Shell Programming

Ansuman Banerjee
Arijit Bishnu
Debapriyo Majumdar

Data and File Structures Lab
M.Tech. Computer Science 1st Year, Semester I
Indian Statistical Institute Kolkata

September 18, 2014

Shell Programming

- When you open a terminal, some shell runs
  - bash: Bourne-Again Shell
  - csh: C-Shell
  - tcsh: TENEX C Shell
- Which one is better?
  - Everyone has own preferences and habits
  - Read up articles, or learn all and figure it out if you have to

A Bash Script

- A Shell script: **first.sh**

A script must start with this (#!). Called the "Shebang"

```
#! /bin/bash

echo "I am working under:"
pwd
echo "The folder contains:"
ls
```

Run the script

```
$ chmod 755 first
$ ./first
```

User input

- Read user input with “read”
- Rest of the input line → the last variable

```
read variable1 [variable2 variable3 …]

read -p “prompt” var1 [var2 var3 …]
```

User input

- A Shell script: **input.sh**

```
#! /bin/bash

read -p "Input a few numbers:" num1 num2
```

What does it do?

```
$ ./input.sh
Input a few numbers:34 56 87
```

Exercise:

Write a shell script that asks me my name. I input my firstname and lastname, and it then tells me my lastname.

Special variables

Parameter	*Meaning*
`$0`	Filename of the shell script itself
`$1-$9`	Command line arguments, numbered 1 to 9
`$#`	The number of command line arguments
`$*`	All arguments into one string; “$*” is one string
`$@`	All arguments into a set of strings
`$?`	Return status of most recently executed command
`$$`	Process id of current process

User input

- **input2.sh**

```
#! /bin/bash
echo “The second input is: “
echo $2
```

What does it do?

```
$ ./input2.sh 45 65
The second input is: 65
```

Exercise:

Write a shell script that takes a directory name as the command line input and lists the contents of the directory.

If – then – else if – else

```
if [ condition ]; then
     statements
elif [ condition ]; then
    statement
else
    statements
fi
```

AND, OR, NOT

AND	&&	Must be enclosed within [[ ]]
OR	||	Must be enclosed within [[ ]]
NOT	!	

Relational operators

Meaning	*Numeric*	*String*
Greater than	-gt	
Greater than or equal	-ge	
Less than	-lt	
Less than or equal	-le	
Equal	-eg	= or ==
Not equal	-ne	!=
str1 is less than str2		[[str1 < str2]]
str1 is greater str2		[[str1 > str2]]
String length is greater than zero		-n str
String length is zero		-z str

If – then – else if – else: example

```
#!/bin/bash

if [[ $1 -ge 0 && $1 -lt $2 ]]; then
   echo "First is positive but smaller."
elif [ $2 -lt $1 ]; then
   echo "Second is smaller."
else
   echo ”First is neg or both are equal."
fi
```

Testing files

`-d file` : True if 'file' is a directory

`-f file` : True if 'file' is an ordinary file

`-r file` : True if 'file' is readable

`-w file` : True if 'file' is writable

`-x file` : True if 'file' is executable

`-s file` : True if length of 'file' is nonzero

Exercise:

Write a shell script that takes a string as the command line input and reports whether or not the string represents an ordinary file which is writable.

Loops

```
while [ condition ]
  do
      commands
  done
```

```
until [ condition ]
  do
      commands
  done
```

```
for VAR in LIST
  do
      commands
  done
```

```
select WORD in LIST
  do
      commands
  done
```

```
for parm
  do
      echo $parm
  done
```

Functions

```
#!/bin/bash

function_name () {
    commands
}

command1
function_name args
…
command_n
```

- Declaration first, call later
- Parameters need not be specified beforehand
- Arguments provided via function call are accessible inside function as $1, $2, $3, …
- But $0 is still the script name, not the function name

- Variables inside functions are global to the script
- Define “local” for local variables
- Exercise: Write a shell script which takes 3 files as command line arguments and outputs their contents, using function.

Acknowledgements

- Reva Freedman's slides:
  - http://faculty.cs.niu.edu/~freedman/
- Tutorial on shell
  - http://linuxcommand.org/writing_shell_scripts.php