**LAB REPORT NO 2**



Submitted by:  **Muhammad Ali**

Registration No:- **19PWCSE1801**

Class Section: A

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

**Engr.Mian Ibad Ali shah**

Data:(28,04,2021)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Unix file: -**

Unix file system is a logical method of organizing and storing large amounts of information in a way that makes it easy to manage. A file is a smallest unit in which the information is stored. Unix file system has several important features. All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system.

Files in Unix System are organized into multi-level hierarchy structure known as a directory tree. At the very top of the file system is a directory called “root” which is represented by a “/”. All other files are “descendants” of root.

**Directories: -**

A directory is a group of files. A directory is divided into two types:

* Root directory – Strictly speaking, there is only one root directory in your Linux and Unix-like system, which is denoted by / (forward slash). It is root of your entire file system and can not be renamed or deleted.
* Sub directory – Directory under root (/) directory is subdirectory which can be created, renamed by the user.

Directories are used to organize your data files, programs more efficiently.

**Directories or Files and their description –**

* **/ :** The slash / character alone denotes the root of the filesystem tree.
* **/bin :**Stands for “binaries” and contains certain fundamental utilities, such as ls or cp, which are generally needed by all users.
* **/boot :**Contains all the files that are required for successful booting process.
* **/dev :**Stands for “devices”. Contains file representations of peripheral devices and pseudo-devices.
* **/etc :**Contains system-wide configuration files and system databases. Originally also contained “dangerous maintenance utilities” such as init,but these have typically been moved to /sbin or elsewhere.
* **/home :**Contains the home directories for the users.
* **/lib :**Contains system libraries, and some critical files such as kernel modules or device drivers.
* **/media :**Default mount point for removable devices, such as USB sticks, media players, etc.
* **/mnt :**Stands for “mount”. Contains filesystem mount points. These are used, for example, if the system uses multiple hard disks or hard disk partitions. It is also often used for remote (network) filesystems, CD-ROM/DVD drives, and so on.
* **/proc :**procfs virtual filesystem showing information about processes as files.
* **/root :**The home directory for the superuser “root” – that is, the system administrator. This account’s home directory is usually on the initial filesystem, and
* hence not in /home (which may be a mount point for another filesystem) in case specific maintenance needs to be performed, during which other filesystems are not available. Such a case could occur, for example, if a hard disk drive suffers physical failures and cannot be properly mounted.
* **/tmp :**A place for temporary files. Many systems clear this directory upon startup; it might have tmpfs mounted atop it, in which case its contents do not survive a reboot, or it might be explicitly cleared by a startup script at boot time.
* **/usr :**Originally the directory holding user home directories,its use has changed. It now holds executables, libraries, and shared resources that are not system critical, like the X Window System, KDE, Perl, etc. However, on some Unix systems, some user accounts may still have a home directory that is a direct subdirectory of /usr, such as the default as in Minix. (on modern systems, these user accounts are often related to server or system use, and not directly used by a person).
* **/usr/bin :**This directory stores all binary programs distributed with the operating system not residing in /bin, /sbin or (rarely) /etc.
* **/usr/include :**Stores the development headers used throughout the system. Header files are mostly used by the #include directive in C/C++ programming language.
* **/usr/lib :**Stores the required libraries and data files for programs stored within /usr or elsewhere.
* **/var :**A short for “variable.” A place for files that may change often – especially in size, for example e-mail sent to users on the system, or process-ID lock files.
* **/var/log :**Contains system log files.

**1.Ordinary files :-**

An ordinary file is a file on the system that contains data, text, or program instructions.

* Used to store your information, such as some text you have written or an image you have drawn. This is the type of file that you usually work with.
* Always located within/under a directory file.
* Do not contain other files.
* In long-format output of ls -l, this type of file is specified by the “-” symbol.

**2. Directories : -**

Directories store both special and ordinary files. For users familiar with Windows or Mac OS, UNIX directories are equivalent to folders. A directory file contains an entry for every file and subdirectory that it houses. If you have 10 files in a directory, there will be 10 entries in the directory. Each entry has two components.  
(1) The Filename  
(2) A unique identification number for the file or directory (called the inode number)

* Branching points in the hierarchical tree.
* Used to organize groups of files.
* May contain ordinary files, special files or other directories.
* Never contain “real” information which you would work with (such as text). Basically, just used for organizing files.
* All files are descendants of the root directory, ( named / ) located at the top of the tree.

In long-format output of ls –l , this type of file is specified by the “d” symbol.

