KORN SHELL 93 – BUILT-IN COMMANDS

Mt Xia publishes information on a variety of topics such as Business Continuity, Disaster Recovery, High Availability, AIX, and Shell Programming.

Mt Xia Inc.
113 East Rich
Norman, OK 73069

Dana French, President
dfrench@mtxia.com
615.556.0456

Built-in Commands

alias [-ptx] [name[=value] ...]
create an alias; with no arguments, display all aliases
-p print "alias" before each alias
-t set or print a tracked alias
-x obsolete: does nothing

bg [*jobid*]
put *jobid* in the background

break [*n*]
exit from enclosing loop (for, while, until, select), if *n* is specified, exit from *n*th enclosing loop

builtin [-ds] [-f *file*] [*name* ...]
with no arguments, display all built-in commands
-d delete built-in *name*
-f load new built-ins from shared library *file*
-s display special built-ins

cd [-LP] [*dir*]
if *dir* not specified, change CWD to $HOME
-L use logical path for cd
-P use physical path for cd
- change CWD to $OLDPWD

command [-pvV] *name* [*arg* ...]
if -v or -V not specified, execute *name* with *arg*
-p use a default search path, not ${PATH}
-v behave like whence
-V behave like whence -v

conditional commands

Built-in Commands

*see "**conditionals and looping**" quick reference*
case
if then else
[...] [[...]] ((...))
|| && !

continue [*n*]
if *n* not specified, go to next iteration of smallest enclosing loop. if *n* is specified, go to next iteration of *n*th enclosing loop.

disown [*jobid* ...]
do not send HUP signal to specified *jobids* when shell exits, if none specified use all active jobs.

echo [*words*]
send *words* to STDOUT followed by newline, if no *words* specified, send newline.

eval [*words*]
re-evaluate *words* using shell processing and execute result.

exec [-a name] [-c] [*words*]
execute *words* in place of shell
-a use name for argv[0]
-c clear the environment first
if *words* equal:
n>file : open output *file* using file descriptor *n*
n<file : open input *file* using file descriptor *n*
n<>file : open input/output file using *file* descriptor *n*
n<&- : close input file opened on file descriptor *n*
n>&- : close output file opened on file descriptor *n*
n>&m : redirect file descriptor *n* to file descriptor *m*

exit [*n*]
exit from current shell. If *n* is not specified use ${?}.

export [-p] [name[=value] ...]
export a shell variable to the environment. if no name specified, display all exported variables
-p print the word "export" before each variable

false
performs no action, but has a return value of 1

fc [-e editor] [-lnr] [first [last]]
display shell history from first to last

Built-in Commands

-e editor run editor if specified, else run ${FCEDIT}
-l display on STDOUT
-n don't display line numbers
-r reverse order

fg [*jobid*]
put *jobid* in the foreground

getconf [name [pathname]]
display POSIX configuration parameters for name and pathname

getopts [-a *name*] optstring name [arg ...]
parse command line options and arguments
-a *name* use *name* for error and usage messages

hist [-e *editor*] [-lnr] [-N *num*] [*first* [*last*]]
display shell history from *first* to *last*
-e *editor* run *editor* if specified, else run ${FCEDIT}
-l display on STDOUT
-n don't display line numbers
-N *num* *num* is relative to current command
-r reverse order

jobs [-lnp] [*jobid* ...]
display job information, if *jobid* not specified, display all job information
-l also list process ID's
-n list jobs whose status has changed
-p only list process groups

kill [-n *signum*] [-s *signame*] *jobid*
send signal identified by *signum* or *signame* to *jobid*. if no signal specified send SIGTERM
-n *signum* send signal corresponding to *signum*
-s *signame* send signal corresponding to *signame*

let *arg* ...
evaluate *arg* as an arithmetic expression. exit value is 0 if result of expression is non-zero. exit value is 1 if result of expression is zero.

looping commands
*see "**conditionals and looping**" quick reference*
for (word list)
for (arithmetic)
select (menu generator)

