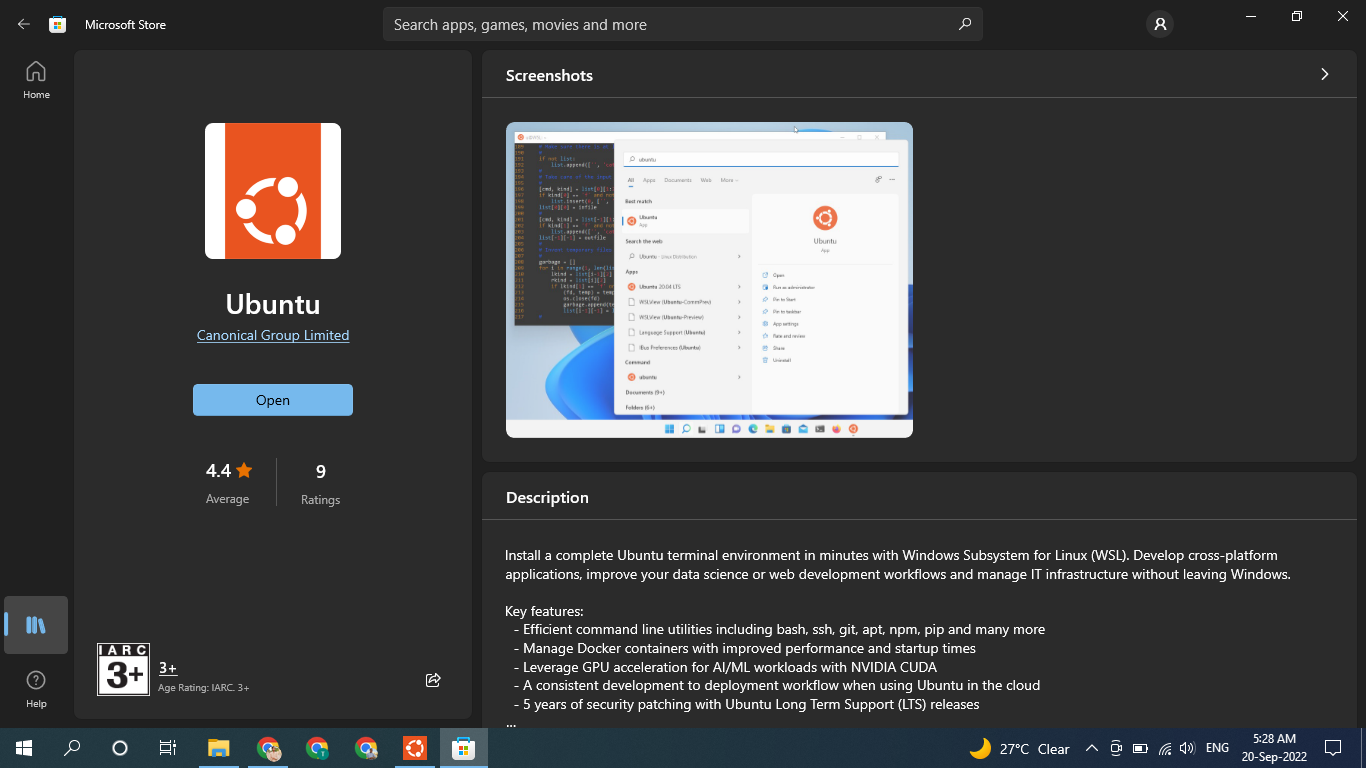
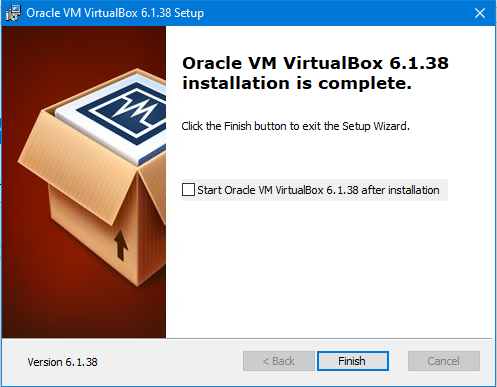
**Lab Session # 01**

