LAB MANUAL

Submitted by :Tanzeela Asghar
Submitted to: Mr.Majid Shafique.
Roll no : 2021-BSE-032
Dated: January 27,2023
Course: Operating System Lab

Lab 01

Summary

Items	Description
Course Title	Operating System
Lab Title	**Introduction to Linux**
Duration	3 Hours
Operating System /Tool/Language	Linux Operating System
Objective	To get familiar with the Linux operating system, and also their installation.

LAB TASK

Question 1:

What is the difference between FAT / NTFS / Ext3.

SOLUTION:

NTFS:

- Support for files and disks of larger capacity that any other file systems can deal with.
- Support for extended file names and foreign characters from difficult languages.
- Severe decrease in system performance when «chkdsk» is used to check the local hard disk or an external disk.
- The standard system maintenance app <<chkdsk>> is notorious for its sluggishness.
- Increased security with the new file encryption feature.
- Much faster operation on drives smaller than 40 GB.
- Smaller files clusters .

- User permission for files and folders.
- File copies are <<undone>> if the interrupted cluster is cleaned.
- Small files are kept in the Master File Table at the beginning of the drive.

FAT :

- It is incompatible with the latest version of <<windows>>, and supports hard disk from 32 MB to 2TB.
- More powerful and efficient recovery utilities.
- Support for quick <<chkdsk>> operation .
- Simple operating system allocation and quicker file read algorithm .
- Faster operation with disks less than 10 GB in capacity.
- Cluster chains containing data from interrupted copies are marked as damaged.
- Master File table is separated from other files.

EXT3:

Ext3 (the third extended file system) is the most commonly used files system on Linux Ext3 file system was introduced in 2001 and same was integrated in Kernel .2.4.15 journaling feature, which is to improve reliability and eliminates need to check file system after unclean shutdown .[1]

.

Cons:

- The main benefit of ext3 is that it allows journaling.
- Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling.
- There are three types of journaling available in the ext3 file system.

1. Journal – Metadata and content are saved in the journal.
2. Ordered – Only metadata is saved in the journal. Metadata are journaled only after writing the content to disk. This is the default.
3. Writeback – Only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to the disk. Max file size 16GB – 2TB. Provide facility to upgrade from Ext2 to Ext3 file systems without having to back up and restore data.

QUESTION 02:

Differentiate CLI & GUI

- A GUI is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators. A CLI is an interface for the user to issue commands in the form of successive lines of text or command lines to perform the tasks.
- GUI stands for Graphical User Interface whereas CLI stands for Command Line Interface.
- It is easy to use the GUI. It is not necessary to have a vast knowledge to operate the system using the GUI. Even a beginner can easily handle the tasks using the GUI. On the other hand, CLI is complex. The user should have good knowledge of the commands. Misspelt commands are of no use.

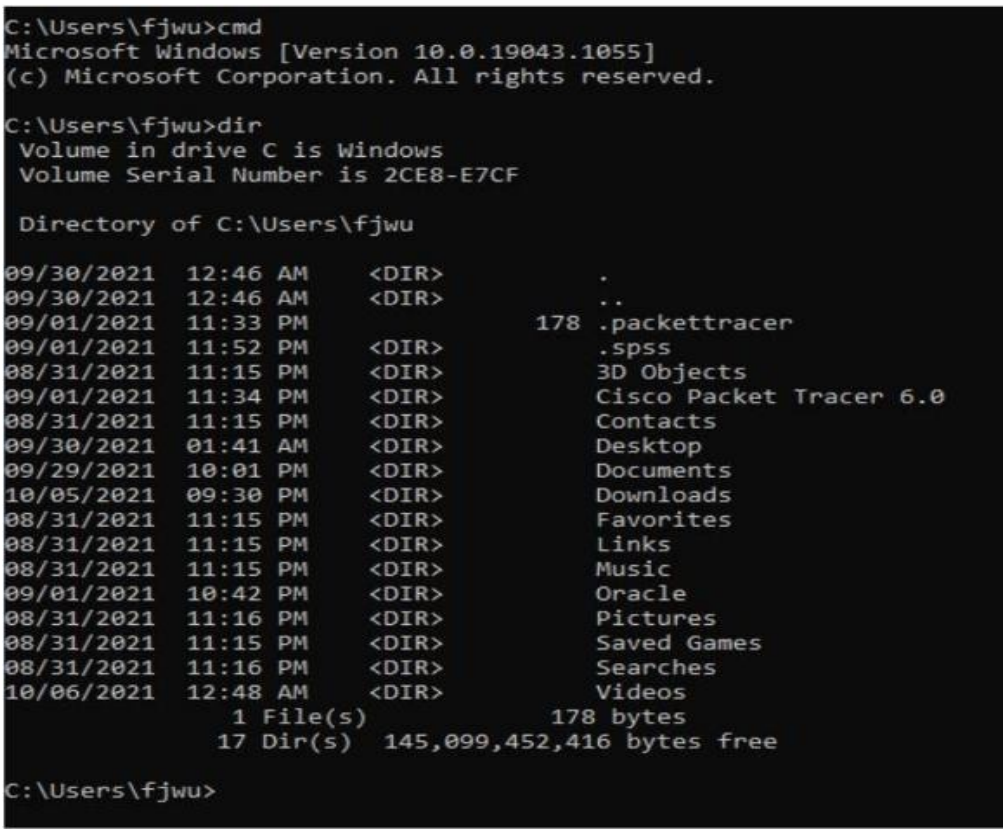
1

- GUI requires more memory as it contains a lot of graphical components. CLI is a command interface, and it does not require more memory. GUI is slower but the CLI is fast. The user can change the appearance of the GUI. There are customizable options to change the appearance.

Question 3

Implement DOC Commands with output screenshot as per task document No 2.

- **Dir**



- **DIR /W**

```
C:\Users\fjwu>dir /w
 Volume in drive C is Windows
 Volume Serial Number is 2CE8-E7CF

 Directory of C:\Users\fjwu

[.]             [..]            .packettracer   [.spss]         [3D Objects]        [Cisco Packet Tracer 6.0]
[Contacts]      [Desktop]       [Documents]     [Downloads]     [Favorites]         [Links]
[Music]         [Oracle]        [Pictures]      [Saved Games]   [Searches]          [Videos]
               1 File(s)            178 bytes
              17 Dir(s)  145,074,536,448 bytes free
```

- **DIR /P**

```
C:\Users\fjwu>dir /p
 Volume in drive C is Windows
 Volume Serial Number is 2CE8-E7CF

 Directory of C:\Users\fjwu

09/30/2021  12:46 AM    <DIR>          .
09/30/2021  12:46 AM    <DIR>          ..
09/01/2021  11:33 PM               178 .packettracer
09/01/2021  11:52 PM    <DIR>          .spss
08/31/2021  11:15 PM    <DIR>          3D Objects
09/01/2021  11:34 PM    <DIR>          Cisco Packet Tracer 6.0
08/31/2021  11:15 PM    <DIR>          Contacts
10/06/2021  02:47 AM    <DIR>          Desktop
10/06/2021  02:45 AM    <DIR>          Documents
10/05/2021  09:30 PM    <DIR>          Downloads
08/31/2021  11:15 PM    <DIR>          Favorites
08/31/2021  11:15 PM    <DIR>          Links
08/31/2021  11:15 PM    <DIR>          Music
09/01/2021  10:42 PM    <DIR>          Oracle
08/31/2021  11:16 PM    <DIR>          Pictures
08/31/2021  11:15 PM    <DIR>          Saved Games
08/31/2021  11:16 PM    <DIR>          Searches
10/06/2021  12:48 AM    <DIR>          Videos
               1 File(s)            178 bytes
              17 Dir(s)  145,073,127,424 bytes free
```

- **DIR /W/P**

```
C:\Users\fjwu>dir /w/p
 Volume in drive C is Windows
 Volume Serial Number is 2CE8-E7CF

 Directory of C:\Users\fjwu

[.]                 [..]                .packettracer       [.spss]             [3D Objects]        [Cisco Packet Tracer 6.0]
[Contacts]          [Desktop]           [Documents]         [Downloads]         [Favorites]         [Links]
[Music]             [Oracle]            [Pictures]          [Saved Games]       [Searches]          [Videos]
               1 File(s)            178 bytes
              17 Dir(s)  145,072,705,536 bytes free
```

- **CD**

```
C:\Users\fjwu>cd\

C:\>
```

- **MD fruit**

```
C:\>md fruit

C:\>dir
 Volume in drive C is Windows
 Volume Serial Number is 2CE8-E7CF

 Directory of C:\

09/01/2021  01:27 AM    <DIR>          app
09/29/2021  09:50 PM    <DIR>          boot
09/01/2021  11:41 PM    <DIR>          Certiport
09/02/2021  12:48 AM    <DIR>          emu8086
11/07/2007  08:00 AM            17,734 eula.1028.txt
11/07/2007  08:00 AM            17,734 eula.1031.txt
11/07/2007  08:00 AM            10,134 eula.1033.txt
11/07/2007  08:00 AM            17,734 eula.1036.txt
11/07/2007  08:00 AM            17,734 eula.1040.txt
11/07/2007  08:00 AM               118 eula.1041.txt
11/07/2007  08:00 AM            17,734 eula.1042.txt
11/07/2007  08:00 AM            17,734 eula.2052.txt
11/07/2007  08:00 AM            17,734 eula.3082.txt
10/06/2021  02:54 AM    <DIR>          fruit
11/07/2007  08:00 AM             1,110 globdata.ini
08/31/2021  11:09 PM    <DIR>          ImDisk
11/07/2007  08:03 AM           562,688 install.exe
11/07/2007  08:00 AM               843 install.ini
11/07/2007  08:03 AM            76,304 install.res.1028.dll
11/07/2007  08:03 AM            96,272 install.res.1031.dll
11/07/2007  08:03 AM            91,152 install.res.1033.dll
11/07/2007  08:03 AM            97,296 install.res.1036.dll
11/07/2007  08:03 AM            95,248 install.res.1040.dll
11/07/2007  08:03 AM            81,424 install.res.1041.dll
11/07/2007  08:03 AM            79,888 install.res.1042.dll
11/07/2007  08:03 AM            75,792 install.res.2052.dll
11/07/2007  08:03 AM            96,272 install.res.3082.dll
08/31/2021  11:12 PM    <DIR>          Intel
12/07/2019  02:14 AM    <DIR>          PerfLogs
09/29/2021  09:49 PM    <DIR>          Program Files
09/02/2021  10:28 PM    <DIR>          Program Files (x86)
10/06/2021  02:53 AM    <DIR>          TEMP
09/01/2021  11:43 PM    <DIR>          TurboC++
```

```
08/31/2021  11:34 PM    <DIR>          Users
11/07/2007  08:00 AM             5,686 vcredist.bmp
11/07/2007  08:09 AM         1,442,522 VC_RED.cab
11/07/2007  08:12 AM           232,960 VC_RED.MSI
09/02/2021  12:44 AM    <DIR>          Windows
09/02/2021  11:28 PM    <DIR>          xampp
              24 File(s)      3,169,847 bytes
              15 Dir(s)  145,069,985,792 bytes free
```

