

csH-reference

COLLABORATORS

	<i>TITLE :</i> csH-reference		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 10, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ssh-reference	1
1.1	C-Shell documentation	1
1.2	commands	1
1.3	abortline	2
1.4	action	3
1.5	addbuffers	3
1.6	alias	3
1.7	ascii	4
1.8	aset	4
1.9	assign	4
1.10	basename	5
1.11	cat	5
1.12	cd	5
1.13	class	6
1.14	close	6
1.15	cls	6
1.16	copy	6
1.17	cp	7
1.18	date	7
1.19	dec	8
1.20	delete	8
1.21	dir	8
1.22	diskchange	9
1.23	echo	9
1.24	else	10
1.25	endif	10
1.26	error	10
1.27	exec	10
1.28	fault	10
1.29	filenote	11

1.30	flist	11
1.31	ftlower	11
1.32	ftupper	11
1.33	foreach	11
1.34	forever	12
1.35	forline	12
1.36	fornum	12
1.37	getenv	12
1.38	goto	13
1.39	head	13
1.40	help	13
1.41	history	13
1.42	howmany	14
1.43	htype	14
1.44	if	14
1.45	inc	15
1.46	info	15
1.47	input	16
1.48	join	16
1.49	keymap	16
1.50	label	16
1.51	linecnt	16
1.52	local	16
1.53	ls	17
1.54	md	17
1.55	man	17
1.56	mem	17
1.57	menu	18
1.58	mkdir	18
1.59	mv	18
1.60	open	18
1.61	path	19
1.62	pri	19
1.63	protect	19
1.64	ps	19
1.65	pwd	20
1.66	qsort	20
1.67	quit	20
1.68	rback	20

1.69 readfile	20
1.70 rename	21
1.71 resident	21
1.72 return	21
1.73 rm	21
1.74 rpn	22
1.75 run	22
1.76 rxrec	22
1.77 rxsend	23
1.78 search	23
1.79 set	24
1.80 setenv	24
1.81 sleep	25
1.82 split	25
1.83 stack	25
1.84 strhead	25
1.85 strings	25
1.86 strleft	25
1.87 strlen	26
1.88 strmid	26
1.89 strright	26
1.90 strtail	26
1.91 source	26
1.92 tackon	27
1.93 tail	27
1.94 tee	27
1.95 touch	28
1.96 truncate	28
1.97 type	28
1.98 unalias	28
1.99 uniq	28
1.100unset	29
1.101usage	29
1.102version	29
1.103waitforport	29
1.104whereis	29
1.105window	29
1.106writefile	30
1.107variables	30

1.108_abbrev	30
1.109_bground	31
1.110_clinumber	31
1.111_cwd	31
1.112_debug	31
1.113_every	31
1.114_except	31
1.115_failat	31
1.116_hilite	31
1.117_history	32
1.118_insert	32
1.119_ioerr	32
1.120_kick2x	32
1.121_lasterr	32
1.122_lcd	33
1.123_man	33
1.124_maxerr	33
1.125_minrows	33
1.126_nobreak	33
1.127_noreq	33
1.128_passed	33
1.129_path	34
1.130_prompt	34
1.131_pipe	34
1.132_qcd	34
1.133_rback	34
1.134_rxpath	35
1.135_scroll	35
1.136_terminal	35
1.137_titlebar	35
1.138_verbose	35
1.139_version	35
1.140functions	35
1.141@abbrev	36
1.142@abs	36
1.143@age	36
1.144@appsuff	37
1.145@arg	37
1.146@availmem	37

