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: (*list*)

Current shell: { *list* }

function *word* [()] ...{ *list* }

word ...() { list }

word ...() *sublist*

time [*pipeline*]

Condition: [[*exp*]]

Other constructs depend on the options **NO_SHORT_LOOPS** and **CSH_JUNKIE_LOOPS** and should be avoided in scripts.

2 Globbing

See also options **GLOB**, **EXTENDED_GLOB**, **KSH_GLOB**, **NULL_GLOB**, **NOMATCH**, **SH_GLOB GLOB_DOTS**. `X`, `Y`, ... are any pattern. `#` and `##` require grouping 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`	Case sensitive: cancel `i` and `I`
`b`	Activate backreferences for parentheses. `$match`, `$mbegin`, `$mend` arrays give matched string, beginning/end indices
`B`	Deactivate backreferences, negating `b`
`m`	Set `$MATCH`, `$MBEGIN`, `$MEND` for string
`M`	Deactivate `m`.
`a`*num*	Allow *num* errors in matches (0 to turn off)
`s`	Match only at start of string (use in param expn)
`e`	Match only at end of string

Globbing modifiers appear in parentheses after a pattern (usually *and*'ed):

`/`	directory
`.`	plain file
`@`	symbolic link
`=`	socket
`p`	named pipe (FIFO)

`*`	executable plain file (0100)
`%`	device file (character or block)
`%b`	block special
`%c`	character special
`r`	readable (0400)
`w`	writable (0200)
`x`	executable (0200)
`A`	group-readable (0040)
`I`	group-writable (0020)
`E`	group-executable (0010)
`R`	world-readable (0200)
`W`	world-writable (0200)
`X`	world-executable (0200)
`s`	setuid (04000)
`S`	setgid (02000)
`t`	files with the sticky bit (01000)
`f`*spec*	chmod-like access permissions e.g. `f70?` or `f:u+w,go-w:`
`e`*str*	eval *str*, use file (`$REPLY`) if status 0 or set `$reply` to file array
`d`*dev*	on device number *dev*
`l[-|+]`*ct*	link count *ct* or less (`+`) or more (`-`) than *ct*
`U`	owned by current effective uid
`G`	owned by current effective gid
`u`*uid*	owned by uid *uid*; may also take forms `.`*name*`.`, `!`*name*`!`, ... or `(`*name*`)`, `{`*name*`}`, ...
`g`*gid*	owned by *gid*, as for `u`*uid*.
`a[Mwhm][-|+]`*n*	accessed (less than, more than) *n* days (months, weeks, hours, minutes) ago
`m[Mwhm][-|+]`*n*	modified ditto
`c[Mwhm][-|+]`*n*	inode changed ditto
`L[kKmMpP][-|+]`*n*	size in bytes (or kb, mb, blocks) = (or <, >) *n*
`^`	negate following qualifiers
`,`	'*or*' lists of qualifers together
`-`	toggle following links (off by default)
`M`	set **MARK_DIRS**, this pattern only
`T`	set **LIST_TYPES**, this pattern only
`N`	set **NULL_GLOB**, this pattern only
`D`	set **GLOB_DOTS**, this pattern only
`n`	set **NUMERIC_GLOB_SORT**, this pattern only
`o[nLlamcd]`	sort order of resulting files: by name, size, no. of links, access/modification/inode time, depth-first order
`o[nLlamcd]`	same but reversed order; `Od` depth-last
`[`*beg*`[,`*end*`]]`	Index of matched file(s) to select
`:...`	remainder treated as history modifiers (each with own `:`)

