

CS-333 - Unix Operating System Fundamentals

Lecture 11 Shell Programming

Based on slides created by Dr. Bangalore for the Spring 2005 offering of the course

Shell Programming Con't

A Quick Review...

Passing Arguments to a Script

- Similar to passing command-line arguments in other programming languages, we can pass arguments to a shell script

Example: ./myscript arg1 arg2

- Shell interprets the arguments as follows:
 - \$1 – argument 1
 - \$2 – argument 2 and so on
 - \$0 – the name of the current script
 - \$* – all arguments
 - \$# – no. of arguments
 - \$\$ – process id (PID) of process running the script

11/10/2006

4

Example: script with complex arguments

```
$ cat > cmdscript.sh
echo "The name of the script is: '$0'"
echo "No. of command-line arguments = '$#' "
echo "First three arguments are: '$1 $2 $3'"
echo "Complete argument list is: '$*' "
echo "PID of this process = '$$'"

$ chmod +x cmdscript.sh

$ ./cmdscript.sh "Hello World" "$USER" "$USER"
The name of the script is: ./cmdscript.sh
No. of command-line arguments = 3
First three arguments are: Hello World puri $USER*
Complete argument list is: Hello World puri $USER*
PID of this process = 5828

$ ./cmdscript.sh a b c d
The name of the script is: ./cmdscript.sh
No. of command-line arguments = 4
First three arguments are: a b c
Complete argument list is: a b c d
PID of this process = 5831
```

11/10/2006

5

Obtaining Exit Status of Processes

- To obtain the exit status of a process, type "echo \$?" after the command is entered
- An exit code "0" indicates program completed successfully, otherwise something went wrong
- "test" or "[" command used for making decision
 - [statement] – returns 0 to the parent process if statement is true, otherwise 1

11/10/2006

6

Procedural Constructs

if...then...else

- Syntax:
if *command* (run *command* and obtain exit status)
then *command* (if exit status is 0, execute this)
else *command* (if exit status is not 0, execute this)
fi

OR

if *condition* ; *then* *action* ; *else* *action2* ; *fi*

- Example:

```
if [ $1 ]
then
  if [ -f $1 ]
  then echo File Exists
  else echo File does not exist
  fi
else
  echo "Usage: $0 <filename>"
fi
```

11/10/2006

8

for

- Syntax:
for *word* *in* *wordlist*
do
command
done
OR
for *word* [*in* *wordlist* ...] ; *do* *actions* ; *done*

- Example:

```
for file in *.java
do
  echo $file
  wc -l $file
done
```

OR
for *file* *in* *.java ; *do* *echo \$file* ; *wc -l \$file* ; *done*

11/10/2006

9

while

- Syntax:
while *command* (run *command* and obtain exit status)
do
command (if exit status is 0, execute *command*)
done
OR
while [*conditions*] ; *do* *actions* ; *done*

- Example:

```
a=0
while [ $a -le 10 ]
do
  echo $a
  ((a=$a+1))
done
```

OR
a=0 ; *while* [*\$a -le 10*] ; *do* *echo \$a* ; *((a=\$a+1))* ; *done*

11/10/2006

10

case

- Syntax:
case *word* *in*
 a) *command* ;; (if *word* matches *a*, execute *command*)
 b) *command* ;; (if *word* matches *b*, execute *command*)
esac
OR
case *word* *in* [*pattern* [| *pattern*]) *actions* ;; ...] *esac*

- Example:

```
a=$1
case $a in
  A) date ;;
  B) cal ;;
  C) ls ;;
  D|E) who | wc -l ;;
esac
```

OR
a=\$1 ; *case* *\$a* *in* A) *date* ;; B) *cal* ;; C) *ls* ;; D|E) *who* | *wc -l* ;; *esac*

11/10/2006

11

Reading User Input

- To read input from users in a shell use the "read" command
- The variable name is passed as an argument to the read command
- If a variable exists its value is modified, otherwise a new variable is created and the value entered is assigned
- Syntax: *read* *variable_name*

```
$ read a; echo $a
1234
1234
```

```
a=0
while [ $a -le 5 ]
do
  read b; echo $a $b; ((a=$a+1));
done
```

11/10/2006

12

Complete Shell Script

```
# An example script with various shell constructs
clear
echo "1) grep $USER /etc/passwd"
echo "2) cat passwd | grep $USER"
echo "3) who | sort | awk '{print $1}' | uniq -c"
echo "What command do you want to run?"
echo "Select 1, 2, or 3"
read choice
echo "You have selected option $choice"
echo
echo "The output for the selected command is:"
case $choice in
  1) grep $USER /etc/passwd ;;
  2) ypcat passwd | grep $USER ;;
  3) who | sort | awk '{print $1}' | uniq -c ;;
esac
a=$?
if [ $a -eq 0 ]
then
    echo "Command executed successfully"
else
    echo "Command returned with the exit code $a"
fi
```

11/10/2006

13

Defining and Using Functions

- Similar to functions in other programming languages we can instruct the shell to define functions
- This function is not a file, it is not a script, it is defined in memory
- You can also save these functions to a file

```
$ numfiles()
{
  is | wc $1
}
$ numfiles temp.java
    14  33  245 temp.java
$ numfiles
    143  144  2004
$ is | wc
    143  144  2004
```

```
$ cat > myfunctions
numfiles()
{
  is | wc $1
}
$ ./myfunctions
$ typeset -F
declare -f numfiles
$
```

11/10/2006

14

"getopts" function

- Used to retrieve options and option-arguments from a list of parameters
Usage: *getopts options variable*
Example: *while getopts abc option*
- getopts* function examines the argument list for options following a dash and then assigns the first option that matches to the variable specified
- If one of the options requires an argument, it must be followed by a colon
(e.g., *getopts a:b:c: option*)
- If no matching option is found, the variable specified will be set to ? character

11/10/2006

15

"getopts" example

```
$ cat getopts.sh
while getopts abo: c
do
  case $c in
    a | b) echo "Option selected is $c" ;;
    o)    echo "Option selected is $c and it's arguments are $OPTARG" ;;
    *)    echo $USAGE; exit 2;;
  esac
done

$ ./getopts.sh -o xxx.z.yy -b -a filename
Option selected is o and it's arguments are xxx.z.yy
Option selected is b
Option selected is a
$ ./getopts.sh -bo "xxx.z.yy" filename
Option selected is b
Option selected is o and it's arguments are xxx.z.yy
```

11/10/2006

Note: You can use * as the default value in case statements

16