Creating Simple Shell Scripts

- Conditionally execute code (use of: if, test, [], etc.)
- Use Looping constructs (for, etc.) to process file, command line input
- Process script inputs ($1, $2, etc.)
- Processing output of shell commands within a script
- Processing shell command exit codes

Conditional Execution: if Statement

Syntax of if Statement :

Example 1: Using Command as condition

```
if some command
then
        Command(s) to be executed
else
         Command(s) to be executed
fi
```

If statement decides to execute code based on exit status($?) of command . Zero exit status means SUCCESS and non-zero exit status means FAILURE

Example 2:

```
if [ some test ]    or  if test some test
then
      Command(s) to be executed
elif [ some test ]
then
      Command(s) to be executed
else
       Command(s) to be executed
fi
```

For different tests, We can check man page for test , man test

- Create a script compare.sh to find largest number out of three integers.
  - On execution, It should ask to enter integers.

```
#!/bin/bash
read -p "Enter first number:" num1
read -p "Enter second number:" num2
read -p "Enter third number:" num3
if [ $num1 -gt $num2 -a $num1 -gt $num3 ]; then
        echo "Number 1= $num1 is largest number"
elif [ $num2 -gt $num1 -a $num2 -gt $num3 ]; then
        echo "Number 2=$num2 is largest number"
else
       echo "Number 3= $num3 is largest number"
fi
```

loop: for Statement

Syntax of for Statement:

```
for VAR in VALUE1 VALUE2 …. VALUEn
do
      Some Command(s) using $VAR
done
```

Practical example 1:

```
for username in ex200 ex294
do
      useradd $username
done
```

Practical example 2:

```
for username in `cat userlist`
do
    useradd $username
done
```

- Create a script create_user.sh under root directory to create users.
  - It should read usernames from file /root/userlist and display on STDOUT "User 'username' already exists" if user already exists.
  - Otherwise, It should prompt for password for user.
  - Set password password for each user.

```
#!/bin/bash
for user in `cat /root/userlist`
do
if grep $user /etc/passwd > /dev/null
then
        echo "User '$user' already exists"
else
        useradd $user
        passwd --stdin $user
fi
done
```

Processing Script Inputs Using $1, $2….

We can create script to accept variable values as command line arguments.

$0 - Script Name

$1 - First Argument

$2 - Second Argument and so on

$# - Total number of arguments provided

$@ - List of all arguments provided

We will understand this with the help of simple example.

- Create a script users.sh under root directory to create users.
  - It should accept username(s) as command line arguments.
  - It should display "No username provided" if executed without any argument.
  - It should display "User already exists, or something wrong happened" if user is not created.

```
#!/bin/bash
if [ "$#" -eq 0 ]; then
    echo "No username provided"
else
    for user in "$@"
    do
        useradd $user 2> /dev/null
          if [ "$?" = "0" ]; then
                echo "User $user has been created"
           else
                echo "User $user already exists, or something wrong happened"
           fi
      done
fi
```