CS183
Introduction to Shell Scripting

Ali Davanian

Some slides were taken from WeeSan Lee

For loops

- **Syntax**

```
for <var> in <list>; do
    <statemants>
done
```

Values separated by a white space “ “

- Example:

```
for i in 1 2 3 4 5 6 7 8 9 10; do
    echo $i
done
```

- <list> can be generated from a command execution; the command should be $()
  - $(ls .)
  - <seq start end> command can be used to generated a sequence automatically
- <list> can also be files in a directory
  - Example: /* a list of all files under the root directory

While loops

- while loop syntax

  while *<condition>*; **do**

  <stmts>

  done

- Example:

```
i=1;
while [ $i -le 5 ]; do
    echo “i=$i”;
    i=$((i+1);
done
```

Read input

- “read” can be used to read from standard input
  - “read line;echo $line” reads a line stores it in $line and outputs it to the standard output
  - “read col1 col2; echo $col1” reads a line and stores words based on the white space in the listed words
    - Columnizing based on white space can be changed by modifying $IFS variable
- “read” can also be used to a line read from a file:
  - read line < cut1.txt;echo “line is $line”
- How can we read multiple lines?
- How can we read all lines of a file?

```
#!/bin/sh
while read line; do
   echo "$line"
done < /etc/passwd
```

What does this script do?

```
#!/bin/sh
OIFS=$IFS; IFS=:
while read name passwd uid gid fullname ignore; do
  echo "$name ($fullname)"
done < /etc/passwd
IFS=$OIFS
```

Function call

- **Syntax**:
  - Definition:
    ```
    function_name(){
            <statements>
    }
    ```
  - Call:
    ```
    function_name
    ```
- Parameters:
  - The parameters can be accessed based on their position $1 $2 $3 …

Examples:

```
start() {
   echo "Start"
}
stop() {
   echo "Stop"
}
restart() {
   echo "Restart"
}
start
stop
restart
```

Regular expressions

- A regular expression (shortened as regex or regexp; also referred to as rational expression) is a sequence of characters that specifies a search pattern[1].
- Most unix commands support regex patterns
- Examples:
  - ^83$: the exact 83 value (not instances having 83 like 183 or 830)
    - grep –E ^83$ file_lines.txt
  - *.txt: all files ending with the extension ".txt"
  - ?.txt: all files have a single character name and the extension txt
    - for i in ./?.txt;do echo $i;done

[1] https://en.wikipedia.org/wiki/Regular_expression

Regular expressions

Letters	
123…	*Digits*
\d	*Any Digit*
\D	*Any Non-digit character*
.	*Any Character*
\\.	*Period*
[abc]	*Only a, b, or c*
[^abc]	*Not a, b, nor c*
[a-z]	Characters a to z
[0-9]	Numbers 0 to 9
\w	Any Alphanumeric character
\W	Any Non-alphanumeric character

{m}	*m Repetitions*
{m,n}	*m to n Repetitions*
*	*Zero or more repetitions*
+	*One or more repetitions*
?	*Optional character*
\s	*Any Whitespace*
\S	*Any Non-whitespace character*
^…$	*Starts and ends*
(…)	*Capture Group*
(a(bc))	*Capture Sub-group*
(.*)	*Capture all*
(abc|def)	*Matches abc or def*

Please see https://regexone.com (the above tables are also taken from this website)

Regular expressions

Write a script that matches the following match group and skips the other words?

Match	Ana
	Bob
	Cpc
Skip	aax
	bby
	ccz

```
grep –E [A-Wa-w]{3} ./regx.txt
```

Regular expressions

Write a script that matches the following match group and skips the other words?

Match	aaaabcc
	aabbbbc
	aacc
Skip	a

```
grep –E aa+b*c+./regx2.txt
```

Additional Resources

- Quick Introduction:
  - http://www.panix.com/~elflord/unix/bash-tute.html
- Advance Bash-Script Guide:
  - http://www.tldp.org/LDP/abs/abs-guide.pdf
- Shell Script Examples:
  - http://www.macs.hw.ac.uk/~hwloidl/Courses/LinuxIntro/x864.html
- Interactive Regular Expression tutorial
  - https://en.wikipedia.org/wiki/Regular_expression
- Unix Power Tools
  - Shelley Powers, Jerry Peek, Tim O'Reilly, Mike Loukides