stdin and writes
it to stdout.
```

1.54 ln

```
  LN (or MAKELINK)


Usage : ln [-s] filename [linkname]
Example : ln stuff/data newname

Options :
 -s make soft link (default is hard link)
```

ln creates an additional directory entry, called a link, to a file or directory. Any number of links can be assigned to a file.

filename is the name of the original file or directory. linkname is the new name to associate with the file or filename. If linkname is omitted, the last component of filename is used as the name of the link.

A hard link (the default) is a standard directory entry just like the one made when the file was created. Hard links can only be made to existing files. Hard links cannot be made across file systems (disk partitions, mounted file systems). To remove a file, all hard links to it must be removed, including the name by which it was first created; removing the last hard link releases the inode associated with the file.

A symbolic link, made with the -s option, is a special directory entry that points to another named file. Symbolic links can span file systems and point to directories. In fact, you can create a symbolic link that points to a file that is currently absent from the file system; removing the file that it points to does not affect or alter the symbolic link itself.

NOTE: Symbolic links (also known as "soft links") are
currently NOT SUPPORTED by AmigaOS. DO NOT USE!

1.55 local

LOCAL

Usage: local [var...var]

Creates one or more local variables. Those variables disappear
at the end of their alias or source file, and cannot be accessed
from inside other aliases or source files.
With no arguments, shows all top level variables and their values.

1.56 ls

LS (or DIR)

Equivalent to DIR.

1.57 makelink

```
  MAKELINK (or LN)


Equivalent to LN.
```

1.58 man

```
  MAN


Usage : man command(s)
Example : man mkdir

Get  info about a Shell command, or others keywords.  These include
all  special  _variables, plus various keywords:  WILDCARDS, PIPES,
EDITING, STARTUP and more.
See special alias manlist to get a list of ALL keywords supported
by man.
You  must set _man to the paths of your .doc files:
   set _man dh1:docs/aliases.doc dh1:docs/csh.doc

To create your own .doc files, precede all your keywords by four
blanks. 'man' will then display lines until the first character
of a line is alphanumeric or has four leading blanks.
```

1.59 md

```
  MD (or MKDIR)


Equivalent to MKDIR.
```

1.60 mem

```
  MEM


Usage : mem [-cfqsl]

Options:
  -c shows the free chip mem only
  -f shows the free fast mem only
  -q outputs just a number without titles
  -s stores current free memory
  -r shows memory used relative to last stored
  -l flushes all unneeded memory
```

1.61 menu

```
  MENU


Usage : menu [-mn] [ title item...item ]
Example : menu Shell   JrComm,,j   Rename,"rename ",r  quit

Options:
 -n clear all existing menus.
 -m use monospaced font.

Appends one pull down in the current console window. Up to 31
menus with 63 items each (including title) can be installed.

If the item is just a string, that string will be in the menu item.
When you select it, it will be put into the prompt and executed.

If there is a comma and after that comma a second string, this
will be the command will be inserted at the prompt. This time you
have to add the ^M yourself if you want the command to be executed.

If there is a second comma, the letter after that comma will be
the keyboard shortcut for that menu item. (This will be case
sensitive some day, use lowercase).

If for any reason your current menu is corrupt, just enter an
empty 'menu' command.

When the first menu is installed you can use option -m to choose
a monospaced font (System Default Font) instead of the default
Intuition Font (which may be a proportional font). This is useful
for user-formatted menus (like in the example script "menu.sh").
```

1.62 mkdir

```
  MKDIR (or MD)


Usage : mkdir [-p] name name name...
Example : mkdir df0:stuff

Options:
 -p create all dirs in path if necessary.

Create the specified directories.
If "name" ends with trailing slash it will be stripped off.

mkdir now supports the -p option.  mkdir -p followed by a full path
name will create all directories necessary to make the path.  For
example, suppose that the directory ram:foo exists and is empty.
"mkdir -p ram:foo/bar/tst/a" would create ram:foo/bar,
ram:foo/bar/tst, and ram:foo/bar/tst/a all in one step.
In addition, it will issue no error codes for directories it cannot
```

make.

1.63 mv

MV (or RENAME)

Equivalent to RENAME.

1.64 open

OPEN

```
Usage : open filename filemode filenumber
Example : open RAM:data w 1

This  allows you to open a file, redirect to it as many commands as
you like, then close it.
Filename is any valid AmigaDOS filename, filemode is either "r" for
read or "w" for write, filenumber is a number between 1 and 10.
To  redirect  a  program to or from an open file, use as your redir
filename a dot followed by the filenumber.
Here is a complete example:

  open RAM:data w 1
  echo -n 2+2= >.1
  rpn 2 2 + . CR >.1
  close 1
  type RAM:data # will display 2+2=4

See also close, flist.
```

1.65 path

PATH

```
Usage : path [-gr] [dir...dir]

Without arguments, lists AmigaDOS path. Otherwise adds given
directories to the path, preventing duplicate entries.

Options:
 -r Resets the path
 -g Global path modifications; operations (add, reset) apply to all
     CLI processes instead of only the current one

Note:
It's not perfectly "legal" to modify the path-list of other
```

```
processes. Adding entries (option -g) works fine in most cases.
But the removal of entries (options -gr together) may crash the
system, because CSH doesn't know about the memory handling of
other processes (it doesn't know how they allocated the memory
for the path-list entries).

