1 Constructs

List is any sequence of commands separated by **;** or **newline**, which are always interchangeable.

if *list*; **then** *list*
[**elif** *list*; **then** *list*] ...
[**else** *list*]
fi

for *name* [**in** *word* ...]
do *list*
done

for *name* **in** *word* ...; { list }

foreach *name* (*word* ...)

list
end

while *list*; **do** *list*; **done**

until *list*; **do** *list*; **done**

repeat *word*; **do** *list*; **done**

repeat *word sublist*

case *word* **in**
[*pattern*) *list* ;;] ...
esac

case *word* { [*pattern*) *list* ;;] ... }

select *name* [**in** *word* ...]; **do** *list*; done

Subshell: (

Current she

funct

word

word

time

Condition:

Other c
NO_SH
and sho

2 Globbing

See also options **GLOB**, **EXTENDED_GLOB**, **KSH_GLOB**, **NULL_GLOB**, **NOMATCH**, **SH_GLOB GLOB_DOTS**. `X`, of previous characters; those and `~`, `^` require **EXTENDED_GLOB**.

`*`	Any string
`?`	Any character
`[...]`	Any of the enclosed characters
`[[:X:]]`	Character classes where X may be:
`alnum`	Alphanumeric,
`alpha`	Alphabetic,
`blank`	Space or tab,
`cntrl`	Control character,
`digit`	Decimal digit,
`graph`	Printable non-whitespace character,
`lower`	Lowercase character,
`print`	Printable character,
`punct`	Printable, not alnum or space,
`space`	Whitespace character,
`upper`	Uppercase character,
`xdigit`	Hexadecimal digit.
	Above use locales, may be combined with other characters e.g. `[-+[:xdigit:]]`
`[^...]`	Any character except those enclosed
`<x-y>`	Any number between x and y inclusive: both optional, defaults 0, ∞
`^X`	Anything not matching `X`
`(X|Y)`	Either `X` or `Y`
`X~Y`	Pattern `X`, but not `Y`
`(X|Y~Z)`	Either `X` or (`Y` but not `Z`)
`X#`	Zero or more occurences of `X`
`X##`	One or more occurences of `X`
`(X)`	Grouping of (part of) pattern.
`**/`	(As path segment) short for `(*/)`: match all subdirectories
`***/`	The same, following symbolic links

Globbing flags appear in the form `(#X)` and require the `EXTENDED_GLOB` option. They may appear in groups. `X` may be:

`i`	Match case insensitively
`l`	Lower case matches upper case
`I`	
`b`	
`B`	
`m`	
`M`	
a*num*	
`s`	
`e`	

Globbin
(usually

`/`	
`.`	
`@`	
`=`	
`p`	

*	executable plain file (0100)		or set $reply to file array	,
%	device file (character or block)	d*dev*	on device number *dev*	-
%b	block special	l[-|+]*ct*	link count *ct* or less (+) or more (-) than *ct*	M
%c	character special	U	owned by current effective uid	T
r	readable (0400)	G	owned by current effective gid	N
w	writable (0200)	u*uid*	owned by uid *uid*; may also take forms	D
x	executable (0200)		.*name*., !*name*!, ... or	n
A	group-readable (0040)		(*name*), {*name*}, ...	o[nLlam
I	group-writable (0020)	g*gid*	owned by *gid*, as for u*uid*.	
E	group-executable (0010)	a[Mwhm][-|+]*n*		
R	world-readable (0200)		accessed (less than, more than) *n* days	
W	world-writable (0200)		(months, weeks, hours, minutes) ago	o[nLlam
X	world-executable (0200)	m[Mwhm][-|+]*n*		
s	setuid (04000)		modified ditto	[*beg*[,*e*
S	setgid (02000)	c[Mwhm][-|+]*n*		
t	files with the sticky bit (01000)		inode changed ditto	:...
f*spec*	chmod-like access permissions	L[kKmMpP][-|+]*n*		
	e.g. f70? or f:u+w,go-w:		size in bytes (or kb, mb, blocks) = (or <, >) *n*	
e*str*	eval *str*, use file ($REPLY) if status 0	^	negate following qualifiers	

3 Options

†means set by default: these options appear with no in front in option listings; +*o* turns single-letter option off (shown in parenthe

