Unix Shell Programming

Chapter 4: Decisions

Roshan Chitrakar, PhD
roshanchi@gmail.com

if

- enables you to test a condition and then change the flow of program execution.

  if command$_t$

  then

  command

  command

  ...

  fi

- where command$_t$ is executed and its ***exit status*** is tested. If the exit status is ***zero***, the commands that follow between the then and the fi are executed; otherwise, they are skipped.
- ***if ,,, then ,,, else ,,,, fi*** construct can also be used.
- ***….. elif….*** clause is also supported.

Exit Status

- Whenever any program completes execution, it ***returns an exit status*** back to the system.
- an exit status of ***zero indicates that a program succeeded***, and nonzero indicates that it failed.
- In a pipeline, the exit status is that of the last command in the pipe. e.g. ***who | grep fred***
- ***The $? Variable:*** set by the shell to the exit status of the last command executed.

  $ who | grep fred # fred does not exist

  $ echo $? returns 1
- the next ***echo $?*** returns 0 (why?)

test

- used for testing one or more conditions
- Syntax : ***test expression***
  - ***$ test "$name" = julio*** # string operator
  - ***$ test -n "$nullvar"***# string is not null
  - ***$ test -z "$nonnullvar"*** # zero length
- Alternative way of test
  - ***test expression*** is equivalent to ***[expression]***
  - ***["$count" -eq 0]*** # integer operator equality
  - ***["$choice" -lt 5]*** # less than
  - ***["$index" -ne "$max"]***
- File operators may also be used in ***test***
  - ***[-f /users/steve/phonebook] # file exists or not***
  - ***[-r /users/steve/phonebook] # read only status***
  - ***[-d /users/home/roshanchi/xshellprg] # file is a directory***

Logical Operators

- [! -r /users/steve/phonebook] # NOT
- [! "$x1" = "$x2"]
- ["$x1" != "$x2"]
- [-f "$mailfile" -a -r "$mailfile"] # AND
- ["$count" -ge 0 -a "$count" -lt 10]
- [-n "$mailopt" -o -r $HOME/mailfile] # OR

exit

- Syntax : exit n
- where n is the exit status that you want returned. If none is specified, the exit status used is that of the last command executed before the exit.

case

- allows you to compare a single value against other values and to execute one or more commands when a match is found.
- The format is –->

```
case value in
pat 1 )
        command
        ...
        command;;
        ...
pat n )
        command
        ...
        command;;
esac
```

The Null Command :

- **:** command does nothing
- it's only used to satisfy the requirement that a command appear
- Example: -

```
if grep "^$system" /users/steve/mail/systems > /dev/null
then
        :
else
        echo "$system is not a valid system"
        exit 1
fi
```

&& and ||

- In case of &&, command2 will be executed only if command1 returns an exit status of zero
  - ***sort bigdata > /tmp/sortout && mv /tmp/sortout bigdata***
- Opposingly in ||, the second command gets executed only if the exit status of the first is nonzero.
  - ***grep "$name" phonebook* || *echo "Couldn't find $name"***
- They may well be combined: -
  - ***who* | *grep "^$name " > /dev/null && echo "$name's not logged on"* || *echo "$name is logged on"***
- When used in an if, the effect of && and || are like ***"AND"*** and ***"OR"***