So use option -g always with care (at least together with -r).
If it works, it's okay. If not, you lose ;-)  There's no 100%
reliable way for global path modifications.
```

1.66 pri

```
  PRI


Usage : pri clinumber pri
Example : pri 3 5 # set priority of cli #3 to 5

Change the priority of the specified task (use PS command to
determine clinumber). If you specify 0 as clinumber you can
change priority of "this" task (the one executing shell).
```

1.67 protect

```
  PROTECT


Usage : protect file1 ... filen [u|g|o|a][+|-|=][flags]
Example : protect myfile u+rwe

Set AmigaDOS file protection flags for the file specified.
Valid flags are h, s, p, a, r, w, e, d.  (x is the same as e)

Ownership:
  u  Set specified bits for User
  g  Set specified bits for Group
  o  Set specified bits for Other (not User, not Group)
  a  all, alias for "ugo" (User/Group/Other)
Specifying no ownership is equal to 'u'.

Modes:
  +  Set specified bits, leave all others
  -  Clear specified bits, leave all others
  =  Set specified bits, clear all others
Specifying no mode is equal to '='.

Archive bit cleared by default!

Note: This command is equivalent to "chmod" except that the
      arguments for filename(s) and flag(s) are reversed.
```

1.68 ps

```
   PS


 Usage : ps [-les] [commandname...commandname]

 Options:
  -l shows full pathnames of commands
  -e excludes the given command names from the list
  -s don't show stacksize and type, use old output-format instead

 Gives status of CLI processes.  eg:

 Proc Command Name         Typ  Stack  Pri.  Address  Directory
 * 1  csh                  fr   10000    0      97b0  Stuff:shell
   2  clock                bw    4096  -10    2101a8  Workdisk:
   3  emacs                bw   30000    0    212f58  Stuff:shell
   4  VT100                bw    4000    0    227328  Workdisk:

 Address is the address of the task, directory is the process
 currently CD'd directory. My default, only the BaseNames of
 the commands are shown. Your own CLI will be marked by an
 asterisk in the first column.

 Stack size is the _real_ size of a command's stack. It's _not_
 the size a program gets if it's launched by this command.  Use
 "Status" instead if you need the size of the Default Stack.
 (be aware: "Status" does _not_ show the stack size used by a prog!)

 Typ are two letters.  The first is either "f" (foreground) or
 "b" (background).  The second is one of:
   i: invalid
   a: added
   r: running / ready to run
   w: waiting
   e: except
   d: removed
```

1.69 pwd

```
   PWD


 Usage : pwd

 Rebuild _cwd by backtracing from your current directory.
```

1.70 qsort

```
   QSORT
```

```
Usage : qsort [-cr] <in >out

Options :
 -c case-sensitive
 -r reverse sort

Quick sorts from stdin to stdout (case-insensitive).
```

1.71 quit

```
  QUIT


Usage : quit

Quit out of Shell back to CLI.
```

1.72 rback

```
  RBACK


Usage : rback command

Start a new process executing the specified command, but can't do
input/output. Equivalent to 'run command >NIL: <NIL:'. Instead of
using rback, you can add a '&' at the end of the command line.

Note: rback cannot start builtin commands. You have to start a
      subshell: rback csh -c "copy ram:temp prt:;rm ram:temp
```

1.73 readfile

```
  READFILE


Usage : readfile varname [filename]

Completely reads an ASCII file and assigns it to a variable. Each
line becomes one word in the resulting string. Embedded blanks
are no problem. If file name is omitted, stdin is used. See also
'writefile', @subfile and @flines
```

1.74 rehash

```
  REHASH


Usage : rehash [-cglos]

Options :
 -c clear local program hash list
 -g clear global program hash list
 -l load global program hash list into local buffer
 -o output local program hash list
 -s save local program hash list to disk

Scans the complete DOS search path (see also 'path') and builds a
program hash list. This can be used for program name completion from
command line (default: ESC-p, ESC-P). And when running commands
CShell does not scan DOS search path any more for every command but
instead scans the program hash list in memory (minimizes disk access
and speeds up running commands significantly).

Of course, when you add a directory to your path, when you add
programs to the existing path or when you remove directories/programs
then you have to rebuild the program hash list.

Each invocation of CShell has its own local buffer to hold that
program hash list.  Use option -s to save the local list to disk
(csh:csh-prgs).  With option -l the list is loaded into memory.
The first CShell loading the list puts a copy of it into a global
buffer so that next time a CShell wants to load it the global buffer
is used and not the disk file.

The global list stays in memory -- even if you quit all CShells.
Use option -g to free that global list (if you are low on memory),
but this does not affect local lists of any currently running CShell.
Option -c clears the local list.

If you run this command without options the scanned list is not only
put into local buffer but also in global buffer.

There's no need to clear the local/global list before
loading/building a new one. This is done automatically.

The variable $_prghash (default: 'csh:csh-prgs') holds the filename
where the program hash list is loaded from and saved to.

Programs from the hash list are case in-sensitive and may be
abbreviated. This can be toggled with the variable "_abbrev".

Recommended usage:
First run "rehash" from your shell to build the program hash list.
Save this list to disk with "rehash -s".  Now include "rehash -l"
in "s:.cshrc" to load this list on every invocation of CShell.
```

1.75 relabel

```
  RELABEL


Usage : relabel drive name
Example : relabel DH0: Picard

Change the volume name of the disk in the given drive to the name
specified. Volume names are set initially when you format a disk.

