Job Control

allows you to work on several jobs at once, switching back and forth between them as you desire:

Format: **command [optional job number]**

Command	Meaning
bg [%*job*]	execute the job (specified by %*job* - default: current job) in the **b**ack**g**round
fg [%*job*]	run job (specified by %job) in **f**ore**g**round
kill [signal] [%job/PID]	iterminate a process or send a signal. If it fails use **kill -9** [%job/PID]
stop [%*job*]	**stop** the job (specified by %*job*)
suspend	stop the current shell (like for foreground jobs)
notify [%*job*]	**notify** user immediately on change of status of job (specified by %*job*)
jobs [-l]	list active and stopped background **jobs** (with process Ids). Current background job is indicated by '+', next by '-'
<CTRL>-c	terminate foreground job
<CTRL>-z	suspend foreground job

Job Number	Meaning
PID	**p**rocess **ID**
%	current job
%*n*	job *n*
%*str*	job with *str* as command name
%?*str*	job with *str* anywhere in command string
%-	previous job

Example:

```
% jobs
% echo $status
0
% prog1 &
```

```
[1] 5913
% prog2
% -z
suspended
% bg
[2] 5924
% jobs
[1] + 5913 Suspended (tty output) prog1
[2]   5924 Running                prog2
% fg
prog1
bin fertig
% jobs
[2] + 5924 Running                prog2
% %2
prog2
```

C Shell Variables

There are two kinds of C shell variables:

- user defined variables
- predefined variables

User Defined Variables

Variable modifiers apply to the following (replace *var* with **argv** to refer to command line arguments):

variable	meaning
$*var*	value of *var*
${*var*}	value of *var*, insulate *var* string
$*var*[*n*]	value of *n*th word from *var* wordlist
${*var*[*n*]}	value of *n*th word from *var* wordlist, insulate *var* string
$*var*[*]	same as $*var*
$*var*[n-m]	words *n* through *m* from *var* wordlist
${*var*[*n*-*m*]}	words *n* through *m* from wordlist, insulate *var* string
$*var*[$#*var*]	last word from *var* wordlist
$*var*{[$#*var*]}	last word from *var* wordlist, insulate *var* string
$*n*	same as $argv[*n*] (*n* restricted to 1-9)
$*	same as $argv[*]
$#*var*	number of words in *var*
${#*var*}	number of words in *var*, insulate *var* string

Variable modifiers do not apply to the following:

variable	meaning
$0	name of script file
$?*var*	1 if *var* is defined; 0 if not
${?*var***}**	1 if *var* is defined; 0 if not; insulate **var** string
$$	process ID of parent shell
$!	process ID of last started background job
$<	read a line from **stdin** (BSD)

Variable modifiers:

Modifier	Meaning
:e	extension (BSD)
:h	header name
:r	root name
:t	tail
:q	quote
:x	quote an expand into individual words
:g[**hrtes**]	modify all words in wordlist using specified modifier

Example:

```
% set e = 2.718282
% set p = $HOME
% echo e
e
% echo $e
2.718282
% echo $e5
e5 Undefined variable
% echo 5$e
52.718282
% echo $p$e
/disk1/users/peter2.718282
% echo $p5$e
p5 Undefined variable
% echo ${p}5$e
/disk1/users/peter52.718282
```

Predefined C Shell Variables

Variable	Meaning
argv	argument vector Wordlist variable containing the argument list passed to shell scripts. Contains the empty string ( ) by default.
cdpath	change directory path (unset by default) Wordlist variable containing the full pathnames of alternate directories to search for arguments to **cd** (and **pushd** and **popd**).
child	child process (non-BSD, unset by default) Contains the process ID of the most recently invoked background process. When the process terminates, variable **child** is undefined.
cwd	current working directory Contains the full pathname of the current working directory.
echo	echo mode (set/unset) When set, each command is displayed just before execution. Commands reflect history, alias, command, filename, and variable substitutions. May enable in a script with the **csh -x** option.
histchars	history substitution characters Contains the two history substitution characters. If unset, these characters are **!** and **^**
history	history list size. (unset by default) Contains the number of past commands the shell will store in the history list
home	home directory Contains the full pathname of the user's home directory. This variable is initialized by the C shell from the environment variable **HOME.**
ignoreeof	ignore end-of-file character (set/unset) When set, the shell will not terminate by reading an end-of-file character from the keyboard (i.e., <CTRL-d>). To logout, use the **logout** command. To exit a child shell, use the **exit** command.
mail	mail file Wordlist or single variable containing the pathnames where the C shell checks for mail. If the first word is numeric, the shell checks for mail in that many seconds.

The default interval is 10 minutes. If the mail variable contains more than one mail file, the mail message is "New mail in ..."; otherwise the message is "You have new mail."
Unset by default, and the shell uses mail file **/usr/spool/spool/mail/$USER** (or **$LOGNAME**)

noclobber do not clobber files (set/unset)
When set, the shell prevents redirection commands from overwriting an existing file. It also prevents append commands from creating a file. Use **!** to override the `noclobber` option on a single command.

noglob do not allow file expansion (set/unset)
When set, filename expansion is inhibited.

nonomatch no error on nonmatching file expansion characters (set/unset)
When set, a command containing file expansion characters that do not match any files does not produce an error. If no files match, the command is invoked with the characters unexpanded.
When unset, the shell reports an error and does not invoke the command.

notify notify of job completions (set/unset) (BSD)
When set, the shell notifies users of job completions asynchronousfy.
When unset, notification is just before the prompt.

path command path list
Wordlist variable containing the pathnames the shell should search to find commands. The C shell sets path to (**. /bin /usr/bin**) by default.
The C shell maintains **path** and the environment variable **PATH** together (**path** is forwarded to **PATH**).

prompt C shell prompt
Contains the C shell prompt string. Default value is '**%** '.

savehist save commands in history list (BSD, unset by default)
Contains the number of commands the shell should save upon logout. The shell places these commands back into the active history list automatically at login without executing. them. The commands are stored in file **~/.history**.

shell default shell file
Contains the full pathname of the default shell. The shell invokes this program to execute shell scripts.
Default value is **/bin/csh**.

status last command status
Contains the completion status of the last invoked command. Built-in commands return 0 if successful and 1 if unsuccessful.

term	terminal ID (BSD) Contains the name of the terminal type. Initialized by default to the value in file **/etc/ttytype** corresponding to the **tty** line.
time	automatic timing control (unset by default) Contains the maximum number of seconds in CPU time the shell allows a command to consume without reporting usage statistics.
user	user's name (BSD) Contains the user's login name. The shell initializes it from the environment variable **USER** (or **LOGNAME**).
verbose	verbose mode (set/unset) When set, the shell displays the command after history substitutions but before alias, command, filename, and variable substitutions. May be invoked in shell scripts with the **csh -v** option.