ALL_EXPORT	Export all new shell params (-a)	BEEP†	Beep on errors etc. (+B)	CSH_NUL
ALWAYS_LAST_PROMPT	Back to prompt after list	BG_NICE†	Lower priority of bg jobs (-6)	DVORAK
ALWAYS_TO_END	End of word after completion	BRACE_CCL	foo{ab} → fooa foob	EQUALS†
APPEND_HISTORY	Append history to file	BSD_ECHO	Builtin echo works like in BSD	ERR_EXI
AUTO_CD	Directory as command does cd (-J)	CDABLE_VARS	cd foo like cd ~foo (-T)	EXEC†
AUTO_LIST	List on ambiguous completion (-9)	CHASE_DOTS	Resolve links when .. in dir	EXTENDE
AUTO_MENU	Menu after second TAB	CHASE_LINKS	Resolve symlinks in directories (-w)	EXTENDE
AUTO_NAME_DIRS	Params with paths become names	CHECK_JOBS†	Report job status at exit	FLOW_CO
AUTO_PARAM_KEYS	Clever del after param completion	CLOBBER†	> to existing file needs >| (+C)	FUNCTIO
AUTO_PARAM_SLASH	$path<TAB> → $path/	COMPLETE_ALIASES	Completion uses unexpanded aliases	GLOB†
AUTO_PUSHD	Make cd act like pushd (-N)	COMPLETE_IN_WORD	Complete at cursor point in word	GLOBAL_
AUTO_REMOVE_SLASH	Strip slash after completion	CORRECT	Correct command spelling (-0)	GLOBAL_
AUTO_RESUME	cmd can behave like %cmd (-W)	CORRECT_ALL	Correct spelling of all args (-O)	GLOB_AS
BAD_PATTERN†	Error on bad glob pattern (+2)	CSH_JUNKIE_HISTORY	Single ! is last command	GLOB_CO
BANG_HIST†	Use !hist on cmd line (+K)	CSH_JUNKIE_LOOPS	Lists can be list; end	GLOB_DO
BARE_GLOB_QUAL†	Use glob quals with just parens	CSH_JUNKIE_QUOTES	No unescaped newlines in quotes	GLOB_SU
BASH_AUTO_LIST†	List only on second tab	CSH_NULLCMD	Don't use $NULLCMD, $READNULLCMD	HASH_CM

HASH_DIRS† Hash directory when cmd runs
HASH_LIST_ALL† Hash all cmds on completion
HIST_ALLOW_CLOBBER Allow clobbering redirects in hist
HIST_BEEP† Beep on bad !-history
HIST_EXPIRE_DUPS_FIRST
Trim duplicate lines to squeeze history
HIST_FIND_NO_DUPS Never show duplicates in history
HIST_IGNORE_ALL_DUPS
Never save duplicate of existing hist entry
HIST_IGNORE_DUPS No adjacent duplicates in history (-h)
HIST_IGNORE_SPACE ‘ cmd’ lines not saved (-g)
HIST_NO_FUNCTIONS Don’t store function definitions
HIST_NO_STORE No history commands in history
HIST_REDUCE_BLANKS Trim excess whitespace in history
HIST_SAVE_NO_DUPS Trim duplicates if saving history
HIST_VERIFY Edit after ! expansion
HUP† Send SIGHUP to jobs on exit
IGNORE_BRACES No {...,...} expansion (-I)
IGNORE_EOF No exit on first ten eof’s (-7)
INC_APPEND_HISTORY Save history as it happens
INTERACTIVE Shell is interactive (not settable) (-i)
INTERACTIVE_COMMENTS
Use comments interactively (-k)
KSH_ARRAYS Array syntax more like ksh
KSH_AUTOLOAD Emulate ksh function loading
KSH_GLOB Emulate ksh patterns, *(...) etc.
KSH_OPTION_PRINT Print options like ksh does
LIST_AMBIGUOUS Only list ambiguous completions
LIST_BEEP Beep on ambiguous completion
LIST_PACKED Squeeze completion listings
LIST_ROWS_FIRST List rows first in completion
LIST_TYPES File types in completion list (-X)
LOCAL_OPTIONS Options set in functions are local
LOCAL_TRAPS Reset traps on leaving func
LOGIN Shell is login (not settable) (-l)
LONG_LIST_JOBS Always use jobs -l (-R)
MAGIC_EQUAL_SUBST Any var=expr file-expands expr
MAIL_WARNING Warn if mail file accessed (-U)
MARK_DIRS Append / to globbed directories (-8)
MENU_COMPLETE Cycle completions on TAB (-Y)
MONITOR Allow job control (-m)
MULTIOS† Implicitly tee/cat multiple <, >
NOMATCH† Error on unmatched globs (+3)
NOTIFY† Report bg jobs on change (-5)
NULL_GLOB Remove unmatched globs (-G)
NUMERIC_GLOB_SORT Numbers sorted in glob
OCTAL_ZEROES 0 introduces octal in math expn
OVER_STRIKE Editor starts in overstrike mode
PATH_DIRS Search path for dir/cmd (-Q)
POSIX_BUILTINS builtin command is specialer
PRINT_EIGHT_BIT Show chars with high bit in listings
PRINT_EXIT_VALUE Show non-zero exit status (-1)
PRIVILEGED Privileged mode: safety first (-p)
PROMPTT_BANG ! is special in prompts
PROMPT_CR† Print CR just before prompt (+V)
PROMPT_PERCENT† Do % expansions in prompt
PROMPT_
PUSHD_I
PUSHD_M
PUSHD_S
PUSHD_T
RC_EXPA
RC_QUOT
RCS†
REC_EXA
RESTRIC
RM_STAR
RM_STAR
SHARE_H
SH_FILE
SH_GLOB
SHIN_ST
SH_NULL
SH_OPTI
SHORT_L
SH_WORD
SINGLE_
SINGLE_
SUN_KEY
UNSET†
VERBOSE
XTRACE
ZLE