If you have a floppy disk system with only one disk drive, be sure
to specify the disks by volume name, instead of drive name.
```

1.76 rename

```
  RENAME (or MV)


Usage : rename [-fv] from to
or  : rename [-fv] from from from ... from todir

Options :
 -f don't abort on errors
 -v verbose mode (print renamed filenames)

Allows you to rename a file or move it around within a disk.
Allows you to move 1 or more files into a single directory.
The archive bit of the file(s) will be cleared.
```

1.77 resident

```
  RESIDENT


Usage : resident [-dr] [files]
Example : resident lc1 lc2 blink  # load these as resident
    resident -d lc1 lc2 blink # defer load when needed
    resident -r lc1 lc2 blink # remove these
    resident      # list resident programs


Options :
 -d deferred load;
 -r remove files from resident list

This is DOS resident. Commands are searched by Shell in resident
list BEFORE of searching on any external device.
Only PURE programs can run as resident, see DOS docs for more info.
Option -d is very useful: you can say, in your startup file,
resident -d file...file; programs will not be loaded immediately,
but only when you will try to load them. This way, you will not
```

```
waste memory and startup time if you don't use the programs.
Old option -a has no more effect.
```

1.78 return

```
  RETURN


Usage : return [n]
Example : return 10

Exit from a script file, or quit from shell with optional
exit code.
```

1.79 rm

```
  RM (or DELETE)


Equivalent to DELETE.
```

1.80 rpn

```
  RPN


Usage : rpn expression
Example : rpn 3 7 * # Prints the value 21

Evaluate an RPN expression, using 32-bit values. In older versions
of Shell RPN contained string functions too, but now that strings
are handled by specifical commands, these are no more needed.
At end of evaluation, RPN prints values on stack, so you can
say for instance "rpn $x 2 * | input x" to double the value of
variable x.
Functions implemented are:

    + - * / Obvious meaning; / means integer division, of course
    %   Module operator e.g. "rpn 7 3 %" answers 1
    & | ~ Bitwise and, or, not operators
    > < ==  Tests for greater-than, lower-than, equal. To get
    a test for >= (or <=), you can use < ! (or > !)
    !   Logical not operator
    DUP   Duplicate value on top of stack
    DROP  Drop value on top of stack
    SWAP  Swap two values on top of stack

To avoid confusion with redirections, > and < operators must be
enclosed in quotes e.g.
```

```
3 2 ">"   # Prints 1
```

1.81 run

```
RUN


Usage : run prgm args
Example : run emacs test.c

Start a new process executing the specified command. This command is
not fully reliable: use at your own risk.  See also rback.
```

1.82 rxrec

```
RXREC


Usage : rxrec [portname]

Create an AREXX-compatible port of the specified name (defaults to
"rexx_csh"), then puts Shell to sleep waiting for messages on it.

CAUTION: the only way to exit from this status is to send to the
port the message "bye".

Example:
Open two Shell's in two separate CLI's. From the first, type:

  rxrec

Now first Shell doesn't respond to keyboard input; instead, it waits
for messages on a port called "rexx_csh". Now, from the other, type:

  rxsend rexx_csh "dir df0:"

You will see the listing of df0: in the first Shell.  Experiment as
you like, then:

  rxsend rexx_csh bye

And all will return to normal.
```

1.83 rxsend

```
RXSEND


Usage : rxsend [-lr] portname command...command
```

```
Options :
 -r set the variable _result to the result string of the
    AREXX command.
 -l send the whole line as *one* command.

Send commands to any program with an AREXX-compatible port. Be aware
that every word is sent as a single command!

You don't have to load anything to use these command (or rxrec):
all you need is a program with the right port.

An example is CygnusEdProfessional: here is, for instance, a command
to wake it up, load the file test.c and jump to line 20:

  rxsend rexx_ced cedtofront "open test.c" "jmp to line 20"
  # rexx_ced is the name of AREXX port for CygnusEd


Refer to your application manual for details and for the names
of the commands and the port.
```

1.84 search

```
  SEARCH


Usage : search [-abceflnoqrvw] file...file string

Search specified files for a string. Only lines containing the
specified strings are displayed.

If the filename is STDIN (in uppercase), the standard input is
used, so you can use search as the destination for a pipe.
Example:
  strings myprog * | search STDIN .library
Lists all libraries used in "myprog".

Search is very fast if none of the options -w, -e and STDIN was
specified and the file fits into memory.

Options:
  -a (abort)   stops search as soon as the pattern was found once
  -b (binary)  shows only byte offsets instead of lines. If combined
               with -n, shows naked numbers.
  -c (case)    turns ON case sensitivity
  -e (exclude) lists lines NOT containing the pattern
  -f (files)   causes only the names of the files in which the pat-
               tern was found to be displayed.
  -l (left)    pattern must be at beginning of line (this is faster
               than using a wild card)
  -n (number)  turns off line numbering
  -o (only)    finds only whole words
  -q (quiet)   suppresses printing of file names.
  -r (recurse) if you specify any directory, all files in that di-
               rectory are recursively searched.
```

```
  -v (verbose) shows each file name on a single line. this is auto-
               matically turned on if search is redirected
  -w (wild)    wild card matching. see notes below

Notes to wild card matching;
- Uses Shell standard matching.
- All standard DOS wildcards are allowed * ? [] () | ~ ' #
- The WHOLE line must match the string, not only a substring.
- String MUST be enclosed in quotes to avoid wildcard expansion

Examples:
  search -cr df0:include ACCESS
Find all occurrences of ACCESS (in uppercase) in all files
contained in include directory.
  search -w shell.h "'#define*"
Lists only lines of file beginning with (not simply containing)
#define. Note the use of ' to escape the special symbol #.
```

1.85 set

```
  SET


Usage : set [name] [=] [string]
Example : set abc hello

Set with no args lists all current variable settings.
Set with one arg lists the setting for that particular variable.
Specifying name and string, stores the string into variable name.