**Setup Environment:**

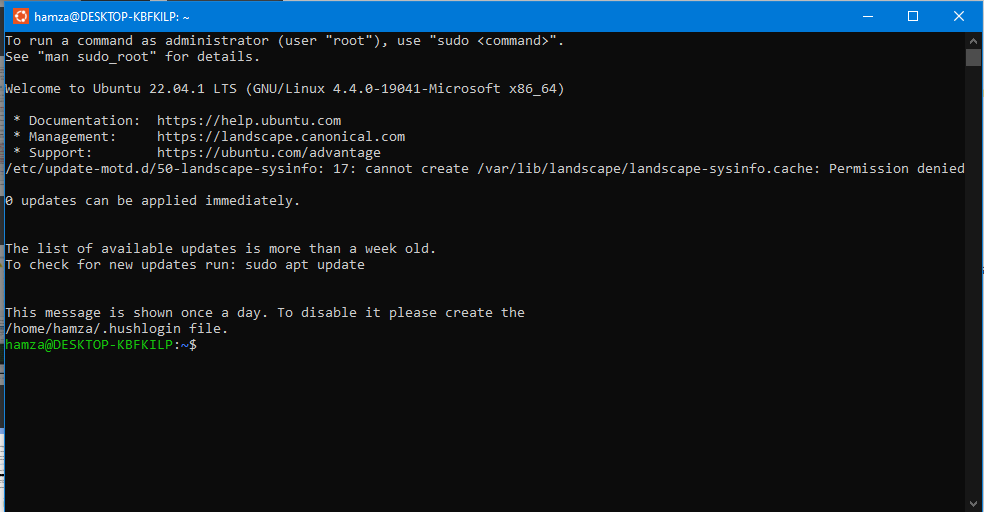
Using WSL sub-system;



Using Virtual Box;



**Launching Terminal (CLI):**

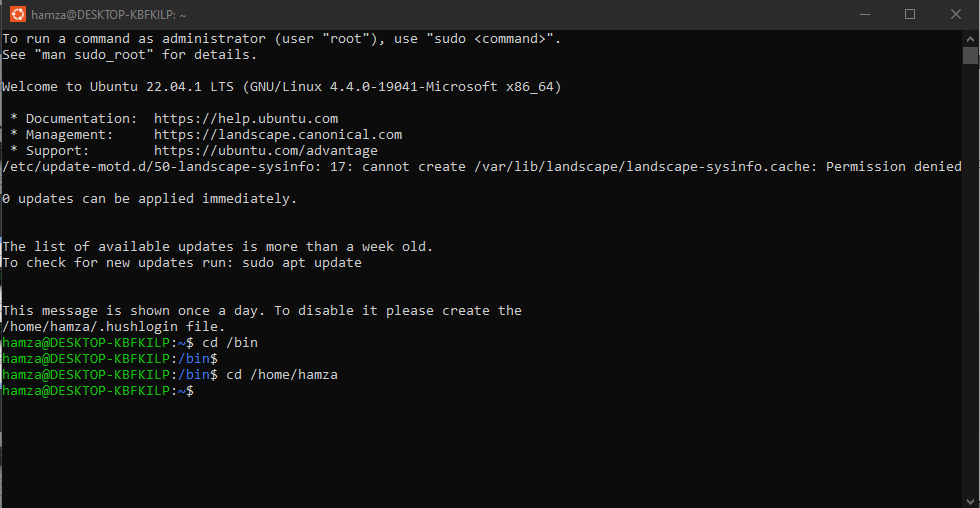


1) The first part of this line is the name of the user (**hamza**).

2) The second part is the computer name (**DESKTOP-KBFKILP**) or the host name. 3) The '**:**' is a simple separator.

4) The tilde '**~**' sign shows that the user in working in the home directory. If you change the directory, this sign will vanish.

**Home directory:**



In the above illustration, we have moved from the **/home** directory to **/bin** using the '**cd**' command. The **~** sign does not display while working in /bin directory. It appears while moving back to the home directory.

5) The '**$**' sign suggests that you are working as a regular user in Linux.

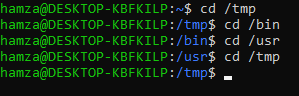
**Present Working Directory:**

The directory that you are currently browsing is called the Present working directory.



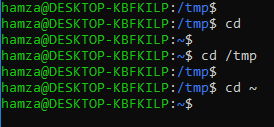
**Changing Directories:**

If you want to change your current directory use the '**cd**' command. **cd /tmp**



**Navigating to home directory:**

If you want to navigate to the home directory, type **cd**. You can also use the **cd ~**



**Moving to root directory:**

The root of the file system in Linux is denoted by '**/**'. Similar to '**c:\**' in Windows.

Type '**cd /**' to move to the root directory.



**Relative and Absolute Paths:**

A path in computing is the address of a file or folder.