- **CD fruit:**

```
C:\>cd fruit

C:\fruit>md grapes

C:\fruit>dir
 Volume in drive C is Windows
 Volume Serial Number is 2CE8-E7CF

 Directory of C:\fruit

10/06/2021  02:58 AM    <DIR>          .
10/06/2021  02:58 AM    <DIR>          ..
10/06/2021  02:58 AM    <DIR>          grapes
               0 File(s)              0 bytes
               3 Dir(s)  143,720,574,976 bytes free
```

- **CD grapes:**

```
C:\fruit>cd grapes

C:\fruit\grapes>
```

- **CD.. :**

```
C:\fruit\grapes>cd..

C:\fruit>
```

Lab 02

Introduction to Basic Shell Commands

Examples

Command: cat

```
tanzeela@ubuntu:~$ cat > file1
im file 1
tanzeela@ubuntu:~$ cat > file2
im file2
tanzeela@ubuntu:~$ cat file1 file2 > file3
```

Command: pwd

```
tanzeela@ubuntu:~$ pwd
/home/tanzeela
tanzeela@ubuntu:~$
```

Command: ls

- **ls**

```
tanzeela@ubuntu:~$ ls
Desktop           file2            labs              Public
Documents         file3            Music             students.txt
Downloads         foo              OSLAB             Templates
examples.desktop  gstudentsort.txt pgstudentsort.txt trimf
fib               gstudents.txt    pgstudent.txt     trmif
file1             gstudent.txt     Pictures          Videos
tanzeela@ubuntu:~$
```

- **ls -al**

```
tanzeela@ubuntu:~$ ls -al
total 148
drwxr-xr-x 17 tanzeela tanzeela 4096 Nov  1 05:49 .
drwxr-xr-x  3 root     root     4096 Oct 27 00:05 ..
-rw-------  1 tanzeela tanzeela 3138 Nov  1 05:48 .bash_history
-rw-r--r--  1 tanzeela tanzeela  220 Oct 27 00:05 .bash_logout
-rw-r--r--  1 tanzeela tanzeela 3771 Oct 27 00:05 .bashrc
drwx------ 10 tanzeela tanzeela 4096 Oct 30 03:36 .cache
drwx------ 13 tanzeela tanzeela 4096 Oct 29 23:52 .config
drwxr-xr-x  2 tanzeela tanzeela 4096 Nov  1 01:59 Desktop
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Documents
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Downloads
```

Command: mv

```
tanzeela@ubuntu:~$ mv laboslab
mv: missing destination file operand after 'laboslab'
Try 'mv --help' for more information.
tanzeela@ubuntu:~$
```

Command: rm

```
tanzeela@ubuntu:~$ rm oslab
rm: cannot remove 'oslab': No such file or directory
tanzeela@ubuntu:~$ cat > oslab
lab os
tanzeela@ubuntu:~$ rm oslab
tanzeela@ubuntu:~$
```

LAB TASKS

Verify that you are in your home directory. Make the directory L*ABS* using the following command. List the files in the current directory to verify that the directory *LABS* has been made correctly. Change directories to *LABS*. Create the file named *file1*. List the contents of the file *file1* to the screen. Make a copy of the file *file1* under the name *file2*. Verify that the files *file1* and *file2* both exist. List the contents of both *file1* and *file2* to the monitor screen. Then delete the file *file1*. Clear the window. Rename *file2* to *thefile*. Copy *thefile* to your home directory. Remove *thefile* from the current directory. Copy *thefile* from your home directory to the current directory. Change directories to your home directory. Remove *thefile* from your home directory and from directory *LABS*. Verify *thefile* is removed from the directory *LABS*. Remove the

directory *LABS* from your home directory with the following command. Verify that *thefile* and *LABS* are gone from your home directory.

Solution

Verify that you are in your home directory

```
tanzeela@ubuntu:~$ pwd
/home/tanzeela
tanzeela@ubuntu:~$
```

Make the directory LABS using the following command .List the files in the current directory to verify that the directory LABS has been correctly.

```
tanzeela@ubuntu:~$ mkdir labs
tanzeela@ubuntu:~$ ls -al
total 124
drwxr-xr-x 17 tanzeela tanzeela 4096 Oct 30 04:54 .
drwxr-xr-x  3 root     root     4096 Oct 27 00:05 ..
-rw-------  1 tanzeela tanzeela 1823 Oct 30 04:17 .bash_history
-rw-r--r--  1 tanzeela tanzeela  220 Oct 27 00:05 .bash_logout
-rw-r--r--  1 tanzeela tanzeela 3771 Oct 27 00:05 .bashrc
drwx------ 10 tanzeela tanzeela 4096 Oct 30 03:36 .cache
drwx------ 13 tanzeela tanzeela 4096 Oct 29 23:52 .config
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 30 00:07 Desktop
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Documents
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Downloads
-rw-r--r--  1 tanzeela tanzeela 8980 Oct 27 00:05 examples.desktop
-rw-rw-r--  1 tanzeela tanzeela   95 Oct 30 04:04 file1
drwx------  3 tanzeela tanzeela 4096 Oct 29 23:12 .gnupg
-rw-rw-r--  1 tanzeela tanzeela  108 Oct 29 11:30 gstudentsort.txt
-rw-rw-r--  1 tanzeela tanzeela    0 Oct 29 10:53 gstudents.txt
-r-xr-xr-x  1 tanzeela tanzeela  108 Oct 29 11:00 gstudent.txt
-rw-------  1 tanzeela tanzeela 1272 Oct 30 03:26 .ICEauthority
drwxrwxr-x  2 tanzeela tanzeela 4096 Oct 30 04:54 labs
drwx------  3 tanzeela tanzeela 4096 Oct 27 00:55 .local
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Music
drwxrwxr-x  2 tanzeela tanzeela 4096 Oct 29 11:15 OSLAB
```

```
drwxrwxr-x  2 tanzeela tanzeela 4096 Oct 29 11:15 OSLAB
-rw-rw-r--  1 tanzeela tanzeela  104 Oct 29 11:26 pgstudentsort.txt
-r-xr-xr-x  1 tanzeela tanzeela  104 Oct 29 11:07 pgstudent.txt
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Pictures
-rw-r--r--  1 tanzeela tanzeela  807 Oct 27 00:05 .profile
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Public
drwx------  2 tanzeela tanzeela 4096 Oct 29 23:12 .ssh
-r--r--r--  1 tanzeela tanzeela   96 Oct 29 11:38 students.txt
-rw-r--r--  1 tanzeela tanzeela    0 Oct 29 23:14 .sudo_as_admin_successful
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Templates
drwxr-xr-x  2 tanzeela tanzeela 4096 Oct 27 00:55 Videos
tanzeela@ubuntu:~$ ^C
tanzeela@ubuntu:~$
```

Change directories to LABS. And create the file.

```
tanzeela@ubuntu:~$ cd labs
tanzeela@ubuntu:~/labs$
```

```
tanzeela@ubuntu:~/labs$ ls -al
total 8
drwxrwxr-x  2 tanzeela tanzeela 4096 Oct 30 04:54 .
drwxr-xr-x 17 tanzeela tanzeela 4096 Oct 30 04:54 ..
tanzeela@ubuntu:~/labs$
```

Make the copy of file file1 under the name file2.

```
tanzeela@ubuntu:~/labs$ cp file1.txt file2.txt
```

Verify that the files file1 and file2 both exits.

```
tanzeela@ubuntu:~/labs$ ls -al
total 16
drwxrwxr-x  3 tanzeela tanzeela 4096 Oct 30 05:22 .
drwxr-xr-x 17 tanzeela tanzeela 4096 Oct 30 04:54 ..
drwxrwxr-x  2 tanzeela tanzeela 4096 Oct 30 05:06 file1
-rw-rw-r--  1 tanzeela tanzeela   12 Oct 30 05:22 file2
tanzeela@ubuntu:~/labs$
```

List the contents of both files file1 and file2 to monitor screen.

```
tanzeela@ubuntu:~/labs$ la -al file1
total 8
drwxrwxr-x 2 tanzeela tanzeela 4096 Oct 30 05:06 .
drwxrwxr-x 3 tanzeela tanzeela 4096 Oct 30 05:22 ..
tanzeela@ubuntu:~/labs$ ls -al file2
-rw-rw-r-- 1 tanzeela tanzeela 12 Oct 30 05:22 file2
tanzeela@ubuntu:~/labs$
```

Then delete the file1

```
tanzeela@ubuntu:~/labs$ rmdir file1
tanzeela@ubuntu:~/labs$ ls -al
total 12
drwxrwxr-x  2 tanzeela tanzeela 4096 Oct 30 05:36 .
drwxr-xr-x 17 tanzeela tanzeela 4096 Oct 30 04:54 ..
```