Also see the section on special _variables.
```

1.86 setenv

```
  SETENV


Usage : setenv envvar value

Sets an ENV: variable to the given value. The value must be
enclosed in quotes if it contains spaces. To retrieve an ENV:
variable, just use $envvar anywhere on a command line.
```

1.87 sleep

```
  SLEEP


Usage : sleep timeout
Example : sleep 10
```

```
Sleep for 'timeout' seconds, or until ^C typed.
```

1.88 source

```
  SOURCE


Usage : source file [arguments]
Example : source mymake.sh all
Result  : batch file 'mymake.sh' called with var _passed = 'all'

Execute commands from a file.  You can create SHELL programs in
a file and then execute them with this command.  Source'd files
have the added advantage that you can have loops in your command
files (see GOTO and LABEL).  You can pass SOURCE files arguments
by specifying arguments after the file name.  Arguments are passed
via the _passed variable (as a single string, a set of words).
See _failat variable for script aborting.

Long lines may be split by appending a backslash (\) at end of
first part. One single line must be shorter than 512 bytes, but
the concatenated line can be as long as you want. There is no
limit on the length of the concatenated line.

Automatic 'sourcing' is accomplished by appending a .sh suffix to
the file (no need to set the s-bit) and executing it as you would
a C program:

--------- file hello.sh ---------
foreach i ( $_passed ) "echo yo $i"
---------------------------------

$ hello a b c
yo a
yo b
yo c

If the last character of a line in a source file is '{', all
following lines will appended to the current one and separated
by semicolons until the last character of a line is '}'. Those
blocks may be nested. You may use comments and unterminated
strings within.

--------- file login.sh ---------
alias complex {
  echo -n "this alias
  echo " works!"
}
---------------------------------

$ login
$ complex
this   alias   works!
```

1.89 split

```
  SPLIT


Usage : split srcvar dstvar...dstvar

Assigns one word of srcvar to every dstvar, the rest of srcvar to
the last dstvar.
Note: You enter variable NAMES, not variables.
```

1.90 stack

```
  STACK


Usage : stack [number]
Example : stack [-s] 8000

Options :
 -s prints size only (pure number, no text).

Changes the default stack for this CLI.
Without arguments, just prints it.
```

1.91 strhead

```
  STRHEAD


Usage : strhead varname breakchar string
Example : strhead x . foobar.bas  # Will set x to "foobar"

Remove everything after and including the breakchar in 'string' and
place in variable 'varname'.
```

1.92 strings

```
  STRINGS


Usage : strings [-bnrv] [file1..fileN] [minlength]
Example : strings [-bnrv] c:dir c:list shell 7

Options :
 -r  if you specify any directory, all files in that directory
     are recursively searched for strings
 -n  print name of current file in front of each string
 -b  shows each string enclosed by  '|' characters, so as to expose
```

```
     leading and trailing spaces or tabs.
 -v  verbose output before each file (filename, minlength)

Prints strings contained in specified files (usually binary)
with length >= minlength.  Default is 4.

You cannot use a filename that represents a number as last
argument. If there are no filenames specified, standard input
is used, so you can use strings as the destination for a pipe.
```

1.93 strleft

```
  STRLEFT


Usage : strleft varname string n
Example : strleft x LongString 5  # Will set x to "LongS"

Place leftmost n chars of string in variable varname.
```

1.94 strlen

```
  STRLEN


Usage : strlen varname string
Example : strlen x Hello    # Will set x to "5"

Puts len of string in variable varname.
```

1.95 strmid

```
  STRMID


Usage : strmid varname string n1 [n2]
Example : strmid x LongString 5 3 # Will set x to "Str"

Places n2 chars from string, starting at n1, in variable varname.
By omitting n2, you get all chars from n1 to end of string.
```

1.96 strright

```
  STRRIGHT


Usage : strright varname string n
```

```
Example : strright x LongString 5 # Will set x to "tring"

Place rightmost n chars of string in variable varname.
```

1.97 strtail

```
  STRTAIL


Usage : strtail varname breakchar string
Example : strtail x . foobar.bas  # Will set x to "bas"

Remove everything before and including the breakchar in 'string' and
place in variable 'varname'.
```

1.98 tackon

```
  TACKON (or ADDPART)


Usage : tackon var pathname filename
Example : tackon x df0:c Dir  # sets x to "df0:c/Dir"
or  : tackon x df0: Preferences #sets x to "df0:Preferences"

Correctly  adds  a  filename  to a pathname, and puts the result in
variable specified.
```

1.99 tail

```
  TAIL


Usage : tail [filename] [num]
Example : tail readme 20

Display last "num" lines of "filename".    If num is not specified,
10 is assumed. If filename is not specified, standard input (stdin)
is taken instead.
```

1.100 tee

```
  TEE


Usage : tee [file]
Example : cc test.c | tee >error.list
```

```
Copies stdin to stdout and the given file.
If file is omitted, stderr is used.
```

1.101 touch

```
  TOUCH


Usage : touch file1 .. fileN

Sets DateStamp of the specified files to the current date & resets
archive bit.

If a file doesn't exist, touch will create an empty one for you.
```

1.102 truncate

```
  TRUNCATE


Usage : truncate [n]
Example : alias | qsort | truncate

A filter that truncates the width of stdin to the specified number,
trying to account for tab's and escape sequences. If the number is
omitted, the current window width is used.
```

1.103 type

```
  TYPE (or CAT)


Equivalent to CAT.
```

1.104 unalias

```
  UNALIAS


Usage : unalias name .. name
Example : unalias vt

Delete aliases..
```

1.105 uniq

```
  UNIQ


Usage : uniq

This is a  filter that removes consecutive, duplicated lines in a
file.  It is most useful on a sorted file.
```

1.106 unset

```
  UNSET


Usage : unset name .. name
Example : unset abc

Unset one or more variables.  Deletes them entirely.
```

1.107 usage

```
  USAGE


Usage : usage [command...command]

If called without arguments, usage gives you a short information
on the special characters used. Otherwise, usage shows you the
usage of the given commands. Calling a command with a '?' as
the only argument will show its usage, too.
```

1.108 version

```
  VERSION


Usage : version

Show current version name, & authors.
```

1.109 waitforport

```
  WAITFORPORT


Usage : waitforport portname [seconds]
```

```
Example : waitforport rexx_ced 5

Waits for a port to come up. Default time is 10 seconds.
```

1.110 whereis

```
  WHEREIS


Usage : whereis [-r] filename [ device1...deviceN ]


Options :
 -r look on all drives.