1.147 @basename	37
1.148 @center	37
1.149 @checkport	37
1.150 @clinum	37
1.151 @complete	38
1.152 @concat	38
1.153 @confirm	38
1.154 @console	38
1.155 @dectohex	38
1.156 @delword	38
1.157 @delwords	38
1.158 @dirname	38
1.159 @dirs	39
1.160 @dirstr	39
1.161 @drive	39
1.162 @drives	39
1.163 @exists	39
1.164 @fileblks	39
1.165 @filelen	39
1.166 @fileprot	39
1.167 @filereq	40
1.168 @files	40
1.169 @flines	40
1.170 @freebytes	40
1.171 @freeblks	40
1.172 @freestore	40
1.173 @getenv	40
1.174 @getclass	40
1.175 @howmany	41
1.176 @index	41
1.177 @info	41
1.178 @intersect	41
1.179 @ioerr	41
1.180 @lookfor	41
1.181 @lower	41
1.182 @match	41
1.183 @max	42
1.184 @megs	42
1.185 @member	42

1.186@min	42
1.187@mix	42
1.188@mounted	42
1.189@nameext	42
1.190@nameroot	42
1.191@opt	43
1.192@pathname	43
1.193@pickargs	43
1.194@pickopts	43
1.195@rnd	43
1.196@rpn	43
1.197@scrheight	43
1.198@scrwidth	43
1.199@sortargs	44
1.200@sortnum	44
1.201@split	44
1.202@strcmp	44
1.203@strhead	44
1.204@strleft	44
1.205@strmid	44
1.206@stright	44
1.207@strtail	45
1.208@subfile	45
1.209@subwords	45
1.210@tackon	45
1.211@trim	45
1.212@unique	45
1.213@union	45
1.214@upper	45
1.215@volume	46
1.216@wincols	46
1.217@winheight	46
1.218@winleft	46
1.219@winrows	46
1.220@wintop	46
1.221@winwidth	46
1.222@without	46
1.223@word	47
1.224@words	47

Chapter 1

ssh-reference

1.1 C-Shell documentation

```
C-SHELL 5.19          1991-1992 by U. DOMINIK MUELLER          REFERENCE

@{ " Commands " link Commands }
@{ " Variables " link Variables }
@{ " Functions " link Functions }
```

1.2 commands

```
@{ "abortline " link abortline } @{"dir " link dir } @{" " ←
  head " link head } @{"menu " link menu } @{"run ←
    " link run } @{"tail " link tail }
@{ "action " link action } @{"diskchange " link diskchange } @{" " ←
  help " link help } @{"mkdir " link mkdir } @{" " ←
  rxrec " link rxrec } @{"tee " link tee }
```

```

@{ "addbuffers " link addbuffers } @{"echo " link echo } @{" " ←
  history " link history } @{"mv " link mv } @{" " ←
  rxsend " link rxsend } @{"touch " link touch }
@{"alias " link alias } @{"else " link else } @{" " ←
  howmany " link howmany } @{"open " link open } @{" " ←
  search " link search } @{"truncate " link truncate }
@{"ascii " link ascii } @{"endif " link endif } @{" " ←
  htype " link htype } @{"path " link path } @{"set ←
    " link set } @{"type " link type }
@{"aset " link aset } @{"error " link error } @{"if ←
    " link if } @{"pri " link pri } @{" " ←
  setenv " link setenv } @{"unalias " link unalias }
@{"assign " link assign } @{"exec " link exec } @{" " ←
  inc " link inc } @{"protect " link protect } @{" " ←
  sleep " link sleep } @{"uniq " link uniq }
@{"basename " link basename } @{"fault " link fault } @{" " ←
  info " link info } @{"ps " link ps } @{" " ←
  split " link split } @{"unset " link unset }
@{"cat " link cat } @{"filenote " link filenote } @{" " ←
  input " link input } @{"pwd " link pwd } @{" " ←
  stack " link stack } @{"usage " link usage }
@{"cd " link cd } @{"flist " link flist } @{" " ←
  join " link join } @{"qsort " link qsort } @{" " ←
  strhead " link strhead } @{"version " link version }
@{"class " link class } @{"fltlower " link fltlower } @{" " ←
  keymap " link keymap } @{"quit " link quit } @{" " ←
  strings " link strings } @{"waitforport" link waitforport }
@{"close " link close } @{"fltupper " link fltupper } @{" " ←
  label " link label } @{"rback " link rback } @{" " ←
  strleft " link strleft } @{"whereis " link whereis }
@{"cls " link cls } @{"foreach " link foreach } @{" " ←
  linecnt " link linecnt } @{"readfile " link readfile } @{" " ←
  strlen " link strlen } @{"window " link window }
@{"copy " link copy } @{"forever " link forever } @{" " ←
  local " link local } @{"rename " link rename } @{" " ←
  strmid " link strmid } @{"writefile " link writefile }
@{"cp " link cp } @{"forline " link forline } @{"ls ←
    " link ls } @{"resident " link resident } @{" " ←
  strright " link strright }
@{"date " link date } @{"fornum " link fornum } @{"md ←
    " link md } @{"return " link return } @{" " ←
  strtail " link strtail }
@{"dec " link dec } @{"getenv " link getenv } @{" " ←
  man " link man } @{"rm " link rm } @{" " ←
  source " link source }
@{"delete " link delete } @{"goto " link goto } @{" " ←
  mem " link mem } @{"rpn " link rpn } @{" " ←
  tackon " link tackon }