3 Options

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

ALL_EXPORT	Export all new shell params (-a)
ALWAYS_LAST_PROMPT	Back to prompt after list
ALWAYS_TO_END	End of word after completion
APPEND_HISTORY	Append history to file
AUTO_CD	Directory as command does `cd` (-J)
AUTO_LIST	List on ambiguous completion (-9)
AUTO_MENU	Menu after second `TAB`
AUTO_NAME_DIRS	Params with paths become names
AUTO_PARAM_KEYS	Clever del after param completion
AUTO_PARAM_SLASH	`$path<TAB>` → `$path/`
AUTO_PUSHD	Make `cd` act like `pushd` (-N)
AUTO_REMOVE_SLASH	Strip slash after completion
AUTO_RESUME	`cmd` can behave like `%cmd` (-W)
BAD_PATTERN†	Error on bad glob pattern (+2)
BANG_HIST†	Use `!hist` on cmd line (+K)
BARE_GLOB_QUAL†	Use glob quals with just parens
BASH_AUTO_LIST†	List only on second tab
BEEP†	Beep on errors etc. (+B)
BG_NICE†	Lower priority of bg jobs (-6)
BRACE_CCL	`foo{ab}` → `fooa foob`
BSD_ECHO	Builtin `echo` works like in BSD
CDABLE_VARS	`cd foo` like `cd ~foo` (-T)
CHASE_DOTS	Resolve links when `..` in dir
CHASE_LINKS	Resolve symlinks in directories (-w)
CHECK_JOBS†	Report job status at `exit`
CLOBBER†	`>` to existing file needs `>|` (+C)
COMPLETE_ALIASES	Completion uses unexpanded aliases
COMPLETE_IN_WORD	Complete at cursor point in word
CORRECT	Correct command spelling (-0)
CORRECT_ALL	Correct spelling of all args (-O)
CSH_JUNKIE_HISTORY	Single `!` is last command
CSH_JUNKIE_LOOPS	Lists can be `list; end`
CSH_JUNKIE_QUOTES	No unescaped newlines in quotes
CSH_NULLCMD	Don't use `$NULLCMD`, `$READNULLCMD`
CSH_NULL_GLOB	Only one glob must match
DVORAK	Use Dvorak keyboard for spelling
EQUALS†	Perform `=cmd` expansion
ERR_EXIT	Exit shell on error (-e)
EXEC†	Execute commands (+n)
EXTENDED_GLOB	Use `#`, `~` and `^` in patterns
EXTENDED_HISTORY	Save timestamp to history file
FLOW_CONTROL†	`^S`, `^Q` do flow control
FUNCTION_ARGZERO†	Set `$0` on function or source
GLOB†	Perform globbing (+F)
GLOBAL_EXPORT†	`typeset -x` applies globally
GLOBAL_RCS†	Use `/etc` startup files
GLOB_ASSIGN	`scalar=*` globs on right
GLOB_COMPLETE	Complete globbing with menu
GLOB_DOTS	Leading dots match wildcards (-4)
GLOB_SUBST	Text from params can glob
HASH_CMDS†	Hash commands when run

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_SUBST	Expand substitutions in prompts
PUSHD_IGNORE_DUPS	Only one instance of dir on stack
PUSHD_MINUS	Swap plus and minus in pushd
PUSHD_SILENT	Don’t print directory stack (-E)
PUSHD_TO_HOME	With no args, pushd goes home (-D)
RC_EXPAND_PARAM	A${array}Z → Aa1Z Aa2Z ... (-P)
RC_QUOTES	echo '''' → '
RCS†	.-files, else just /etc/zshenv (+f)
REC_EXACT	Prefer exact match in completion (-S)
RESTRICTED	Can’t cause as much damage
RM_STAR_SILENT	No query on rm * (-H)
RM_STAR_WAIT	Don’t believe first RMSTAR reply
SHARE_HISTORY	Read/write history as it happens
SH_FILE_EXPANSION	Perform ~file, =cmd first
SH_GLOB	Disable (, |,), < in patterns
SHIN_STDIN	Read commands from stdin (-s)
SH_NULLCMD	Null commands assume : behaviour
SH_OPTION_LETTERS	Letter options work like in ksh
SHORT_LOOPS†	Short verb!for!, select, if, function
SH_WORD_SPLIT	Split words like lesser shells do (-y)
SINGLE_COMMAND	Read a command and exit (-t)
SINGLE_LINE_ZLE	Editor only uses one line (-M)
SUN_KEYBOARD_HACK	Ignore unmatched trailing ‘ (-L)
UNSET†	Unset parameters cause error (+u)
VERBOSE	Print input lines as read (-v)
XTRACE	Print cmds and args when run (-x)
ZLE	Use the shell’s line editor (-Z)

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*#*pattern*}
${*name*##*pattern*}
    $*name* with shortest (longest) match of *pattern* removed from head. Patterns as globbing; original parameter unchanged
${*name*%*pattern*}
${*name*%%*pattern*}
    As for #, but remove from tail of match
${*name*/*pattern*/*repl*}

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*:
Search/substitute *expr*th match

M	Include matched portion
R	Include unmatched portion (Rest)
B	Include index of beginning
E	Include index of end
N	Include length of match

Summary of rules for substitution

1	Nested substitution, ${${...}}
2	Subscript of parameter by name, ${*name*[*i*]}
3	(P) flag
4	"${...}" joining
5	Nested subscript, ${${...}[*i*]}
6	#, %, /. : modifications
7	(j) flag or space joining
8	(s), (f), (z) or = splitting
9	Shell word splitting (no flags)
10	(e) flag
11	(l) or (r) padding

Flags in indexing: usage $*name*[(i)*index*] etc.

e Backward compatability only
w Index by words of scalar
s:string:
Separate words with string
p Use print escapes in following s
f Index by lines: same as pws:\n:
r Reverse index array/substring/word
For assocs, match against values
R As r, but last match (all for assocs)
k In assoc, keys are patterns; get first
K In assoc, keys are patterns; get all
i As r, but return index
For assocs match against keys
I As I, but last match (all for assocs)
n:*expr*:
Use expr'th first/last match b:*expr*:
r, R, i, I start search at *expr*th elt.