4 Parameter expansion

$*name*
${*name*}
Basic parameter substitution
${+*name*}
1 if *name* set, 0 otherwise
${*name*:-*word*}
$*name* if non-null, else *word*
${*name*-*word*}
$*name* if set, else *word*
(Similar for others with/without colon.)
${*name*:=*word*}
$*name* if non-null, else use *word*
and set *name* to that
${*name*:=*word*}
Unconditional assignment ${*name*:?*word*}
$*name* if non-null, else print *word* and exit
${*name*:+*word*}
word if $*name* non-null, else nothing
${*name*
${*name*
$*na*
pat
glo
${*name*
${*name*
As
${*name*

Substitute longest match of *pattern* by *repl*

`${(S)`*name*`/`*pattern*`/`*repl*`}`

Substitute shortest match

`${`*name*`//`*pattern*`/`*repl*`}`

Substitute all non-overlapping longest matches

`${`*name*`/#`*pattern*`/`*repl*`}`

Subst if *pattern* at start of string

`${`*name*`/%`*pattern*`/`*repl*`}`

Subst if *pattern* at end of string

`${`*name*`:/`*pattern*`/`*repl*`}`

Subst if *pattern* matches entire string

`${#`*spec*`}`

Count length of scalar or words of array

`${^`*spec*`}`

`${^^`*spec*`}`

Turn on (off) **RC_EXPAND_PARAM**

`${=`*spec*`}`

`${==`*spec*`}`

Turn on (off) **SH_WORD_SPLIT**

`${~`*spec*`}`

`${~~`*spec*`}`

Turn on (off) **GLOB_SUBST**

`${`*spec*`:`*mod*`}`

Apply history modifier *mod*

`${${name\dots}...}`

Perform both sets of modifications on value
N.B. does not do extra lookup, see `(P)`

Flags: usage `${(o)`*name*`}` etc.

`A` `${...:=...}` creates array
`AA` ... creates associative array
`@` Split into words in double quotes
`e` Use shell expansion on result
`P` Force `$`*name* to be re-used as name
`o` sort words in ascending order
`O` sort words in descending order
`i` case-independent with `o` or `O`
`L` all letters lower case
`U` all letters upper case
`C` capitalise words
`V` make special characters visible
`q` quote result with `\`
`qq` quote result with `'`
`qqq` quote result with `"`
`qqqq` quote result with `$'...'`
`Q` remove one level of shell quoting
`%` Expand prompt escapes
`%%` Expand as prompt with current settings
`X` Report parse errors with quotes, patterns
`c` `${#`*name*`}` counts characters
`w` `${#`*name*`}` counts words
`W` As `w`, but count empty words
`k` With assoc include keys
`v` With assoc include values
`p` Use print escapes in args below
`F` Join words with newlines
`f` Split on newlines
`z` Split using ordinary parsing
`t` Subsituted description, not value

Flags with delimiters; use any pair of chars in place of colon, also matched `<>`, `()`, `{}`, `[]`

`l:`*expr*`::`*string1*`::`*string2*`:`
Pad words on left to *expr* chars using *string1* repeated (default space), *string2* appears just once

`r:`*expr*`::`*string1*`::`*string2*`:`
Ditto padded on right

`j:`*string*`:`
Join words using *string* (occurs before splitting)

`s:`*string*`:`
Split words at *string*

Flags applying with `${...#...}` or `${...%...}`

`S` search substrings too
`I:`*expr*`:`
Sea
`M` Incl
`R` Incl
`B` Incl
`E` Incl
`N` Incl