```

1.3 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.

1.4 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

1.5 addbuffers

Usage : addbuffers drive buffers [drive buffers ...]

Example : addbuffers df0: 24

Just like AmigaDOS addbuffer command, causes new buffers to be allocated for disk I/O. Each buffer costs 512 bytes of memory, CHIP memory if a disk drive.

1.6 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

Usage : ascii

```
ascii 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 aset

Usage : aset name value

Example : aset INCLUDE include:

Set a variable in a way that is compatible with ARP/old Aztec set command; this is completely different from ENV: Shell variable.

This is obsolete. Use setenv instead.

1.9 assign

Usage : assign

```
assign logical
```

```
assign [-lnp] logical1 physical1 [logical2 physical2 ... ]
```

The first form shows all assigns.
The second form kills one assign.
The third form assigns logical1 to physical1 and so on. Options:
-l creates a late-binding assign under kick 2.0, normal otherwise
-n creates a non-binding assign under kick 2.0, normal otherwise
-p creates a path-assign under kick 2.0, cancelled otherwise
For definition of late/nonbinding, refer to your AmigaDOS manual.

1.10 basename

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

Sets var specified to basenames of paths.

1.11 cat

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

Type the specified files onto the screen. If no file is specified, STDIN is used (note: ^\ is EOF). CAT is meant to output text files only. Specifying -n option you will output numbered lines.

1.12 cd

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

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'll no more 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 directories 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.

1.13 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

1.14 close

Usage : close [filenumber]

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

1.15 cls

Usage : cls

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

1.16 copy

Usage : copy [-udfpm] file file
 or : copy [-udfpm] file1 file2...fileN dir
 or : copy [-rudfp] 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

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 override 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.

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.

1.17 cp

Equivalent to copy.

1.18 date

Usage : date [-sr] [new date and/or time]

Example : date Wednesday # this refers to NEXT wed, of course

Options:

- 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.

1.19 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.

1.20 delete

Usage : delete [-p][-r][-q] file file file...
 Example : delete foo.txt test.c

Remove (delete) the specified files. Remove always returns errorcode 0. You can remove empty directories. The '-r' option will remove non-empty directories by recursively removing all sub directories.

You can remove delete-protected files specifying -p option. If you specify any wildcard deletes the files will be listed as they are deleted. This can be suppressed by redirecting to nil: If you specify -q, no error message will be supplied if the file to be removed didn't exist or couldn't be deleted.

1.21 dir

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

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.

The lformat string is used to create your own directory format. It will override all else and 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	

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.

1.22 diskchange

Usage : diskchange drive

Like AmigaDOS diskchange.

1.23 echo

Usage : echo [-en] string

Example : echo hi there

Results : hi there

Echo the string given. If -n switch given no newline is appended. If -e is on, echo to stderr.

1.24 else

Usage : else ; command

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

Else clause, must follow an IF statement.

1.25 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.

1.26 error

Usage : error n

Generates return code n.

1.27 exec

Usage : exec [-i] command [args]

Example : set cmdline "dir ram:"

exec \$cmdline # would not work without exec

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.

1.28 fault

Usage : fault error1 .. errorN

Example : fault 205 212

Like AmigaDOS fault, prints specified error messages.

1.29 filenote

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

The first form sets AmigaDOS comment of the specified file.
The second form displays the file notes of the given files.

1.30 flist

