CSci 4061: Introduction to Operating Systems

Recitation 2

September 18, 2

By: Manu Khandelwal

Agenda

- What are Shell Scripts
- Variables, Operators
- Decision Making, Loops
- Example Shell Scripts
- Practice Script Programs

Shell scripts

What are shell scripts?

- Shell script is a list of command, which are listed in the order of execution.

- There are
  - Variables (to read and store data)
  - Conditional Tests (e.g. value A greater than value B)
  - Loops (iteration)

- Shell scripts are interpreted and not compiled.

Example Scripts (Hello World!)

hello_world.bash

```
#!/bin/bash
# comments start with # symbols in shells
echo "Hello, world!"
```

Execution of the above script

chmod u+x hello_world.bash
./hello_world.bash

Hello, world!

Variable Names

- The name of a variable can contain
  - letters (a to z or A to Z)
  - numbers (0 to 9)
  - underscore character (_)

- Valid variables
  - _ALI
  - VAR_1

- Invalid variables
  - 2_VAR
  - -VARIABLE
  - VAR1-VAR2
  - VAR_A!

Example Scripts (Read from Shell)

user_input.bash

```
#!/bin/bash

# Read the name of the user

echo "What is your name?"
read PERSON
echo "Hello, Your name is : ${PERSON:=John
Doe}"
```

Variable Modifiers (bash)

Modifier	Semantics
${variable:-word}	If *variable* is unset or null, the expansion of *word* is substituted. Otherwise, the value of *variable* is substituted.
${variable:=word}	If *variable* is unset or null, the expansion of *word* is assigned to *variable*. The value of *variable* is then substituted.
${variable:+word}	If *variable* is null or unset, nothing is substituted, otherwise the expansion of *word* is substituted.

`variable_modifier.bash`

Special Variables

- $0 represents name of the script filename
- $# number of arguments
- $1 through $9, up to nine arguments
- $$ prints the process PID
- $* All the arguments passed

Example scripts – (Argument Passing)

argument_passing.bash

```
#!/bin/bash
echo "File Name: $0"
echo "Total Number of Parameters : $#"
echo "Quoted Values: $*"

echo "Argument 1 equals to $1 "
echo "Argument 2 equals to $2 "

let sum=$1+$2    #remove let and try!
echo "Sum = $sum"
```

Execution of the above script

chmod u+x argument_passing.bash

./argument_passing.bash 10 20

Example scripts – (Arrays)

using_arrays.bash

```
#!/bin/bash
arr=(one two three)
NAME[0]="Albert"
NAME[1]="Daisy"
NAME[2]="Sebastian"
NAME[3]="Arnold"
NAME[4]="Ali"
echo "First Index of NAME: ${NAME[0]}"
echo "Second Index of NAME: ${NAME[1]}"
echo "First Index of arr: ${arr[0]}"
```

Basic Operators

- Arithmetic Operators
  - + (Addition)
  - - (Subtraction)
  - * (Multiplication)
  - / (Division)
  - % (Modulus)
  - = (Assignment)
  - == (Equality)
  - != (Not Equality)

Basic Operators

- Relational Operators

String Test	Test for
string1 = string2	Equality test
string1 != string2	Non-equality test
string	String not null
-z string	String length is zero

Integer Test	
int1 -eq int2	Equality test
int1 -ne int2	Not equal
int1 -gt int2	int1 greater than int2
int1 -ge int2	int1 greater than equal to int2
int1 -lt int2	int1 less than int2
int1 -le int2	int1 less than equal to int2

Basic Operators

- Logical Operators
  - ! This is logical negation
  - -o This is logical **OR**
  - -a This is logical **AND**

Example scripts - (Arithmetic)

arithmetic.bash

```
#!/bin/bash

#input two numbers
#output the sum and the product

  echo "Enter one number"
  read number1
  echo "Enter a second number"
  read number2
  let sum=$number1+$number2
  echo "Sum equals to $sum"
  let product=$number1*$number2
  echo "product equals to $product"
```

Decision Making

- Unix Shell supports following forms of **if...else** statement
  - if...fi statement
  - if...else...fi statement
  - if...elif...else...fi statement

- Unix Shell supports **case...esac** statement

Example scripts (file tests, if - else)

file_test.bash

```
#!/bin/bash

echo "Enter File"
read datafile
if [ -e $datafile ]
then
  echo "We found $datafile"
else
  echo "$datafile not found"
fi
```

Other tests

- `-f` file is a regular file
- `-s` file is not zero size
- `-d` file is a directory
- `-r` can read
- `-w` can write
- `-x` can execute

... and many more

Example scripts - "case"

case.bash

```
#!/bin/bash
echo -n "Enter the name of an animal: "
read ANIMAL

echo -n "The $ANIMAL has "
case $ANIMAL in
     'horse’)     echo -n "four";;
      'dog’ )     echo -n "four";;
     'cat’)       echo -n "four";;
     'man’ )      echo -n "two";;
     'kangaroo’ ) echo -n "two";;
     *)           echo -n "an unknown number of";;
esac
echo " legs."
```

Shell Loop Types and Controls

- Shell Loop Types
  - while loop
  - for loop
  - until loop
  - select
- Shell Loop Controls
  - break
  - continue

Shell Loop Types Examples

loops.bash

```
#!/bin/bash

# Usage of for loop
for i in $( ls ); do
    echo item: $i
done

# Usage of while loop
COUNT=0
while [ $COUNT -lt 10 ]; do
    echo The counter value is $COUNT
    let COUNT+=1
done

# Usage of until loop
COUNT=20
until [ $COUNT -lt 10 ]; do
    echo The counter value is $COUNT
    let COUNT-=1
done
```

Select Examples

select.bash

```
#!/bin/bash

# Usage of select loop type

select i in mon tue wed thur fri sat sun exit
do
  case $i in
     mon) echo "Monday";;
     tue) echo "Tuesday";;
     wed) echo "Wednesday";;
     thur) echo "Thursday";;
     fri) echo "Friday";;
     sat) echo "Saturday";;
     sun) echo "Sunday";;
     exit) exit;;
  esac
done
```

Print all command line arguments

all_argument.bash

```
#!/bin/bash
echo "Argument are $*"
echo "Number of arguments is $#"
num=0
for x  in  $*
   do
      #num=`expr $num + 1`;
      #num=$(expr $num + 1);
      let num=$num+1
      echo "Arg $num is $x."
   done
```

Example scripts - (command line arguments, loops)

find_loops.bash

```
#!/bin/bash

echo "Enter the name of the folder"
read folder

list=$(find $folder -type f)
for file in $list;
  do
      fileSize=$(stat -c%s $file)
      echo "Size of $file = $fileSize bytes."
  done
```

Practice Shell Script

Write a bash script for safe copying.

./safe_copying sourceFile destinationFile

If destinationFile exist, take permission from the user whether to overwrite it or not

safe_copying.bash

Practice Shell Script

Write a shell script to print following pattern.

```
0
1 0
2 1 0
3 2 1 0
4 3 2 1 0
5 4 3 2 1 0
6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
8 7 6 5 4 3 2 1 0
9 8 7 6 5 4 3 2 1 0
```

pattern_print.bash

Practice Shell Script

Write a shell script to create a calculator supporting below operation

1. Addition
2. Subtraction
3. Multiplication
4. Division

calculator.bash