If just a file name is given, whereis searches all subdirectories of
the current directory for that file. An asterisk '*' is appended to
the file. Wild cards are allowed for the file (no asterisk will be
appended then), but no path names. If additional arguments are
given, whereis searches only these paths, not the current directory.
```

1.111 window

```
  WINDOW


Usage : window [-fblsaq] [dimensions]

Options :
  -f  (front) Window to front
  -b  (back)  Window to back
  -l  (large) Window to maximum size
  -s  (small) Window to minimum size
  -a  (activate)
  -q  (query) Lists screens and windows open
  -w  (width) Ignore window width for option "-q" (query)

   Various  operations on CLI window.  If dimensions are specified,
they must be in the form x y width height, with values separated by
spaces.
   The  command  "window  -l" may be very useful on PAL machines to
get  a  full  PAL  window  from  your login sequence, or if you use
overscan WorkBench.
   Option  -q  gives,  for  each  Screen and Window currently open,
title, left edge, top edge, width, height, (depth).
```

1.112 writefile

```
  WRITEFILE

Usage: writefile varname

Writes a set of words to stdout, one word per line. Note that the
name of the variable (var) must be supplied, not the value ($var).
```

1.113 commands

```
abortline

action

addbuffers

addpart

alias

ascii

assign

basename

cat

cd

chgrp

chmod

chown

class

close

cls

copy

cp

date

dec

delete
```

```
dir
diskchange
echo
else
endif
error
exec
fault
filenote
flist
fltlower
fltupper
foreach
forever
forline
fornum
getenv
goto
head
help
history
howmany
htype
if
inc
info
input
join
keymap
```

```
label

linecnt

ln

local

ls

makelink

man

md

mem

menu

mkdir

mv

open

path

pri

protect

ps

pwd

qsort

quit

rback

readfile

rehash

relabel

rename

resident

return

rm
```

```
rpn

run

rxrec

rxsend

search

set

setenv

sleep

source

split

stack

strhead

strings

strleft

strlen

strmid

strright

strtail

tackon

tail

tee

touch

truncate

type

unalias

uniq

unset

usage

version
```

```
waitforport

whereis

window

writefile
```

1.114 _abbrev

```
  _abbrev

Holds a number which lets you select the various modes of
command-abbreviation:

  0   internal commands and commands buffered with "rehash"
      can no longer be abbreviated  (same as "unset _abbrev")
  1   internal commands can be abbreviated
  2   commands buffered with "rehash" can be abbreviated,
      the first (partially) matching command from the list is taken
  4   commands buffered with "rehash" can be abbreviated,
      if the command matches a buffered command completely,
      then this is taken, else the first partially matching
      command from the list is taken
  8   if command wasn't found in Cshell's internal program list
      (built with "rehash"), then search DOS path-list

Numbers can be added to combine modes.

The main difference between '2' and '4' is that '2' does not
recognize a completely matching command if it has already
found a partially matching command earlier in the list.  Eg,
you type "ed" and you have (in this order) "EdPlayer" and
"Ed" in your list, then "Ed" can never be called (except with
absolute path). So '4' first seeks for a completely matching
command -- and if it cannot find anything then (and only then)
it searches for an abbreviated command.

Thus it makes obviously no sense to use '2' and '4' together.

By default, this variable is set to '5' (1+4).
```

1.115 _bground

```
  _bground

True if the shell was started with a non-interactive input.
```

1.116 _clinumber

```
  _clinumber

Contains the number (1-20) of current CLI.
```

1.117 _clipri

```
  _clipri

Task priority while editing command line.
(also affects filename completion)
```

1.118 _cquote

```
  _cquote

If set to some value, quotation marks are handled as
Commodore-Shell does, so that they are parsable by ReadArgs()
(function call in dos.library). Commodore-Shell treats quotes
within a string "as as". Only leading quotation marks "quote"
other special chars (like spaces). In contrast to UNIX shells
where quotation marks always "quote" other chars - regardless
of their position. UNIX behavior (_cquote unset) is default.
```

1.119 _cwd

```
  _cwd

Holds a string representing the current directory we are in from
root.  The SHELL can get confused as to its current directory if
some external program changes the directory.  Use PWD to rebuild
the _cwd variable in these cases.
```

1.120 _debug

```
  _debug

Debug mode... use it if you dare.  must be set to some value
```

1.121 _dirformat

```
  _dirformat

Holds a format string for option -z of builtin command "dir".
Used to keep aliases short and to bypass problems with dir's
```

```
format option "-z" (eg, it's not possible to specify other options
after -z without using @pickargs/@pickopts).