**3. Special Files : -**

 Used to represent a real physical device such as a printer, tape drive or terminal, used for Input/Output (I/O) operations. Device or special files are used for device Input/Output(I/O) on UNIX and Linux systems. They appear in a file system just like an ordinary file or a directory.  
On UNIX systems there are two flavors of special files for each device, character special files and block special files :

* When a character special file is used for device Input/Output(I/O), data is transferred one character at a time. This type of access is called raw device access.
* When a block special file is used for device Input/Output(I/O), data is transferred in large fixed-size blocks. This type of access is called block device access.

For terminal devices, it’s one character at a time. For disk devices though, raw access means reading or writing in whole chunks of data – blocks, which are native to your disk.

* In long-format output of ls -l, character special files are marked by the “c” symbol.
* In long-format output of ls -l, block special files are marked by the “b” symbol.

**4. Pipes : -**

UNIX allows you to link commands together using a pipe. The pipe acts a temporary file which only exists to hold data from one command until it is read by another.A Unix pipe provides a one-way flow of data.The output or result of the first command sequence is used as the input to the second command sequence. To make a pipe, put a vertical bar (|) on the command line between two commands.

For example: who | wc -l

In long-format output of ls –l , named pipes are marked by the “p” symbol.

**Classes of Accounts: -**

The files and directories in the home directory of your CLAS Linux account can be accessed on computers running the Linux operating system. Linux is a type of [UNIX](http://www.unix.org/) and uses UNIX file and directory permissions. For purposes of permissions,

**UNIX divides accounts into three classes:**

**1.user:**  
Your account.

**2.group:**  
Any permissions group that your account belongs to.

**3.other:**  
Any account that is not yours and that does not belong to a permissions group that your account belongs to.

**Types of Permissions: -**

There are three basic types of permissions which can be assigned to each of these three classes of accounts:

1.read  
2.write  
3.execute

**File Permissions: -**

These three types of permissions mean slightly different things for files than for directories. For files, these permissions grant these rights:

**1.read:**  
Allowed to read the contents of the file

**2.write:**  
Allowed to modify or delete the file

**3.execute**:  
Allowed to run the file as a process, if possible

**Directory Permissions: -**

For directories, the permissions grant these rights:

**1.read:**Allowed to list the contents of the directory

**2.write**:  
Allowed to create, modify or delete files in the directory

**3.execute:**  
Allowed to acess a file in the directory if you know the name of the file.

**Viewing File Permissions: -**

The ls command is used to list files and the contents of directories. The -l parameter displays permissions. For example, to see the permissions of a file named foo in the directory /usr/bin/bar, you would execute:

ls -l /usr/bin/bar/foo

And the command would return something like this:

-rwxr-xr-- 1 muhammad guest 3072 Feb 11 09:25 /usr/bin/foo

In the example, muhammad is the account that owns foo, and guest is the name of the group that owns /usr/bin/foo. The -rwxr-xr-- at the left indicates the permissions. The first character, the -, indicates that /usr/bin/foo is a file, not a directory. The rwx shows the permissions for the user class of accounts - in this case, muhammad. The r indicates read permission; the w, write permission; and the x, execute permission. The next three characters, r-x, show permissions for the group class of accounts, which is guest in this example. Finally, the last three characters, r--, display permissions for the other class - any account that is not muhammad and is not in the guest group.

**Viewing Directory Contents: -**

If you want to see the contents of a directory, you also use ls. Suppose that /usr/bin/bar is a directory. Then the command:

ls -l /usr/bin/bar

may return something like this:

drwxr-x--x 5 muhammad guest 4096 Jan 23 2008 foodir  
-rw-r----- 1 muhammad guest 48128 Sep 14 2004 WhatToDo.doc  
-rw-rw-r-- 1 muhammad guest 464 Jul 6 2005 WinCA.txt

This shows us the contents of the directory bar. The d at the left of the entry for foodir indicates that foodir is a directory.

**Viewing Directory Permissions: -**

If you want to see the permissions of the /usr/bin/bar directory itself, not its contents, then you need to use the -d command-line argument for ls. So, you'd execute this command:

ls -ld /usr/bin/bar

and you'd see something like this:

dr-xrwxr-x 3 muhammad guest 4096 Jan 23 2008 /usr/bin/bar

## Default Access Permissions : -

When you create a file or directory its access permissions are set to a default value. These are usually:  rw-------

gives you read and write permission for your files; no access permissions for the group or others. rwx------

gives you read, write and execute permission for your directories; no access permissions for the group or others.