Clear the window

```
tanzeela@ubuntu:~/labs$ clear
```

The following screen shows its clear

```
tanzeela@ubuntu:~/labs$
```

Rename the file2.

```
tanzeela@ubuntu:~/labs$ ls -al
total 12
drwxrwxr-x  2 tanzeela tanzeela 4096 Oct 30 05:36 .
drwxr-xr-x 17 tanzeela tanzeela 4096 Oct 30 04:54 ..
-rw-rw-r--  1 tanzeela tanzeela   12 Oct 30 05:22 Renamedfile
tanzeela@ubuntu:~/labs$
```

LAB NO 3

LAB TASKS

Examples :

Command : date

```
tanzeela@ubuntu:~$ date
Sun Oct 30 03:54:38 PDT 2022
tanzeela@ubuntu:~$
```

Command : cal

```
tanzeela@ubuntu:~$ cal
    October 2022
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
tanzeela@ubuntu:~$
```

Command : head

```
tanzeela@ubuntu:~$ cat > file1
tanzeela
ayesha
mariyam
bisma
tanzeela@ubuntu:~$ head -n2 file1
tanzeela
ayesha
tanzeela@ubuntu:~$
```

Command : head

```
tanzeela@ubuntu:~$ tail -n2 file1
mariyam
bisma
tanzeela@ubuntu:~$
```

Command : ls -l

```
tanzeela@ubuntu:~$ ls -l
total 88
drwxr-xr-x 2 tanzeela tanzeela 4096 Nov  1 01:59 Desktop
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Documents
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Downloads
-rw-r--r-- 1 tanzeela tanzeela 8980 Oct 27 00:05 examples.desktop
-rw-rw-r-- 1 tanzeela tanzeela   30 Nov  1 05:24 file1
-rw-rw-r-- 1 tanzeela tanzeela   18 Nov  1 01:29 foo
-rw-rw-r-- 1 tanzeela tanzeela  108 Oct 29 11:30 gstudentsort.txt
-rw-rw-r-- 1 tanzeela tanzeela    0 Oct 29 10:53 gstudents.txt
-r-xr-xr-x 1 tanzeela tanzeela  108 Oct 29 11:00 gstudent.txt
drwxrwxr-x 2 tanzeela tanzeela 4096 Oct 30 05:36 labs
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Music
drwxrwxr-x 2 tanzeela tanzeela 4096 Oct 29 11:15 OSLAB
-rw-rw-r-- 1 tanzeela tanzeela  104 Oct 29 11:26 pgstudentsort.txt
-r-xr-xr-x 1 tanzeela tanzeela  104 Oct 29 11:07 pgstudent.txt
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Pictures
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Public
-r--r--r-- 1 tanzeela tanzeela   96 Oct 29 11:38 students.txt
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Templates
-rw-rw-r-- 1 tanzeela tanzeela    4 Nov  1 01:23 trimf
-rwxr-xr-x 1 tanzeela tanzeela   20 Nov  1 01:29 trmif
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Videos
tanzeela@ubuntu:~$
```

Command : chmod

```
tanzeela@ubuntu:~$ chmod 770 file1
tanzeela@ubuntu:~$ ls -l
total 88
drwxr-xr-x 2 tanzeela tanzeela 4096 Nov  1 01:59 Desktop
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Documents
drwxr-xr-x 2 tanzeela tanzeela 4096 Oct 27 00:55 Downloads
-rw-r--r-- 1 tanzeela tanzeela 8980 Oct 27 00:05 examples.desktop
-rwxrwx--- 1 tanzeela tanzeela   30 Nov  1 05:24 file1
```

Command : ls

```
tanzeela@ubuntu:~$ ls >> file1
tanzeela@ubuntu:~$ cat file1
tanzeela
ayesha
mariyam
bisma
Desktop
Documents
Downloads
examples.desktop
file1
foo
gstudentsort.txt
gstudents.txt
gstudent.txt
labs
```

Task 1:

Create file name students.txt, gstudent.txt, pgstudents. Enter students' names in gstudent.txt and pgstudent.txt. Now create directory having name OSLAB and copy all files to this directory. Now append the file students.txt with first five sorted names from pstudent.txt and last five sorted names from gstudent.txt. Then show the contents of sorted names from file students.txt. Change the permissions of file students.txt read only and both other files read and execute only.

Solution:

Create file name students.txt, gstudent.txt, pgstudents.

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tanzeela@ubuntu:~$ touch students.txt
tanzeela@ubuntu:~$ touch gstudents.txt
tanzeela@ubuntu:~$ touch pgstudent.txt
tanzeela@ubuntu:~$ ls
Desktop    examples.desktop  pgstudent.txt  students.txt
Documents  gstudents.txt     Pictures       Templates
Downloads  Music             Public         Videos
tanzeela@ubuntu:~$
```

Enter students' names in gstudent.txt and pgstudent.txt

```
tanzeela@ubuntu:~$ cat > gstudent.txt
Tanzeela
Bisma
Sameer
Ayesha
Minahil
Usman
Abeera
Fiza
Abdullah
^Z
[1]+ Stopped                 cat > gstudent.txt
```

```
tanzeela@ubuntu:~$ cat > pgstudent.txt
Sajid
Majid
Warda
Aiza
Rehan
Hamza
Amna
Tanzeela
Jiya
^Z
[2]+ stopped                 cat > pgstudent.txt
^C
```

Now create directory having name OSLAB

```
tanzeela@ubuntu:~$ mkdir OSLAB
tanzeela@ubuntu:~$
```

Copy all files to this directory.

```
tanzeela@ubuntu:~$ cp students.txt gstudent.txt pgstudent.txt OSLAB
tanzeela@ubuntu:~$
```

Now append the files student.txt with first five sorted names from pgstudent.txt and the last five sorted names with gstudent.txt.

```
tanzeela@ubuntu:~$ sort < pgstudent.txt > pgstudentsort.txt
tanzeela@ubuntu:~$ head -n5 pgstudent.txt
Sajid
Majid
Warda
Aiza
Rehan
tanzeela@ubuntu:~$ sort < gstudent.txt > gstudentsort.txt
tanzeela@ubuntu:~$ tail -n5 gstudent.txt
Abeera
Fiza
Abdullah
^Z
[1]+ Stopped                 cat > gstudent.txt
tanzeela@ubuntu:~$ head -n5 pgstudentsort.txt > students.txt
tanzeela@ubuntu:~$
```

```
tanzeela@ubuntu:~$ tail -n5 gstudentsort.txt >> students.txt
tanzeela@ubuntu:~$
```

Show the contents of sorted names from file students.txt.

```
tanzeela@ubuntu:~$ ^C
tanzeela@ubuntu:~$ cat students.txt
[2]+ stopped                 cat > pgstudent.txt
Aiza
Amna
^C
Hamza
Minahil
Sameer
Tanzeela
Usman
^Z
tanzeela@ubuntu:~$
```

Change the permissions of file students.txt read only and both other files read and execute only.

```
tanzeela@ubuntu:~$ chmod 444 students.txt
tanzeela@ubuntu:~$ ls -l
total 68
```

```
tanzeela@ubuntu:~$ chmod 555 gstudent.txt pgstudent.txt
tanzeela@ubuntu:~$ ls -l
total 68
```

Task 2:

Define variables x and y with value 20 and 30 and print it on screen, then sum, subtract, divide these numbers and print it on the screen.

Define variable x and y with value 20 and 30 and print it on screen, then sum, subtract,

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tanzeela@ubuntu:~$ x=20
tanzeela@ubuntu:~$ y=30
tanzeela@ubuntu:~$ echo $x
20
tanzeela@ubuntu:~$ echo $y
30
tanzeela@ubuntu:~$

tanzeela@ubuntu:~$ ^C
tanzeela@ubuntu:~$ expr 20 + 30
50
tanzeela@ubuntu:~$ expr 20 - 30
-10
tanzeela@ubuntu:~$ expr 20/30
20/30
tanzeela@ubuntu:~$ expr 20 / 30
0
tanzeela@ubuntu:~$
```

Task 3:

Write a program that will displays the fibanocaii series of any given number. Take the limit from the user

```
tanzeela@ubuntu:~$ echo "How many number of terms to be generated ?"
How many number of terms to be generated ?
tanzeela@ubuntu:~$ read n
9
tanzeela@ubuntu:~$ function fib
> {
> x=0
> y=1
> i=2
> echo "Fibonacci Series up to $n terms :"
> echo "$x"
> echo "$y"
> while [ $i -lt $n ]
> do
> i='expr $i + 1 '
> z='expr $x + $y '
> echo "$z"
> x=$y
> y=$z
> done
> }
tanzeela@ubuntu:~$ r='fib $n'
tanzeela@ubuntu:~$ echo "$r"
```

Lab 04

LAB TASKS

Task 1:

Write shell script as follows:

```
$ chmod 755 trmif
```

```
tanzeela@ubuntu:~$ cat > trmif
# script to test rm command and exist status
if rm $1
then
echo "$1 file detected"
fi
tanzeela@ubuntu:~$ chmod 755 trmif
tanzeela@ubuntu:~$
```

```
Fibonacci series upto 9 terms:
0
1
1
2
3
5
8
13
21
tanzeela@ubuntu:~$
```

Press Ctrl + d to save

Answer the following question in reference to above script:

1. foo file exists on your disk and you give command, **$ ./trmfi foo** what will be output?

```
tanzeela@ubuntu:~$ cat > foo
```

```
tanzeela@ubuntu:~$ cat > foo
I'm in file foo
tanzeela@ubuntu:~$ ./trmif foo
foo file detected
tanzeela@ubuntu:~$
```

2. If bar file not present on your disk and you give command, **$ ./trmfi bar** what will be output?

```
tanzeela@ubuntu:~$ ./trmif bar
rm: cannot remove 'bar': No such file or directory
tanzeela@ubuntu:~$
```

3. And if you type **$ ./trmfi** What will be output?

```
tanzeela@ubuntu:~$ ./trmif
rm: missing operand
Try 'rm --help' for more information.
tanzeela@ubuntu:~$
```

