OPERATING SYSTEM

Lab Manual
[Fall/ Spring 2019_]

Student Name: Muhammad Fazeel Rabbani
Student Id: 022-16-14020

Prepared By: *Dr. Noman Islam*

Instructor: *Dr. Noman Islam*

LIST OF EXPERIMENTS

S. No	Date	Experiment
1	__/__/__	To study and implement file I/O in Java
2	__/__/__	To study and implement socket programming in Java
3	__/__/__	To study and implement multi-threading in Java
4		To study and implement concurrency control techniques in Java
5	__/__/__	To study and execute basic Linux commands on a terminal
6	__/__/__	To study and execute system administration commands on a terminal
7	__/__/__	To study and implement shell programming in Linux
8	__/__/__	To study and implement containers and dockers
9	__/__/__	To study and setup a kubernetes cluster
10	__/__/__	To study and implement process scheduling algorithms in Java

To study and implement File I/O in Java

Instructions:

Type the following program and save.

```
//FileTest.java
import java.io.*;
class FileTest
{
   public void fileWrite()
   {
      File dstFile = new File("K:\\myOutput\\outputfile.txt");
      PrintWriter out = new PrintWriter
                (new BufferedWriter(new FileWriter(dstFile)));
      out.print("Hello ");
      out.println("world");
      out.close();
   }
}
//FileTestMain.java
import java.io.*;

class FileTestMain
{
  public static void main(String[] args)
  {
      FileTest fileTest = new FileTest();
      fileTest.fileWrite();
   }
}
```

Lab Tasks

1. Try to compile the class FileTest. What goes wrong? This is because opening up a file could throw an IOException, which is a checked exception. This means you have to tell Java how to deal with it, or the program won't compile
2. Run your program again. If all went successfully, open up "My Computer", and find your FilePractice folder on your K drive. You should be able to find the file "outputfile.txt". Double click on it, and take a look. What do you see?
3. Modify your program to write to the file five lines, each of which contains your name or a friend's name, followed by a space and then an age, then another space and a gpa. For example:

   Arlene 19 3.8
   Bill 22 3.5
   Marilyn 15 3.9
   Bryan 35 1.1
   Buzz 6 4.0
4. Add the following method to your FileTest class:

   public void consoleRead() throws IOException

```
{
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("What is your first name? ");
    String first = in.readLine();
    System.out.print("What is your last name? ");
    String last = in.readLine();
    System.out.println("Your name is " + last + ", " + first + ".");
}
```

Compile it. Add "throws" statements as necessary. Modify your main to run the consoleRead method, and recompile. Run your program. What does it do?

5. Add the following method to FileTest.

```
public void fileRead() throws IOException
  {
    File srcFile = new File("K:\\FilePractice\\outputfile.txt");
    BufferedReader in = new BufferedReader(new FileReader(srcFile));
    String text = in.readLine();
    System.out.println(text);
    in.close();
  }
```

Run the method. What do you see? Modify this method to print out the names, ages, and gpas of the five people you stored back in Task 1.

6. In reality, you would want to be able to separate each item on each line into different variables, rather than keeping all the information on name, age, and gpa in one string. To break it up, use a StringTokenizer.

Seudo Code In Java::

```
package FileTest;
// @author Fazeel

import java.io.*;
import static java.lang.System.in;
import java.util.*;
public class FileTest {
  public void fileWrite()
  {
    try{
    File dstFile = new File("FileTest2.txt");
    PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(dstFile)));
    out.println("BASIM   19 3.8");
    out.println("FAZEEL  22 3.5");
    out.println("DANIYAL 25 3.9");
    out.println("AHMAD   35 1.1");
    out.println("NIDA    16 4.0");
```

```
      out.close();
   }
    catch (Exception ex){
       System.out.println(ex);
      }
   }
  public void consoleRead() throws IOException
{
      BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
      System.out.print("What is your first name :: ");
      String first = in.readLine();
      System.out.print("What is your last name  :: ");
      String last = in.readLine();
      System.out.println("Your name is " + first + ", " + last + ".");
}
      public void fileRead() throws IOException
   {
      try{
         File srcFile = new File("FileTest2.txt");
      BufferedReader in = new BufferedReader(new FileReader(srcFile));
      String text=null;
         System.out.println("::   Display Record   ::");
      while((text=in.readLine()) != null){
         StringTokenizer st= new StringTokenizer(text);
         System.out.println("Name  :: "+ st.nextToken());
         System.out.println("Age   :: "+ st.nextToken());
         System.out.println("CGPA  :: "+ st.nextToken());
         System.out.println("\n");
      }}
      catch(Exception s){
            System.out.println(s);
            }
             in.close();
   }
public static void main(String[] args) throws IOException {
      // TODO code application logic here
FileTest  fileTest = new FileTest( );
      fileTest.fileWrite();
fileTest.consoleRead();
fileTest.fileRead();
   }}
```

Output & Display ::

```
run:
What is your first name :: Fazeel
What is your last name  :: Rabbani
Your name is Fazeel, Rabbani.
::   Display Record   ::
Name  :: FAZEEL
Age   :: 22
CGPA  :: 3.7


Name  :: BASIM
Age   :: 19
CGPA  :: 3.8


Name  :: DANIYAL
Age   :: 25
CGPA  :: 3.9


Name  :: AHMAD
Age   :: 35
CGPA  :: 3.1


Name  :: NIDA
Age   :: 18
CGPA  :: 3.0
```

Lab 2

To study and implement socket programming in Java

Sockets provide the communication mechanism between two computers using Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). This lab will demonstrate how to implement TCP sockets using Java. Before starting the lab, download and install Java and Eclipse IDE by following the instructions below:

1. Download and Install Java Development Kit (JDK)'s latest version
2. Download 'Eclipse' on your computer
3. Go to Eclipse folder and Run eclipse.exe file
4. The Eclipse environment will start. Now perform the lab tasks.

Lab Tasks:

1. Find the IP address of a local host using java program. Use the InetAddress class.

Seudo Code ::

```
package ip.address;
import java.net.*;
    /**
      @author Fazeel
     */
    public class IPADDRESS {
       public static void main(String[] args)  {
          try{
            InetAddress IP = InetAddress.getLocalHost();
          System.out.println("IP of my system is :: "+IP.getHostAddress());
          }catch(Exception ex){
          ex.printStackTrace();
          }
       }
    }
```

Output & Display ::

```
run:
IP of my system is :: 192.168.10.5
BUILD SUCCESSFUL (total time: 0 seconds)
```

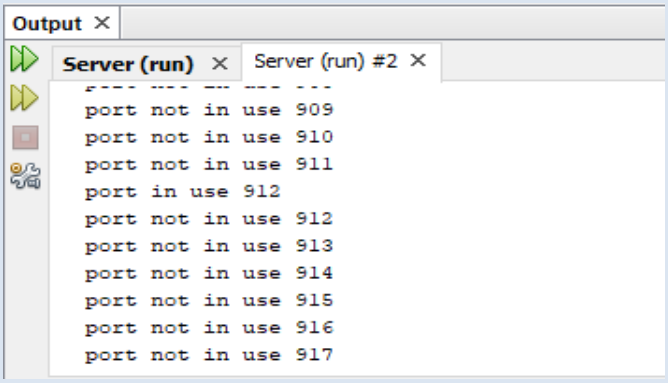
2. Write a small port scanner application. The program usage is as follows:

E:\ >java PortScanner 132 137
Port not in use : 132
Port not in use : 133
Port not in use : 134
Port in use : 135
Port not in use : 136
Port not in use : 137

Seudo Code ::

```
package server;

import java.io.*;
import java.net.*;

/**
 *
 * @author Fazeel
 */
public class Server {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        // TODO code application logic here
        for(int i=1;i<=65000;i++){
            try{
            Socket s=new Socket("127.0.0.1",i);
            System.out.println("port in use "+i);
            s.close();}
            catch(IOException s){

            }
            System.out.println("port not in use "+i);
        }
    }
}
```

Output & Display ::



3. Write a small server that accepts socket connection on port 2020. Develop a client application that connects to the server.
    a. Using BufferedOutputStream, write to the server “Hello”
    b. The server should respond with the word Hello
4. Modify the Task 3 to develop an echo server

Seudo Code ::

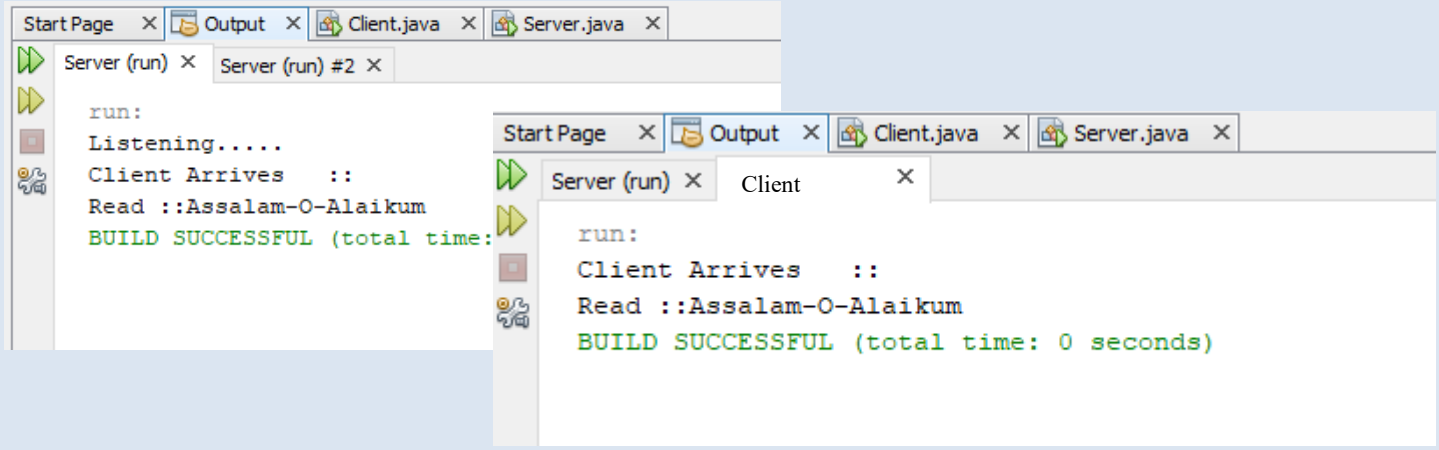
```
package server;
/**
 * @author 7500
 // Server
 */
import java.net.*;
import java.io.*;
public class Server {
public static void main(String[] args){
      // TODO code application logic here
      try{
      ServerSocket Seve=new ServerSocket(5000);
         System.out.println("Listening.....");
      Socket sp=    Seve.accept();
         DataOutputStream dos=new DataOutputStream(sp.getOutputStream());
         DataInputStream dis=new DataInputStream(sp.getInputStream());
         System.out.println("Client Arrives   ::");

           // FristPart
//           System.out.println("Read ::"+dis.readUTF());
//           dos.writeUTF("Wa Alaikum Assalam");
           //Second part
           String data=dis.readUTF();
           System.out.println("Read ::"+data);
           dos.writeUTF(data);

         Seve.close();
      }catch(Exception ex){
         System.out.println(""+ex);
        ex.printStackTrace();
      }   }   }
```

//Cleint

```
package server;
```

```
import java.io.DataInputStream;
import java.io.*;
import java.net.*;

/**
 *
 * @author 7500
 */
public class Client {
   public static void main(String[] args) {
        try{
      Socket sp=new Socket("localhost",5000);
        System.out.println("Client Arrives   ::");
         DataOutputStream dos=new DataOutputStream(sp.getOutputStream());

         DataInputStream dis=new DataInputStream(sp.getInputStream());

         dos.writeUTF("Assalam-O-Alaikum");
         System.out.println("Read ::"+dis.readUTF());
           //System.out.println(""+dos.writeUTF(dis.readUTF()));

         sp.close();
      }catch(Exception ex){
         System.out.println(""+ex);
        ex.printStackTrace();
      }
   }
   }
```

Output & Display ::



To study and implement multi-threading in Java

Instructions:

1. A thread is an independent unit of execution.
2. In Java, the Runnable interface and Thread class of package java.lang are used for implementation of thread
3. To implement a thread, the desired class must implement the Runnable interface and provide the run() method.

```
public class MyThread implements Runnable {
  public void run() {
        //implementation of thread
  }
}
```

4. The Thread class can then be used to start a thread as follows:

```
public class TestThread
{
   public static void main( String[] args )
   {
        MyThread m = new MyThread();
        Thread t = new Thread(m);
        m.start();
   }
}
```

Lab Tasks:

1. Write a class that implements Runnable. Define a constructor that takes the name of the thread as argument. The thread upon execution will print the name of the thread in a while loop. Define and run 5 thread objects. What output do you see?
2. In task 1, modify the run method to randomly sleep the thread for few milliseconds. Observe the output.
3. Create a multi-threaded client server application in Java.

Seudo Code ::

```
package thread;
/**
 * @author 7500
 // Thread
 */

public class MyThread implements Runnable{
   private  String name;
   @Override
   public void run() {
   //   while(true){
         System.out.println(name);
      //     Thread.sleep(1000)   }
  }       public MyThread(String name) {        this.name = name ;   }
```

```
   public static void main(String[] args) {
  try{
     for(int i=1;i<=5;i++){
      MyThread m = new MyThread("Test"+i);
            Thread t = new Thread(m);
 t.start();
     }
    }catch(Exception ex){
      System.out.println("");
     }
    }
}
```

Output & Display ::

```
run:
Test1
Test2
Test3
Test4
Test5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Seudo Code::

```
Package thread;
import java.io.*;
import java.net.*;
import java.nio.CharBuffer;
public class Myrunable implements Runnable{
   private Socket s;
    public  Myrunable(Socket s) {
       this.s = s;
    }
   @Override
    public void run() {
         try  {
         DataInputStream dis = new DataInputStream(s.getInputStream());
         DataOutputStream dos = new DataOutputStream(s.getOutputStream());
         dos.writeUTF(dis.readUTF());
         s.close();
        }
```

```
      catch (Exception e)
      {
         e.printStackTrace();
      }
}
   public static void main(String[] args) {
       try{
     ServerSocket ss = new ServerSocket(4000);
   System.out.println("Server Running....::");
     while(true)
     {
        Socket s = ss.accept();
       Myrunable st = new Myrunable(s);
        Thread t = new Thread (st);
        t.start();
     }

      }catch(Exception ex){
         System.out.println(""+ex);
      }
   }
}
```

Output & Display ::