* In Windows; **C:\documentsandsettings\user\downloads**
* In Linux; **/home/user/downloads**

There are two kinds of paths:

**1. Absolute Path:**

Let's say you have to browse the images stored in the Pictures directory of the home folder. The absolute file path of Pictures directory **/home/Pictures**, to navigate to this directory, you can use command **cd /home/hamza/Pictures**.



**2. Relative Path:**

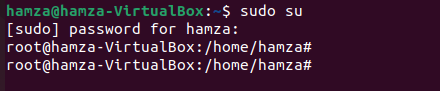
You can directly come into download by using ‘**cd Downloads**’ command if your pwd is home.



**Lab Session # 02**

**Executing a Linux Command:**

By using ‘**sudo su**’ you can access to the root directory of the Linux.



‘**pwd**’ displays the name of the current directory. **pwd** stands for present working directory. By typing this command, you are informed of which directory you are currently in.

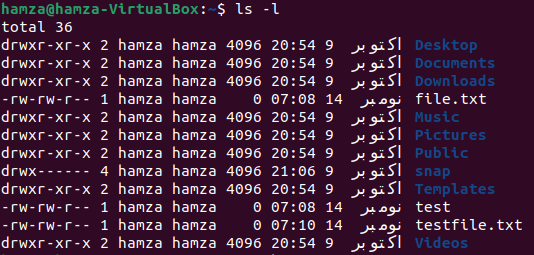


‘**ls**’ Displays the listing of files and directories. If no file or directory is specified, then the current directory’s contents are displayed. By default, the contents are sorted alphabetically.



**Lab Session # 03**

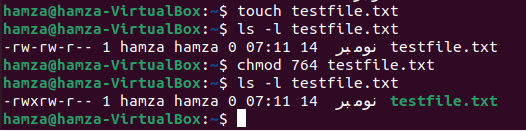
‘**ls -l**’ is used to list the content of your current working directory.



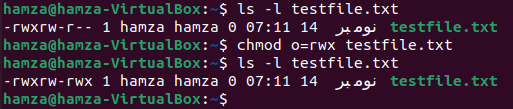
In the case of no file in home directory Linux prompt got returned but.

* We can change file/directory permission through ‘**chmod**’. There are two methods as following:

**Absolute (numeric) Mode**; in this we use assigned number for permission as for here we have used 7(Read, Write, Execute) 6 (Read, Write) 4 (Read) which results in giving new permission to testfile.txt.

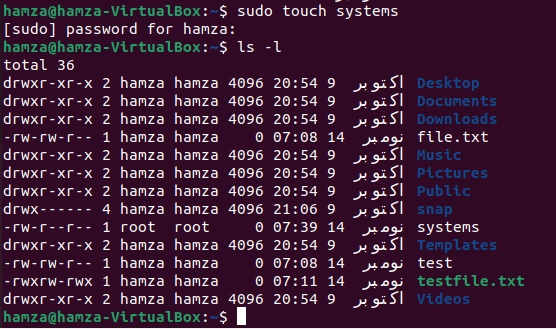


**Symbolic Mode**; we can use +, -, = for changing mode of permission. Here firstly ls -l memo1 is used to get current permission and then **chmod o=rwx** is used to change permission ‘o’ shows that it is implies to other.

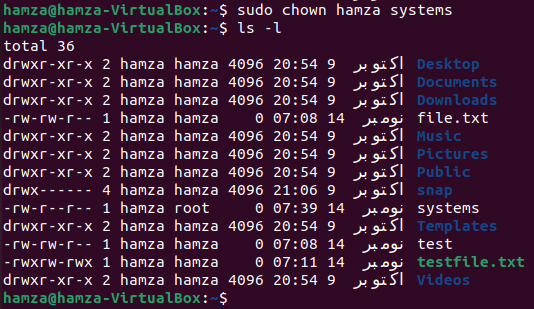


**Changing ownership through command.**

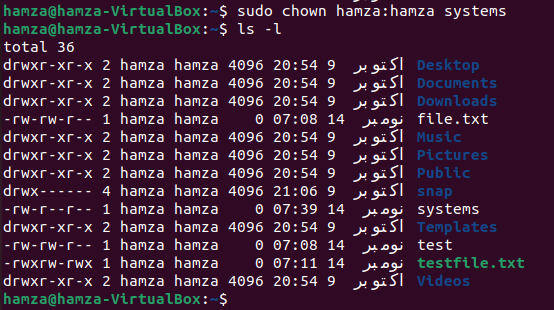
first, we have reached/touch the ‘**systems file**’ using ‘**sudo touch systems**’ and by using ‘**ls -l**’ we have listed the files. We can clearly see the “**root root**” ownership.