Task 2:

Write a shell script that computes the gross salary of an employee according to the following:

1) if basic salary is <1500 then HRA=10% of the basic salary.
2) if basic salary is >1500 then HRA=20% of the basic salary.

```
echo "Enter Basic Salary"
read basic
if [ $basic -gt 1500 ]
then
        HRA=$(((($basic*10)/(100)))
        echo "Gross Salary= $HRA"
elif [ $basic -lt 1500 ]
then
        HRA=$(((($basic*20)/(100)))
        echo "Gross Salary= $HRA"
else
        echo "Must enter Basic salary."
fi
```

Terminal

```
tanzeela@ubuntu:~$ cd Desktop
tanzeela@ubuntu:~/Desktop$ chmod 755 lab4.sh
tanzeela@ubuntu:~/Desktop$ ./lab4.sh
Enter Basic Salary
1350
Gross Salary= 270
tanzeela@ubuntu:~/Desktop$ ./lab4.sh
Enter Basic Salary
1850
Gross Salary= 185
tanzeela@ubuntu:~/Desktop$
```

Task 3:

Write a shell script to ADD two numbers taken from argument?

```
if [ $# -eq 2 ]
then
add=$(($1+$2))
echo "Sum = $add"
else
echo "Must enter only two numbers . "
fi
```

TERMINAL

```
tanzeela@ubuntu:~/Desktop$ chmod 755 lab4.sh
tanzeela@ubuntu:~/Desktop$ ./lab4.sh 6 7
Sum = 13
tanzeela@ubuntu:~/Desktop$ ./lab4.sh 3 6 9
Must enter only two numbers .
tanzeela@ubuntu:~/Desktop$
```

Lab 05

Example:

```c
#include<stdio.h>
#include<unistd.h>//contains prototype for fork
#include<sys/types.h>//contains prototype for pid_t
int main()
{
pid_t pid;
pid = fork();
if (pid==0)
printf("\n I'm the child process");
else if (pid>0)
printf("\n I'm the parent process.My child pid is %d",pid);
else
perror("error in fork");
return 0;
}
```

Output

```
tanzeela@ubuntu:~/Desktop$ gcc -o pid pid.c
tanzeela@ubuntu:~/Desktop$ ./pid 0

 I'm the parent process.My child pid is 45582tanzeela@ubuntu:~/Desktop$
 I'm the child process
```

Example

```c
#include<stdio.h>
#include<sys/wait.h>//contains prototype for wait
#include<stdlib.h>
void main()
{
int pid , status;
printf("Hello World!\n");
pid = fork();
if(pid==-1)//check for error in fork
{
printf("fork failed\n");
exit(1);
}
if(pid==0)
{
printf("Child!\n");
}
else
{
wait(&status);//parent waits for child to finish
printf("Parent wait Finish!\n");
}
}
```

Output

```
tanzeela@ubuntu:~/Desktop$ gcc -o fork fork.c
fork.c: In function 'main':
fork.c:8:7: warning: implicit declaration of function 'fork' [-Wimplicit-functi
on-declaration]
 pid = fork();
       ^~~~
tanzeela@ubuntu:~/Desktop$ ./fork
Hello World!
Child!
Parent wait Finish!
tanzeela@ubuntu:~/Desktop$
```

Example 1

```
1 #include <unistd.h>
2 #include <stdio.h>
3 int main()
4 {
5 printf("1\n");
6 execl("/bin/ls","ls",NULL);
7 }
```

Output :

```
fjwu@fjwu-virtual-machine :~/Desktop$ sudo g++ abc.cpp
fjwu@fjwu-virtual-machine :~/Desktop$ ./a.out
1
abc.cpp  a.out
fjwu@fjwu-virtual-machine :~/Desktop$
```

Example 2

```
#include<stdio.h>
void main()
{
printf("2/n");
}
```

Output

```
tanzeela@ubuntu:~/Desktop$ gcc -o exp1 exp1.c
tanzeela@ubuntu:~/Desktop$ ./exp1
2/ntanzeela@ubuntu:~/Desktop$
```

LAB TASKS

Task 1:

Write a program where a child is created to execute a command.

```
#include<unistd.h>
#include<stdio.h>
#include<sys/types.h>
int main()
{
pid_t pid;
pid=fork();
if(pid==0);
printf("\n I'm in the child process");
}
```

Output

```
tanzeela@ubuntu:~$ cd Desktop
tanzeela@ubuntu:~/Desktop$ gcc -o pid2 pid2.c
tanzeela@ubuntu:~/Desktop$ ./pid2

 I'm in the child processtanzeela@ubuntu:~/Desktop$
 I'm in the child process
```

Task 2:

Write a program that spawns two children, then wait for their completion, and behaves differently according to which one is finished.

```
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<stdlib.h>
#include<sys/types.h>
void main()
{
int pid_t,pid,pid2,status;
pid=fork();
{ if (pid==-1)
{
printf("error in fork \n");
exit(1);}
if (pid==0){
printf("No error : child: \n");}
else
{ wait(& status);
{ printf("Waiting for child 1 : \n");}}}
pid2==fork();
{ if (pid2==-1)
{ printf("error in the fork \n");
exit(1);}
if (pid2==0)
{ printf("\n no error :child: \n");}
else{
wait(& status);
{ printf ("Waiting for child 2 \n");}}}}
```

Output

```
tanzeela@ubuntu:~/Desktop$ gcc -o waiting waiting.c
tanzeela@ubuntu:~/Desktop$ ./waiting
No error : child:
Waiting for child 2
Waiting for child 2
Waiting for child 1 :
Waiting for child 2
Waiting for child 2
tanzeela@ubuntu:~/Desktop$
```

Lab 06 & 07

LAB TASKS

Task 1:

Create a file named “oslab.txt” present in directory named “os” located in home directory.

Shell Script :

```
#include<stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
void main()
{
  int fd;
  fd=creat("oslab.txt",S_IREAD |S_IWRITE);
  if(fd==-1)
  printf("Error in operating file \n");
  else
  {
    printf("oslab.txt is open for read/write access\n");
    printf("oslab.txt is currently empty\n");
  }
  close(fd);
}
```

Terminal :

```
tanzeela@ubuntu:~$ cd Desktop
tanzeela@ubuntu:~/Desktop$ mkdir os
tanzeela@ubuntu:~/Desktop$ ls
cplus.c  exp2        fork.c   pid     pid.c     waiting
exp1     fibser.cpp  lab4.sh  pid2    salary.c  waiting.c
exp1.c   fork        os       pid2.c  task1.c
tanzeela@ubuntu:~/Desktop$
```

```
tanzeela@ubuntu:~/Desktop$ ./task1
oslab.txt is open for read/write access
oslab.txt is currently empty
tanzeela@ubuntu:~/Desktop$
```

Task 2:

Open a file for reading named “oslab.txt” present in directory named “os” located in home directory.

Shell Script :

```c
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
void main()
{ int fd;
fd= open ("oslab.txt" , O_RDONLY);
if (fd==-1)
{
printf("\n Error in operating file");
printf("\n File does not Exists");
}
else
{
printf("\n oslab.txt open for read only");
printf("\n oslab.txt is currently empty");
}
close(fd);
}
```

Terminal :

```
tanzeela@ubuntu:~/Desktop$ gcc -o task2 task2.c
```

```
tanzeela@ubuntu:~/Desktop$ ./task2

 oslab.txt open for read only
 oslab.txt is currently emptytanzeela@ubuntu:~/Desktop$
```

Open a new file for reading and writing, whose location is provided by the user.

Shell Script :

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
void main()
{
char path[80];
printf("\n Enter the location for file creation :\n");
scanf("%s" , path);
int fd;
fd = open(path,O_RDWR | O_CREAT | O_EXCL, S_IREAD | S_IWRITE);
if (fd==-1)
{
printf("\n Error in operating file \n");}
if (fd!=-1)
{
printf("\n File opened for read/write acess\n");
}
close(fd);
}
```

Terminal :

```
tanzeela@ubuntu:~/Desktop$ gcc -o task2 task2.c
```

```
tanzeela@ubuntu:~/Desktop$ ./task2

 Enter the location for file creation :
/home/tanzeela/task2

 File opened for read/write acess
tanzeela@ubuntu:~/Desktop$ ^C
tanzeela@ubuntu:~/Desktop$
```

Task 3:

Read at most 100 bytes into buffer from standard input.

Shell Script :

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/types.h>
int main()
{
char buffer[100];
int fd;
fd = open("fileA.txt" , O_RDONLY);
if (fd==-1)
{
printf("\n fileA.txt opened for read acess\n");
lseek(fd,0l,0);
read(fd , buffer ,100);
printf("\n""%s(read into buffer from fileA.txt)\n", buffer);}
else
printf("\n Error : fileA.txt does not exist\n");
close(fd);
}
```

Terminal :

```
tanzeela@ubuntu:~/Desktop$ gcc -o task3 task3.c
```

```
tanzeela@ubuntu:~/Desktop$ ./task3

 fileA.txt opened for read acess

(read into buffer from fileA.txt)
tanzeela@ubuntu:~/Desktop$
```

Read bytes into a temporary buffer from file whose location and name is entered by the user as an argument at the time of execution.