```
run:
Server Running....::
```

To study and execute basic Linux commands on a terminal

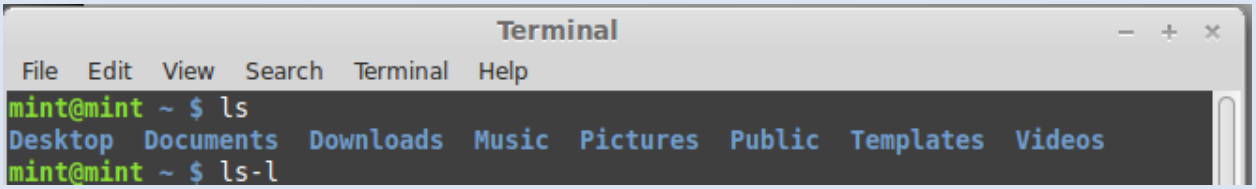
Linux is a Unix-like and mostly POSIX-compliant computer operating system (OS) assembled under the model of free and open-source software development and distribution. In this lab, we will work on Ubuntu, one of the flavors of Linux. For this purpose, we will use virtualization environment.

Lab Tasks:

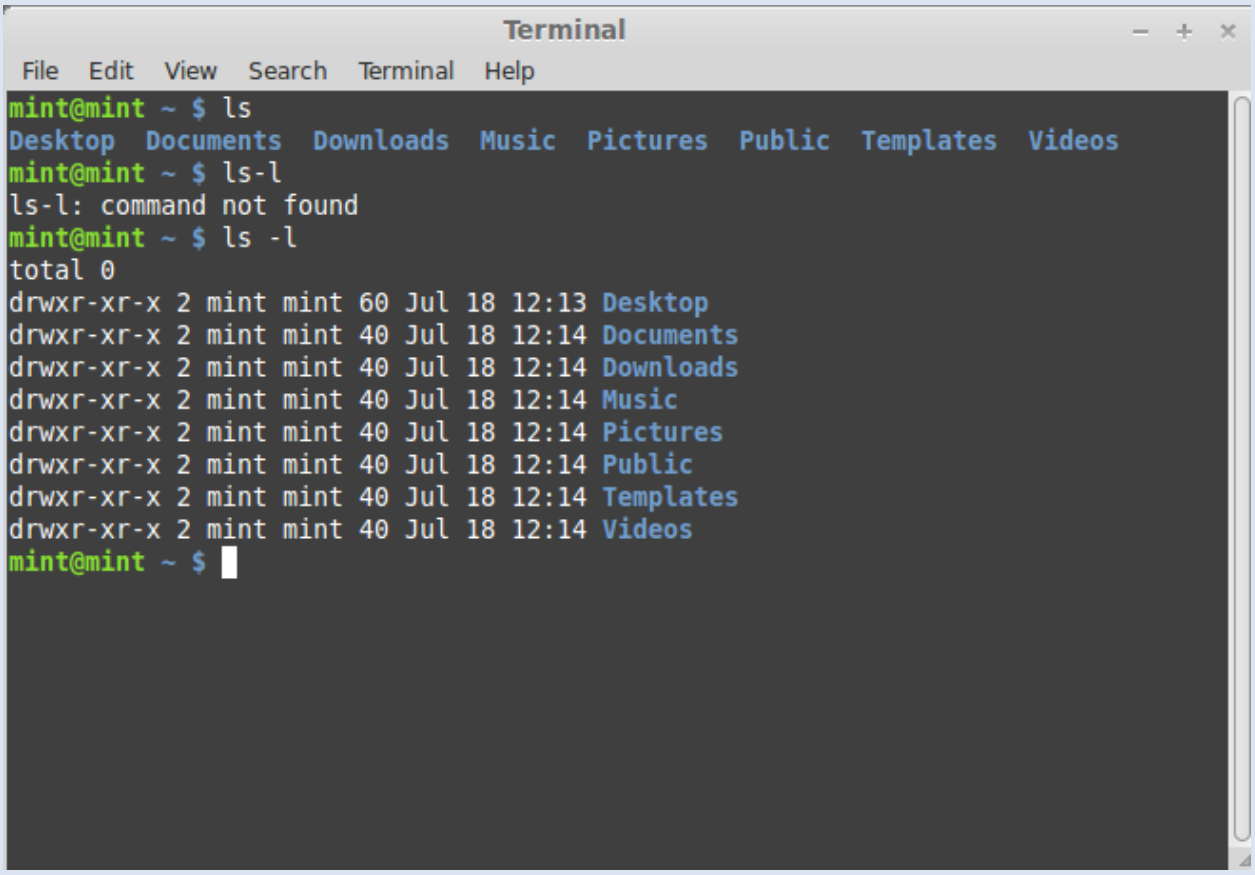
1. Using ls command find out the contents of current directory
2. What are the permissions for normal user, group and world for each file
3. Find out the name of current working directory
4. Create a new folder named “lab os” using the mkdir command
5. Switch to the directory “lab os”
6. Create a file in the directory named “lab4.txt” using touch command
7. List down the contents of file using cat command. Try using “more” and “less” option
8. Find out the space consumed by directory using “du” command
9. Copy the file to parent directory using cp command
10. Remove the file using rm command
11. Remove the directory using rmdir command
12. Check the free space on disk using df command
13. Change the password of the user using passwd command
14. Switch to super user, using the command “su
15. Using the history command, list down the commands run on the terminal window

---

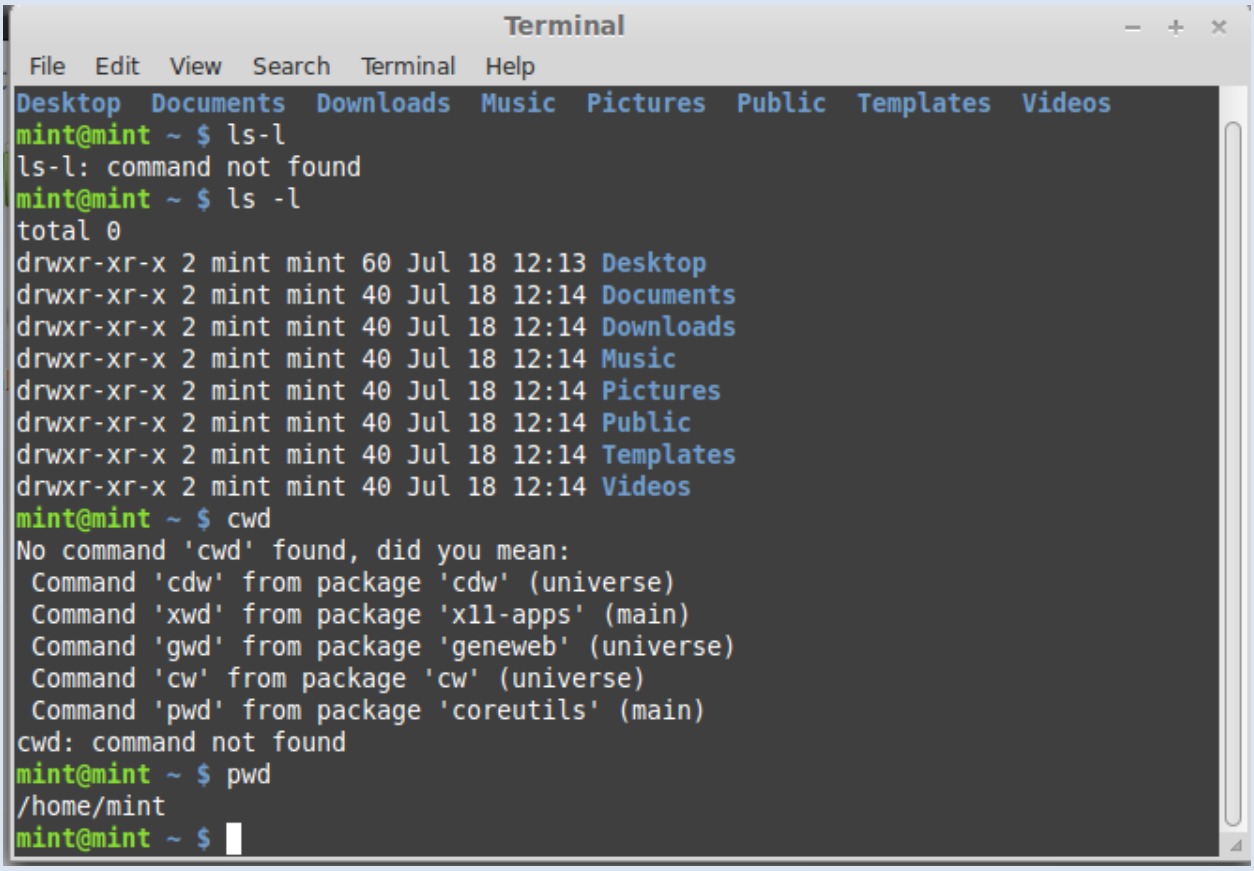
1. Using ls command find out the contents of current directory



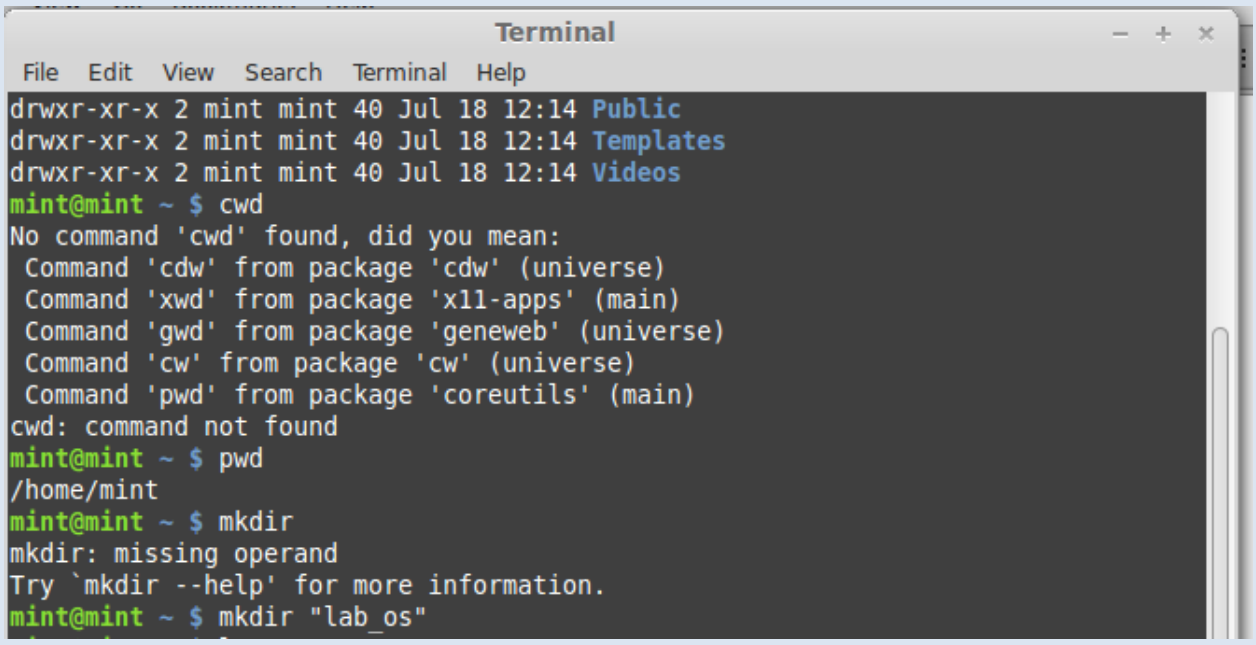
2. What are the permissions for normal user, group and world for each file



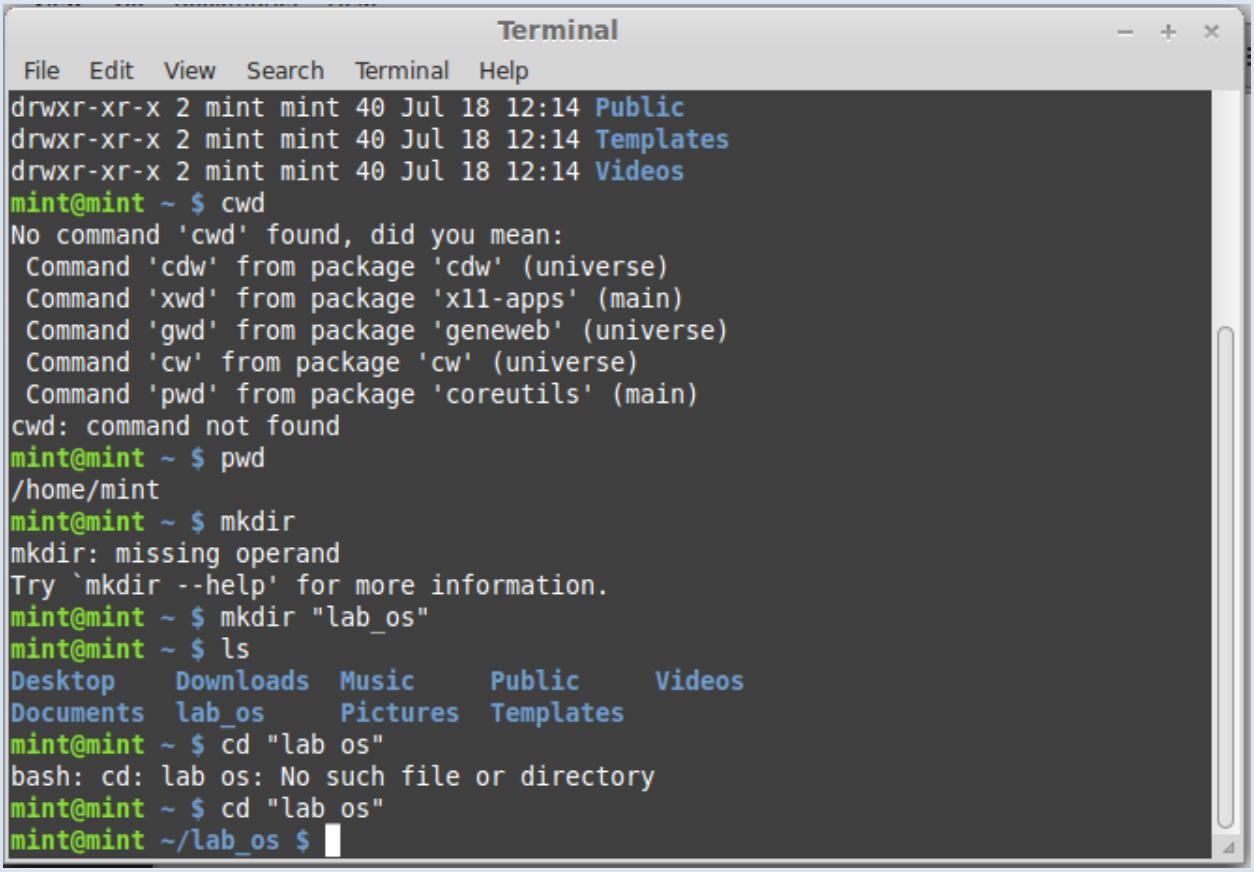
3. Find out the name of current working directory



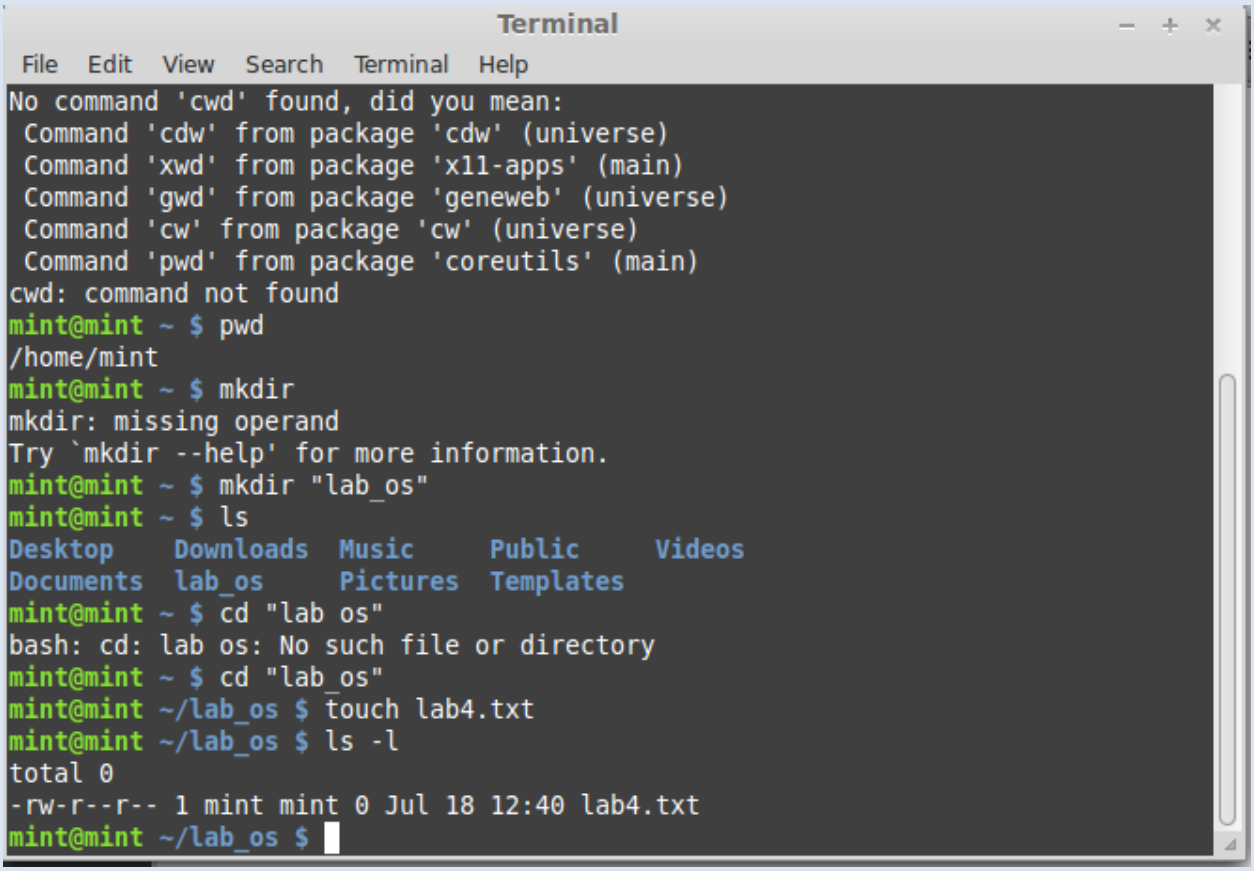
4. Create a new folder named “lab os” using the mkdir command



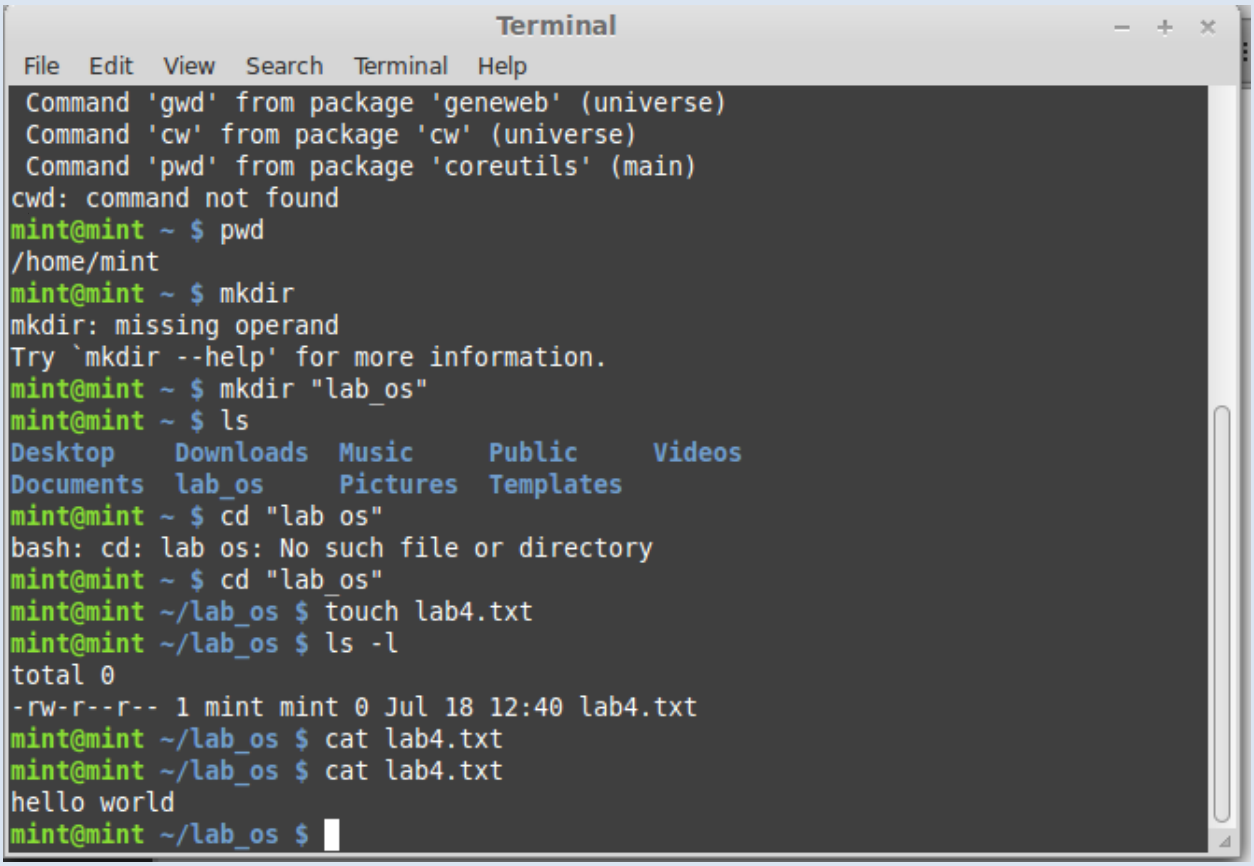
5. Switch to the directory “lab os”



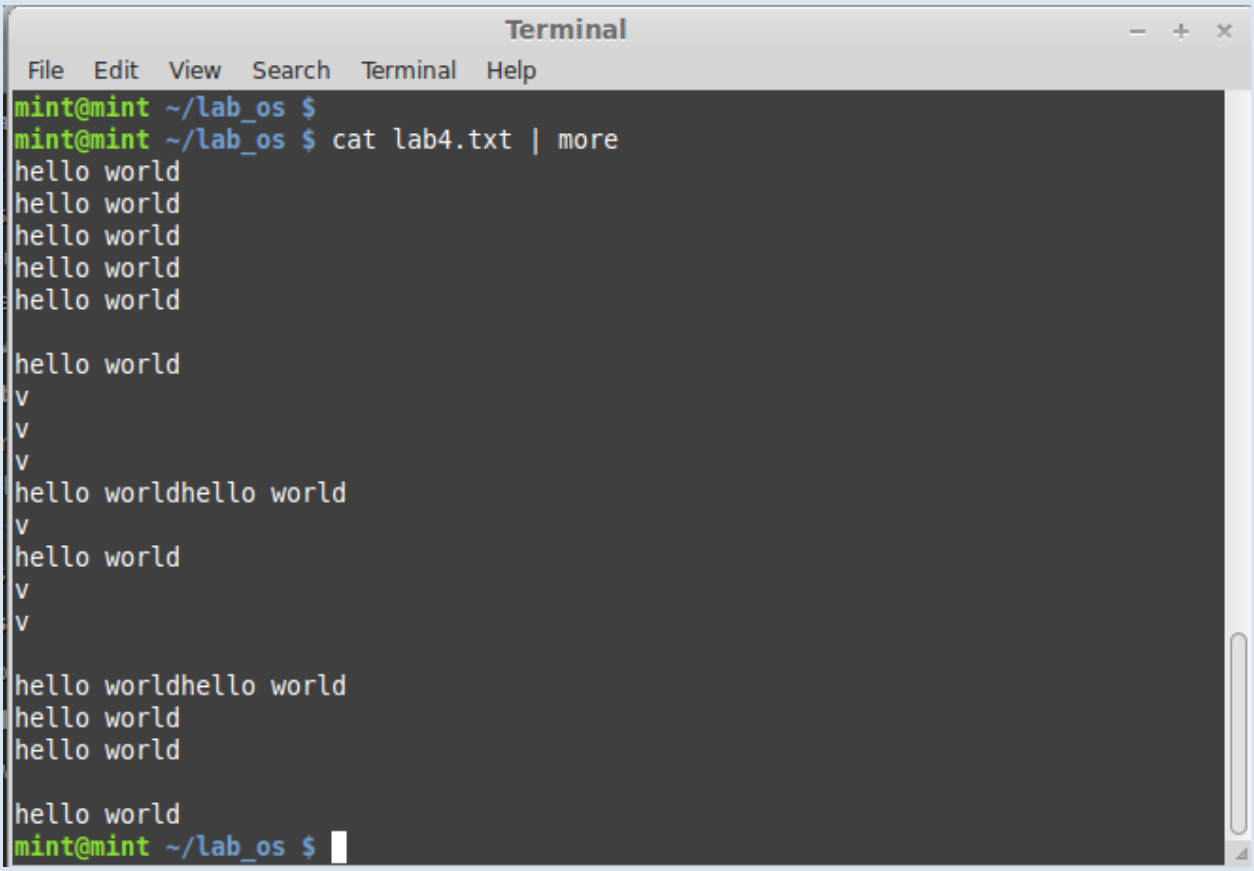
6. Create a file in the directory named “lab4.txt” using touch command



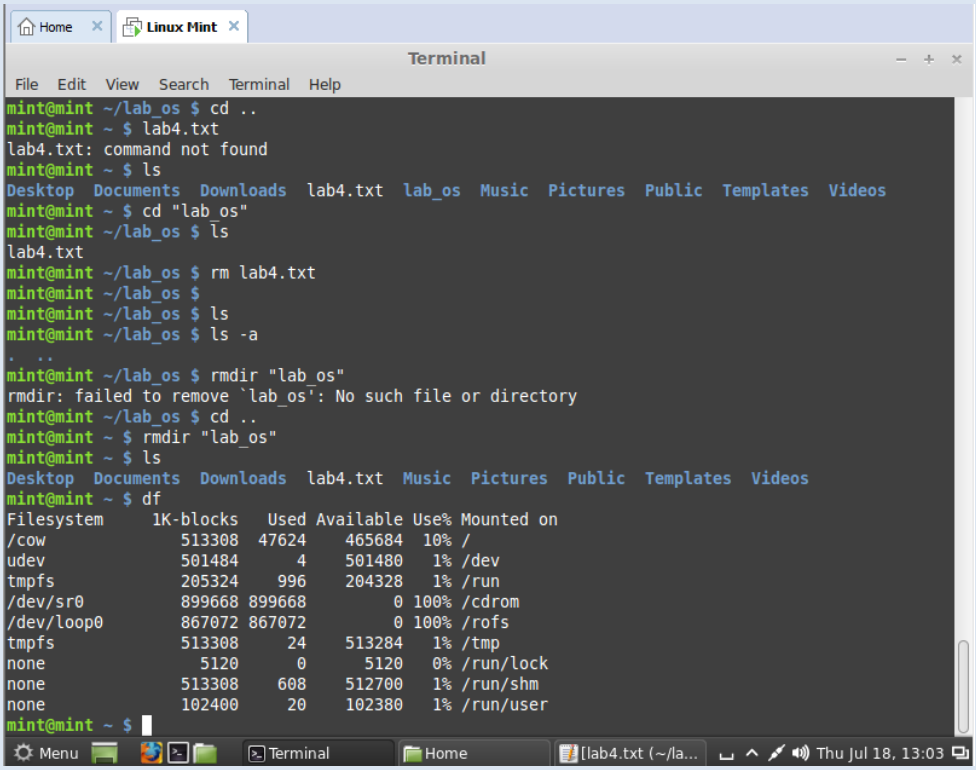
7. List down the contents of file using cat command. Try using “more” and “less” option


Terminal
File Edit View Search Terminal Help
Command 'gwd' from package 'geneweb' (universe)
Command 'cw' from package 'cw' (universe)
Command 'pwd' from package 'coreutils' (main)
cwd: command not found
mint@mint ~ $ pwd
/home/mint
mint@mint ~ $ mkdir
mkdir: missing operand
Try `mkdir --help' for more information.
mint@mint ~ $ mkdir "lab_os"
mint@mint ~ $ ls
Desktop Downloads Music Public Videos
Documents lab_os Pictures Templates
mint@mint ~ $ cd "lab os"
bash: cd: lab os: No such file or directory
mint@mint ~ $ cd "lab_os"
mint@mint ~/lab_os $ touch lab4.txt
mint@mint ~/lab_os $ ls -l
total 0
-rw-r--r-- 1 mint mint 0 Jul 18 12:40 lab4.txt
mint@mint ~/lab_os $ cat lab4.txt
mint@mint ~/lab_os $ cat lab4.txt
hello world
mint@mint ~/lab_os $

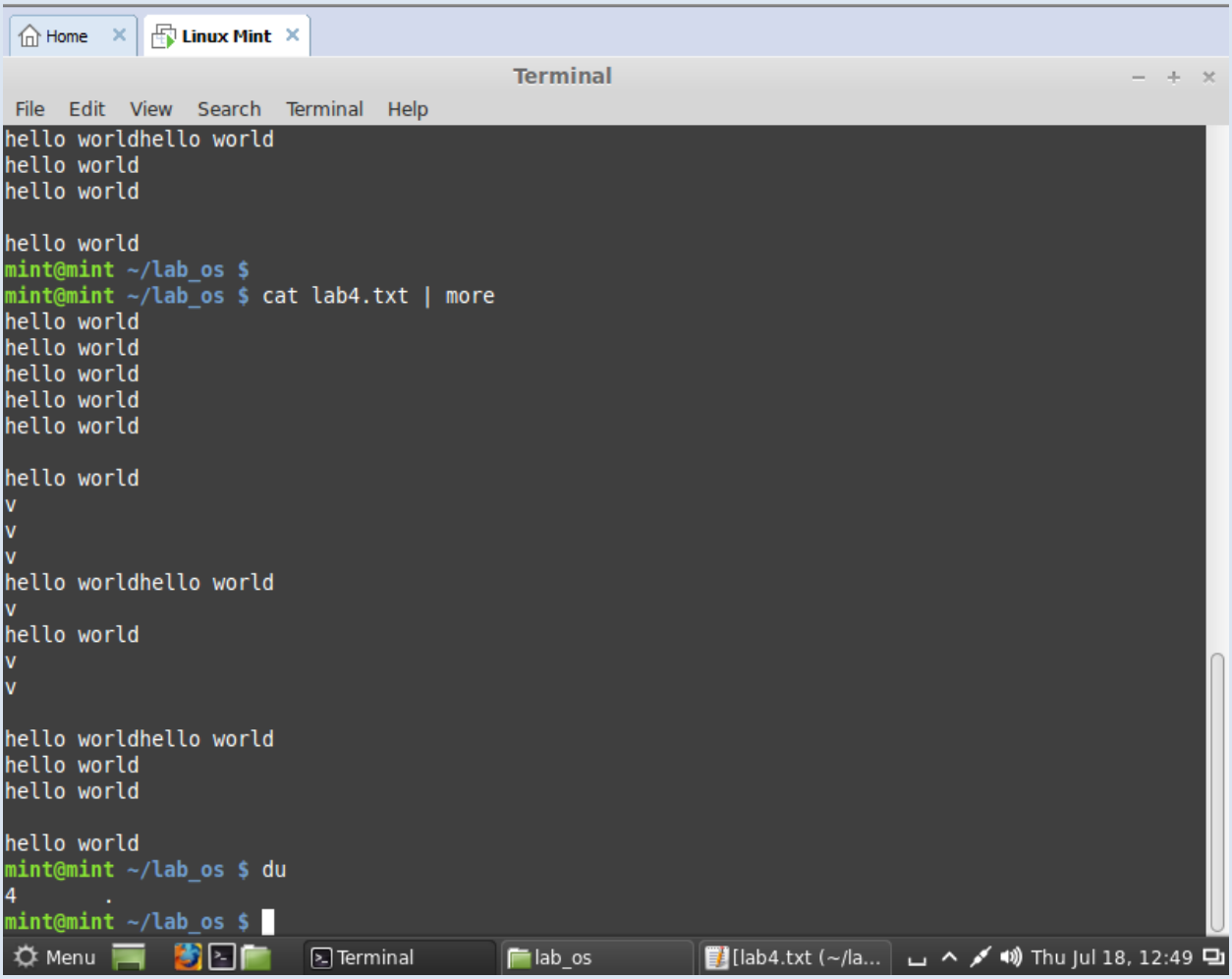
8. More:



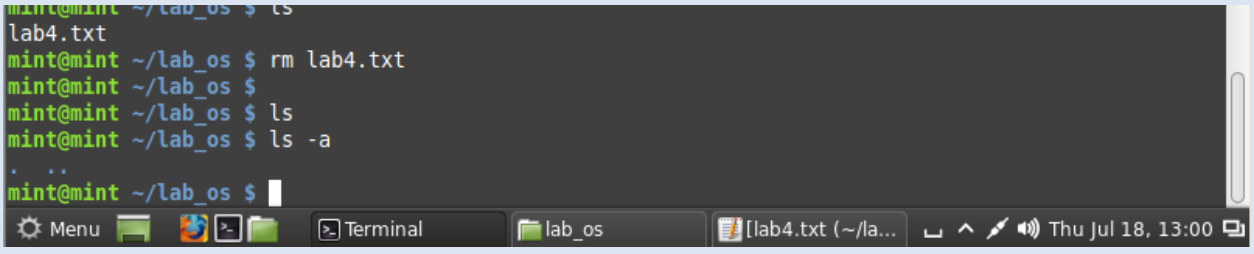
9. Find out the space consumed by directory using “du” command



10. Copy the file to parent directory using cp command



11. Remove the file using rm command