After using ‘**chown**’ we have assigned it to ‘**hamza**’ which is clearly seen afterwards by listing command.



Now we have changed both “**user**” and “**group**” from ‘**root**’ to “**hamza**” by using command “**chown usr:grp file name**”.



**Lab Session # 04**

**Input/Output Redirection in Linux:**

**Output Redirection:**

The '**>**' symbol is used for output (STDOUT) redirection.

Example: **ls -al > listings**

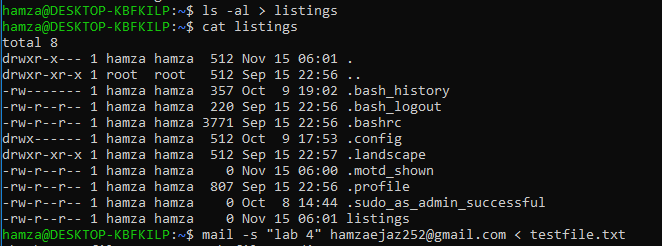
Here the output of command **ls -al** is re-directed to file "**listings**" instead of your screen.

**Input redirection:**

The '**<**' symbol is used for input (STDIN) redirection.

Example: The mail program in Linux can help you send emails from the Terminal. You can type the contents of the email using the standard device keyboard. But if you want to attach a File to email you can use the input re-direction operator in the following format.

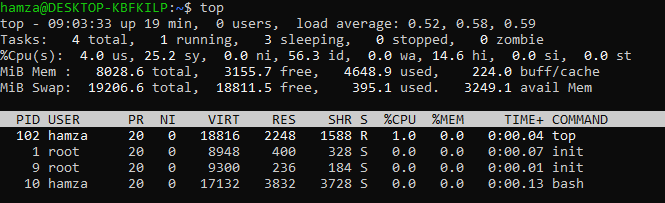
**mail -s "Subject" to-address < Filename**



**Lab Session # 05**

**Top**:

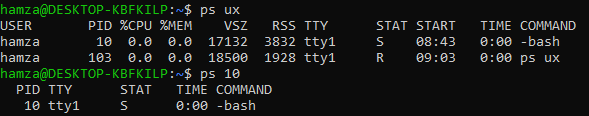
This utility tells the user about all the running processes on the Linux machine.



**PS:**

This command stands for **'Process Status'**. It is similar to the "**Task Manager**" that pop- ups in a Windows Machine when we use Cntrl+Alt+Del.

* To check all the processes running under a user, use the command - **ps ux**
* You can also check the process status of a single process, use the syntax - **ps PID**



**Kill**:

This command terminates running processes on a Linux machine. To use these utilities, you need to know the PID (process id) of the process you want to kill. Syntax - **kill PID**

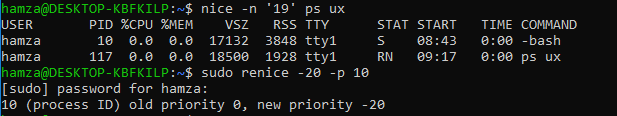
To find the PID of a process simply type **pidof Process name**



**NICE:**

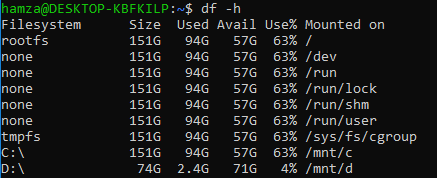
Linux can run a lot of processes at a time, which can slow down the speed of some high priority processes and result in poor performance. This priority is called Niceness in Linux, and it has a value between -20 to 19.

* To start a process with a niceness value other than the default value use the syntax **nice -n 'Nice value' process name**
* If there is some process already running on the system, then you can 'Renice' its value using syntax. **renice 'nice value' -p 'PID'**



**DF:**

This utility reports the free disk space (Hard Disk) on all the file systems. If you want the information in a readable format, then use the command **'df -h'**

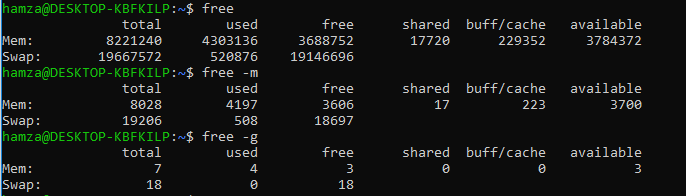


**Free:**

This command shows the free and used memory (RAM) on the Linux system. You can also use the following arguments:

**free -m** to display output in MB

**free -g** to display output in GB



**Lab Session # 06**

**OBJECT: Shell Scripting**

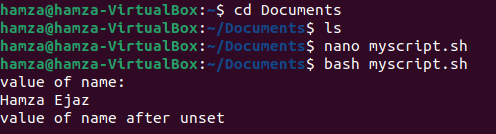
What is Shell Scripting?

