Review

- Shell Scripts
  - How to make executable
  - How to change mode
- Shell Syntax
  - Variables
  - Quoting
  - Environment Variables
  - Parameter Variables

Preview

- Conditions
- The test, or `[` Command
- Control Structures
  - if statement
  - if-else-if statement
  - for loop statement
  - while loop statement
  - until loop statement
  - case statement

Conditions

- Every programming language has the ability to test condition and perform different action based on the test result.
- Since a shell scrip condition can test the exit code of any command and script written by a programmer.
- It is **important to include exit** command at the end of any scripts that you write.

The test, or [, command

- On most systems, the **[** and **test** command are synonymous. When **[** is used for testing condition, a trailing **]** is also used just for readability.
- There must be a space after the [command.
- The condition types with the test command fall into three types
  - String comparison
  - Arithmetic comparison
  - File conditionals

The test, or [, command
(String Comparison)

- [string1 = string2] : true if two strings are equal
- [string1 != string2]: true if two strings are not equal
- [–n string] : true if the string is not null
- [–z string]: true if the string is null

The test, or [, command
(String Comparison)

```
#!/bin/sh
# condition.sh

myname="Sang-Eon Park"
echo "What is your name?"
read yourname

if [ -z "$yourname" ]; then
        echo "no input from keyboard"
        exit 1
elif ["$myname" = "$yourname" ]; then
        echo "We have same name $myname"
else
        echo "My name is $myname. Your name is $yourname"
fi
exit 0;
```

The test, or [, command
(Arithmetic Comparison)

- [expression1 **–eq** expression2]: true if two expression are equal
- [expression1 **–ne** expression2]: true if two expression are not equal
- [expression1 **–gt** expression2]: true if expression1 is greater than expression2
- [expression1 **–ge** expression2]: true if expression1 is greater or equal to expression2
- [expression1 **–lt** expression2]: true if expression1 is less than expression2
- [expression1 **–le** expression2]: true if expression1 is less than or equal to expression2

The test, or [, command
(Arithmetic Comparison)

```
# arithmatic.sh
# shows [ commend
#!/bin/sh

if [ 4 -eq 4 ]; then
   echo " 4 is equal to 4"
else
   echo " 4 is not equal to 4"
fi

a=4
b=4
if [ $a -eq $b ]; then
   echo " a is equal to b"
else
   echo " a is not equal to b"
fi

exit 0
```


The test, or [, command
(file Conditional)

- [–d file]: true if the file is a directory
- [–e file]: true if the file is exist
- [–s file]: true if the file has nonzero size
- [–f file]: true if the file is a regular file
- [–g file]: true if set-group-id is set on the file
- [–u file]: true if set-user-id is set on the file
- [–r file]: true if the file is readable
- [–w file]: true if the file is writable
- [–x file]: true if the file is executable

The test, or [, command
(file Conditional)

```
#!/bin/sh
# testCond.sh
# testing condtion with [
echo "file name to check?"
read fname
if [ -e $fname ]; then
   echo "the file $fname exist!"
else
   echo "There is no sush a $fname file exist"
   exit 1;
fi
exit 0;
```


The test, or [, command
(file Conditional)

```
#!/bin/sh
# testCond1.sh
# testing condtion with [

echo "file name to check?"
read fname
if [ -e $fname ]; then
          echo "the file $fname exist in the current directory"
else
          echo "There is no sush a $fname file exist in the current directory"
          exit 1;
fi

if [ -f $fname ]; then
          echo " the file $fname is regular file"
else
          echo " the file $fname is not regular file"
fi

exit 0;
```


Control Structures
(if statement)

- The if statement tests the condition and execute a statement or group of statements.
- SYNTAX

  if *test-commands*; **then**
  consequent-commands
  else
  alternate-consequents
  fi

Control Structures
(if statement)

```
#!/bin/sh
#greeting.sh testing if statement

echo "Is it morning? Please answer yes or no"
read yesorno
if [ $yesorno = "yes" ]; then
     echo "Good morning!"
else
    echo "Good afternoon!"
fi
exit 0;
```


Control Structures
(if –else-if Statement)

```
#!/bin/sh
#ifelseif.sh: test if-else-if statement

echo "Is it morning? Please answer yes or no"
read yesorno
if [ $yesorno = "yes" ]; then
     echo "Good morning!"
elif [ $yesorno = "no"]; then
     echo "Good afternoon!"
else
   echo "Sorry $yesorno not recognized. Enter yes or no"
   exit 1
fi
exit 0;
```


A problem with variable

- If user just press enter key instead of enter any string for an input of a variable, we will get a error message:
  [:=: unery operator expected]
- Since if [= "yes"] is illegal statement.
- To avoid this, we need use double quotation around a variable

```
if ["$yesorno" ="yes" ]; then
   echo "Good morning!"
elif ["$yesorno"="no" ]; then
   echo "Good afternoon!"
```