```
Usage : flist
```

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

1.31 ftolower

```
Usage : ftolower
Example : dir | ftolower
Or : ftolower <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.

1.32 ftupper

The same of ftolower, only this converts to upper case.

1.33 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
```

'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

```

Flag -v causes arguments to be displayed every time command is executed. All 'for...' commands can be interrupted using CTRL-D.

1.34 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.

1.35 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.

1.36 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.

Switch -v (verbose) causes printing of progressive numbers.

Switch -s allows you to specify a step; if this is negative, the count will be backwards.

1.37 getenv

Usage : getenv [shellvar] envvar

Gets the value of an ARP or ENV: variable (ARP list searched first) 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.

1.38 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.

1.39 head

Usage : head filename [num]

Example : head readme 20

Display first "num" lines of "filename". If num is not specified, 10 is assumed.

1.40 help

Usage : help

Example : help

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.

1.41 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.

1.42 howmany

Usage : `howmany`

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

1.43 htype

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

Displays the specified files in hex and ASCII, just like the system command `'Type file opt h'`. Especially suitable for binary files. If `-r` is specified, `htype` displays all files in a directories.

1.44 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`

Makes the following instructions up to the next `endif` conditional. 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 `if 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 existence of

the specified file. -e is the same as -f

Switch -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.

Switch -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 exist, 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 -r evaluates a given RPN expression (see under RPN for more info): if value on top of stack is 0, then FALSE, else TRUE.

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

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

Switch -n (NOT) reverses the result.

To test if a given variable is defined, use if -v varname.

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.

1.45 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.

1.46 info

Usage : info [path1 path2 ... pathN]

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

1.47 input

Usage : input [-s] var var ... var

Example : input abc

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. If -s is turned on, 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 this with care.

1.48 join

Usage : join [-r] file1..fileN destfile

Example : join part1 part2 part3 total

Joins the specified files to get destfile. If destfile already exists, an error message is generated and operation is aborted, unless you specify -r (replace) option.

1.49 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.

1.50 label

Usage : label name

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

1.51 linecnt

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

1.52 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.

1.53 ls

Equivalent to dir.

1.54 md

Equivalent to mkdir.

1.55 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 dhl:docs/aliases.doc dhl: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.

1.56 mem

Usage : mem [-cfqu]

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
-

1.57 menu

Usage : menu [-n] [title item...item]

Example : menu Shell JrComm,,j Rename,"rename ",r quit

Appends one pull down in the current console window. Up to six menus with 16 items every 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.

To clear all existing menus use option -n.

1.58 mkdir

Usage : mkdir name name name...

Example : mkdir df0:stuff

Create the specified directories.

1.59 mv

Equivalent to rename.

1.60 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.

1.61 path

Usage : path [-r] [dir...dir]

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

Options:

-r Resets the path

1.62 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).

1.63 protect

Usage : protect file1 ... fileN [+|-|=][flags]

Example : protect myfile +rwe

Set AMIGADOS file protection flags for the file specified. Valid flags are h, s, p, a, r, w, e, d. x ist the same as e. 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.