Shell scripting is writing a series of command for the shell to execute, can be stored as file and executed anytime. A shell script or shell program has its own syntax like other programming languages. It allows user to define variables, assign various values, and so on.

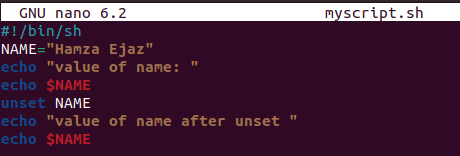
* Check existing folders



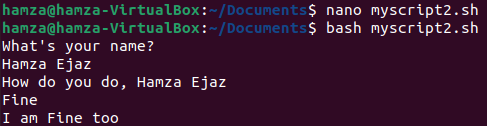
* Go to any folder using **cd**
* Make script using **‘nano filename.sh’**. write in script file as show in 2nd pic then **CTRL + O** saves a Nano file. **CTRL + X** exits Nano.
* Execute nano script file using **‘bash filename.sh’**

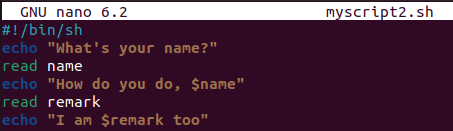


* Nano script file having hard coded variable



* Another example with dynamic variable where assigning value at run time

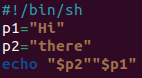




Positional Parameters:

The shell has knowledge of a special kind of variable called a positional parameter.

# Program reverse, prints the command line parameters out in reverse order echo "$p2" "$p1"





**Lab Session # 07**

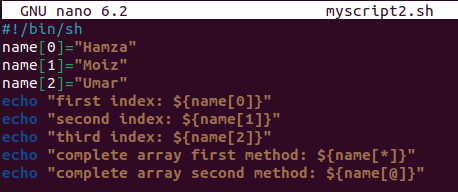
**OBJECT: Arrays for Shell**

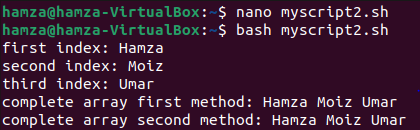
Following is the simplest method of creating an array variable. This helps assign a value to one of its indices.

**array\_name[index]=value**

You can access all the items in an array in one of the following ways : **${array\_name[\*]}**

**${array\_name[@]}**

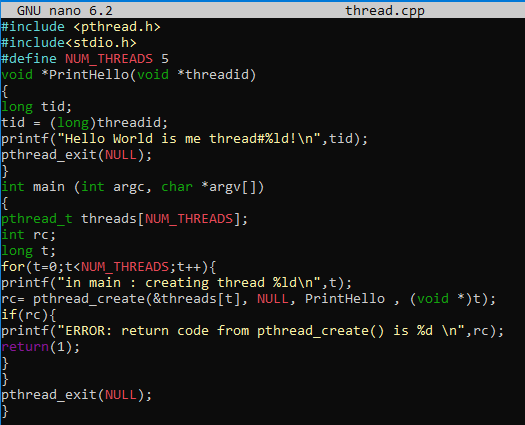




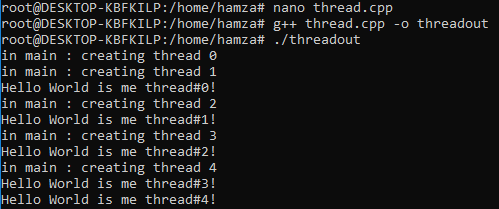
**Lab Session # 11**

**OBJECT: Demonstrate multi-threading using POSIX thread (Pthread) Library**

We create file using ‘**nano filename.ext**’ then write thread program in it.



Now save program and compile using ‘**g++ filename -o threadout**’ then run using ‘**./threadout**’



**Lab Scheduling**

* We have to create file using ‘**touch filename.ext**’
* Goto file explorer, open the file which created by above command
* Write program of scheduling then save it
* Compile that program file using ‘**gcc filename -o test**’ and run using ‘**./test**’