**The Current Working Directory: -**

The current working directory is the directory that, by default, a UNIX command will use when it is executed. For example, if you do not specify a file or directory when you run the ls command, then ls will assume that you want to see the contents of your current working directory. So,

ls

will return a list of the files and directories in your current working directory. To see the absolute path of your current working directory, use the pwd command.

UNIX provides a shorthand for your current working directory. A single period (.) indicates the current working directory. Two periods (..) indicate the directory immediately above your current working directory.

**Hidden Files and Directories:-**

In UNIX, if a file or directory name begins with a period (.) then by default, ls will not display the file or directory in a directory listing. To see all the files in a directory, including hidden files, use the -a command-line argument. The command:

**ls -a:**

will show all files and directories in a directory, including hidden files. The command:

**ls -al:**

will display all files and directories, and also show their permissions.

**Home Directories: -**

Each Linux account is associated with a home directory. When you login to your Linux account, by default, your current working directory will be your home directory. UNIX provides a short-hand symbol for your home directory, the tilde character, ~. So, to see a list of files in your home directory, you can execute:

ls ~

## Creating a File:-

Many people create files using a text editor, but you can use the command **cat** to create files without using/learning to use a text editor. To create a practice file (called **firstfile**) and enter one line of text in it, type the following at the **%** prompt:

**cat > firstfile**: -

Terminate file entry by typing **Control-d** on a line by itself. (Hold down the Control key and type d.) On your screen, you will see:

**% cat > firstfile**

To examine the contents of a file you have just created, enter this at the **%** prompt:

**cat firstfile**

## Copying a File:-

To make a duplicate copy of a file, use the command **cp**. For example, to create an exact copy of the file called **firstfile,** you would type:

**cp firstfile secondfile**

This results in two files with different names, each containing the same information. The **cp** command works by overwriting information. If you create a different file called **thirdfile** and then type the following command:

**cp thirdfile firstfile**

you will find that the original contents of **firstfile** are gone, replaced by the contents of **thirdfile**.

## 

## Renaming a File:-

Unix does not have a command specifically for renaming files. Instead, the **mv** command is used both to change the name of a file and to move a file into a different directory.

To change the name of a file, use the following command format (where **thirdfile** and **file3** are sample file names):

**mv thirdfile file3**

This command results in the complete removal of **thirdfile**, but a new file called **file3** contains the previous contents of **thirdfile.**

Like **cp,** the **mv** command also overwrites existing files. For example, if you have two files, **fourthfile** and **secondfile,** and you type the command

**mv fourthfile secondfile**

**mv** will remove the original contents of **secondfile** and replace them with the contents of **fourthfile**. As a result, **fourthfile** is renamed **secondfile**, but in the process **secondfile** is deleted.

## Removing a File:-

Use the **rm** command to remove a file. For example,

**rm file3**

deletes **file3** and its contents. You may remove more than one file at a time by specifying a list of files to be deleted. For example,

**rm firstfile secondfile**

You will be prompted to confirm whether you really want to remove the files:

**rm: remove firstfile (y/n)? y**  
**rm: remove secondfile (y/n)? N**

Type **y** or **yes** to remove a file; type **n** or **no** to leave it intact.

#### **Creating a Directory: -**

Creating directories permits you to organize your files. The command

**mkdir project1**

creates a directory called **project1,** where you can store files related to a particular project. The directory that you create will be a subdirectory within your current directory. For details on how to navigate directories and display the files and directories they contain, see [List Contents and Navigate Unix Directories.](https://documentation.its.umich.edu/node/297)

## Moving and Copying Files Into a Directory: -

The **mv** and **cp** commands can be used to put files into a directory. Assume that you want to place some files from your current directory into a newly created directory called **project1.** The command

**mv bibliography project1**

will move the file **bibliography** into the directory **project1**. The command

**cp chapter1 project1**

will place a copy of the file **chapter1** in the directory **project1**, but leave **chapter1** intact in the current directory. There will now be two copies of **chapter1**, one in the current directory and one in **project1**.

## Renaming a Directory: -

You can also use the **mv** command to rename and move directories. When you type the command

**mv project1 project2**

the directory called **project1** will be given the new name **project2** as long as a directory called **project2** did not previously exist. If directory **project2** already existed before the mv command was issued,

**mv project1 project2**

would move the directory **project1** and its files into the directory **project2.**

## Copying a Directory:-

You can use the **cp** command to create a duplicate copy of a directory and its contents. To copy directory **project1** to directory **proj1copy,** for example, you would type

**cp -r project1 proj1copy**

If directory **proj1copy** already exists, this command will put a duplicate copy of **directory project1** into directory **proj1copy\**.