1.64 ps

Usage : ps [commandname...commandname]

Gives status of CLI processes. eg:

Proc	Command	Name	CLI Type	Pri.	Address	Directory
* 1	csH		Initial CLI	0	97b0	Stuff:shell
2	clock		Background	-10	2101a8	Workdisk:
3	emacs		Background	0	212f58	Stuff:shell

```
4 VT100      Background    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.

Options:

- l shows full pathnames of commands
- e excludes the given command names from the list

1.65 pwd

Usage : pwd

Rebuild `_cwd` by backtracing from your current directory.

1.66 qsort

Usage : qsort [-r] <in >out

Quick sorts from stdin to stdout. -r causes reverse sorting.

1.67 quit

Usage : quit

Quit out of Shell back to CLI.

1.68 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. The variable `Under 1.3`, `$_newproc` will hold the cli number of the newly created process.

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

1.69 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

1.70 rename

Usage : rename from to
 or : rename from from from ... from todir

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.

1.71 resident

Usage : resident [-r][-d] [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

This is ARP resident. Commands are searched by Shell in resident list BEFORE of searching on any external device. Only PURE programs can run as resident, see ARP 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.

1.72 return

Usage : return [n]
 Example : return 10

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

1.73 rm

Equivalent to delete.

1.74 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 specific 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
```

1.75 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. Under 1.3, \$_newproc will hold the cli number of the new process. See also rback.

1.76 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.

1.77 rxsend

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

The option -r sets the variable _result to the result string of the AREXX command.

The option -l send the whole line as *one* command.

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

1.78 search

Usage : search [-abcefnqrvw] 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 pattern 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 directory are recursively searched.
- v (verbose) shows each file name on a single line. this is automatically 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 ARP 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 #.

1.79 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`.

1.80 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.

1.81 sleep

Usage : sleep timeout

Example : sleep 10

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

1.82 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.

1.83 stack

Usage : stack [number]

Example : stack 8000

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

1.84 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'.

1.85 strings

Usage : strings file1..fileN minlength

Example : strings c:dir c:list shell 7

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

1.86 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.

1.87 strlen

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

Puts len of string in variable varname.

1.88 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.

1.89 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.

1.90 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'.

1.91 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!
```

1.92 tackon

```
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.

1.93 tail

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

Display last "num" lines of "filename". If num is not specified, 10 is assumed.

1.94 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.

1.95 touch

Usage : touch file1 .. fileN

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

1.96 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.

1.97 type

Equivalent to CAT.

1.98 unalias

Usage : unalias name .. name

Example : unalias vt

Delete aliases..

1.99 uniq

Usage : uniq

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

1.100 unset

Usage : unset name .. name

Example : unset abc

Unset one or more variables. Deletes them entirely.

1.101 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.

1.102 version

Usage : version

Show current version name, & authors.

1.103 waitforport

Usage : waitforport portname [seconds]

Example : waitforport rexx_ced 5

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

1.104 whereis

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

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. whereis -r looks on all drives.

1.105 window

Usage : window [-q][-f][-b][-l][-s] [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

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).

1.106 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).

1.107 variables

```
@{ "_abbrev" " link _abbrev } @{" _except" " link _except } @{" " ←
  _kick2x " link _kick2x } @{" _nobreak " link _nobreak } @{" "_qcd ←
    " link _qcd } @{" _verbose " link _verbose }
@{ "_bground" " link _bground } @{" _failat " link _failat } @{" " ←
  _lasterr " link _lasterr } @{" _noreq " link _noreq } @{" "_rback ←
    " link _rback } @{" _version " link _version }
@{ "_clinumber" link _clinumber } @{" _hilite " link _hilite } @{" "_lcd ←
  " link _lcd } @{" _passed " link _passed } @{" "_rxpath " ←
  link _rxpath }
@{ "_cwd" " link _cwd } @{" _history " link _history } @{" "_man ←
  " link _man } @{" _path " link _path } @{" "_scroll " ←
  link _scroll }
@{ "_debug" " link _debug } @{" _insert " link _insert } @{" " ←
  _maxerr " link _maxerr } @{" _prompt " link _prompt } @{" " ←
  _terminal " link _terminal }
@{ "_every" " link _every } @{" _ioerr " link _ioerr } @{" " ←
  _minrows " link _minrows } @{" _pipe " link _pipe } @{" " ←
  _titlebar " link _titlebar }
```

1.108 _abbrev

If set to 'n', internal commands can no longer be abbreviated.