Summar

1 Nes
2 Sub
3 `(P)`
4 `"${`
5 Nes
6 `#`, `%`
7 `(j)`
8 `(s)`
9 She
10 `(e)`
11 `(l)`

Flags in

`e` Bac
`w` Ind
`s:strir`
Sep
`p` Use
`f` Ind
`r` Rev
For
`R` As
`k` In a
`K` In a
`i` As
For
`I` As
`n:`*expr*`:`
Use
`r`, `R`

5 History

See also parameters **histchars**, **HISTFILE**, **HISTSIZE**, **SAVEHIST** and options **APPEND_HISTORY**, **CSH_JUNKIE_H**
HIST_ALLOW_CLOBBER, **HIST_IGNORE_DUPS**, **HIST_IGNORE_SPACE**, **HIST_NO_STORE**, **HIST_VERIFY**,
HIST_EXPIRE_DUPS_FIRST, **HIST_FIND__NO_DUPS**, **HIST_IGNORE_ALL_DUPS**, **HIST_NO_FUNCTIONS**, **H**
HIST_SAVE_NO_DUPS, **INC_APPEND_HISTORY**, **SHARE_HISTORY**.
Events:

`!`	start history substitution unless after space, newline, `=`, `(`	`!-`*n*	line *n* before current	`!{...}`
		`!`*str*	last line beginning with *str*	`!"`
`!!`	immediately previous command	`!?`*str*`[?]`	last line containing *str*	
`!`*n*	command line *n*	`!#`	current command so far	

Words: separated from event by ':'

`0`	first word on line (command)	`%`	word matched by `?s`	*x*`*`
n	*n*th argument of command	*x*`-`*y*	range of words	*x*`-`
`^`	first argument of command	`-`*y*	same as `0-`*y*	
`$`	last argument of command	`*`	all arguments	

Modifiers: also with globbing and parameters

`h`	(head) strip last path cpt	`Q`	remove one level of quotes	`f`
`r`	remove suffix `.`*suf*	`x`	same but split words at space	`F:`*expr*`:`
`e`	leave only suffix *suf*	`l`	all letters lower case	`w`
`t`	(tail) leave only last path cpt	`u`	all letters upper case	`W:`*sep*`:`
`&`	repeat last substitution	`s/`*old*`/`*new*`[/]`		
`p`	don't execute new command		replace *old* by *new* (string)	
`q`	quote words from further subst	`g`	(before s) change every occurrence	

6 Parameters

Special parameters: arrays are lower case except `status`; those marked† are assignable:

`!`	Last bg PID
`ARGC`	
`#`	Pos. param count
`$`	Current PID
`-`	Shell flags set
`argv`†	
`*`†	Pos. params as array
`@`	Same as `argv[@]`

`status`	
`?`	Last prog status
`pipestatus`	Array of statuses for pipeline
`_`	Last arg of prev cmd
`CPUTYPE`	CPU determined at run time
`EGID`†	Effective GID
`EUID`†	Effective UID
`ERRNO`	System error no.
`GID`†	Current GID
`HOST`	Current host name

`LINENO`
`LOGNAME`
`MACHTYP`
`OLDPWD`
`OPTARG`
`OPTIND`
`OSTYPE`
`PPID`
`PWD`
`RANDOM`†

SECONDS†	Seconds since start of shell
SHLVL	Incremented for each zsh
signals	Names of signals
TTY	Name of shell terminal
TTYIDLE	Idle time of tty (secs.) or -1
UID†	UID
USERNAME†	username
VENDOR	Machine manufacturer
ZSH_NAME	Shell invocation name
ZSH_VERSION	ID of zsh version

Other parameters used by shell (†colon-separated path)