Shell Script :

```c
#include<stdio.h>
#include<stdlib.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
int main()
{ int fd;
char buffer[100] , path[50];
printf("\n Enter path and name of the file\n");
scanf("%s" ,path);
fd=open(path ,O_RDONLY);
if (fd!=-1)
{
printf("\n file is opened for read acess\n");
lseek(fd,0l,0);
read(fd , buffer ,100);
printf("\n""%s(read into buffer from file)\n", buffer);
}
else
printf("\n Error : file does not exist\n");
close(fd);
}
```

Terminal :

```
tanzeela@ubuntu:~/Desktop$ gcc -o task3b task3b.c
```

```
tanzeela@ubuntu:~/Desktop$ ./task3b

 Enter path and name of the file
/home/tanzeela/file

 Error : file does not exist
tanzeela@ubuntu:~/Desktop$ ^C
tanzeela@ubuntu:~/Desktop$
```

Task 4:

Write "read" and "write" commands for the following conditions

- ***Read data from standard input and write it on standard output.***
- ***Read data from the file, given as an argument by the user and write it on standard output. Start reading from the end of the file.***
- ***Read data from standard input and write it on the file given as an argument by the user.***
- ***Read and write data from the files named "lab3.c" and "lab4.c" respectively. The files are present in the directory "lab", located in the home directory.***

Shell Script :

Open task4.cpp

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
int main()
 {
        int fd1, fd2, num;
        char buffer[100],path1[50],path2[50];
        printf("\n Enter the input file path \n");
        scanf("%s", path1);
        fd1-open (path1,O_RDONLY);
        if(fd1 != -1)
        {
        printf("\n Input file opened for read acces only");
        printf("\n Enter the output file path");
        scanf("%s", path2);
        fd2-open (path2,O_RDWR |O_CREAT|O_EXCL<S_IREAD | S_IWRITE);
        if(fd2 != -1)
        {
                printf("\n Output file opened for read/write access only \n");
                printf("\n Enter the maximum number of bytes to read from Input file:");
                scanf("%d",&num);
                lseek(fd1,0l,0);
                read(fd1,buffer, num);
                printf("\n %s\t(read fromInput file)\n",buffer);
                lseek(fd2,0l,0);
                read(fd2, buffer,num);
                printf("\n %s\t(was written to output file)\n" ,buffer);
        }
        else
        close(fd1);
        close(fd2);
        }
        else
        printf("\n Error: Input file does not exists \n");
}
```

Terminal :

```
ubuntu@ubuntu:~$ gcc -o task4 task4.cpp
ubuntu@ubuntu:~$ ./task4

 Enter the input file path
/home/ubuntu

 Input file opened for read acces only
 Enter the output file path
/home/ubuntu/output

 Output file opened for read/write access only

 Enter the maximum number of bytes to read from Input file:32



        (read fromInput file)



        (was written to output file)
ubuntu@ubuntu:~$
```

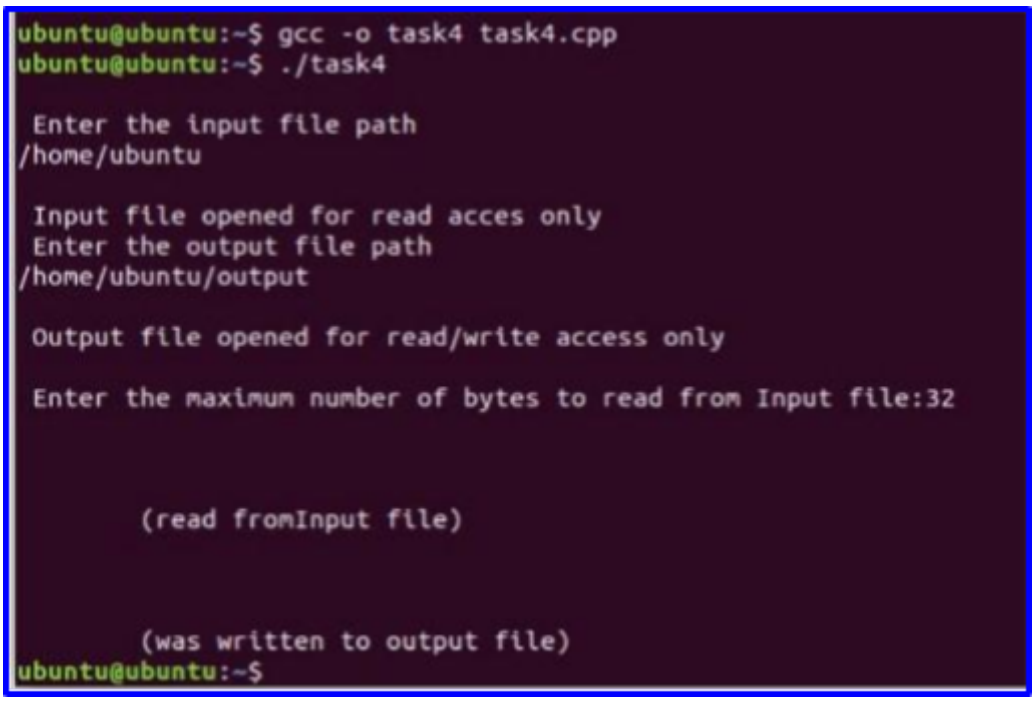
Task 5:

Write complete code to copy the contents of an input file to an output file using system calls where the input and output files names are entered by the user.

Shell Script:

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
int main()
 {
        int fd1, fd2, num;
        char buffer[100],path1[50],path2[50];
        printf("\n Enter the input file path \n");
        scanf("%s", path1);
        fd1-open (path1,O_RDONLY);
        if(fd1 != -1)
        {
        printf("\n Input file opened for read acces only");
        printf("\n Enter the output file path");
        scanf("%s", path2);
        fd2-open (path2,O_RDWR |O_CREAT|O_EXCL<S_IREAD | S_IWRITE);
        if(fd2 != -1)
        {
                printf("\n Output file opened for read/write access only \n");
                printf("\n Enter the maximum number of bytes to read from Input file:");
                scanf("%d",&num);
                lseek(fd1,0l,0);
                read(fd1,buffer, num);
                printf("\n %s\t(read fromInput file)\n",buffer);
                lseek(fd2,0l,0);
                read(fd2, buffer,num);
                printf("\n %s\t(was written to output file)\n" ,buffer);
        }
        else
        close(fd1);
        close(fd2);
        }
        else
        printf("\n Error: Input file does not exists \n");
}
```

Terminal :



Task 6:

Unlink system call:

Unlink() deletes a name from the filesystem. If that name was the last link to a file and no processes have the file open the file is deleted and the space it was using is made available for reuse. If the name was the last link to a file but any processes still have the file open the file will remain in existence until the last file descriptor referring to it is closed. If the name refers to a symbolic link the link is removed. If the name refers to a socket, fifo or device the name for it is removed but processes which have the object open may continue to use it. link system call: link() creates a new link (also known as a hard link) to an existing file. If a newpath exists it will not be overwritten. This new name may be used exactly as the old one for any operation; both names refer to the same file (and so have the same permissions and ownership) and it is impossible to tell which name was the `original'.

Lab 08

Inter Process Communication

LAB TASKS

Task 1:

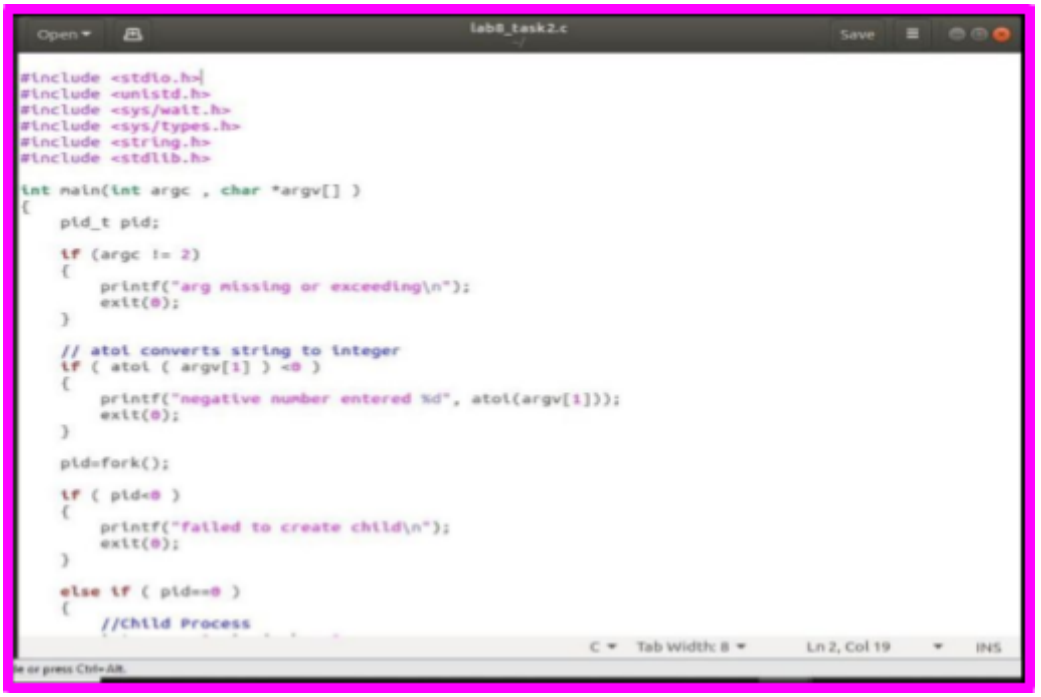
Compute the Factorial of a number using IPC (PIPE implementation).

- ***Parent creates pipe***
- ***Forks a child***
- ***Parent writes into pipe (the number whose factorial is to be calculated, take the number from the user)***
- ***Child reads from pipe and compute the Factorial of a number written by Parent***

Answer :

Shell

Script :



```
        for (i=0; i<n; i++)
        {
            if (sum[i] > 0)
                printf(" %d ", sum[i]);
        }
        exit(0);
    }

    // parent process
    else
    {
        wait(NULL);

        // waiting for child process to end
        printf("Done\n");
    }
}
```

Terminal :

```
File  Edit  View  Search  Terminal  Help
tanzeela@ubuntu:~$ gcc lab8_task2.c -o lab8_task2
tanzeela@ubuntu:~$ ./lab8_task2 4
 1
 1  2
 1  2  6
 1  2  6  24
After deletion sum
 1  3  9  33 Done
tanzeela@ubuntu:~$
```

Task 2:

Using pipes, parent read data from one file, and child write data into another file.