```
mint@mint ~ $ df
Filesystem     1K-blocks   Used Available Use% Mounted on
/cow              513308  47624    465684  10% /
udev              501484      4    501480   1% /dev
tmpfs             205324    996    204328   1% /run
/dev/sr0          899668 899668         0 100% /cdrom
/dev/loop0        867072 867072         0 100% /rofs
tmpfs             513308     24    513284   1% /tmp
none                5120      0      5120   0% /run/lock
none              513308    608    512700   1% /run/shm
none              102400     20    102380   1% /run/user
mint@mint ~ $
```

Menu | Terminal | Home | [lab4.txt (~/la... | Thu Jul 18, 13:04

12. Remove the directory using rmdir command

```
passwd: password unchanged
mint@mint ~ $ passwd
Changing password for mint.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
mint@mint ~ $
```

Menu | Terminal | Home | [lab4.txt (~/la... | Thu Jul 18, 13:12

13. Check the free space on disk using df command

mint@mint: ~

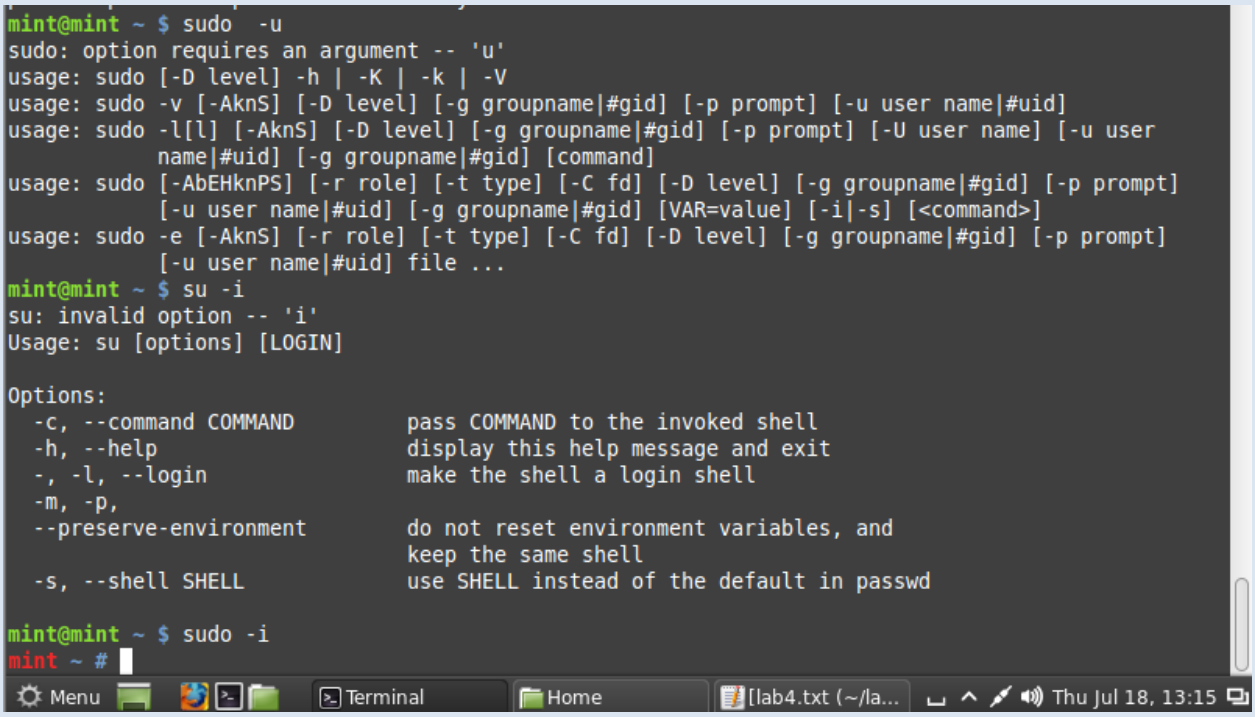
File Edit View Search Terminal Help