ARGV0	Export to change argv[0]
BAUD	Line speed (zero to ignore)
cdpath, CDPATH†	Directories search for **cd** command
COUMNS	No. of columns on terminal
DIRSTACKSIZE	Max size of dir. stack
FCEDIT	Default editor for **fc** cmd.
fignore, FIGNORE†	Suffixes ignored for completion
fpath, FPATH†	Path to search for autoload fns.
histchars	three chars: 1) start of history (!), 2) quick history sub (^), 3) comment (#)
HISTCHARS	same as histchars
HISTFILE	Where to save shell history
HISTSIZE	Max history lines internally
HOME	Default target for **cd** cmd.
IFS	Word seperators for input
KEYTIMEOUT	Time to waits for key in sequence
LANG	General locale setting
LC_ALL	Overrides LANG and other LC_*
LC_COLLATE	Determines character ordering
LC_CTYPE	Determines types of characters
LC_MESSAGES	For messages: not used by zsh
LC_NUMERIC	For decimal point, number separator
LC_TIME	Date and time format
LINES	No. of lines on terminal
LISTMAX	No. of files to list without asking
LOGCHECK	How often to check watch (secs.)
MAIL	File to check for mail
MAILCHECK	How often to check MAIL (secs.)
mailpath, MAILPATH†	List of files to check for new mail. Can follow each with ?'message to print'
manpath, MANPATH†	Not used by shell, probably used by **man** cmd.
module_path, MODULE_PATH†	Path for dynamic modules; not imported
NULLCMD	Used for redirs. with no cmd.
path, PATH†	Where to search for commands
POSTEDIT	Output when line editor exits
PROMPT, prompt	
PS1	
PROMPT2	
PROMPT3	
PROMPT4	
psvar,	
READNUL	
REPORTT	
RPROMPT	
RPS1	
SAVEHIS	
SPROMPT	
STTY	
TERM	
TIMEFMT	
TMOUT	
TMPPREF	
watch,	
WATCHFM	
WORDCHA	
ZBEEP	
ZDOTDIR	

Prompt escape sequences: those with †can use integer count *n*, which must immediately follow %. Default is 1 except for %_.

%%	A '%'
%)	A ')'
%d %/†	$PWD
%~†	$PWD, but use ~-abbrevs
%h %!	Current history event no.
%L	The current value of $SHLVL
%M	Full hostname
%m†	Host up to *n*'th dot
%S %B %U	Start standout, bold, underline
%s %b %u	Stop corresponding mode
%t %@	Time in 12 hour format
%T	Time in 24 hour format
%*	Same with seconds
%n	$USERNAME
%N	Name of script, sourced file, function
%i	Line number inside %N
%w	Date as day-dd
%W	Date as mm/dd/yy
%D	Date as yy-mm-dd
%D{*string*}	Use strftime to format *string*
%l	Current tty
%?	
%_†	
%E	
%#	
%v†	
%{...%}	
%<strin	
%c† %.†	
%C	

Codes for ternary expressions in prompts, format `%(`*char*`.`*true-text*`.`*false-text*`)`, integer count *n* may proceded or follow '`(`'. Test is true if:

`c . ~`	Tilde'd path has >= *n* elts
`/ C`	Ditto for absolute path
`t`	Current minute is *n*
`T`	Current hour is *n*
`d`	Current day of month is *n*
`D`	Month is *n* (Jan = 0)
`w`	Weekday is *n* (Sun = 0)
`?`	Last exit status was *n*
`#`	Running as uid *n*
`g`	Running as gid *n*
`L`	`$SHLVL` >= *n*
`S`	`$SECONDS` >= *n*
`v`	`${#psvar}` >= *n*
`_`	At least *n* shell constructs
`!`	True if shell is priveleged

Escape sequences in `$WATCHFMT`:

`%n`	Name of user
`%a`	
`%l`	
`%M`	
`%m`	
`%S %U %`	
`%s %u %`	
`%t %@`	
`%T`	
`%w`	
`%W`	
`%D`	

Ternary expressions in `$WATCHFMT`, format `%(`*char*`.`*true-text*`.`*false-text*`)`, can be used with `l`, `n`, `m` or `M` (true if non-empty value for logout).

7 Conditions

File tests: followed by a file name

Cond	**true if file**
`-a`	exists
`-b`	block special
`-c`	character special
`-d`	directory
`-e`	exists
`-f`	plain file
`-g`	has setgid bit set
`-h`	symbolic link
`-k`	has sticky bit set
`-p`	FIFO/pipe
`-r`	readable
`-s`	has size > 0
`-u`	has setuid bit set
`-w`	writeable
`-x`	executable/dir. readable:
`-L`	symbolic link
`-O`	owned by UID
`-G`	owned by GID
`-S`	socket
`-N`	access time not newer than mod time
`-ot`	file
`-ef`	nam
`=`	
`==`	*stri*
`!=`	...
`<`	AS
`>`	AS
`-eq`	Nu
`-ne`	Nu
`-lt`	Nu
`-gt`	Nu
`-le`	Nu
`-ge`	Nu

Other tests with single argument:

`-n`	string, length > 0
`-o`	option, is set
`-t`	fd, open to tty
`-z`	string, length zero

Two argument tests ([[*a* `test` *b*]]):

`-nt` file *a* newer than *b*

Also gro
handlin