## Removing a Directory: -

Use the command **rmdir** to remove an empty directory. Multiple empty directories may be removed by listing them after the command:

**rmdir testdir1 testdir2**

If you try to remove a directory that is not empty, you will see

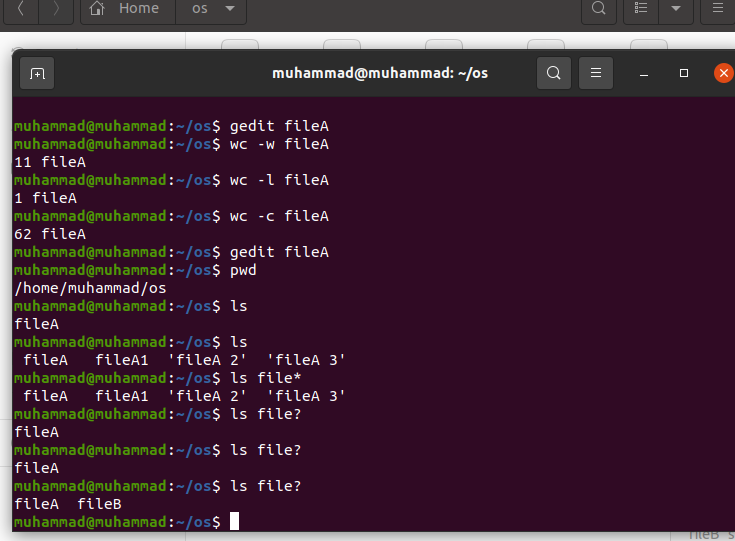
**rmdir: testdir3: Directory not empty**

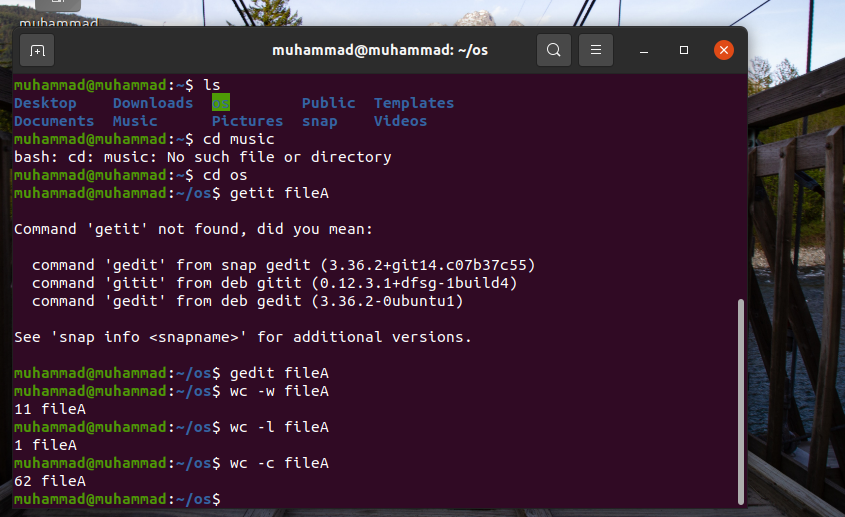
If you are sure that you want to remove the directory and all the files it contains, use the command