```
bash: cd: /Desktop: No such file or directory
mint@mint:~$ cd Desktop
mint@mint:~/Desktop$ mkdir file3.txt
mint@mint:~/Desktop$ cd ..
mint@mint:~$ ls
Desktop    Downloads  Pictures  Templates  file1.txt
Documents  Music      Public    Videos     file2.txt
mint@mint:~$ ls Desktop
file3.txt  ubiquity.desktop
mint@mint:~$ cp /Desktop /CopyRE
cp: cannot stat '/Desktop': No such file or directory
mint@mint:~$ cp /Desktop /file3.txt
cp: cannot stat '/Desktop': No such file or directory
mint@mint:~$ cp /file3.txt /Desktop
cp: cannot stat '/file3.txt': No such file or directory
mint@mint:~$ cp -R file3.txt file1.txt
cp: cannot stat 'file3.txt': No such file or directory
mint@mint:~$ cp -R file1.txt file2.txt
mint@mint:~$ pwd
/home/mint
mint@mint:~$ ls
Desktop    Downloads  Pictures  Templates  file1.txt
Documents  Music      Public    Videos     file2.txt
mint@mint:~$
```

14. Change the password of the user using passwd command

15. Switch to super user, using the command “su”

16. Using the history command, list down the commands run on the terminal window



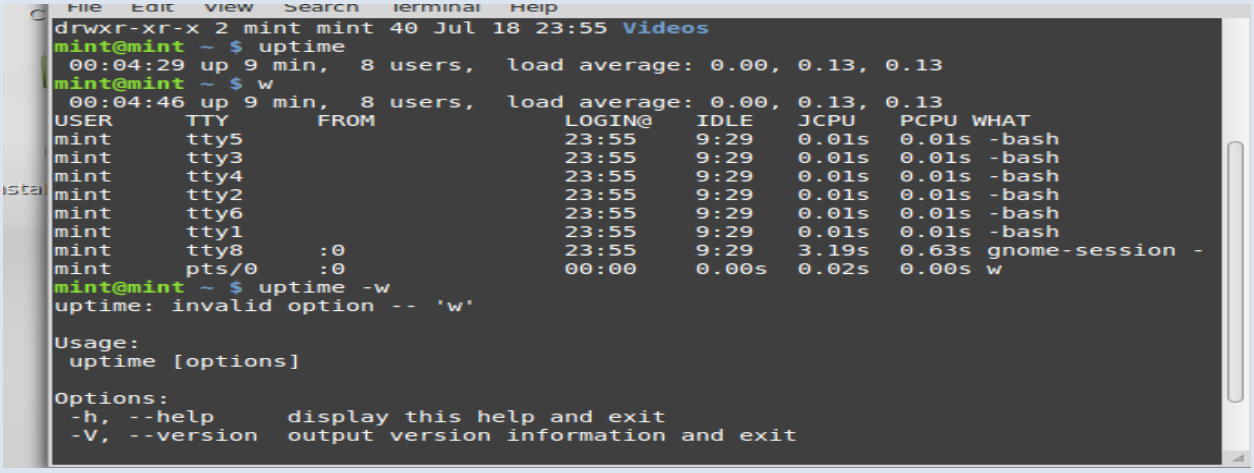
To study and execute system administration commands on a terminal

Instructions:

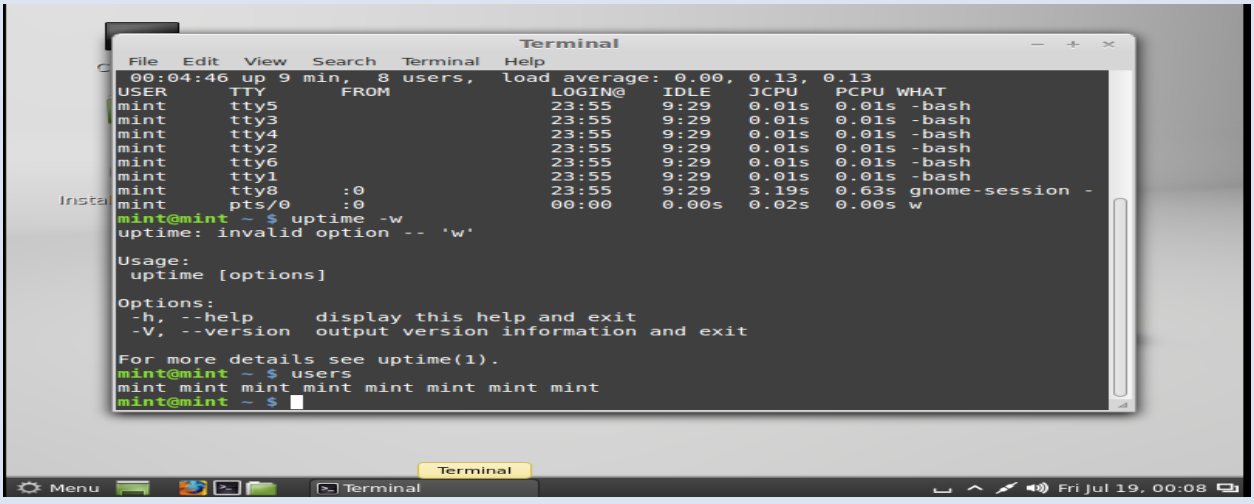
Linux comprises a set of commands for basic system administration. In this lab, we will study these commands.

Lab Tasks:

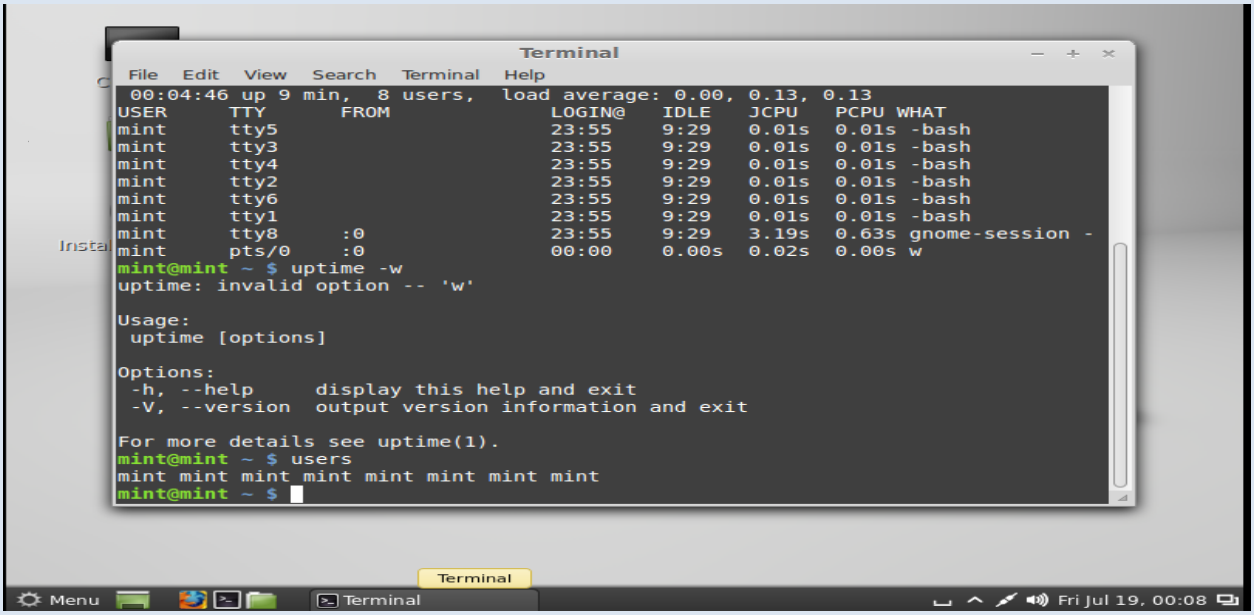
1. Using the ‘uptime' command, since how long your system is running and the number of users that are currently logged in.



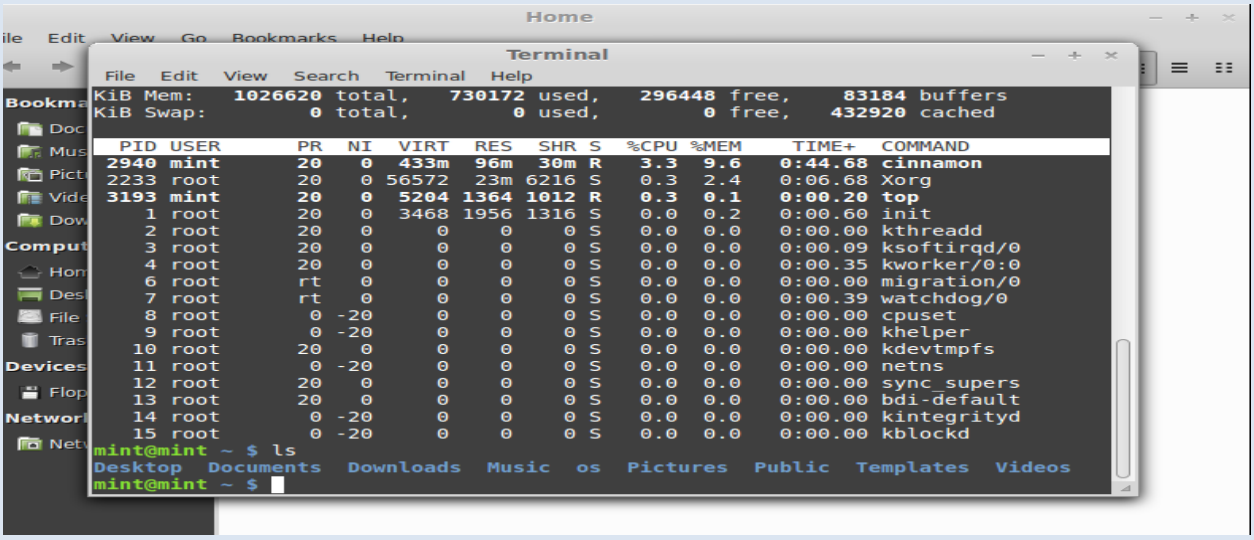
2. Using the ‘w’, display the users currently logged in and their process along-with load averages



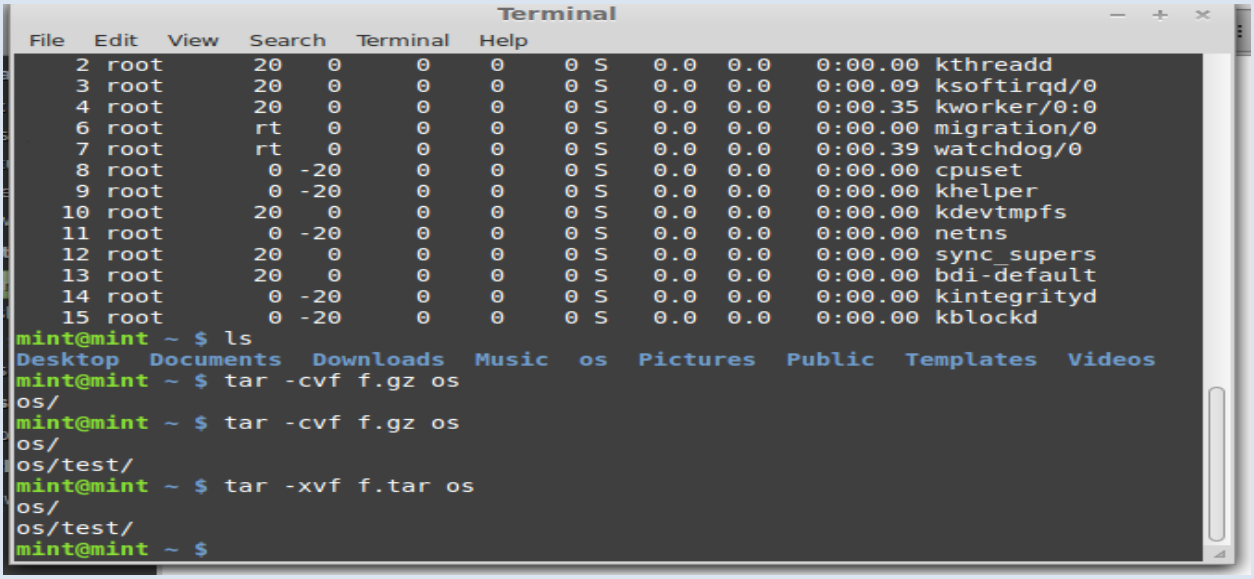
3. Using the ‘users’ command, display the currently logged in users.



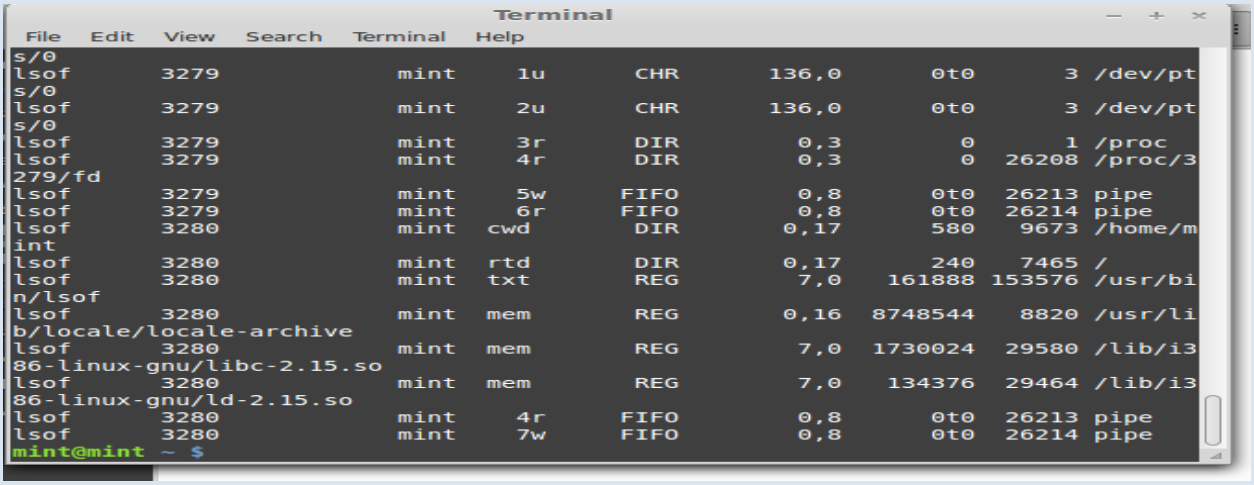
4. Using the ‘top’ command, display processor activity of your system and also displays tasks managed by kernel in real-time.



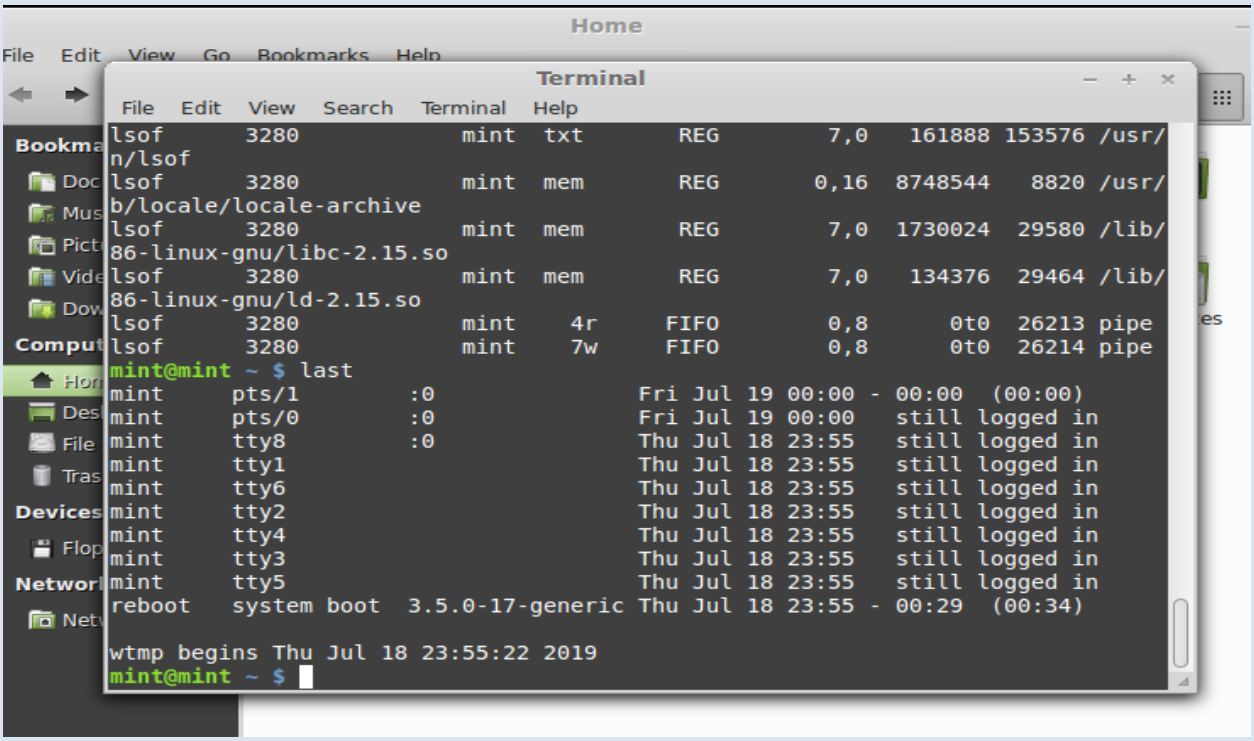
5. Using ‘tar’ command, compress your home directory in Linux.



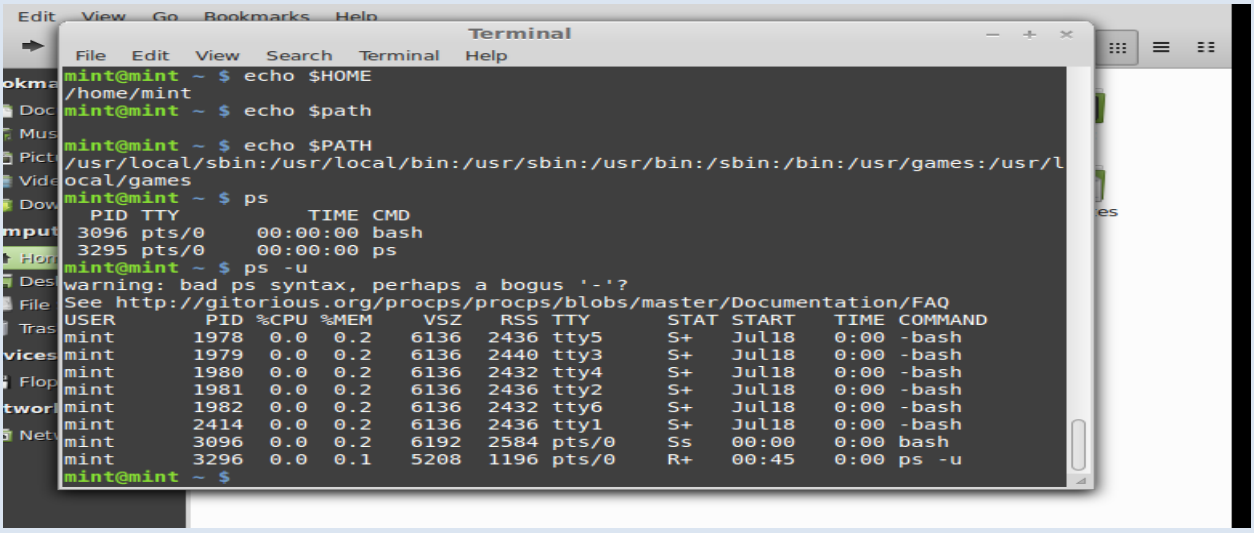
6. ‘lsof’ command to list all open files



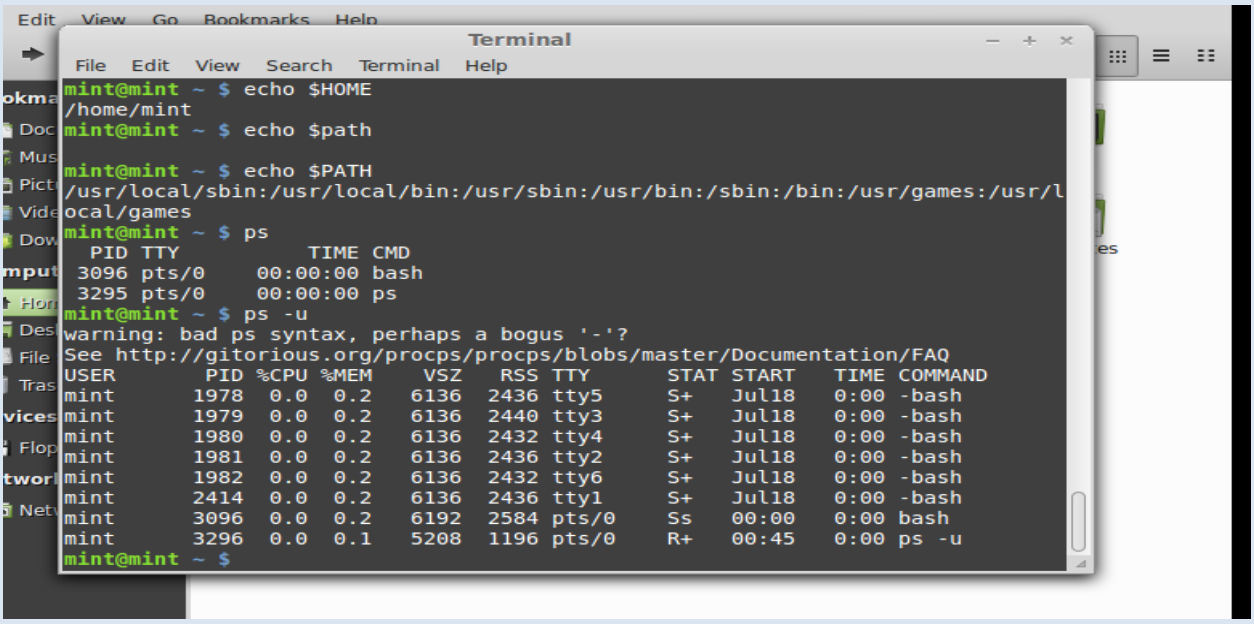
7. Using the ‘last’ command, watch activity of ‘mint’ user in the system



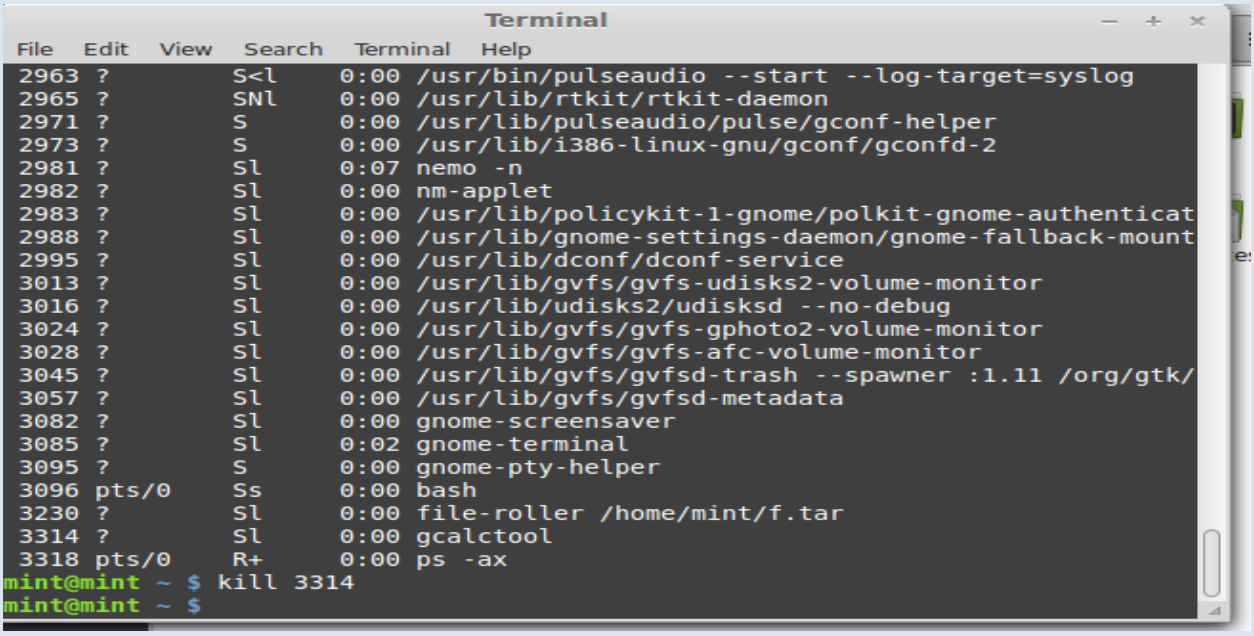
8. Using the ‘env’ command, lists all the environment variables of your system. Use ‘echo’ command to print values of $HOME and $PATH



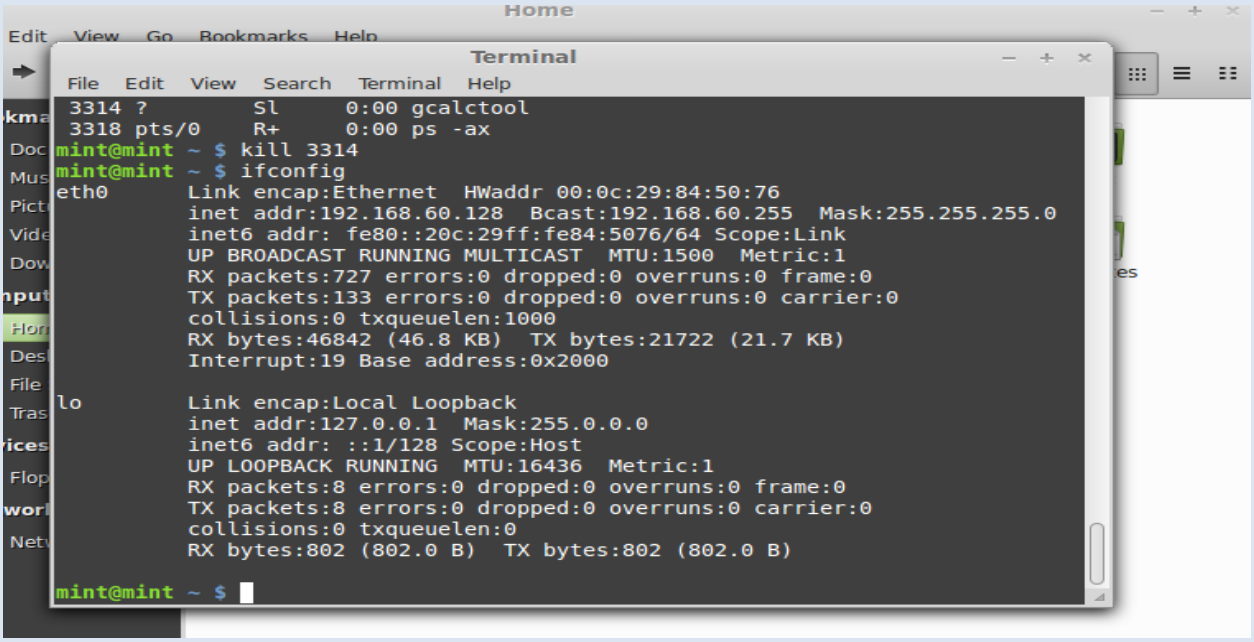
9. The ‘ps’ command displays about processes running in the system. Try option –ax, -u.



10. The ‘kill’ command can be used to terminate process. Using this command terminate some processes of your system



11. ‘ifconfig’ command is used to show the configuration of internet on LINUX. Use this command to find IP and MAC address of your computer



12. Using the 'netstat' command, show the status of your network