Shell

Script:

```c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main(){

    int fd[2],n;
    char buffer[100] ;
    pid_t p;
    pipe(fd);
    p=fork() ;
   if(p>0 ){

            printf("Parent passing value to child\n");
            write(fd[1],"hello\n",6);
        }

else{

  printf("child printing recieved value\n") ;
  n=read(fd[0],buffer,100);
  write(1,buffer,n);

}
}
```

Terminal :

```
File  Edit  View  Search  Terminal  Help
tanzeela@ubuntu:~$ gcc ipc.c -o ipc
tanzeela@ubuntu:~$ ./ipc
Parent passing value to child
child printing recieved value
hello
tanzeela@ubuntu:~$
```

Lab 09

Memory Management Schemes-First Fit Best Fit

LAB TASKS

Example code link (https://www.geeksforgeeks.org/program-first-fit-algorithm-memory-management/)

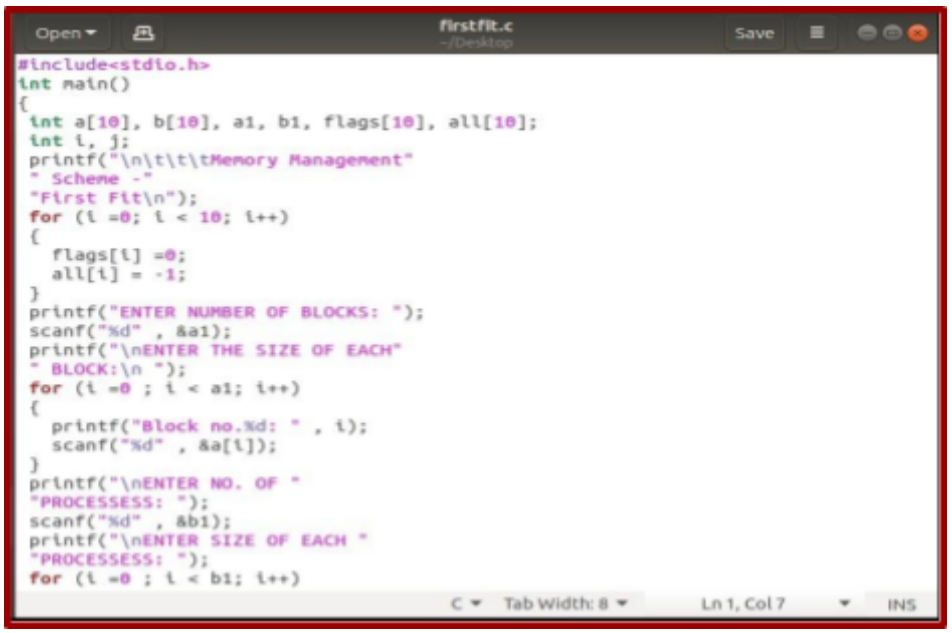
Question

Write a program for first fit and best fit algorithm for memory management.

Solution:

FOR FIRST FIT:

Shell Script:



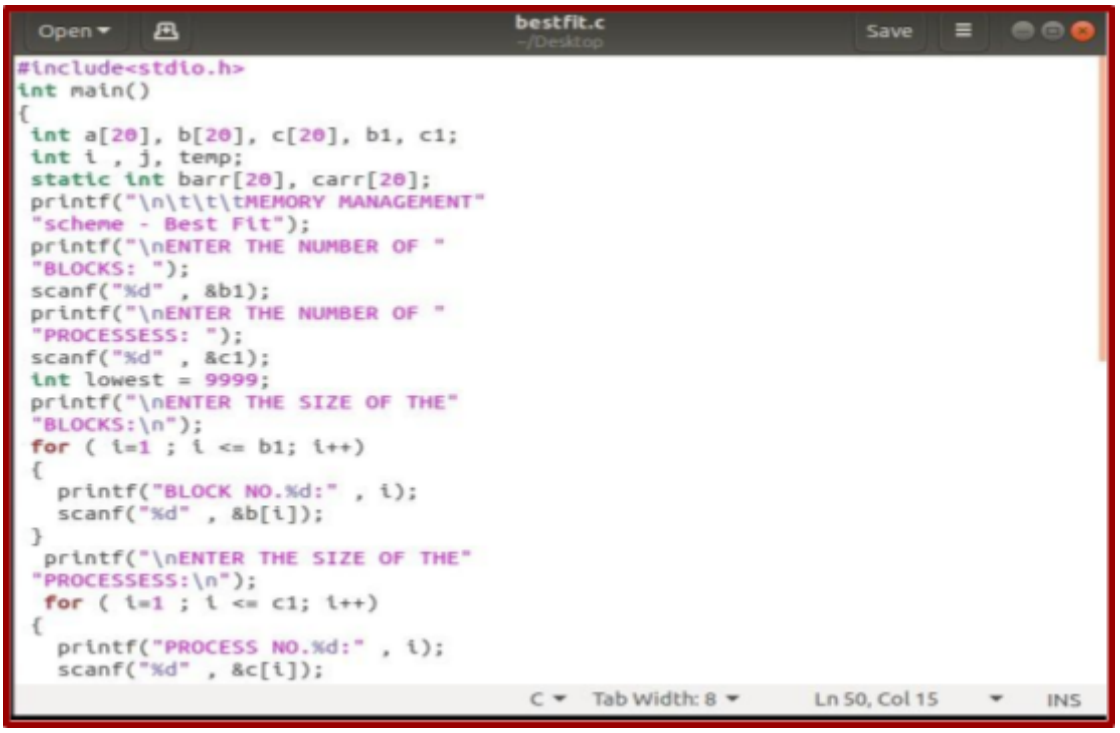
```
Open ▾   firstfit.c   ~/Desktop   Save
{
  printf("process no.%d: " , i);
  scanf("%d" , &b[i]);
}
for (i =0 ; i < b1; i++)
for (j =0 ; j < b1; j++)
if (flags[j] == 0 && a[j] >= b[i])
{
  all[j] = i;
  flags[j] = 1;
  break;
}
printf("\nBlock no.\tsize\t\t"
"process no.\t\tsize");
for (i =0 ; i < a1; i++)
{
  printf("\n%d\t\t%d\t\t" ,
  i + 1 , a[i]);
 if (flags[i] == 1)
{
  printf("%d\t\t\t%d\t\t" , all[i]
  + 1 , b[all[i]]);
}
else
printf("NOT ALLOCATED" );
}
printf("\n");
}
C ▾   Tab Width: 8 ▾   Ln 1, Col 7   INS
```

Output:

```
File  Edit  View  Search  Terminal  Tabs  Help
tanzeela@ubuntu: ~   tanzeela@ubuntu: ~/Desk...   tanzeela@ubuntu: ~/Desk...
tanzeela@ubuntu:~/Desktop$ pwd
/home/tanzeela/Desktop
tanzeela@ubuntu:~/Desktop$ gcc firstfit.c -o firstfit
tanzeela@ubuntu:~/Desktop$ ./firstfit

                        Memory Management Scheme -First Fit
ENTER NUMBER OF BLOCKS: 4

ENTER THE SIZE OF EACH BLOCK:
 Block no.0: 3
Block no.1: 5
Block no.2: 9
Block no.3: 6

ENTER NO. OF PROCESSESS: 4

ENTER SIZE OF EACH PROCESSESS: process no.0: 5
process no.1: 2
process no.2: 9
process no.3: 6

Block no.       size            process no.             size
1               3               2                       2
2               5               1                       5
3               9               3                       9
4               6               4                       6
tanzeela@ubuntu:~/Desktop$
```

FOR BEST FIT:

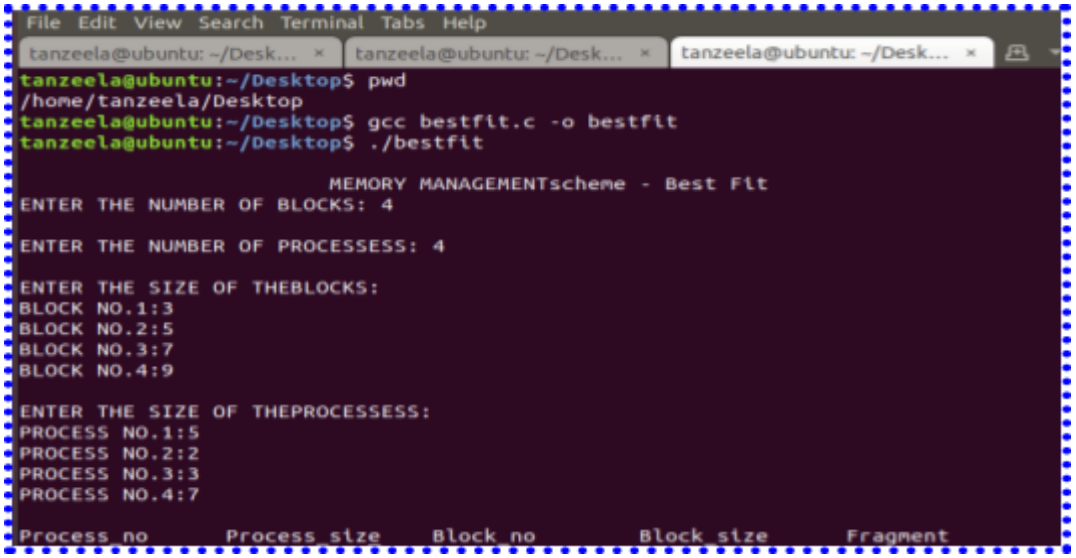
Shell Script

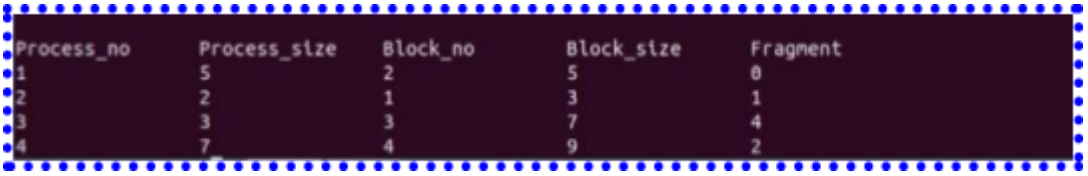


```
#include<stdio.h>
int main()
{
 int a[20], b[20], c[20], b1, c1;
 int i , j, temp;
 static int barr[20], carr[20];
 printf("\n\t\t\tMEMORY MANAGEMENT"
 "scheme - Best Fit");
 printf("\nENTER THE NUMBER OF "
 "BLOCKS: ");
 scanf("%d" , &b1);
 printf("\nENTER THE NUMBER OF "
 "PROCESSESS: ");
 scanf("%d" , &c1);
 int lowest = 9999;
 printf("\nENTER THE SIZE OF THE"
 "BLOCKS:\n");
 for ( i=1 ; i <= b1; i++)
 {
   printf("BLOCK NO.%d:" , i);
   scanf("%d" , &b[i]);
 }
  printf("\nENTER THE SIZE OF THE"
 "PROCESSESS:\n");
  for ( i=1 ; i <= c1; i++)
 {
   printf("PROCESS NO.%d:" , i);
   scanf("%d" , &c[i]);
```

```
  for ( j=1 ; j <= b1; j++)
  {
    if (barr[j] !=1)
      {
         temp = b[j] - c[i];
         if (temp >= 0)
         if (lowest > temp)
           {
             carr[i] = j;
             lowest = temp;
           }
      }
    }
 a[i] = lowest;
 barr[carr[i]] = 1;
 lowest = 10000;
 }
 printf("\nProcess_no\tProcess"
 "_size\tBlock_no\t"
 "Block_size\tFragment");
 for (i = 1 ; i <= c1 && carr[i] != 0 ; i++)
 {
   printf("\n%d\t\t%d\t\t%d\t\t"
   "%d\t\t%d" , i ,
   c[i], carr[i] , b[carr[i]], a[i]);
 }
printf("\n");
```

Output:


File Edit View Search Terminal Tabs Help
tanzeela@ubuntu: ~/Desk...
tanzeela@ubuntu: ~/Desk...
tanzeela@ubuntu: ~/Desk...
tanzeela@ubuntu:~/Desktop$ pwd
/home/tanzeela/Desktop
tanzeela@ubuntu:~/Desktop$ gcc bestfit.c -o bestfit
tanzeela@ubuntu:~/Desktop$ ./bestfit
MEMORY MANAGEMENTscheme - Best Fit
ENTER THE NUMBER OF BLOCKS: 4
ENTER THE NUMBER OF PROCESSESS: 4
ENTER THE SIZE OF THEBLOCKS:
BLOCK NO.1:3
BLOCK NO.2:5
BLOCK NO.3:7
BLOCK NO.4:9
ENTER THE SIZE OF THEPROCESSESS:
PROCESS NO.1:5
PROCESS NO.2:2
PROCESS NO.3:3
PROCESS NO.4:7
Process no Process size Block no Block size Fragment


Process_no Process_size Block_no Block_size Fragment
1 5 2 5 0
2 2 1 3 1
3 3 3 7 4
4 7 4 9 2

LAB#10

Example

Write a program to create a thread.

Program:

```
Open ▾   thread.c   Save
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
void *thread_function(void *arg);
 int i,j;
 int main() {
 pthread_t a_thread;  //thread declaration

 pthread_create(&a_thread, NULL, thread_function, NULL);
//thread is created
 pthread_join(a_thread, NULL);
 printf("Inside Main Program\n");
 for(j=20;j<25;j++)
 {
 printf("%d\n",j);
 sleep(1);
 }
 }
void *thread_function(void *arg) {
// the work to be done by the thread is defined in this function
 printf("Inside Thread\n");
 for(i=0;i<5;i++)
 {
 printf("%d\n",i);
 sleep(1);
 }
 }
```

OUTPUT

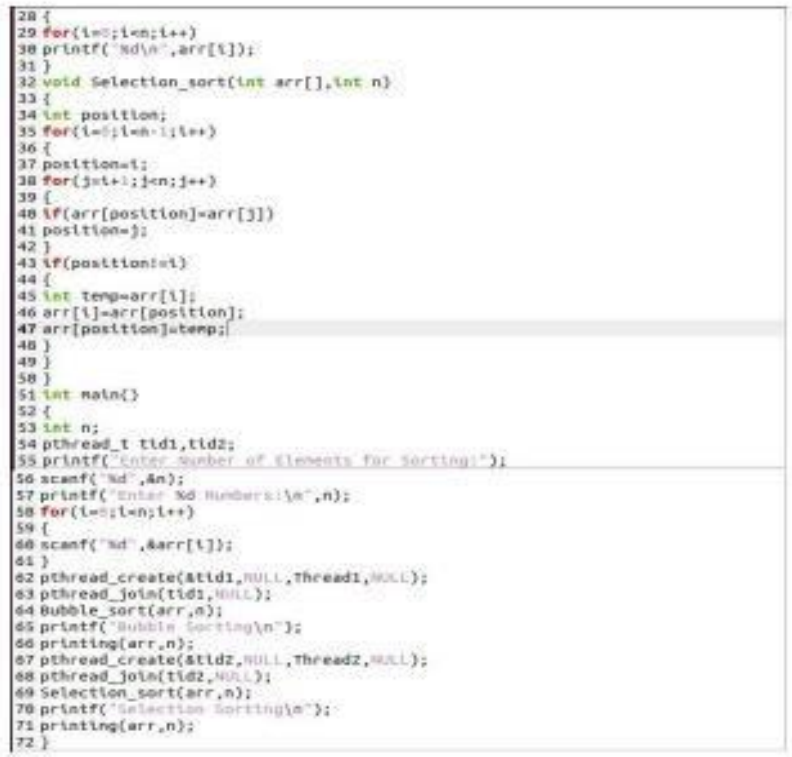
```
File  Edit  View  Search  Terminal  Help
codebind@codebind:~$ gcc thread.c -lpthread
codebind@codebind:~$ ./a.out
Inside Thread
0
1
2
3
4
Inside Main Program
20
21
22
23
24
codebind@codebind:~$
```

Task#1

Write a program that sorts the array in ascending order using bubble sort and descendingorder using selection sort by means of multithreading.

Program:

```
Open   lab10c   Save
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<unistd.h>
int i,j,arr[50];
void *Thread1(void* ptr)
{
printf("Thread # 1\n");
}
void *Thread2(void* ptr)
{
printf("Thread # 2\n");
}
void Bubble_sort(int arr[],int n)
{
for(i=0;i<n-1;i++)
for(j=0;j<n-i-1;j++)
{
if(arr[j]>arr[j+1])
{
int temp =arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}
void printing(int arr[],int n)
```



```
{
for(i=0;i<n;i++)
printf("%d\n",arr[i]);
}
void Selection_sort(int arr[],int n)
{
int position;
for(i=0;i<n-1;i++)
{
position=i;
for(j=i+1;j<n;j++)
{
if(arr[position]>arr[j])
position=j;
}
if(position!=i)
{
int temp=arr[i];
arr[i]=arr[position];
arr[position]=temp;
}
}
}
int main()
{
int n;
pthread_t tid1,tid2;
printf("Enter Number of Elements for Sorting:");
scanf("%d",&n);
printf("Enter %d Numbers:\n",n);
for(i=0;i<n;i++)
{
scanf("%d",&arr[i]);
}
pthread_create(&tid1,NULL,Thread1,NULL);
pthread_join(tid1,NULL);
Bubble_sort(arr,n);
printf("Bubble Sorting\n");
printing(arr,n);
pthread_create(&tid2,NULL,Thread2,NULL);
pthread_join(tid2,NULL);
Selection_sort(arr,n);
printf("Selection Sorting\n");
printing(arr,n);
}
```

OUTPUT

```
Enter Number of Elements for Sorting:5
Enter 5 Numbers:
30
17
15
40
2
Thread # 1
Bubble Sorting
2
15
17
30
40
Thread # 2
Selection Sorting
40
30
17
15
2
```