Control Structures
(if –else-if Statement)

```
#!/bin/sh
#ifelseif1.sh: test if-else-if statement

echo "Is it morning? Please answer yes or no"
read yesorno
if ["$yesorno" ="yes" ]; then
     echo "Good morning!"
elif ["$yesorno" ="no"]; then
     echo "Good afternoon!"
else
   echo "Sorry $yesorno not recognized. Enter yes or no"
   exit 1
fi
exit 0;
```



```
#!/bin/bash
# nestedif.sh
# Declare variable choice and assign value 4
choice=4
# Print to stdout
 echo "1. Bash"
 echo "2. Scripting"
 echo "3. Tutorial"
 echo -n "Please choose a word [1,2 or 3]? "
# Loop while the variable choice is equal 4
# bash while loop
while [ $choice -eq 4 ]; do
# read user input
read choice
# bash nested if/else
if [ $choice -eq 1 ] ; then

        echo "You have chosen word: Bash"

else

        if [ $choice -eq 2 ] ; then
                echo "You have chosen word: Scripting"
        else

                if [ $choice -eq 3 ] ; then
                        echo "You have chosen word: Tutorial"
                else
                        echo "Please make a choice between 1-3 !"
                        echo "1. Bash"
                        echo "2. Scripting"
                        echo "3. Tutorial"
                        echo -n "Please choose a word [1,2 or 3]? "
                        choice=4
                fi
        fi
fi
Done
exit 0
```

Control Structures
(for Loop Statement)

- Syntax

for *name* [in *words* ...]; do
    commands;
done

for ((*expr1* ; *expr2* ; *expr3*)) ; do
  commands ;
done

Control Structures
(for Loop Statement)

```
#!/bin/sh
# first.sh
# This file looks through all the files in the current
# directory for the string "main", and then print the name
# of those files to the standard output.

for file in *
do
       if grep -q main $file
       then
              echo $file
       fi
done
exit 0
```


Control Structures
(for Loop Statement)

```
#!/bin/sh
#  for1.sh Loop through a set of strings:
for m in Samsung LG Nokia Apple "Hewlett Packard"
do
  echo "Manufacturer is:" $m
done
```

```
#!/bin/sh
# for2.sh : Loop 10 times:
for i in $(seq 1 10);
do
        echo "$i  Hello World $i ";
done
```

```
#!/bin/sh
# for3.sh : Loop 10 times:
for ((i=1; i<11; i++));
do
        echo "$i  Hello World $i ";
done
```



```
#!/bin/sh
# forloop.sh
# shows for loop example
#loop through a set of stings:
for m in Samsung Sony Panasonic LG
do
  echo "Manufactuer of LED TV is: $m"
done

# Loop 10 times
echo "You want me say I love you? yes or no"
read answer
if [ "$answer" = "yes" ]; then
    for i in $(seq 1 10);
    do
       echo "I love you $1"
    done
else
    echo "Bye"
fi

exit 0;
```


Control Structures
(for Loop Statement)

```
#!/bin/sh
# foo.sh
# Loop through the arguments passed to a function

foo()
{
    for ARG in "$@"; do
      echo $ARG;
    done
}

foo $@

exit 0
```


Control Structures
(while Loop Statement)

- Execute *consequent-commands* as long as *test-commands* has an exit status of zero
- SYNTAX :

while *test-commands*; do
    consequent-commands
done

Control Structures
(while Loop Statement)

```
#!/bin/sh
# secret.sh
# testing while loop

echo "Enter password"
read password

while [  "$password" != "secret"  ];
do
   echo "Wrong password Try again"
   read password
done
 echo "You got it!"
exit 0
```

Control Structures
(while Loop Statement)

```
#!/bin/sh
# while.sh: to test while loop. let command, which treats its
# argument in a way meant to accommodate numbers

count=1
while [ "$count" -le 20 ]; do
   echo "Here we go again count = $count"
   let count++
# or ((count++))
done

exit 0
```


Control Structures
(while Loop Statement)

```
#!/bin/sh
# while1.sh testing nested while loop

y=1
while [ $y -le 12 ]; do
      x=1
      while [ $x -le 12 ]; do
            printf "% 4d" $(( $x * $y ))
            let x++
      done
      echo ""
      let y++
done
```


Control Structures
(until Loop Statement)

- Execute *consequent-commands* as long as *test-commands* has an exit status which is not zero.
  - While loop – continue if condition is true
  - Until loop – continue if condition is false
- SYNTAX

  until *test-commands*; do
    consequent-commands;
  done

Control Structures
(until Loop Statement)

```
#!/bin/bash
# until.sh example for until loop

Count=10
until [  $Count -lt 0 ]; do
     echo Count $Count
     let Count--
done
```


Control Structures
(until Loop Statement)

```
#!/bin/sh
# until1.sh testing until loop
clear
y=12
until [ $y -lt 1 ]; do
      x=12
      until [ $x -lt 1 ]; do
            printf "% 4d" $(( $x * $y ))
            let x--
      done
      echo ""
      let y--
done
```


Control Structures
(case Statement)

- case will selectively execute the *command-list* corresponding to the first *pattern* that matches *word*.
- SYNTAX

  case *expression* in
    pattern1) *statements* ;;
    pattern2) *statements* ;;
    ...
  esac

;; is for last statement in each case if there are more then one statements

Control Structures
(case Statement)

```
#!/bin/sh
# case.sh: example of case statement

echo -n "Is it morning? Enter yes or no "
read Answer
case $Answer in
yes|y|YES)
   echo "Good Morning!"
   echo "Have a Wonderful Day!";;
[nN]*)
   echo "Good Afternoon!"
   echo "How was your day!";;
*)
   echo "Sorry, answer not recognized"
   echo "Bye"
     exit 1;;
esac
exit 0
```


Control Structures
(case Statement)

```
#!/bin/sh
# case1.sh example of case statement

for filename in *; do
    case $filename in
        *.c ) echo "$filename is a C program";;
        *.cpp ) echo "$filename is a C++ program";;
        *.sh ) echo "$filename is a shell script";;
        *.o ) echo  "$filename is object code";;
        *   ) echo "error: $filename is not a source or
object file.";;
    esac
done
exit 0
```