1.109 `_bground`

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

1.110 `_clinumber`

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

1.111 `_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.

1.112 `_debug`

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

1.113 `_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.

1.114 `_except`

See EXCEPTION

1.115 `_failat`

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

1.116 `_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".

1.117 `_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.

1.118 `_insert`

Sets the default for insert/overtyping 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.

1.119 `_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()`

1.120 `_kick2x`

True if `dos.library V36` could be opened (which means that `kickstart 2.0` is around)

1.121 `_lasterr`

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

1.122 `_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.

1.123 `_man`

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

1.124 `_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.

1.125 `_minrows`

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

1.126 `_nobreak`

If set to some value, disables CTRL-C.

1.127 `_noreq`

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

1.128 `_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"  
-----
```

1.129 `_path`

Tells CShell where to look for executable files. The current directory and the AmigaDOS path will be searched first. The trailing slash for directories is not necessary anymore. 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)

1.130 `_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
%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'
%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).

1.131 `_pipe`

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

1.132 `_qcd`

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

1.133 `_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.

1.134 `_rxpath`

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

1.135 `_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`.

1.136 `_terminal`

Indicates whether or not shell was started in terminal mode.

1.137 `_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.

1.138 `_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 highlight the debug output. Any combination allowed: `set _verbose sah`

1.139 `_version`

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

1.140 `functions`

```
@{ "@abbrev " link @abbrev } @{"@delword " link @delword } @{" " ←
  @freeblks " link @freeblks } @{"@min " link @min } @{"@split ←
    " link @split } @{"@wincols " link @wincols }
@{"@abs " link @abs } @{"@delwords " link @delwords } @{" " ←
  @freestore" link @freestore } @{"@mix " link @mix } @{"@strcmp ←
    " link @strcmp } @{"@winheight" link @winheight }
@{"@age " link @age } @{"@dirname " link @dirname } @{" " ←
  @getenv " link @getenv } @{"@mounted " link @mounted } @{" " ←
  @strhead " link @strhead } @{"@winleft " link @winleft }
```

```

@{ "@appsuff " link @appsuff } @{ "@dirs " link @dirs } @{ " ←
  @getclass " link @getclass } @{ "@nameext " link @nameext } @{ " ←
  @strleft " link @strleft } @{ "@winrows " link @winrows }
@{ "@arg " link @arg } @{ "@dirstr " link @dirstr } @{ " ←
  @howmany " link @howmany } @{ "@nameroot " link @nameroot } @{ "@strmid ←
  " link @strmid } @{ "@wintop " link @wintop }
@{ "@availmem " link @availmem } @{ "@drive " link @drive } @{ "@index ←
  " link @index } @{ "@opt " link @opt } @{ "@strright " ←
  link @strright } @{ "@winwidth " link @winwidth }
@{ "@basename " link @basename } @{ "@drives " link @drives } @{ "@info ←
  " link @info } @{ "@pathname " link @pathname } @{ "@strtail " ←
  link @strtail } @{ "@without " link @without }
@{ "@center " link @center } @{ "@exists " link @exists } @{ " ←
  @intersect" link @intersect } @{ "@pickargs " link @pickargs } @{ " ←
  @subfile " link @subfile } @{ "@word " link @word }
@{ "@checkport" link @checkport } @{ "@fileblks " link @fileblks } @{ "@ioerr ←
  " link @ioerr } @{ "@pickopts " link @pickopts } @{ "@subwords " ←
  link @subwords } @{ "@words " link @words }
@{ "@clinum " link @clinum } @{ "@filelen " link @filelen } @{ " ←
  @lookfor " link @lookfor } @{ "@rnd " link @rnd } @{ "@tackon ←
  " link @tackon }
@{ "@complete " link @complete } @{ "@fileprot " link @fileprot } @{ "@lower ←
  " link @lower } @{ "@rpn " link @rpn } @{ "@trim " ←
  link @trim }
@{ "@concat " link @concat } @{ "@filereq " link @filereq } @{ "@match ←
  " link @match } @{ "@scrheight" link @scrheight } @{ "@unique " ←
  link @unique }
@{ "@confirm " link @confirm } @{ "@files " link @files } @{ "@max ←
  " link @max } @{ "@scrwidth " link @scrwidth } @{ "@union " ←
  link @union }
@{ "@console " link @console } @{ "@flines " link @flines } @{ "@megs ←
  " link @megs } @{ "@sortargs " link @sortargs } @{ "@upper " ←
  link @upper }
@{ "@dectohex " link @dectohex } @{ "@freebytes" link @freebytes } @{ " ←
  @member " link @member } @{ "@sortnum " link @sortnum } @{ "@volume ←
  " link @volume }

```

1.141 @abbrev