Lab 11

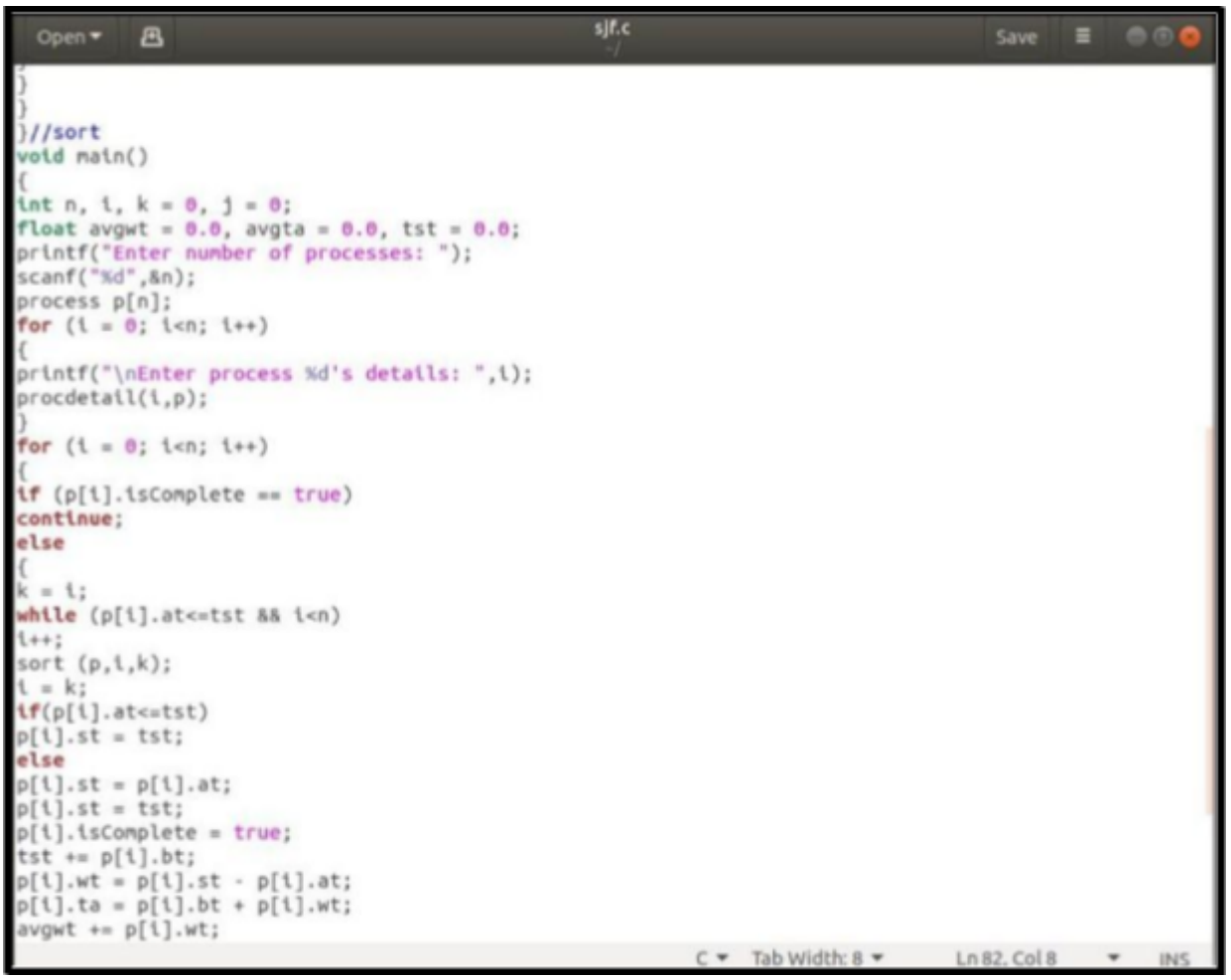
Shortest Job First

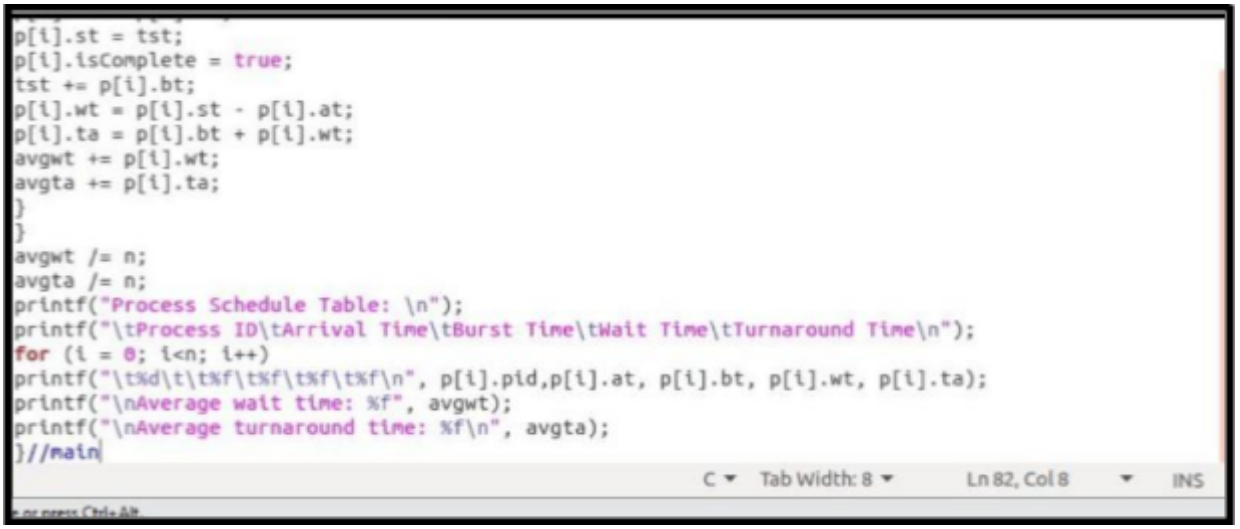
LAB TASK

Question:

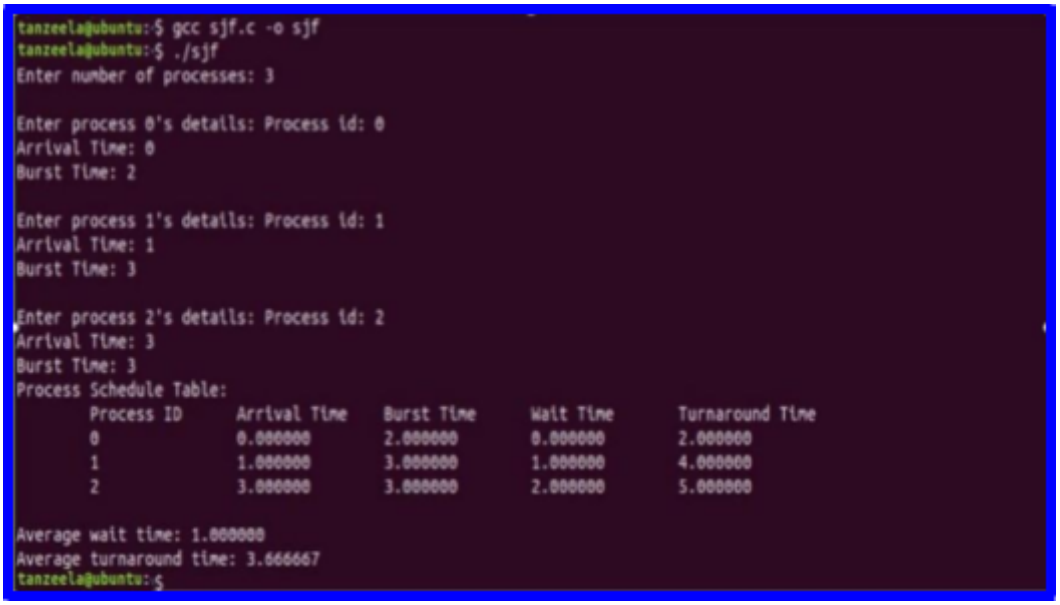
★ **Implement Shortest Job First (Non-Preemptive) CPU Scheduling Algorithm.**

```
#include<stdio.h>
#include<stdbool.h>
typedef struct
{
int pid;
float at, wt, bt, ta, st;
bool isComplete;
}process;
void procdetail(int i, process p[])
{
printf("Process Id: ");
scanf("%d", &p[i].pid);
printf("Arrival Time: ");
scanf("%f", &p[i].at);
printf("Burst Time: ");
scanf("%f", &p[i].bt);
p[i].isComplete = false;
}//procdetail
void sort(process p[], int i, int start)
{
int k = 0, j;
process temp;
for (k = start; k<i; k++)
{
for (j = k+1; j<i; j++)
{
if(p[k].bt > p[j].bt)
continue;
else
{
temp = p[k];
p[k] = p[j];
p[j] = temp;
}
}
}
```


Open
sjf.c
Save
}
}
}//sort
void main()
{
int n, i, k = 0, j = 0;
float avgwt = 0.0, avgta = 0.0, tst = 0.0;
printf("Enter number of processes: ");
scanf("%d",&n);
process p[n];
for (i = 0; i<n; i++)
{
printf("\nEnter process %d's details: ",i);
procdetail(i,p);
}
for (i = 0; i<n; i++)
{
if (p[i].isComplete == true)
continue;
else
{
k = i;
while (p[i].at<=tst && i<n)
i++;
sort (p,i,k);
i = k;
if(p[i].at<=tst)
p[i].st = tst;
else
p[i].st = p[i].at;
p[i].st = tst;
p[i].isComplete = true;
tst += p[i].bt;
p[i].wt = p[i].st - p[i].at;
p[i].ta = p[i].bt + p[i].wt;
avgwt += p[i].wt;
C
Tab Width: 8
Ln 82, Col 8
INS


p[i].st = tst;
p[i].isComplete = true;
tst += p[i].bt;
p[i].wt = p[i].st - p[i].at;
p[i].ta = p[i].bt + p[i].wt;
avgwt += p[i].wt;
avgta += p[i].ta;
}
}
avgwt /= n;
avgta /= n;
printf("Process Schedule Table: \n");
printf("\tProcess ID\tArrival Time\tBurst Time\tWait Time\tTurnaround Time\n");
for (i = 0; i<n; i++)
printf("\t%d\t\t%f\t%f\t%f\t%f\n", p[i].pid,p[i].at, p[i].bt, p[i].wt, p[i].ta);
printf("\nAverage wait time: %f", avgwt);
printf("\nAverage turnaround time: %f\n", avgta);
}//main
C
Tab Width: 8
Ln 82, Col 8
INS

Output :



Lab 12

Round-Robin CPU Scheduling Algorithm

LAB TASK

Question:

Implement Round Robin CPU Scheduling Algorithm.

Shell Script:

```
    {
        temp[i] = temp[i] - quant;
        sum = sum + quant;
    }
    if(temp[i]==0 && count==1)
    {
        y--; //decrement the process no.
        printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-
bt[i]);
        wt = wt+sum-at[i]-bt[i];
        tat = tat+sum-at[i];
        count =0;
    }
    if(i==NOP-1)
    {
        i=0;
    }
    else if(at[i+1]<=sum)
    {
        i++;
    }
    else
    {
        i=0;
    }
}
// represents the average waiting time and Turn Around time
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
printf("\n");

}
```

C ▾ Tab Width: 8 ▾ Ln 1, Col 21 ▾ INS

Output:

```
tanzeela@ubuntu:~$ gcc roundrobin.c -o roundrobin
tanzeela@ubuntu:~$ ./roundrobin
 Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:        1

Burst time is:  2

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:        2

Burst time is:  2

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:        3

Burst time is:  4

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:        5

Burst time is:  3
Enter the Tine Quantun for the process:         2

 Process No              Burst Time              TAT             Waiting Time
Process No[1]           2                               1                       -1
Process No[2]           2                               2                       0
Process No[3]           4                               7                       3
Process No[4]           3                               6                       3
 Average Turn Around Time:      1.250000
 Average Waiting Time:  4.000000
tanzeela@ubuntu:~$
```