```
root@kali:~# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 kali:33244             151.101.38.49:https     ESTABLISHED
tcp        0      0 kali:56346             oscp-router01.gno:https ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    24469    /run/user/0/systemd/n
otify
unix  2      [ ]         DGRAM                    19127    /run/user/131/systemd
/notify
unix  3      [ ]         DGRAM                    14532    /run/systemd/notify
unix  18     [ ]         DGRAM                    14542    /run/systemd/journal/
dev-log
unix  2      [ ]         DGRAM                    14551    /run/systemd/journal/
syslog
unix  7      [ ]         DGRAM                    14577    /run/systemd/journal/
```

13. Using the ping command, to ping your localhost

```
root@kali:~# ping localhost
PING localhost(localhost (::1)) 56 data bytes
64 bytes from localhost (::1): icmp_seq=1 ttl=64 time=0.037 ms
64 bytes from localhost (::1): icmp_seq=2 ttl=64 time=0.050 ms
64 bytes from localhost (::1): icmp_seq=3 ttl=64 time=0.048 ms
64 bytes from localhost (::1): icmp_seq=4 ttl=64 time=0.052 ms
64 bytes from localhost (::1): icmp_seq=5 ttl=64 time=0.059 ms
64 bytes from localhost (::1): icmp_seq=6 ttl=64 time=0.065 ms
64 bytes from localhost (::1): icmp_seq=7 ttl=64 time=0.095 ms
64 bytes from localhost (::1): icmp_seq=8 ttl=64 time=0.106 ms
64 bytes from localhost (::1): icmp_seq=9 ttl=64 time=0.059 ms
64 bytes from localhost (::1): icmp_seq=10 ttl=64 time=0.041 ms
```

14. Create a group named ‘student’ using groupadd
15. Create a file named ‘hello.txt’
16. Using the ‘useradd’ command create a user with your name in the group student
17. Change the owner of hello.txt to user you just created
18. Change the group owner of hello.txt to group student

```
root@kali:~# sudo groupadd std
root@kali:~# touch hello.txt
root@kali:~# ls
core     Documents  hello.txt  Pictures  Templates   Videos
Desktop  Downloads  Music      Public    Update.apk
root@kali:~# sudo useradd fazeel
root@kali:~# ls -l
total 1628
-rw------- 1 root root 1921024 Nov 11 13:13 core
drwxr-xr-x 4 root root    4096 Oct 22 09:36 Desktop
drwxr-xr-x 2 root root    4096 Aug 30 14:30 Documents
drwxr-xr-x 2 root root    4096 Aug 30 14:30 Downloads
-rw-r--r-- 1 root root       0 Nov 11 13:19 hello.txt
drwxr-xr-x 2 root root    4096 Aug 30 14:30 Music
drwxr-xr-x 2 root root    4096 Aug 30 14:30 Pictures
drwxr-xr-x 2 root root    4096 Aug 30 14:30 Public
drwxr-xr-x 2 root root    4096 Aug 30 14:30 Templates
-rw-r--r-- 1 root root   10087 Aug 31 07:40 Update.apk
drwxr-xr-x 2 root root    4096 Aug 30 14:30 Videos
```