**rm -r testdir3**

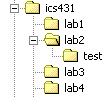
**Assignment Problems:**

**1.Run all the commands given in the Lab Notes, and observe the output for each command.**

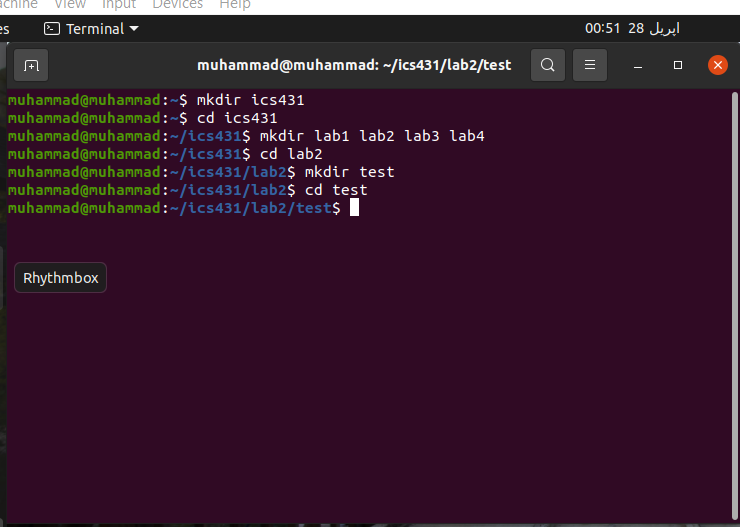
**Answer:**

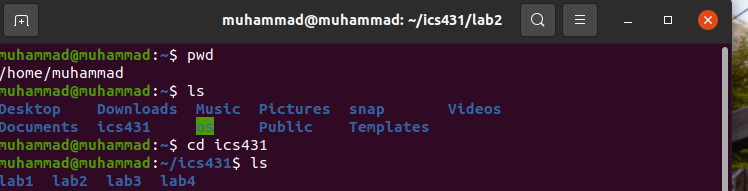


**2. Create a directory named ics431 under  your home directory. Then create directories named lab1 lab2 lab3 ... inside the ics431 directory. Also create a directory named test inside the directory lab2. Write all the commands.**



**Answer:**

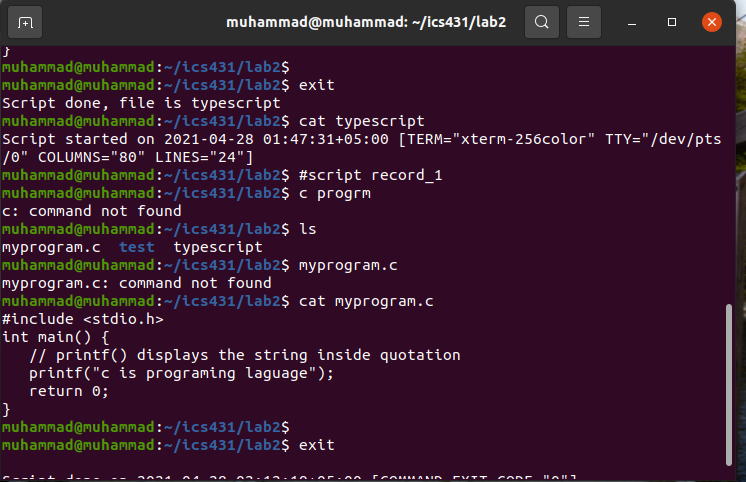
****

****

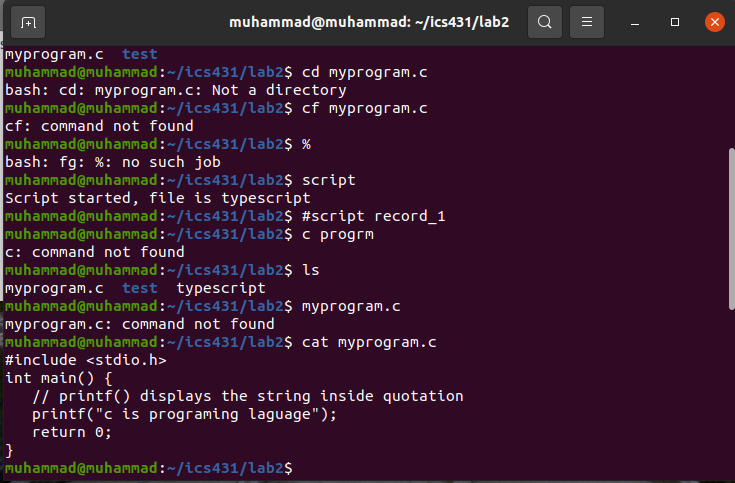
**3. a. Now create a C program file named myprogram.c which displays "C is a programming Language". This file should be in lab2 directoty.**

**Answer:**

I have not install c library yet in unix operating system so it will not be run.