```

@abbrev( str1 str2 [len] )
true if the first <len> chars of str1 are an abbreviation of str2

```

1.142 @abs

```

@abs( num )
returns absolute value of <num>

```

1.143 @age

`@age(file)`
the age of that file in days, 99999 if file not found

1.144 @appsuff

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

1.145 @arg

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

1.146 @availmem

`@availmem([type])`
returns free 'chip', 'fast' or otherwise total memory

1.147 @basename

`@basename(path ... path)`
returns the file name parts of the paths

1.148 @center

`@center(word len)`
returns a string of length <len> with <word> centered in it

1.149 @checkport

`@checkport(portname)`
indicates if given port exists

1.150 @clinum

`@clinum(procname)`
returns the number of the cli identified by a name or a number

1.151 @complete

@complete(abbrev word ... word)
returns the first word <abbrev> is an abbreviation of

1.152 @concat

@concat(word word ... word)
concat all words in one blank separated string, see @split

1.153 @confirm

@confirm(title item ... item)
asks for confirmation of every item and returns the confirmed ones

1.154 @console

@console(STDIN|STDOUT)
tells whether stdin or stdout are interactive (not redirected)

1.155 @dectohex

@dectohex(number)
returns a string representing <number> in hex

1.156 @delword

@delword(word word ... word n)
returns a string with the n-th word deleted.

1.157 @delwords

@delwords(word word ... word n m)
deletes the next m words from the n-th.

1.158 @dirname

@dirname(path)
strips the base name from a path, just returns the directory

1.159 @dirs

@dirs(name name name name)
 returns the directories among the given file names, see @files

1.160 @dirstr

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

1.161 @drive

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

1.162 @drives

@drives()
 outputs all available drives

1.163 @exists

@exists(file)
 tells whether a file exists or not

1.164 @fileblks

@fileblks(file file ... file)
 returns the # of blocks needed for the files, incl. dir blocks

1.165 @filelen

@filelen(file file ... file)
 count the total number of bytes of the given files

1.166 @fileprot

@fileprot(file)
 returns a string like ---arwed

1.167 @filereq

@filereq(title path&pattern filename)
brings up the ARP file requester and returns the selected file name

1.168 @files

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

1.169 @flines

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

1.170 @freebytes

@freebytes(path)
the number of free bytes on the given path

1.171 @freeblks

@freeblks(path)
the number of free blocks on the given path

1.172 @freestore

@freestore(path)
the amount of free store on that path, given in K, M and G

1.173 @getenv

@getenv(varname)
returns the value of the named env: variable

1.174 @getclass

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

1.175 @howmany

@howmany()
indicates the # of shells running

1.176 @index

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

1.177 @info

@info(path)
the corresponding line from the 'info' command, each entry a word

1.178 @intersect

@intersect(word1 word2 word3 , word4 word5 word6)
returns all words which are in both lists. see @union, @member

1.179 @ioerr

@ioerr(num)
returns the corresponding error string to num

1.180 @lookfor

@lookfor(file paths)
looks for a file in the current directory and the paths. See \$_path

1.181 @lower

@lower(word ... word)
lowercases its arguments. see @upper

1.182 @match

@match(word ... word "pattern")
returns the words in the list that match the ARP-pattern

1.183 @max