```
root@kali:~# chown fazeel hello.txt
root@kali:~# chgrp std hello.txt
root@kali:~# ls -l
total 1628
-rw------- 1 root   root 1921024 Nov 11 13:13 core
drwxr-xr-x 4 root   root    4096 Oct 22 09:36 Desktop
drwxr-xr-x 2 root   root    4096 Aug 30 14:30 Documents
drwxr-xr-x 2 root   root    4096 Aug 30 14:30 Downloads
-rw-r--r-- 1 fazeel std        0 Nov 11 13:19 hello.txt
drwxr-xr-x 2 root   root    4096 Aug 30 14:30 Music
drwxr-xr-x 2 root   root    4096 Aug 30 14:30 Pictures
drwxr-xr-x 2 root   root    4096 Aug 30 14:30 Public
drwxr-xr-x 2 root   root    4096 Aug 30 14:30 Templates
-rw-r--r-- 1 root   root   10087 Aug 31 07:40 Update.apk
drwxr-xr-x 2 root   root    4096 Aug 30 14:30 Videos
```

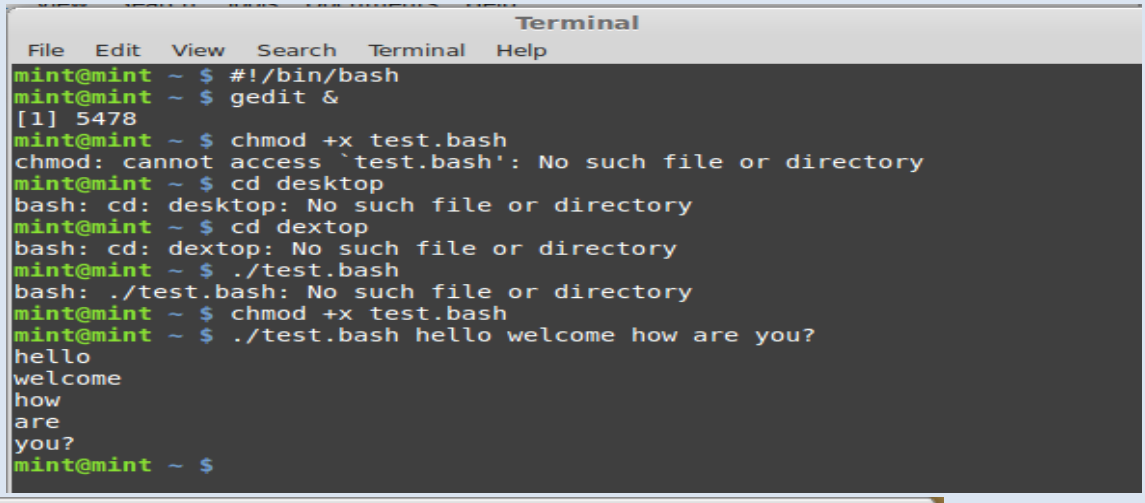
To study and implement shell programming in Linux

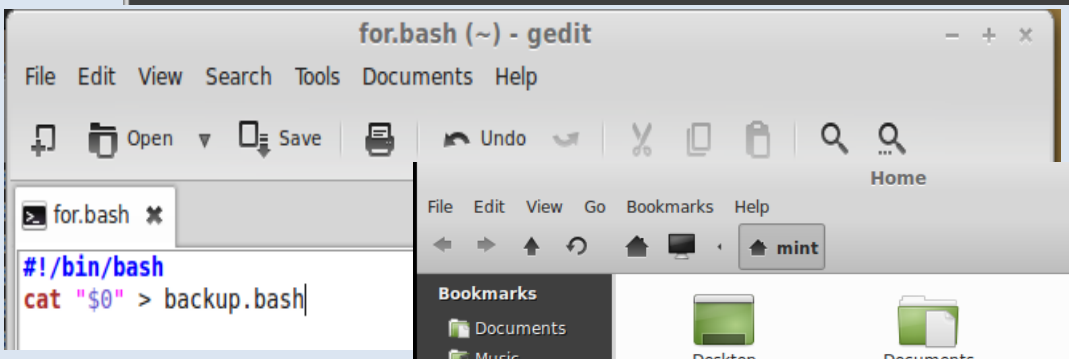
Instructions:

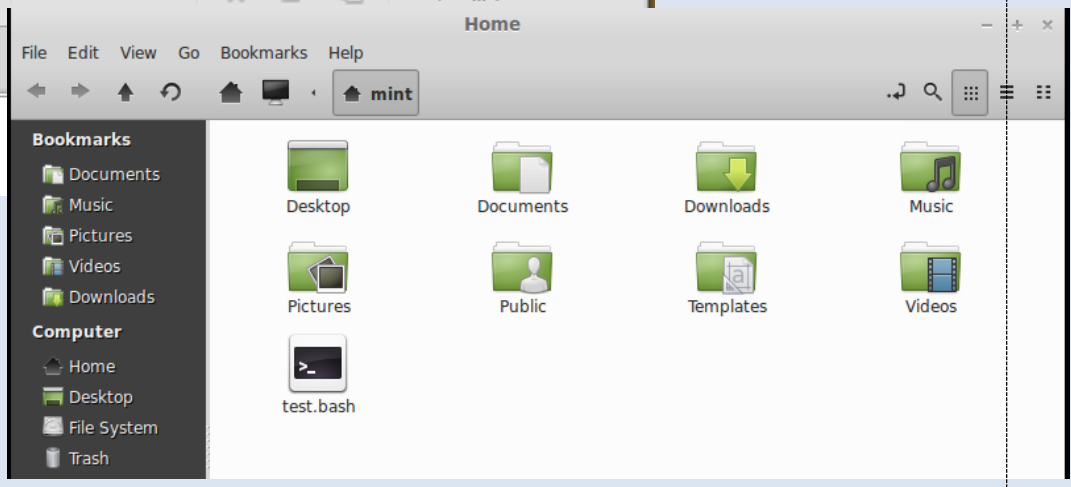
1. A shell script is a computer program designed to be run by the Unix shell, a command-line interpreter
2. The various dialects of shell scripts are considered to be scripting languages.
3. Typical operations performed by shell scripts include file manipulation, program execution, and printing text.

Lab Tasks:

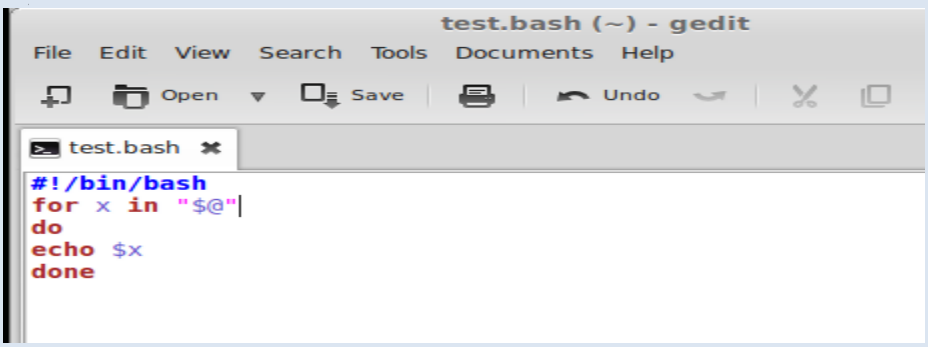
1. Write a script that backs itself up, that is, copies itself to a file named backup.sh.
   Hint: Use the cat command



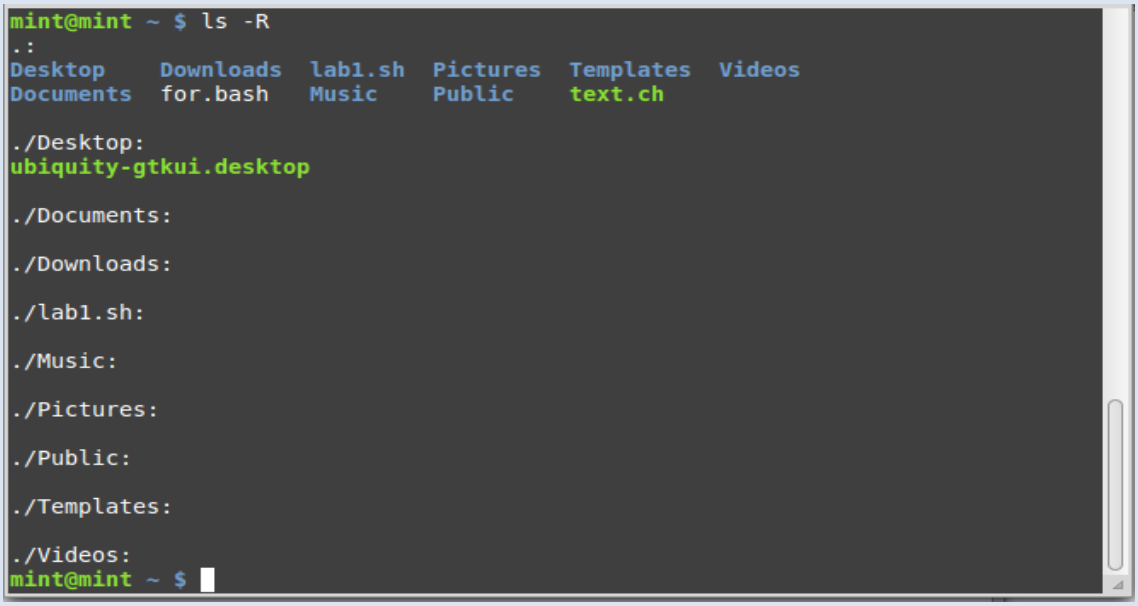




2. Write a script that echoes itself to stdout, but backwards.
   Hint: Use the tac command



3. Perform a recursive directory listing on the user's home directory and save the information to a file.



4. Write a script that reads each line of a target file, then writes the line back to stdout, but with an extra blank line following. This has the effect of double-spacing the file.

Open

double_space.bash
~/Desktop/LAB OS

```
#!/bin/bash
echo "Test Program"

sed '$!G'
```

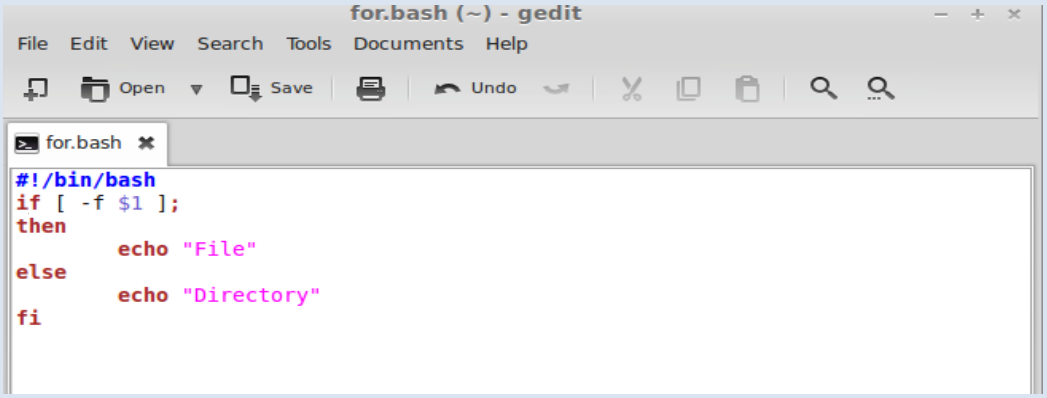
```
root@kali:~/Desktop/LAB OS# ./double_space.bash
Test Program
12
3
12

3
3

2
3

1
2
```

5. Write a shell script that takes a command –line argument and reports on whether it is directory, a file

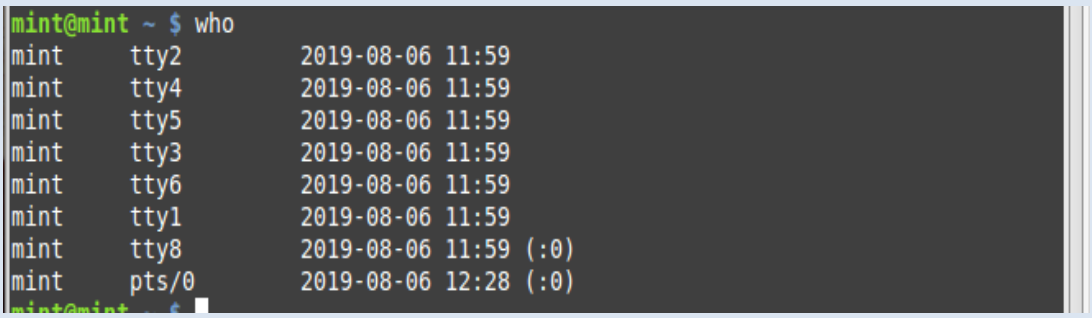
Terminal

File Edit View Search Terminal Help

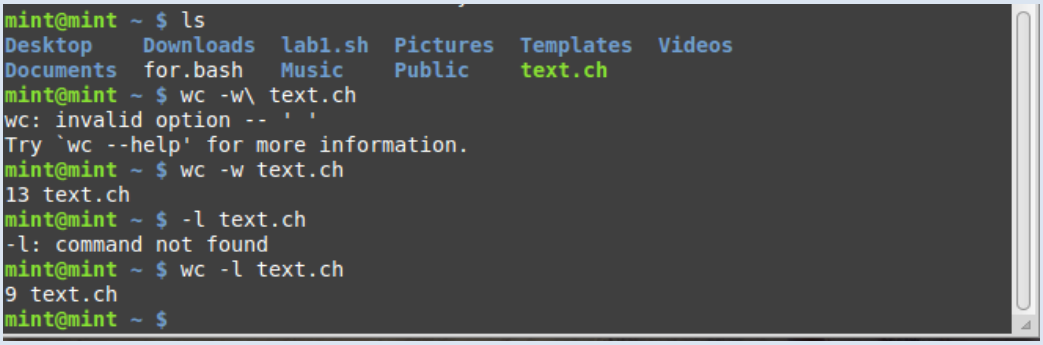
```
mint@mint ~ $ mkdir lab1.sh
mint@mint ~ $ gedit text.ch &
[1] 3329
mint@mint ~ $ chmod +x text.ch
mint@mint ~ $ ./text.ch
./text.ch: line 2: if[-f ]: command not found
./text.ch: line 3: syntax error near unexpected token `then'
./text.ch: line 3: `then'
mint@mint ~ $ ./text.ch
./text.ch: line 2: if[-f ]: command not found
./text.ch: line 3: syntax error near unexpected token `then'
./text.ch: line 3: `then'
mint@mint ~ $ ./text.ch
./text.ch: line 2: if[ -f  ]: command not found
./text.ch: line 3: syntax error near unexpected token `then'
./text.ch: line 3: `then'
mint@mint ~ $ ./text.ch
file
mint@mint ~ $ ./text.ch
file
mint@mint ~ $ ./text.ch lab1.sh
directory
mint@mint ~ $
```

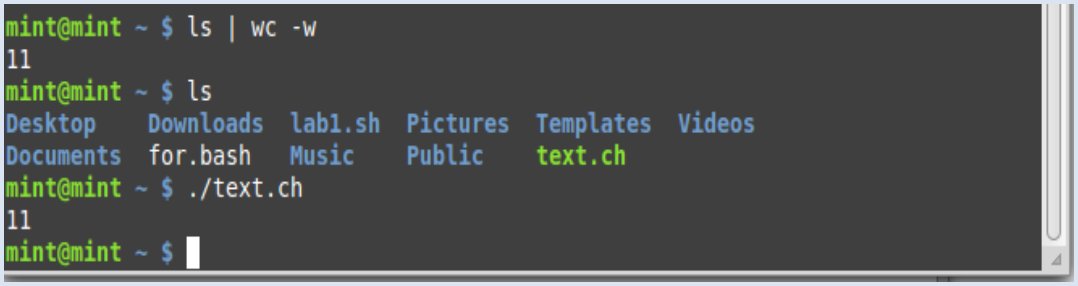


6. Write a shell script program to display list of user currently logged in.



7. Shell script program to count number of files in a Directory.





To study and implement concurrency control techniques in Java

Java provides the synchronized key word for implementing concurrency control while using multi-threaded applications. In this lab, you will learn how to implement these techniques.

Instructions:

Create the following program in Java:

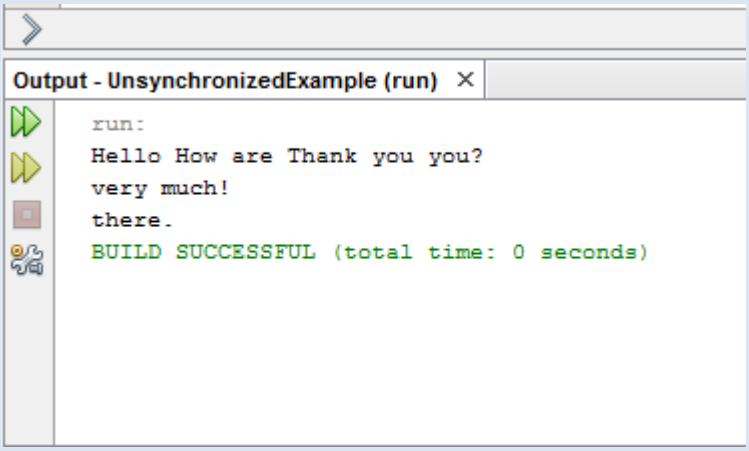
```
public class UnsynchronizedExample {
   public static void main(String[] args) {
      new PrintStringsThread("Hello ", "there.");
      new PrintStringsThread("How are ", "you?");
      new PrintStringsThread("Thank you ", "very much!");
   }
}

public class PrintStringsThread implements Runnable {

   Thread thread;
   String str1, str2;

   PrintStringsThread(String str1, String str2) {
      this.str1 = str1;
      this.str2 = str2;
      thread = new Thread(this);
      thread.start();
   }

   public void run() {
      TwoStrings.print(str1, str2);
   }

}

public class TwoStrings {

   // This method is not synchronized
   static void print(String str1, String str2) {
      System.out.print(str1);
      try {
         Thread.sleep(500);
      } catch (InterruptedException ie) {
      }
      System.out.println(str2);
   }
```

}

Lab Tasks:

1. What output do you see? Explain the output.
2. Now use the synchronized methods to display the desired result.
3. Now use the synchronized keyword on an object to synchronize.

Output & Display ::

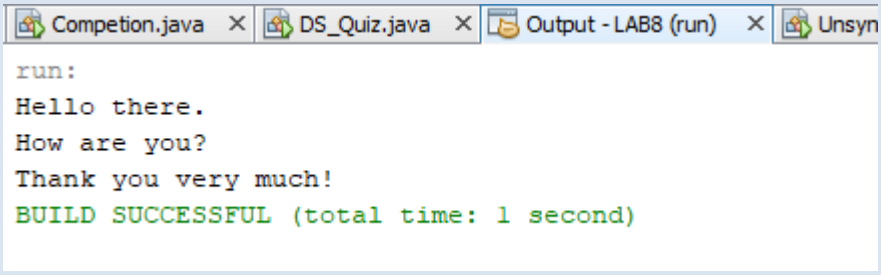


Task 2 ::

Code ::

```
public class TwoStrings {
      static synchronized void print(String str1, String str2) {
       System.out.print(str1);
       try {
          Thread.sleep(500);
       } catch (InterruptedException ie) {      }
       System.out.println(str2);}
}
```

Out Put Display

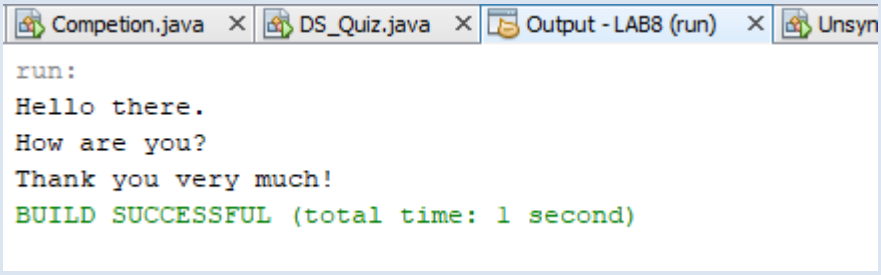


Task #3

Scudo Code in Java ::