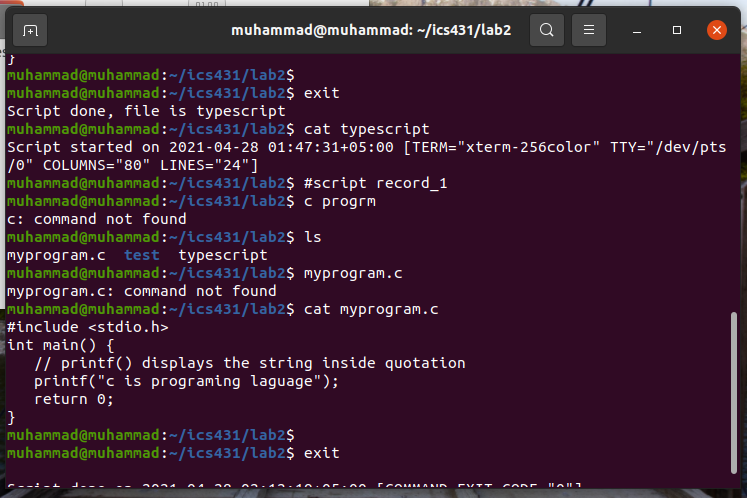
****

**b. Record (using script command) the following operations in a file called rec - in the directory test.**

****

**i.Compile the program**

**ii. Execute the program and see the output**

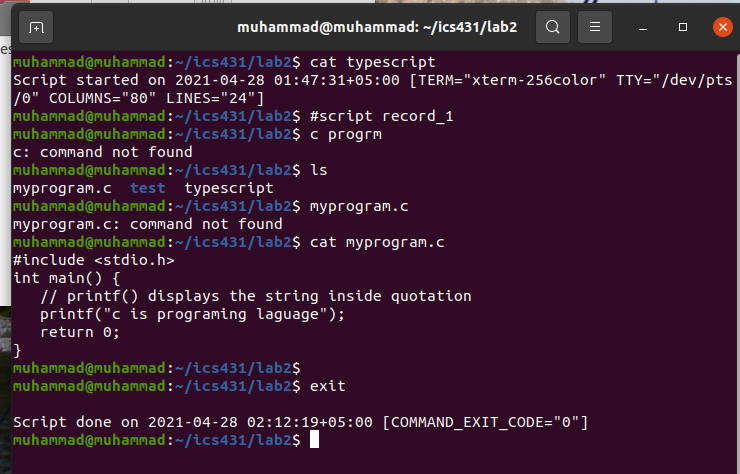


**iii. stop recording and see the rec file contents using cat command**

**iv. redirect the output of this program to a file called out**

**v. What is the content of out and rec. Are They same?**

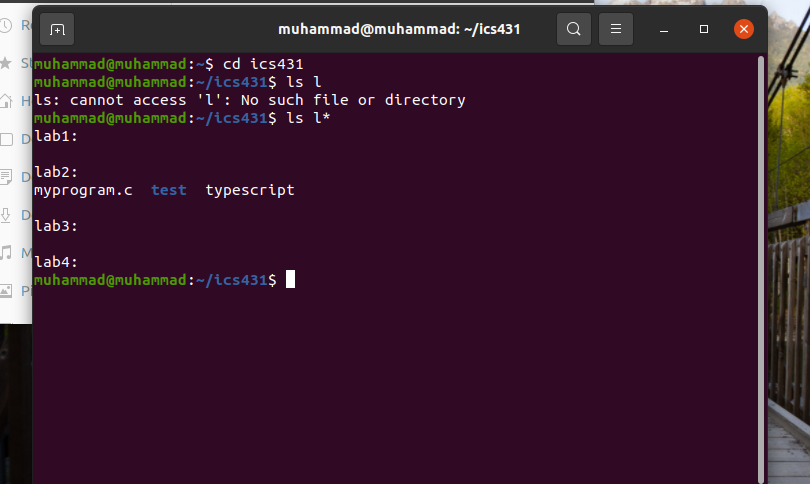
**Answer:**



**4. Go to the ics431 directory and list all the directory names starting with l. Write the commands.**

**Answer:**

**Command:** ls l\*



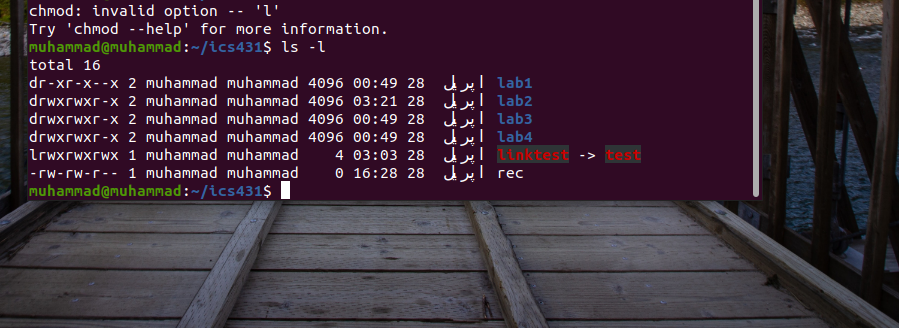
**5. From ics431 directory, create a soft link to the test directory in the name linktest.**

**a. Go to linktest directory and display the files. What Files are displayed?**

**Answer:**

**Command:** ln -r test linktest

it will display that (**linktest -> test)** linktest is Simulink to test directory.