Built-in Commands

getopts (command line option processing)
until (loop until expression is true)
while (loop while expression is true)

newgrp [*word*]
change current group to *word*

print [-f *format*] [-enprRs] [-u n] [*word* ...]
display *words* on STDOUT
-e evaluate escape sequences (default)
-f *format* behave as printf command
-n suppress printing newline character at end of line
-p print to STDIN of co-process
-r suppress evaluation of escape sequences
-R same as -r but also ignore other options
-s print to history file
-u *n* print to file descriptor *n* (default is 1)

printf format [arg ...]
display output like ANSI C printf, format operators:
%b expand escape sequences in strings
%d decimal
%H HTML/XML
%n put number chars printed into variable
%P convert regular expression to ksh pattern
%R convert ksh pattern to regular expression
%q display quoted string that can be re-read
%(fmt)T display date/time in date format *fmt*
%Z evaluates to a null character

pwd [-LP]
display the name of the current working directory (CWD)
-L display logical path (default)
-P display physical path

read [-A *name*] [-d *delim*] [-n *count*] [-t *timeout*] [-prs] [-u *n*] [name?prompt] [*variables* ...]
read from STDIN and assign input to *variables*
-A *name* read words into indexed array *name*
-d *delim* use *delim* to separate words in input
-n *count* read at most *count* bytes
-p read from STDOUT of co-process
-r do not interpret "\" in input
-s save input in history file
-t *timeout* wait *timeout* seconds for input

Built-in Commands

-u *n* read from file descriptor *n* (default=0)
name?*prompt* displays "*prompt*" and reads input into variable identified by *name*

readonly [-p] [*name*[=value] ...]
mark variable *names* as read-only. Displays list of existing read-only variables if *names* not specified.
-p print "readonly" before each variable

return [*n*]
exit a function with return value of *n*. if *n* not specified return value is ${?}. If not in function, behaves like exit.

set
*see "**set and typeset**" quick reference*

shift [*n*]
shifts position of all command line arguments *n* positions to the left. if n not specified, default value of *n* is 1

sleep *seconds*
stop execution for a given number of *seconds*

test
evaluates conditional expressions: *see "**Testing**" quick reference*

times
print accumulated process times

trap [-p] [*word*] [*sigs*]
if a signal is received that is in the list of *sigs*, execute *word*. *sigs* is list of signal numbers or names.
if *word* is null – signal is ignored
if *word* is "-" - default action of signal is performed
if *word* is non-null – *word* is executed

true
performs no action, but has a return value of 0

typeset
*see "**set and typeset**" quick reference*

ulimit [*type*] [*options*] [*limit*]
set or display per-process limits. default action is to set both hard and soft limits, displays soft limits.
-H hard limit *type*

Built-in Commands

-S soft limit *type*
-a display all limits
-c core file size (512 byte blocks)
-d kilobytes of data segment
-f maximum file size (512 byte blocks)
-m kilobytes of physical memory
-n maximum number of file descriptors +1
-p size of pipe buffers (512 byte blocks)
-s kilobytes of stack segment
-t cpu seconds
-v kilobytes of virtual memory
limit can be result of arithmetic expression
limit can be set to "unlimited" to set it to maximum

umask [-S] [mask]
with no arguments, displays the current mask. if mask is specified it is subtracted from default permissions of 666 for files, or 777 for directories.
-S print current mask in symbolic form. symbolic mask is permissions to keep.

unalias [-a] *names*
remove aliases *names*
-a remove all aliases

unset [-fnv] [*names*]
unsets variables, name references, or functions
-v unset variables *names*
-f unset functions *names*
-n unset nameref *names*

wait [*jobid*]
wait for job *jobid* to complete, if *jobid* not specifed, wait for all child processes to complete.

whence [-afpv] *name* ...
indicate how each *name* is treated by the shell
-a display all interpretations of *name*
-f skip the search for functions
-v display verbose information
-p do a path search even if *name* is a keyword, alias, or function