```
public class PrintStringsThread implements Runnable {

      Thread thread;
   String str1, str2;
   static  TwoStrings Printer=new TwoStrings();
  PrintStringsThread (String str1, String str2) {
      this.str1 = str1;
      this.str2 = str2;
      thread = new Thread(this);
      thread.start();
   }
     @Override
   public void run() {
      synchronized(Printer){
      Printer.print(str1, str2);
      }
   }
}
```

Out Put Display



To study and implement process scheduling algorithms in Java

Instructions:

In this lab, we will implement different CPU scheduling techniques.

Lab Tasks

1. **Shortest Job First:** The number of processes and burst time is input from the user. The program should then print total access time, burst time and wait time for every process. Also print the average wait time.
   Hint: Sort the element based on their burst time

2. Simulate the First Come First Serve and Priority scheduling algorithm.

Scudo Code in Java ::

```
package shortest.job;
import java.util.*;
/**
 *
 * @author Fazeel
 */
public class ShortestJob {
    /**
     * @param args the command line arguments
     */
    static Vector v = new Vector();
    public static void main(String[] args) {
        // TODO code application logic here
          System.out.println("Enter number of processes: ");
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        for(int i=0;i<n;i++) {
            System.out.println("Process  "+ i + " Burst Time: ");
            process p = new process();
            p.burst_time = s.nextInt();
            v.add(p);                   }
        v.sort(new Comparator() {
            public int compare(Object  a, Object b) {
                return ((process)a).burst_time - ((process)b).burst_time;
            }

        });         int c = 0 ;
```

```
        for(int i=0;i<v.size();i++) {
            ((process)v.get(i)).wait_time = c;
            c+=((process)v.get(i)).burst_time;
        }

        System.out.println(v);
    }
}
```

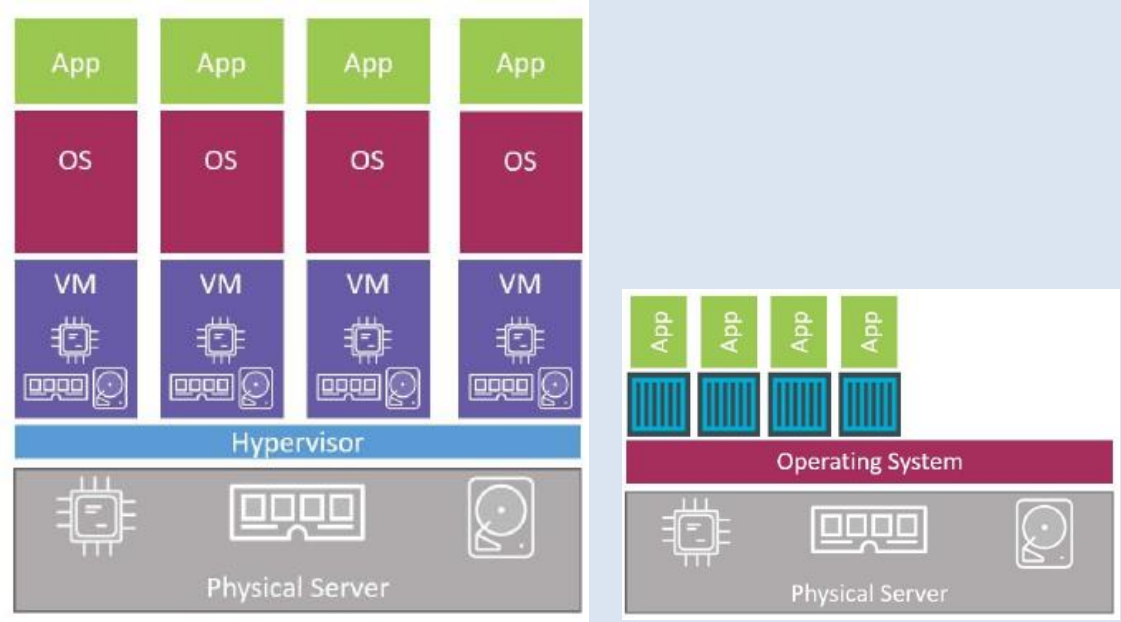
OUTPUT :

```
run:
Enter number of processes:
5
Process  0 Burst Time:
21
Process  1 Burst Time:
45
Process  2 Burst Time:
78
Process  3 Burst Time:
12
Process  4 Burst Time:
32
[0      12      0, 0    21      12, 0   32      33, 0   45      65, 0   78      110]
BUILD SUCCESSFUL (total time: 13 seconds)
```

To study and implement containers and dockers

Instructions:

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.



The VM Approach The container approach

Figure 10.1: The difference between container and VM approach

Lab Tasks

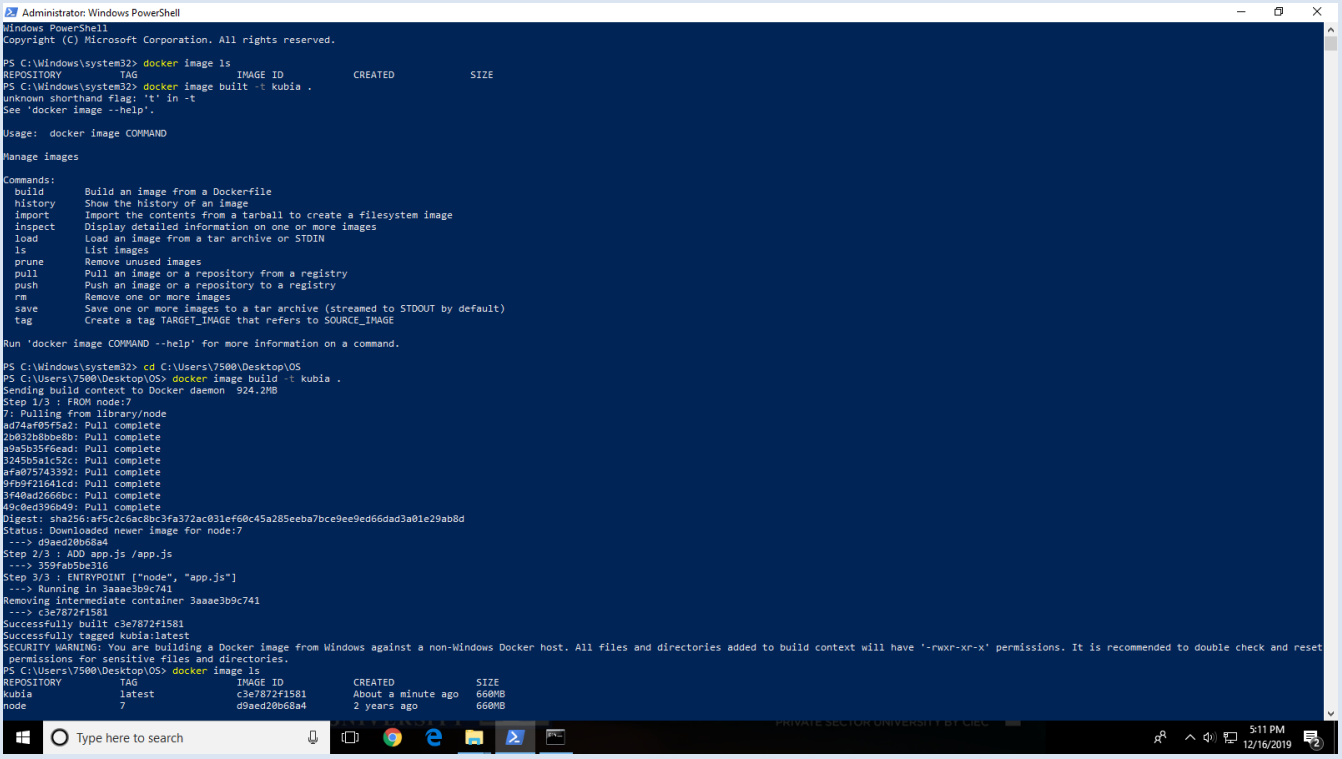
Docker commands

1. Download and install the VMware toolbox for windows
2. List the images available on your system with command “docker image ls”.
3. Goto dockerhub.com and browse for repositories of alpine, python, tensorflow
4. Pull the repository of alpine using docker pull alpine:latet
5. Now list down the images again
6. Start a container using command docker container run -it alpine:latest /bin/bash
7. Run a ps command from inside of the container to list all running processes

8. Press Ctrl-PQ to exit the container without terminating it
9. You can see all running containers on your system using the docker container ls
10. Attach to the running container again with command “docker container exec -it vigilant_borg bash” where vigilant_borg is the name of yhour cotinaer
11. Press Ctl-PQ again to exit the container
12. Stop the running container with command “docker container stop vigilant_borg”.
13. Remove the container using command docker container rm vigilant_borg

Building docker images

14. Clone a repository using command “git clone https://github.com/nigelpoulton/psweb.git”
15. Change your directory using “cd psweb”
16. List the contents of the Dockerfile using “cat Dockerfile”
17. Build the docker image using “docker image build –t os:latest .”
18. Check to make sure that the new os:latest image exists on your host
19. Now run a container with the newly create image “docker container run -d --name web1 -p 8080:8080 os:latest”
20. Open a web browser and navigate to the DNS name or IP address of the host that you are running the container from and point it to port 8080Well done. You’ve taken an application and containerized it (built a Docker image from it).



Lab 11

To study and setup a Kubernetes cluster

Kubernetes abstracts away the hardware infrastructure and exposes your whole datacenter as a single enormous computational resource. It allows you to deploy and run your software components without having to know about the actual servers underneath. When deploying a multi-component application through Kubernetes, it selects a server for each component, deploys it, and enables it to easily find and communicate with all the other components of your application.

Lab Tasks:

Setup

1. Install docker desktop from http://www.dockerhub.com
2. Next, download the kubectl from https://kubernetes.io/

Build an image:

3. Create a sample nodes.js server file (app.js)

```
const http = require('http');
const os = require('os');
console.log("Kubia server starting...");
var handler = function(request, response) { console.log("Received request from " + request.connection.remoteAddress);
response.writeHead(200);
response.end("You've hit " + os.hostname() + "\n"); };
var www = http.createServer(handler);
www.listen(8080);
```

4. Now crate a docker file:

```
FROM node:7
ADD app.js /app.js
ENTRYPOINT ["node", "app.js"]
```

5. Now run the following command to build an image:
   docker build -t kubia .

6. Enable kubernetes on your system (it is available in System tray)
7. Switch the context of kubectl by running:
   kubectl config use-context docker-for-desktop

Getting details about nodes and cluster

8. Run the following command to check if your cluster is working: kubectl cluster-info
9. List all the nodes in your cluster: kubectl get nodes
10. To see more detailed information about an object, you can use the kubectl describe command

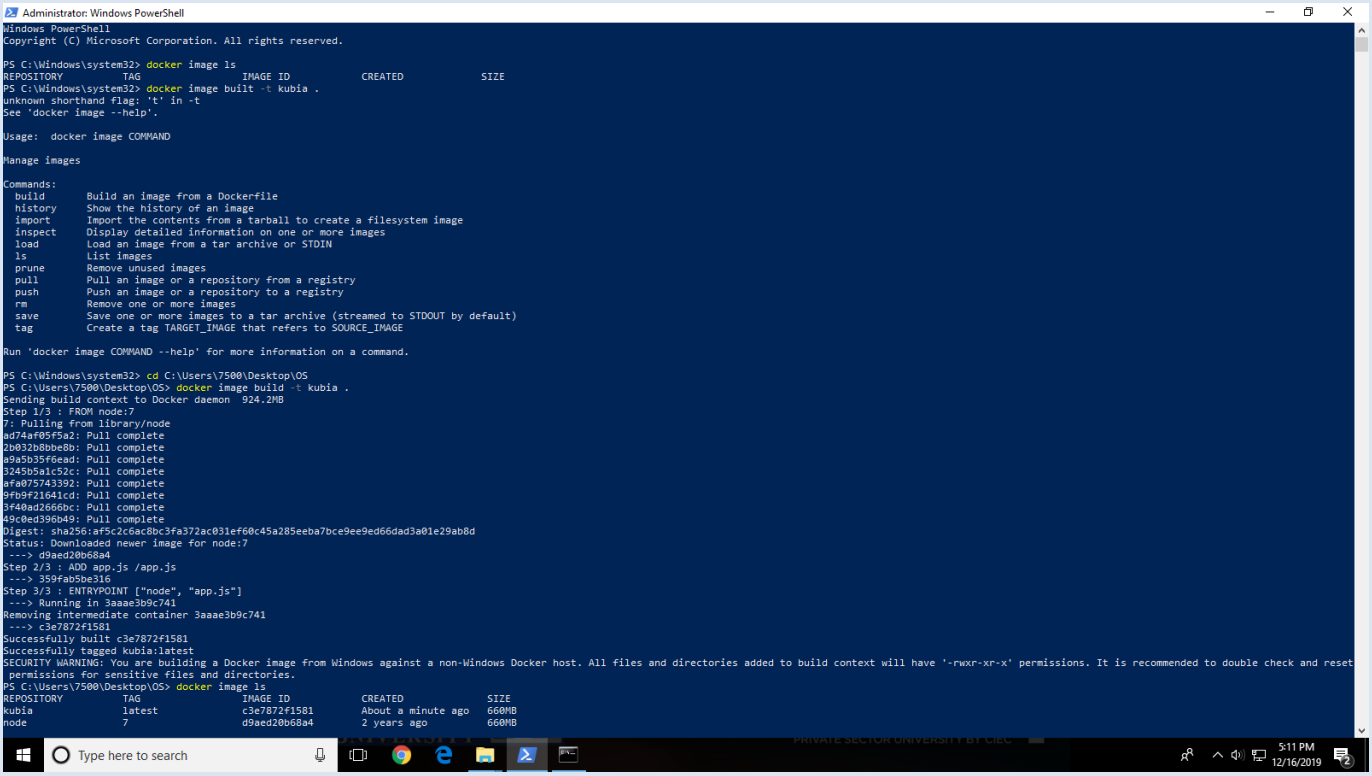
Deploying an application

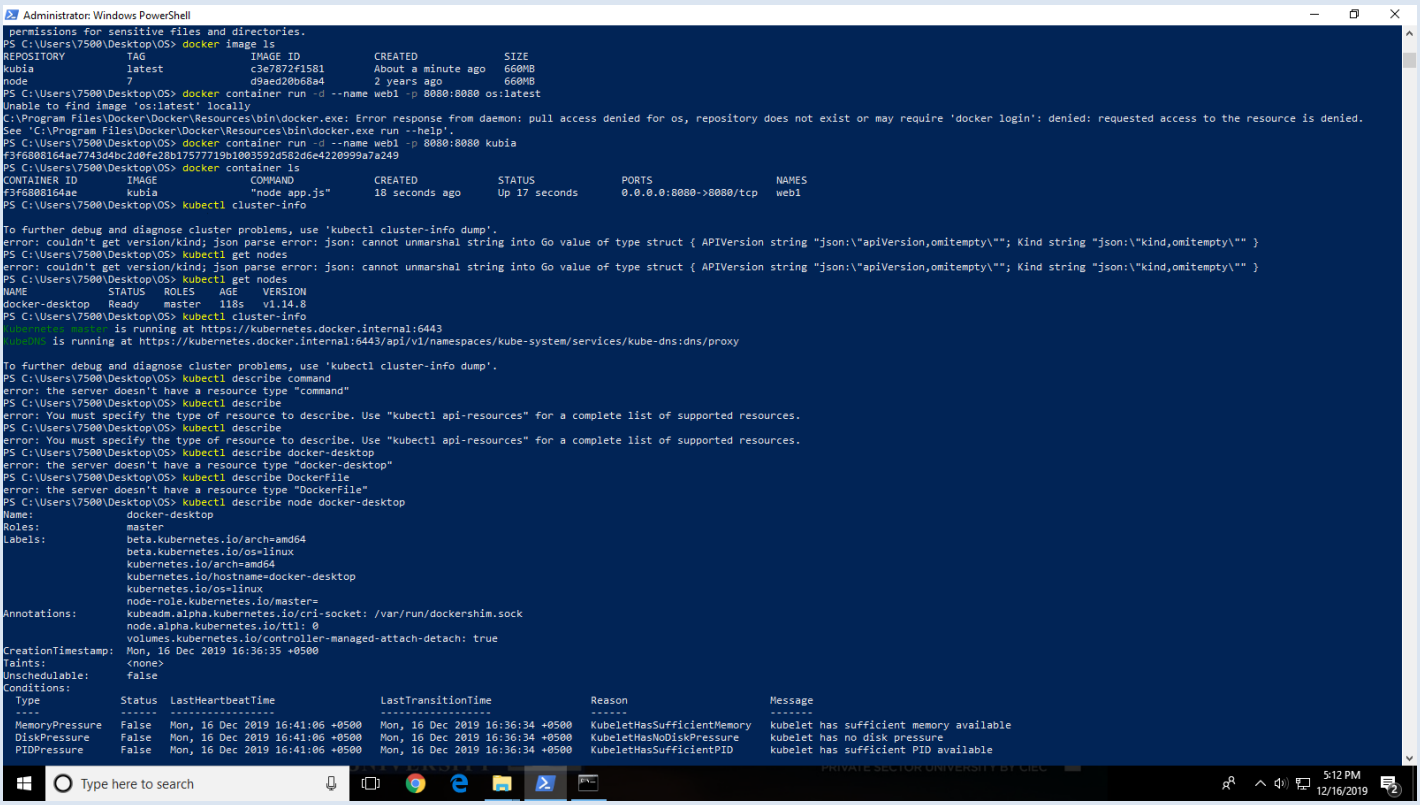
11. Deploy your application using the following command: kubectl run kubia --image=luksa/kubia --port=8080 --generator=run/v1
12. List all the pods using following command: kubectl get pods
13. Expose your replication controller using following command: kubectl expose rc kubia --type=LoadBalancer --name kubia-http
14. Now, list the services using: kubectl get services
15. Access your application from the listed URL.

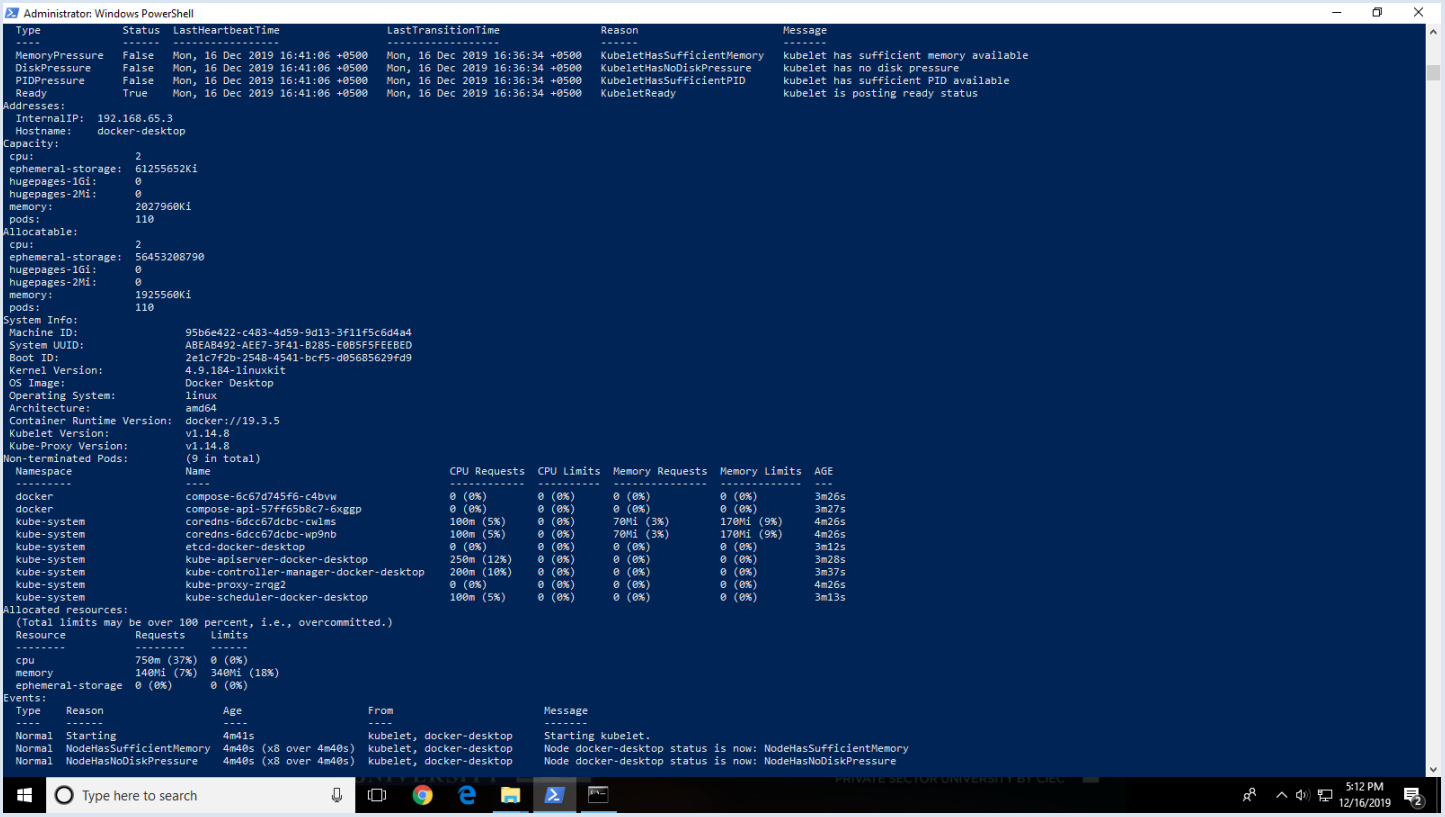
Replication controllers

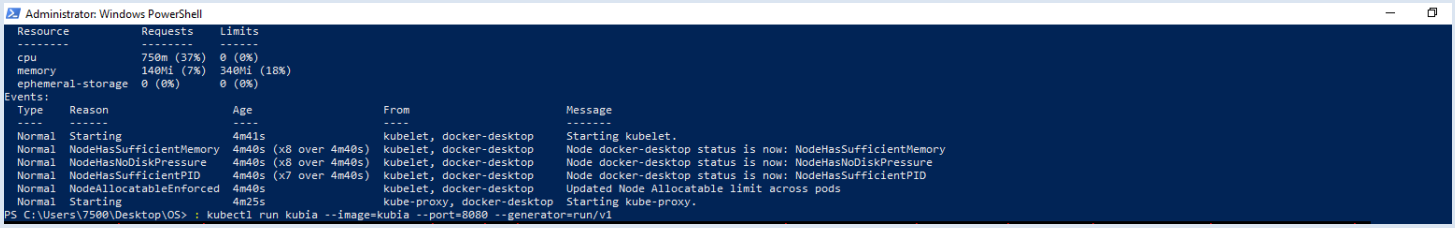
16. Your pod is managed by a ReplicationController. See the replication controller with the following command: kubectl get replicationcontrollers
17. Scale up the number of replicas of your pod with following command: kubectl scale rc kubia --replicas=3
18. Now run the following command: kubectl get rc
19. Run the following command to see number of pods: kubectl get pods
    For more details, run: kubectl get pods -o wide

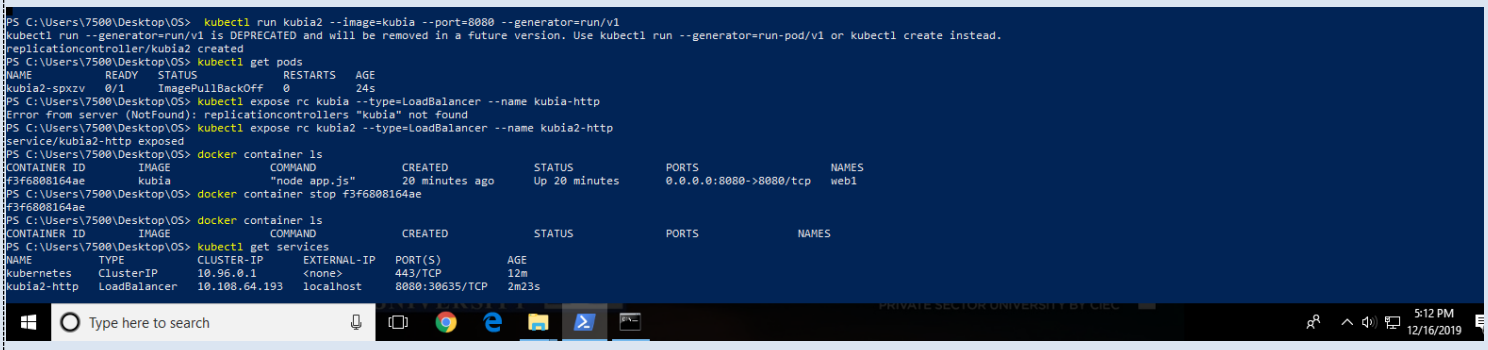
DISPLAY OUTPUT




Administrator: Windows PowerShell
permissions for sensitive files and directories.
PS C:\Users\7500\Desktop\OS> docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
kubia latest c3e7872f1581 About a minute ago 660MB
node 7 d9aed20b68a4 2 years ago 660MB
PS C:\Users\7500\Desktop\OS> docker container run -d --name web1 -p 8080:8080 os:latest
Unable to find image 'os:latest' locally
C:\Program Files\Docker\Docker\Resources\bin\docker.exe: Error response from daemon: pull access denied for os, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'C:\Program Files\Docker\Docker\Resources\bin\docker.exe run --help'.
PS C:\Users\7500\Desktop\OS> docker container run -d --name web1 -p 8080:8080 kubia
f3f6808164ae7743d4bc2d0fe28b17577719b1003592d582d6e4220999a7a249
PS C:\Users\7500\Desktop\OS> docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f3f6808164ae kubia "node app.js" 18 seconds ago Up 17 seconds 0.0.0.0:8080->8080/tcp web1
PS C:\Users\7500\Desktop\OS> kubectl cluster-info
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
error: couldn't get version/kind; json parse error: json: cannot unmarshal string into Go value of type struct { APIVersion string "json:\"apiVersion,omitempty\""; Kind string "json:\"kind,omitempty\"" }
PS C:\Users\7500\Desktop\OS> kubectl get nodes
error: couldn't get version/kind; json parse error: json: cannot unmarshal string into Go value of type struct { APIVersion string "json:\"apiVersion,omitempty\""; Kind string "json:\"kind,omitempty\"" }
PS C:\Users\7500\Desktop\OS> kubectl get nodes
NAME STATUS ROLES AGE VERSION
docker-desktop Ready master 118s v1.14.8
PS C:\Users\7500\Desktop\OS> kubectl cluster-info
Kubernetes master is running at https://kubernetes.docker.internal:6443
KubeDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
PS C:\Users\7500\Desktop\OS> kubectl describe command
error: the server doesn't have a resource type "command"
PS C:\Users\7500\Desktop\OS> kubectl describe
error: You must specify the type of resource to describe. Use "kubectl api-resources" for a complete list of supported resources.
PS C:\Users\7500\Desktop\OS> kubectl describe
error: You must specify the type of resource to describe. Use "kubectl api-resources" for a complete list of supported resources.
PS C:\Users\7500\Desktop\OS> kubectl describe docker-desktop
error: the server doesn't have a resource type "docker-desktop"
PS C:\Users\7500\Desktop\OS> kubectl describe DockerFile
error: the server doesn't have a resource type "DockerFile"
PS C:\Users\7500\Desktop\OS> kubectl describe node docker-desktop
Name: docker-desktop
Roles: master
Labels: beta.kubernetes.io/arch=amd64
beta.kubernetes.io/os=linux
kubernetes.io/arch=amd64
kubernetes.io/hostname=docker-desktop
kubernetes.io/os=linux
node-role.kubernetes.io/master=
Annotations: kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
node.alpha.kubernetes.io/ttl: 0
volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp: Mon, 16 Dec 2019 16:36:35 +0500
Taints: <none>
Unschedulable: false
Conditions:
Type Status LastHeartbeatTime LastTransitionTime Reason Message
MemoryPressure False Mon, 16 Dec 2019 16:41:06 +0500 Mon, 16 Dec 2019 16:36:34 +0500 KubeletHasSufficientMemory kubelet has sufficient memory available
DiskPressure False Mon, 16 Dec 2019 16:41:06 +0500 Mon, 16 Dec 2019 16:36:34 +0500 KubeletHasNoDiskPressure kubelet has no disk pressure
PIDPressure False Mon, 16 Dec 2019 16:41:06 +0500 Mon, 16 Dec 2019 16:36:34 +0500 KubeletHasSufficientPID kubelet has sufficient PID available
Type here to search
5:12 PM
12/16/2019


Administrator: Windows PowerShell
Type Status LastHeartbeatTime LastTransitionTime Reason Message
MemoryPressure False Mon, 16 Dec 2019 16:41:06 +0500 Mon, 16 Dec 2019 16:36:34 +0500 KubeletHasSufficientMemory kubelet has sufficient memory available
DiskPressure False Mon, 16 Dec 2019 16:41:06 +0500 Mon, 16 Dec 2019 16:36:34 +0500 KubeletHasNoDiskPressure kubelet has no disk pressure
PIDPressure False Mon, 16 Dec 2019 16:41:06 +0500 Mon, 16 Dec 2019 16:36:34 +0500 KubeletHasSufficientPID kubelet has sufficient PID available
Ready True Mon, 16 Dec 2019 16:41:06 +0500 Mon, 16 Dec 2019 16:36:34 +0500 KubeletReady kubelet is posting ready status
Addresses:
InternalIP: 192.168.65.3
Hostname: docker-desktop
Capacity:
cpu: 2
ephemeral-storage: 61255652Ki
hugepages-1Gi: 0
hugepages-2Mi: 0
memory: 2027960Ki
pods: 110
Allocatable:
cpu: 2
ephemeral-storage: 56453208790
hugepages-1Gi: 0
hugepages-2Mi: 0
memory: 1925560Ki
pods: 110
System Info:
Machine ID: 95b6e422-c483-4d59-9d13-3f11f5c6d4a4
System UUID: ABEAB492-AEE7-3F41-B285-E0B5F5FEEBED
Boot ID: 2e1c7f2b-2548-4541-bcf5-d05685629fd9
Kernel Version: 4.9.184-linuxkit
OS Image: Docker Desktop
Operating System: linux
Architecture: amd64
Container Runtime Version: docker://19.3.5
Kubelet Version: v1.14.8
Kube-Proxy Version: v1.14.8
Non-terminated Pods: (9 in total)
Namespace Name CPU Requests CPU Limits Memory Requests Memory Limits AGE
docker compose-6c67d745f6-c4bvw 0 (0%) 0 (0%) 0 (0%) 0 (0%) 3m26s
docker compose-api-57ff65b8c7-6xggp 0 (0%) 0 (0%) 0 (0%) 0 (0%) 3m27s
kube-system coredns-6dcc67dcbc-cwlms 100m (5%) 0 (0%) 70Mi (3%) 170Mi (9%) 4m26s
kube-system coredns-6dcc67dcbc-wp9nb 100m (5%) 0 (0%) 70Mi (3%) 170Mi (9%) 4m26s
kube-system etcd-docker-desktop 0 (0%) 0 (0%) 0 (0%) 0 (0%) 3m12s
kube-system kube-apiserver-docker-desktop 250m (12%) 0 (0%) 0 (0%) 0 (0%) 3m28s
kube-system kube-controller-manager-docker-desktop 200m (10%) 0 (0%) 0 (0%) 0 (0%) 3m37s
kube-system kube-proxy-zrqg2 0 (0%) 0 (0%) 0 (0%) 0 (0%) 4m26s
kube-system kube-scheduler-docker-desktop 100m (5%) 0 (0%) 0 (0%) 0 (0%) 3m13s
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
Resource Requests Limits
cpu 750m (37%) 0 (0%)
memory 140Mi (7%) 340Mi (18%)
ephemeral-storage 0 (0%) 0 (0%)
Events:
Type Reason Age From Message
Normal Starting 4m41s kubelet, docker-desktop Starting kubelet.
Normal NodeHasSufficientMemory 4m40s (x8 over 4m40s) kubelet, docker-desktop Node docker-desktop status is now: NodeHasSufficientMemory
Normal NodeHasNoDiskPressure 4m40s (x8 over 4m40s) kubelet, docker-desktop Node docker-desktop status is now: NodeHasNoDiskPressure
Type here to search
5:12 PM
12/16/2019


Administrator: Windows PowerShell
Resource Requests Limits
-------- -------- ------
cpu 750m (37%) 0 (0%)
memory 140Mi (7%) 340Mi (18%)
ephemeral-storage 0 (0%) 0 (0%)
Events:
Type Reason Age From Message
---- ------ ---- ---- -------
Normal Starting 4m41s kubelet, docker-desktop Starting kubelet.
Normal NodeHasSufficientMemory 4m40s (x8 over 4m40s) kubelet, docker-desktop Node docker-desktop status is now: NodeHasSufficientMemory
Normal NodeHasNoDiskPressure 4m40s (x8 over 4m40s) kubelet, docker-desktop Node docker-desktop status is now: NodeHasNoDiskPressure
Normal NodeHasSufficientPID 4m40s (x7 over 4m40s) kubelet, docker-desktop Node docker-desktop status is now: NodeHasSufficientPID
Normal NodeAllocatableEnforced 4m40s kubelet, docker-desktop Updated Node Allocatable limit across pods
Normal Starting 4m25s kube-proxy, docker-desktop Starting kube-proxy.
PS C:\Users\7500\Desktop\OS> : kubectl run kubia --image=kubia --port=8080 --generator=run/v1


PS C:\Users\7500\Desktop\OS> kubectl run kubia2 --image=kubia --port=8080 --generator=run/v1
kubectl run --generator=run/v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
replicationcontroller/kubia2 created
PS C:\Users\7500\Desktop\OS> kubectl get pods
NAME READY STATUS RESTARTS AGE
kubia2-spxzv 0/1 ImagePullBackOff 0 24s
PS C:\Users\7500\Desktop\OS> kubectl expose rc kubia --type=LoadBalancer --name kubia-http
Error from server (NotFound): replicationcontrollers "kubia" not found
PS C:\Users\7500\Desktop\OS> kubectl expose rc kubia2 --type=LoadBalancer --name kubia2-http
service/kubia2-http exposed
PS C:\Users\7500\Desktop\OS> docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f3f6808164ae kubia "node app.js" 20 minutes ago Up 20 minutes 0.0.0.0:8080->8080/tcp web1
PS C:\Users\7500\Desktop\OS> docker container stop f3f6808164ae
f3f6808164ae
PS C:\Users\7500\Desktop\OS> docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
PS C:\Users\7500\Desktop\OS> kubectl get services
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 12m
kubia2-http LoadBalancer 10.108.64.193 localhost 8080:30635/TCP 2m23s
Type here to search
5:12 PM
12/16/2019