**b. From there go to the parent directory. Which parent are you getting? State reasons.**

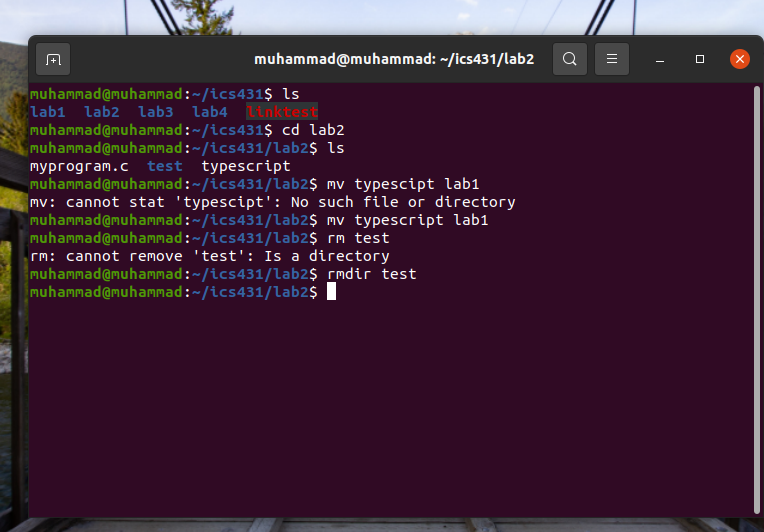
**Answer:** parent is the ics431 directory b/c we Simulink the linktest directory with test from ics431.

**6. Try to open the created file myprogram.c in the notepad of your desktop computer. This can be done by using ftp. Modify the program in notepad to print " Now I like know Unix and windows OS". and execute in the Unix environment (again ftp is needed). Write all the commands to do this.**

**Answer:** Sorry, I have not available notepad in my new OS Unix

**7. Move the file rec to the directory lab1 and delete the directory test and observe what had happened to linktest.** **What is it pointing to?**

**Answer:** move command will the typescript to lab1 and delete test directory will delete linktest directory b/c both are Simulink.

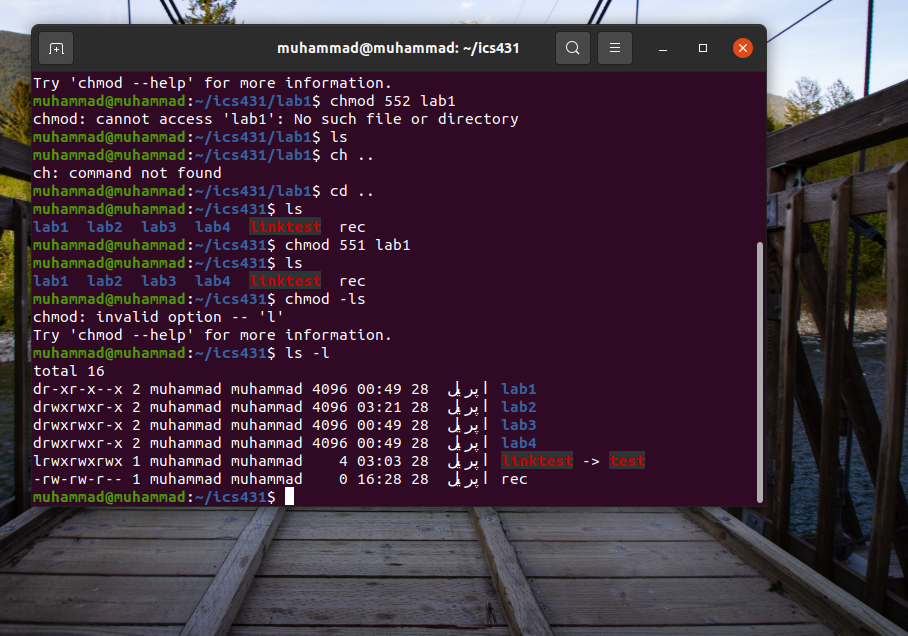
****

**8. Record the following:**

**a. Go to directory lab1**

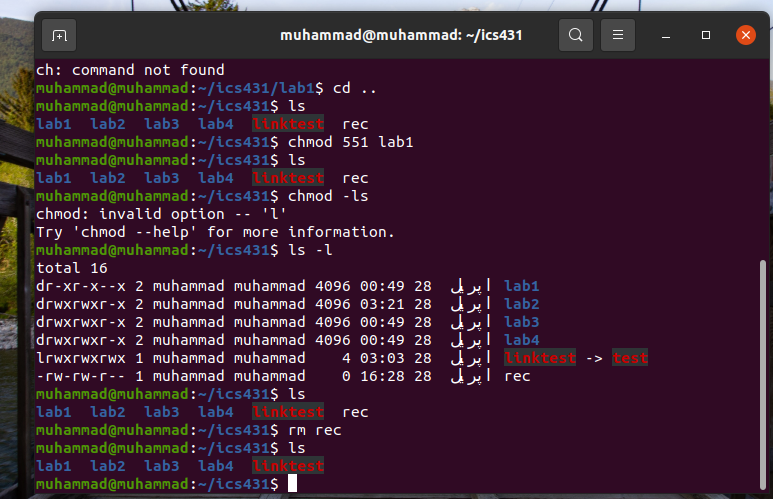
**b. change the modes of all files to [read exec to owner & group and only execute to others]**