The format string is limited to a maximum of 80 characters.
```

1.122 _every

```
  _every

Contains the name of a command that is to be executed every time
just before the prompt is printed. Do not use this to echo the
prompt.
```

1.123 _except

```
  _except

See EXCEPTION
```

1.124 _failat

```
  _failat

If a command returns with an error code higher than this, the
batch file aborts. The default is 20.
```

1.125 _hilite

```
  _hilite

Holds the font attributes used for highlighting. One letter for one
attribute:
  b    for bold
  i    for italic
  u    for underlined
  r    for reverse
  c3   for foreground color 3
  c3,2 for foreground color 3 and background color 2
Any combinations are allowed. _hilite defaults to "c7", in terminal
mode to "r".
```

1.126 _history

```
  _history

This variable is set to a numerical value, and specifies how far
back your history should extend. Set it to 0 to disable history,
for example if you test your programs for memory leaks. Defaults
to 50.
```

1.127 _insert

```
  _insert

Sets the default for insert/overtype mode for command line editing.
ESC-i toggles between, but after <RET> the default is set back as
indicated by this variable. By default _insert is 1, unsetting
_insert will make overtype the default.
```

1.128 _ioerr

```
  _ioerr

Contains the secondary error code for the last command. Will be
changed after every external command and after a failed internal
command. See @ioerr( )
```

1.129 _kick

```
  _kick

holds version number of Operating System (version.library),
eg, 37, 38, 39, 40, ...
```

1.130 _kick2x

```
  _kick2x

True if exec.library V37+ could be opened
(which means that kickstart 2.0 is around)
```

1.131 _kick3x

```
  _kick3x

True if exec.library V39+ could be opened
(which means that kickstart 3.0 is around)
```

1.132 _lasterr

```
_lasterr
```

Return code of last command executed. This includes internal commands as well as external comands, so to use this variables you must check it IMMEDIATELY after the command in question.

1.133 _lcd

```
_lcd
```

Holds the name of the last directory. The builtin alias 'dswap' cd's to that directory. If called again, you're back where you were.

1.134 _man

```
_man
```

The path and name of your .doc files. Defaults to 'csh:csh.doc'

1.135 _mappath

```
_mappath
```

Cshell allows invocation of foreign shells if a script starts with "#!" or ";!" followed by a command to execute. To use unmodified Unix scripts pathname-mapping is necessary to convert pathes like "/usr/..." to, eg, "usr:...". Set _mappath to enable this name-mapping. By default _mappath is unset ("/usr/" would be interpreted as relative AmigaDOS path).

1.136 _maxerr

```
_maxerr
```

The worst (highest) return value to date. To use this, you usually set it to '0', then do some set of commands, then check it.

1.137 _minrows

```
_minrows
```

Gives the minimum number of rows a window must have to turn on quick scrolling. Defaults to 34.

1.138 _nobreak

```
  _nobreak

If set to some value, disables CTRL-C.
```

1.139 _nomatch

```
  _nomatch

If set to some value, don't check patterns if they match.
(By default CSH aborts command execution if all patterns does not
 match. If at least one pattern matches CSH does not abort.)
```

1.140 _noreq

```
  _noreq

If set to some value, disables system requesters ("Please insert
volume ...."). Turned on in vt200 mode.
```

1.141 _passed

```
  _passed

This variable contains the passed arguments when you SOURCE a file
or execute a .sh file.  For instance:

test a b c d

-------- file test.sh ----------
echo $_passed
foreach i ( $_passed ) "echo YO $i"
--------------------------------
```

1.142 _path

```
  _path

Tells CShell where to look for executable files. The current direc-
tory and the AmigaDOS path will be searched first. The trailing
slash for directories is not necessary any more.  The entire path
will be searched first for the <command>, then for <command>.sh
(automatic shell script sourcing). Example:
  set _path ram:c,ram:,sys:system,dh1:tools,df0:c
(This path has the advantage that these directories need not even
 exist, that you can access devices (AmigaDOS path only knows
```

```
 volumes under Kick 1.3) and that no disk seeks happen at startup)

The usage of _path is NOT recommended anymore, use the AmigaDOS
search path instead (builtin command "path").
```

1.143 _prghash

```
  _prghash

The filename where the program hash list (command 'rehash')
is loaded from and saved to.
```

1.144 _prompt

```
  _prompt

This variable now can contain the following control characters:
 %c  for color change. This highlights your prompt. See _hilite
 %e  for elapsed time. The time the last command took to execute.
 %m  for memory. This shows your current memory in K
 %t  for time. This shows your current time in the format HH:MM:SS
 %d  for date. This shows the current date in the format DD-MMM-YY
 %p  for path. This inserts the current path
 %V  for volume. This inserts the current volume
 %n  for number. This inserts the current process number
 %v  for version. This shows the version number of CShell
 %h  for history. This displays the current history number
 %f  for free store. This shows the free store on the current drive
 %r  for pRiority. Inserts the task priority of the current
 %s  for shells open. Inserts the result of 'howmany'
 %U  for user. Shows current user (only with "MultiUser" package)
 %x  for external cmd return code. Yields the last error code
The default for prompt is now "%c%p> "
The  if  command  will  set the prompt to a '_ ' if commands are
disabled while waiting for a 'endif' or 'else' command (interactive
mode only).
```

1.145 _pipe

```
  _pipe

The directory where temporaries are stored. Default: 'T:'
```

1.146 _qcd

_qcd

Holds the name of the file where the all directories on your hard disk are stored. If not set, disables quick cd-ing.

1.147 _rback

_rback

Is the name of the command to be the prepended to the command line when '&' was added to it. Defaults to 'rback', can't be a multi word command yet.

1.148 _rxpath

_rxpath

The same as with _path, but this is where CShell looks for .rexx files. Defaults to REXX:

1.149 _scroll

_scroll

Holds the number of lines to be scrolled at once when quick scrolling is used. If unset or <=1, quick scrolling is off. Defaults to 3.

1.150 _terminal

_terminal

Indicates whether or not shell was started in terminal mode.

1.151 _timeout

_timeout

Set the timeout period (in microsends) for the connected terminal to respond to a WINDOW STATUS REQUEST (special Amiga control sequence to get window bounds). Only used in terminal mode, of course.

For local usage a small value is sufficient, for remote usage (eg, over a serial line) the value should be much higher (eg, 500000, which a 1/2 second).

1.152 _titlebar

