

CS-344 - Unix Operating System Fundamentals

Lecture 6 Shell Programming

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

Shell: Turn Interpretation Off

- Shell interprets characters entered at the command prompt
- To turn off this interpretation we must use
 - Backslash - \
 - Double quotes - ""
 - Single quotes - ''
- What happens when the following commands are used
 - echo '\$HOME' '\$HOME' \ \$HOME
 - echo My files are: "*"
 - echo "My files are: *"
 - echo 'My files are: *'

“echo” command

- Using " " and ' ' with *echo*

```
$ echo 'Hello $USER'
```

```
Hello $USER
```

```
$ echo "Hello $USER"
```

```
Hello puri
```
- When the shell interprets `'` it stops interpretation until a matching `'` is found
- To eliminate new line with *echo* use

```
echo -n <text>
```

```
echo -n "Enter your choice: "
```

7/15/2005

7

Shell Programming



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 are 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

7/15/2005

9

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
The name of the script is: ./cmdscript.sh
No. of command-line arguments = 3
First three arguments are: a b c
Complete argument list is: a b c
PID of this process = 5831
```

7/15/2005

10

Simple Arithmetic

- Using "bc"
 - `a=8; echo "$a * 9" | bc` (result is 8*9 = 72)
- Using "let" (assign values to variables and to perform arithmetic calculations)
 - `a=8`
 - `echo $a`
 - `let a=$a*9` [could also use `((a=$a*9))`]
 - `echo $a` [could use `echo $((a=$a*9))` instead]

7/15/2005

11

Obtaining Exit Status of Processes

- When we type a command and press ENTER
 - the shell interprets the command
 - creates child process (or processes)
 - completes input/output redirection and passes arguments
 - instructs child process to execute appropriate program
 - when program completes, shell receives the exit status code from the child process (every program returns an exit code on completion)
- To obtain this exit status, type "echo \$?" after the command is entered
- An exist code "0" indicates program completed successfully, otherwise something went wrong

7/15/2005

12

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
```

7/15/2005

16

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
```

7/15/2005

17

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
```

7/15/2005

18

Complete Shell Script

```
# An example script with various shell constructs
clear
echo "1) grep $USER /etc/passwd"
echo "2) ypcat 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
```

7/15/2005

19

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

7/15/2005

20

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

“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

7/15/2005

21

“getopts” example

```

$ cat getopts.sh
while getopts abc: 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 -abo "xxx.z.yy" filename
Option selected is a
Option selected is b
Option selected is o and it's arguments are xxx.z.yy

```

7/15/2005 Note: You can use * as the default value in case statements 22

Final Notes:

- Infinite loop:


```

while :
do
...
exit 0
...
done

```
- Selections:


```

case
  1) ... ;;
  2) ... ;;
  ...
  7|8|9) ... ;;
  *) ... ;;
ecase

```

7/15/2005 23