**Answer:**



**c. Try to delete rec file and observe the o/p. Write the o/p**

**Answer:**



**9. Display a file containing all the full names (in sorted order) of the users currently logged in to the Unix server**

**Write the commands to do this.**

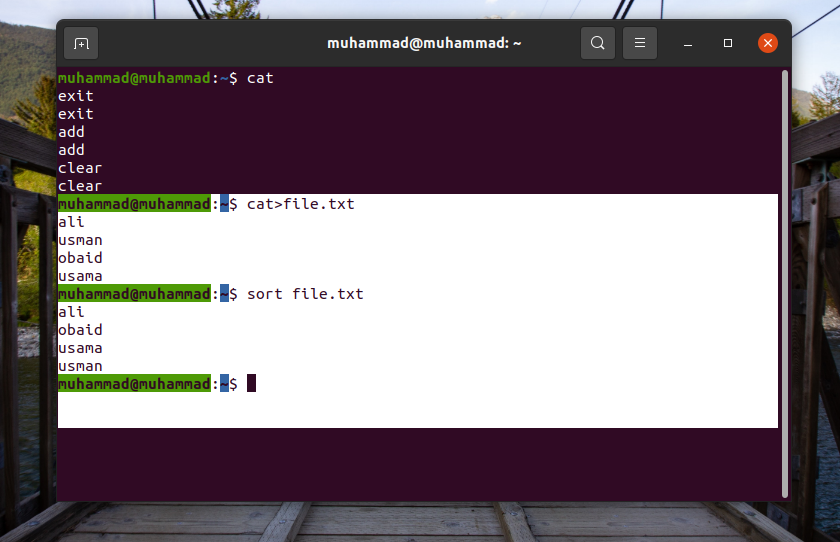
**Answer:**

**Command:** (cat>file.txt)

This command is to enter list of file to be sorted.

**Command:** sort file.txt

This command will display all the file in sorted order.

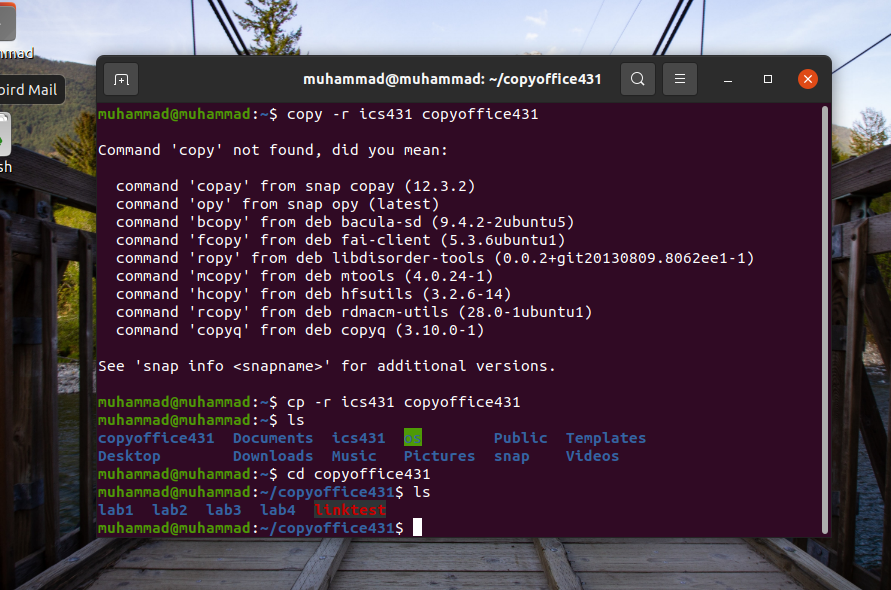


**10. Make a copy of the directory ics431 in the same level and name it as CopyOfics431. (All subdirectories and files inside should be copied.)**

**Answer:**

**Command: cp -r ics431 copyoffice431**

this command will copy ics431 directory and all subdirectory to copyoffice431 new directory.

****