```
  _titlebar
```

The same control characters as for the _prompt can be used for _titlebar, too. The only difference is that %c is ignored. The titlebar is updated every time before the prompt appears.

1.153 _verbose

```
  _verbose
```

If set to 's', turns on verbose mode for source files (every command will be displayed before being executed). If set to 'a', displays all substeps while alias substitution. 'h' will hilight the debug output. Any combination allowed: set _verbose sah

1.154 _version

```
  _version
```

Contains the version number of the shell, e.g. 510.

1.155 variables

_abbrev

_bground

_clinumber

_clipri

_cquote

_cwd

_debug

_dirformat

_every

_except

_failat

```
_hilite

_history

_insert

_ioerr

_kick

_kick2x

_kick3x

_lasterr

_lcd

_man

_mappath

_maxerr

_minrows

_nobreak

_nomatch

_noreq

_passed

_path

_prghash

_prompt

_pipe

_qcd

_rback

_rxpath

_scroll

_terminal

_timeout

_titlebar

_verbose
```

```
_version
```

1.156 @abbrev

```
@abbrev( str1 str2 [len] )
```

true if the first <len> chars of str1 are an abbreviation of str2

1.157 @abs

```
@abs( num )
```

returns absolute value of <num>

1.158 @age

```
@age( file )
```

the age of that file in days, null-string if file not found

1.159 @age_mins

```
@age_mins()
```

the age of that file in minutes, null-string if file not found

1.160 @appsuff

```
@appsuff( name suffix )
```

appends an suffix (.ZOO) to a string if it's not already there

1.161 @arg

```
@arg( arg ... arg )
```

see @pickargs( )

1.162 @ask

```
  @ask( title item ... item )
```

asks for confirmation of every item and returns the confirmed ones
(very similar to @confirm(), but default is negative)

1.163 @availmem

```
  @availmem( [type] )
```

returns free 'chip', 'fast' or otherwise total memory

1.164 @basename

```
  @basename( path ... path )
```

returns the file name parts of the paths

1.165 @center

```
  @center( word len )
```

returns a string of length <len> with <word> centered in it

1.166 @checkport

```
  @checkport( portname )
```

indicates if given port exists

1.167 @clinum

```
  @clinum( procname )
```

returns the number of the cli identified by a name or a number

1.168 @complete

```
  @complete( abbrev word ... word )
```

returns the first word <abbrev> is an abbreviation of

1.169 @concat

```
  @concat( word word ... word )
```

concats all words in one blank separated string, see @split

1.170 @confirm

```
  @confirm( title item ... item )
```

asks for confirmation of every item and returns the confirmed ones
(very similar to @ask(), but default is positive)

1.171 @console

```
  @console( STDIN|STDOUT )
```

tells whether stdin or stdout are interactive (not redirected)

1.172 @dectohex

```
  @dectohex( number )
```

returns a string representing <number> in hex

1.173 @delword

```
  @delword( word word ... word n )
```

returns a string with the n-th word deleted.

1.174 @delwords

```
  @delwords( word word ... word n m )
```

deletes the next m words from the n-th.

1.175 @dirname

```
  @dirname( path )
```

strips the base name from a path, just returns the directory

1.176 @dirs

```
  @dirs( name name name name )
```

returns the directories among the given file names, see @files

1.177 @dirstr

```
  @dirstr( lformat file )
```

returns any info (size, date, file comment) about a file

1.178 @drive

```
  @drive( path )
```

outputs the drive (device) name associated to <path>

1.179 @drives

```
  @drives( )
```

outputs all available drives

1.180 @exists

```
  @exists( file )
```

tells whether a file exists or not

1.181 @fileblks

```
  @fileblks( file file ... file )
```

returns the # of blocks needed for the files, incl. dir blocks

1.182 @filedate

```
  @filedate( file )
```

returns a string representing the date of the given file

1.183 @fileinfo

```
  @fileinfo
```

Equivalent to @dirstr

1.184 @filelen

```
  @filelen( file file ... file )
```

count the total number of bytes of the given files

1.185 @filenote

```
  @filenote( file )
```

returns filenote of given file

1.186 @fileprot

```
  @fileprot( file )
```

returns a string like ---arwed

1.187 @filereq

```
  @filereq( title path&pattern filename )
```

brings up the ASL file requester and returns the selected file name

1.188 @files

```
  @files( file file ... file )
```

gives you the files among those names, no directories. see @dirs

1.189 @filesize

```
  @filesize
```

Equivalent to @filelen

1.190 @flines

```
  @flines( varname )
```

counts the number of lines in a readfile-file (faster than @words)

1.191 @freebytes

```
  @freebytes( path )
```

the number of free bytes on the given path

1.192 @freeblks

```
  @freeblks( path )
```

the number of free blocks on the given path

1.193 @freestore

```
  @freestore( path )
```

the amount of free store on that path, given in K, M and G

1.194 @getenv

```
  @getenv( varname )
```

returns the value of the named env: variable

1.195 @getclass

```
  @getclass( file )
```

returns the class (type) of the file. See chapter XIV

1.196 @hextodec

```
  @hextodec( hex-number )
```

returns a string representing <hex-number> in dec

1.197 @howmany

```
  @howmany( )
```

indicates the # of shells running

1.198 @index

```
  @index( string pattern )
```

returns the index of pattern in string (starting at 1),
0 if not found

1.199 @info

```
  @info( path )
```

the corresponding line from the 'info' command, each entry a word

1.200 @intersect

```
  @intersect( word1 word2 word3 , word4 word5 word6 )
```

returns all words which are in both lists. see @union, @member

1.201 @ioerr

```
  @ioerr( num )
```

returns the corresponding error string to num

1.202 @lookfor