```
@max( num num ... num )  
computes the maximum of all given numbers
```

1.184 @megs

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

1.185 @member

```
@member( word1 word word ... word )  
tells you if word1 is among the remaining words
```

1.186 @min

```
@min( num num ... num )  
computes the minimum of all given numbers
```

1.187 @mix

```
@mix( arg1 ... argn )  
randomly mixes its arguments
```

1.188 @mounted

```
@mounted( device )  
returns a boolean indicating whether the specified device is mounted
```

1.189 @nameext

```
@nameext( filename )  
returns all after the last dot of <filename>.
```

1.190 @nameroot

```
@nameroot( filename )  
returns all before the LAST dot of <filename>.
```

1.191 @opt

@opt(arg ... arg)
see @pickopts()

1.192 @pathname

@pathname(path)
obsolete. use @dirname

1.193 @pickargs

@pickargs(arg ... arg)
picks of its arguments those which don't start with a '--'

1.194 @pickopts

@pickopts(arg ... arg)
picks of its arguments those which start with a '--'

1.195 @rnd

@rnd()
returns a 32 bit random number

1.196 @rpn

@rpn(expression)
computes the rpn expression. See rpn command

1.197 @scrheight

@scrheight()
outputs the current height of the screen the shell is running in

1.198 @scrwidth

@scrwidth()
outputs the current width of the screen the shell is running in

1.199 @sortargs

```
@sortargs( name ... name )  
sorts its arguments alphabetically
```

1.200 @sortnum

```
@sortnum( number ... number )  
sorts its arguments numerically
```

1.201 @split

```
@split( string )  
makes each blank separated part of @string a word, see @concat
```

1.202 @strcmp

```
@strcmp( name name )  
returns -1, 0 or 1 depending of alphabetical comparison
```

1.203 @strhead

```
@strhead( breakchar string )  
see strhead command
```

1.204 @strleft

```
@strleft( string number )  
see strleft command
```

1.205 @strmid

```
@strmid( string n1 n2 )  
see strmid command
```

1.206 @strright

```
@strright( string n )  
see strright command
```

1.207 @strtail

@strtail(breakchar string)
see strtail command

1.208 @subfile

@subfile(varname n m)
like @subwords, but acts on a readfile-file and is faster

1.209 @subwords

@subwords(word ... word n m)
returns the next m words word of the given list starting from n

1.210 @tackon

@tackon(path file)
see tackon command

1.211 @trim

@trim(word word word)
removes all leading and trailing blanks from the words

1.212 @unique

@unique(word ... word)
sorts the arguments and makes each of them unique

1.213 @union

@union(name ... name , name ... name)
returns all names that are in either list. See @intersect, @member

1.214 @upper

@upper(word ... word)
upper cases the given words. see @lower

1.215 @volume

```
@volume( path )  
returns the volume name in that path or ""
```

1.216 @wincols

```
@wincols( )  
returns the number of columns in the current shell window
```

1.217 @winheight

```
@winheight( )  
outputs the height of your window in pixels
```

1.218 @winleft

```
@winleft( )  
returns the left edge of your window
```

1.219 @winrows

```
@winrows( )  
returns the number of lines in the current shell window
```

1.220 @wintop

```
@wintop( )  
returns the top edge of your window
```

1.221 @winwidth

```
@winwidth( )  
outputs the width of your window in pixels
```

1.222 @without

```
@without( name ... name , name ... name )  
returns all names of list 1 that are not in list 2
```

1.223 @word

```
@word( name ... name n )  
picks the n-th word from the list.
```

1.224 @words

```
@words( name ... name )  
returns the number of words in the list.
```