5 History

See also parameters **histchars**, **HISTFILE**, **HISTSIZE**, **SAVEHIST** and options **APPEND_HISTORY**, **CSH_JUNKIE_HISTORY**, **EXTENDED_HISTORY**, **HIST_ALLOW_CLOBBER**, **HIST_IGNORE_DUPS**, **HIST_IGNORE_SPACE**, **HIST_NO_STORE**, **HIST_VERIFY**, **BANG_HIST**, **HIST_BEEP**, **HIST_EXPIRE_DUPS_FIRST**, **HIST_FIND__NO_DUPS**, **HIST_IGNORE_ALL_DUPS**, **HIST_NO_FUNCTIONS**, **HIST_REDUCE_BLANKS**, **HIST_SAVE_NO_DUPS**, **INC_APPEND_HISTORY**, **SHARE_HISTORY**.

Events:

`!`	start history substitution unless after space, newline, `=`, `(`
`!!`	immediately previous command
`!`*n*	command line *n*
`!-`*n*	line *n* before current
`!`*str*	last line beginning with *str*
`!?`*str*[`?`]	last line containing *str*
`!#`	current command so far
`!{...}`	insulate history reference
`!"`	no more expansion this line

Words: separated from event by ':'

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

Modifiers: also with globbing and parameters

`h`	(head) strip last path cpt
`r`	remove suffix `.`*suf*
`e`	leave only suffix *suf*
`t`	(tail) leave only last path cpt
`&`	repeat last substitution
`p`	don't execute new command
`q`	quote words from further subst
`Q`	remove one level of quotes
`x`	same but split words at space
`l`	all letters lower case
`u`	all letters upper case
`s/`*old*`/`*new*[`/`]	replace *old* by *new* (string)
`g`	(before s) change every occurrence
`f`	repeat till no further change
`F:`*expr*`:`	same but max *expr* changes
`w`	(as prefix) apply to each word
`W:`*sep*`:`	same but separate words on *sep*

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`	Input line no.
`LOGNAME`	User name
`MACHTYPE`	Machine type
`OLDPWD`	Previous working dir.
`OPTARG`	
`OPTIND`	Value, index of last **getopts** option
`OSTYPE`	OS type
`PPID`	PID of parent proc.
`PWD`	Current working dir.
`RANDOM`†	Random integer: assign to seed.

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	Prompt used by editor
PROMPT2, PS2	Continuation prompt
PROMPT3, PS3	Prompt used by **select** cmd.
PROMPT4 PS4	Execution trace prompt
psvar, PSVAR†	Replace `%v` in prompts
READNULLCMD	Command used with only input readir.
REPORTTIME	Longer commands print usage (secs.)
RPROMPT	
RPS1	Prompt displayed at right of line
SAVEHIST	Max no. of lines in history file
SPROMPT	Prompt used for spelling correction
STTY	Args. to follow **stty**, export to run before external cmd.
TERM	Type of terminal for editing
TIMEFMT	Format of process time reports
TMOUT	SIGALRM if idle this long (secs.)
TMPPREFIX	Path to temp files (**/tmp/zsh**)
watch, WATCH†	List of users to watch log in/out (also **all**, **notme**, `%` tty, `@` host)
WATCHFMT	Format of `watch` reports
WORDCHARS	Non-alphanumeric characters used as part of a word by editor
ZBEEP	Sequence to output instead of beeping
ZDOTDIR	Where to find `.zshrc` etc.

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	`$USERNAME`
%N	Name of script, sourced file, function
%i	Line number inside `%N`
%w	Date as `day-dd`
%W	Date as `mm/dd/yy`
%D	Date as `yy-mm-dd`
%D{*string*}	Use `strftime` to format *string*
%l	Current tty
%?	Return status of last command
%_†	Parser status, *n* for max level
%E	Clear to end of line
%#	`#` if root, else `%`
%v†	*n*'th elementt of `$psvar`
%{...%}	String which does not move cursor
%<string< %>string> %[<string] %[>string]	Truncate `string` on L or R, *n* gives max length.
%c† %.†	Component of `$PWD` (deprecated)
%C	Same but don't expand ~'s

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`	'logged on' or 'logged off'
`%l`	User's tty
`%M`	Full remote host name
`%m`	Host to first '.'
`%S %U %B`	Start standout, underline, boldface
`%s %u %b`	Stop corresponding mode
`%t %@`	Time in 12-hour format
`%T`	Time in 24-hour format
`%w`	Date as `day-dd`
`%W`	Date as `mm/dd/yy`
`%D`	Date as `yy-mm-dd`

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 corresponding `%`), or `a` (true for login, false 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

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
`-ot`	file a older than b
`-ef`	names refer to same file
`=`	
`==`	*string* matches *pattern*
`!=`	...does not match
`<`	ASCII before
`>`	ASCII after
`-eq`	Numbers equal
`-ne`	Numbers unequal
`-lt`	Numeric $a < b$
`-gt`	Numeric $a > b$
`-le`	Numeric $a \leq b$
`-ge`	Numeric $a \geq b$

Also grouping `(`...`)`, negation `!`, and `&&`, or `||`; special handling of **/dev/fd**.