```
  @lookfor( file paths )
```

looks for a file in the current directory and the paths. See $_path

1.203 @lower

```
  @lower
```

lowercases its arguments. see @upper

1.204 @match

```
  @match( word ... word "pattern" )
```

returns the words in the list that match the DOS-pattern

1.205 @max

```
  @max( num num ... num )
```

computes the maximum of all given numbers

1.206 @megs

```
  @megs( number )
```

expresses a number in K, M and G (-bytes), rounded correctly

1.207 @member

```
  @member( word1 word word ... word )
```

tells you if word1 is among the remaining words

1.208 @min

```
  @min( num num ... num )
```

computes the minimum of all given numbers

1.209 @mix

```
  @mix( arg1 ... argn )
```

randomly mixes its arguments

1.210 @mounted

```
  @mounted( device )
```

returns a boolean indicating whether the specified device is mounted,
(don't add an extra colon ':' at the end)

1.211 @nameext

```
  @nameext( filename )
```

returns all after the last dot of <filename>.

1.212 @nameroot

```
  @nameroot( filename )
```

returns all before the LAST dot of <filename>.

1.213 @opt

```
  @opt( arg ... arg )
```

see @pickopts( )

1.214 @pathname

```
  @pathname( path )
```

obsolete. use @dirname

1.215 @pickargs

```
  @pickargs( arg ... arg )
```

picks of its arguments those which don't start with a '-'

1.216 @pickopts

```
  @pickopts( arg ... arg )
```

picks of its arguments those which start which a '-'

1.217 @rnd

```
  @rnd( seed )
```

returns a 32 bit random number (default seed is 1);
'seed' is optional and can be used to set new seed for @rnd( ),
if you use seed=0 then CSH takes current system time as seed

1.218 @rpn

```
  @rpn( expression )
```

computes the rpn expression. See rpn command

1.219 @scrheight

```
  @scrheight( )
```

outputs the current height of the screen the shell is running in

1.220 @scrwidth

```
  @scrwidth( )
```

outputs the current width of the screen the shell is running in

1.221 @sortargs

```
  @sortargs( name ... name )
```

sorts its arguments alphabetically

1.222 @sortnum

```
  @sortnum( number ... number )
```

sorts its arguments numerically

1.223 @split

```
  @split( string )
```

makes each blank separated part of @string a word, see @concat

1.224 @strcmp

```
  @strcmp( name name )
```

returns -1, 0 or 1 depending of alphabetical comparison
(case-sensitive)

1.225 @stricmp

```
  @stricmp( name name )

returns -1, 0 or 1 depending of alphabetical comparison
(case-insensitive)
```

1.226 @strhead

```
  @strhead( breakchar string )

see strhead command
```

1.227 @strleft

```
  @strleft( string number )

see strleft command
```

1.228 @strmid

```
  @strmid( string n1 n2 )

see strmid command
```

1.229 @strright

```
  @strright( string n )

see strright command
```

1.230 @strtail

```
  @strtail( breakchar string )

see strtail command
```

1.231 @subfile

```
  @subfile( varname n m )

like @subwords, but acts on a readfile-file and is faster
```

1.232 @subwords

```
  @subwords( word ... word n m )
```

returns the next m words word of the given list starting from n

1.233 @tackon

```
  @tackon( path file )
```

see tackon command

1.234 @trim

```
  @trim( word word word )
```

removes all leading and trailing blanks from the words

1.235 @unique

```
  @unique( word ... word )
```

sorts the arguments and makes each of them unique

1.236 @union

```
  @union( name ... name , name ... name )
```

returns all names that are in either list. See @intersect, @member

1.237 @upper

```
  @upper( word ... word )
```

upper cases the given words. see @lower

1.238 @volume

```
  @volume( path )
```

returns the volume name in that path or ""

1.239 @wincols

```
  @wincols( )
```

returns the number of columns in the current shell window

1.240 @winheight

```
  @winheight( )
```

outputs the height of your window in pixels

1.241 @winleft

```
  @winleft( )
```

returns the left edge of your window

1.242 @winrows

```
  @winrows( )
```

returns the number of lines in the current shell window

1.243 @wintop

```
  @wintop( )
```

returns the top edge of your window

1.244 @winwidth

```
  @winwidth( )
```

outputs the width of your window in pixels

1.245 @without

```
  @without( name ... name , name ... name )
```

returns all names of list 1 that are not in list 2

1.246 @word

```
  @word( name ... name n )
```

picks the n-th word from the list.

1.247 @words

```
  @words( name ... name )
```

returns the number of words in the list.

1.248 functions

```
@abbrev
@abs
@age
@age_mins
@appsuff
@arg
@ask
@availmem
@basename
@center
@checkport
@clinum
@complete
@concat
@confirm
@console
@dectohex
@delword
```

```
@delwords

@dirname

@dirs

@dirstr

@drive

@drives

@exists

@fileblks

@filedate

@fileinfo

@filelen

@filenote

@fileprot

@filereq

@files

@filesize

@flines

@freebytes

@freeblks

@freestore

@getenv

@getclass

@hextodec

@howmany

@index

@info

@intersect

@ioerr

@lookfor
```

```
@lower

@match

@max

@megs

@member

@min

@mix

@mounted

@nameext

@nameroot

@opt

@pathname

@pickargs

@pickopts

@rnd

@rpn

@scrheight

@scrwidth

@sortargs

@sortnum

@split

@strcmp

@stricmp

@strhead

@strleft

@strmid

@strright

@strtail
```

```
@subfile

@subwords

@tackon

@trim

@unique

@union

@upper

@volume

@wincols

@winheight

@winleft

@winrows

@wintop

@winwidth

@without

@word

@words
```