NOTES ON UNIX SHELL PROGRAMMING

BOURNE SHELL—sh, prompt is $ ($PS1)

VARIABLES: INDIVIDUAL COMMAND LINE ARGs: $0,$1,...,$9
ALL COMMAND LINE ARGs $1-$9 : $*, @
COMMAND LINE ARG COUNT: #
SHELL PROCESS ID: $$
LAST COMMAND EXIT STATUS: ?
SECONDARY COMMAND LINE PROMPT $PS2

The If Construct

if [ test options ]
then
command
command
...
else (OR elif)
command
command

fi

(Note: else is optional)

The While Construct

while [ test options ]
do
command
command
...
done

The Until Construct

until [ test options ]
do
command
command
...
done

The For Construct

for var in word1 word2 ... wordn
do
command
command
...
done

The Case Construct

case value in
pattern1) command
command
command;

pattern2) command
command
command;

...

(default->) *) command
command
command;

esac

In the If, While and Until constructs, a command can be used in place of [ test options ]. See next page for list of test options.

Special commands meaning

exit n terminates a shell script with exit status = n.
break terminates a loop (while, until, for)
continue skip remaining steps in current loop cycle
shift move command line args to the left (new $1 = old $2)
null command (has exit status=0, i.e. TRUE)
TEST OPTIONS

Operator Returns TRUE (exit status of 0) if

Integer Operators

int1 -eq int2 int1 is equal to int2
int1 -ge int2 int1 is greater than or equal to int2
int1 -gt int2 int1 is greater than int2
int1 -le int2 int1 is less than or equal to int2
int1 -lt int2 int1 is less than int2
int1 -ne int2 int1 is not equal to int2

String Operators

str1 = str2 str1 is equal to str2
str1 != str2 str1 is not equal to str2
string string is not null
-n string string is not null (string must be seen by test)
-z string string is null (string must be seen by test)

File Operators

-d file file is a directory
-f file file is an ordinary file
-r file file is readable by the process
-s file file has nonzero length
-w file file is writable by the process
-x file file is executable

Logical Operators (for compound test conditions)

-a -> And -o -> Or

example: [ "$a" -eq 0 -o "$b" -lt 10 ]

Logical Operators (for compound command evaluation)

&& And
|| Or

command1 && command2 means command2 will be executed
    only if command1 returns exit status of 0.
command1 || command2 means command2 will be executed
    only if command1 returns exit status != 0.

Calculations - use the expr command:
    var=`expr $var1 op $var2` where op is +, -, *, /, %
KORN SHELL - ksh, superset of BOURNE SHELL

VARIABLES: INDIVIDUAL COMMAND LINE ARGUMENTS: $0,$1,...,$9,$(10),$11,...
$RANDOM - random number between 0 and 32767
$SECONDS - number of seconds since shell began.
$PS3 - prompt for the select statement
rest same as BOURNE

EXPRESSIONS - RELATIONAL OPERATORS: >,==,<,<=,>=,<,!=
- ARITHMETIC OPERATORS: +,-,*,/,%,,+-,+-,,=,=,%=,
- use [[ ]] or (( )) for advanced relational expressions.

VARIABLE ASSIGNMENT - STRING ASSIGNMENT: same as Bourne
- NUMERICAL ASSIGNMENT: let var=expression
  var=$($ expression )

VARIABLE ATTRIBUTES - typeset option var

VARIABLE SUBSTRING - ${varname:startpos:clength} expands to the substring
of varname starting from the character at startpos,
for a length up to a max of clength characters.

ARRAYS - set -A arrayvar word list
  typeset arrayvar[numelements] or create dynamically
  arrayvar[pos]=value
  var=${arrayvar[$pos]}]

Special commands meaning:
alias, unalias create, remove new name for a builtin command
fc [-l] N fix [list] the command for event N in the history
  must be preceded by FCEDIT=/usr/bin/vi
COMMAND LINE EDITING - must be preceded by set -o vi
  press <esc>
  use vi commands on the current line
  press return to execute the new command.
  can press y to invoke the full screen vi
  editor and modify the existing line and add
  additional command lines. Commands will be
  executed when you leave the editor.

EXTENDED PATTERN MATCHING:
*(p) - zero or more occurrences of pattern p
+(p) - one or more occurrences of pattern p
?(p) - zero or one occurrences of pattern p
!(p) - matches what does not match pattern p
@(p1|p2|...) - matches patterns p1 or p2 or ...
C SHELL - csh, prompt is % ($prompt)

VARIABLES: INDIVIDUAL COMMAND LINE ARGXS: $0,$argv[1],$argv[2]...
ALL COMMAND LINE ARGXS $argv[1].....: $argv[*]
COMMAND LINE ARG COUNT: $#argv
SHELL PROCESS ID: $$
LAST COMMAND EXIT STATUS: $status

The If Construct

if ( expr ) then
  command
  command
  ...
else (OR else if )
  command
  command
endif

(Note: else is optional)

The While Construct

while ( expr )
  command
  command
  ...
end

The Switch Construct

switch ( string )
  case pattern1:
    command
    ...
    breaksw
  case pattern2:
    command
    ...
    breaksw
  ...
  default:
    command
    ...
    command
  endsw.

The Foreach Construct

dforeach var { wordlist }
  command
  command
  ...
  command
end

EXPRESSIONS - RELATIONAL OPERATORS: >,=,<,<=,>=,!=
- ARITMETIC OPERATORS: +,-,*,/,&,+,-,*,=,/=,%=,

VARIABLE ASSIGNMENT - STRING ASSIGNMENT: set var = value
- NUMERICAL ASSIGNMENT: @ var = expression

ARRAYS - set arrayvar = ( word list )
  set arrayvar[pos] = value
$#arrayvar - the number of elements in the arrayvar

Special commands

exit(n) exit with status = n.
alias,unalias create,remove new name for a builtin command
set history = 20 turn on logging of 20 most recent commands
! N reexecute event N from the history list
pushd,popd stack,unstack directories
set var = `head -1` read a variable from the keyboard