Experiment No. 7

Exp No.7.a: Explore the internal commands of Linux ; Commands for creating files

Theory: Linux is an Operating System's Kernel. It was created by Linus Torvalds from Scratch. Linux is free and open-source, that means that you can simply change anything in Linux and redistribute it in your own name! There are several Linux Distributions, commonly called "distros". A few of them are:

Ubuntu Linux	Red Hat Enterprise Linux
Linux Mint	Debian
Fedora	

Linux Shell or "Terminal"

Shell is a program that receives commands from the user and gives it to the OS to process and it shows the output. Linux's shell is its main part. Its distros come in GUI (Graphical User Interface) but basically, Linux has a CLI (Command Line Interface)

Linux Commands

Basic Commands

Theory: **cat command:** This command is useful for creating, displaying, appending information and concatenating two or more files.

```
Syntax:- $ cat > filename  --- Creating file
             $ cat filename --- Displaying the contents of file
             $ cat >> filename --  Appending information to file
             $ cat file1 file2  >  file3 ----- Copy
```

a) **touch command:** The touch command is used to create an empty file.

```
Syntax:- touch filename
Example:- touch Sunday
```

The above command creates the new file "Sunday" and assigns your username as file owner.

The touch command can also be used to change the access and modification times on an existing file without changing the file contents. If you want to change only the access time, use the -a parameter. To change only the modification time, use the –m parameter. By default touch uses the current time. You can specify the time by using the –t parameter with a specific timestamp:

```
Syntax:- $ touch -t 200812251200 test1
Output:-
$ ls -l test1
 -rw-r--r-- 1 rich rich 0 Dec 25 2008 test1
```

b) **Commands for performing operations on files**

mv command: The mv command renames/moves files. It has two distinct functions; (i) It renames a file or directory or (ii) It moves a group of files to different directory.

Example:- $ mv test1 test5

The above command renames file from test1 to test5

Example:- $ mv linux Ty_linux

The above command renames a directory from linux to Ty_linux

Example:- $ mv file? Ty_linux

The above command moves all files starting from word "file" from current directory to Ty_linux directory.

Note:- '?' and '*' are considered as wild card characters. '?' matches a single character whereas "*" matches zero or more characters

rm command: The rm command deletes one or more files. It normally operates silently and hence should be used with caution

Syntax:- $ rm filename

Example:- $ rm file1

The above command removes/deletes the file from the system without any confirmation.

For removing files that are placed in another directory can be done without using "cd command".

cp command: The destination parameter for cp command can be directory also. If the destination is a directory then cp command will copy the file from the existing folder to the destination folder (i.e. Copy and paste operation)

This command allows copying of files and directories from one location to another.

Syntax:- $ cp source_file destintion_file

When both the source and destination parameters are filenames, the cp command copies the source file to a new file with the filename specified as the destination.

Example: $ cp file1 dest_file

In the above example, the destination file "dest_file" does not exist hence it will be first created then copying will be done. This new file created has a different inode number indicating that it's a completely new file.

If the destination file already exists, then cp command overwrites the destination file without giving any warning.

Example: $ cp file1 file2

In the above example, the destination file "file2" already exists hence; it will be overwritten without any warning from the system.

c) **Listing files and directories**

ls - The most basic feature of the shell is the ability to see what files are available on the system. The list command (ls) is the tool that helps do that. It displays the files and directories located in your current directory. It produces listing in alphabetical order.

–l	It produces long listing of files and folders
–n	Its output is same as "-l option" except username is replaced by UID and group name is replaced by GID.
–R	It will list sub-directory components recursively.
–i	It displays inode number of each file and folder
–r	It displays the output in reverse order
–F	This option is used to easily distinguish files from directories. It flags the directories with a forward slash, to help identify them in the listing. Similarly, it flags executable files (like the myprog file below) with an asterisk, to help you find the files that can be run on the system easier.

OPTIONS of *'ls'* command

cd - This command is used to change directory. It allows us to move from one directory to another**.**

Syntax: cd destination_folder_name

mkdir - This command is used to create a directory. It is followed by the name of the directory to be created.

Syntax: $ mkdir dir_name

Example: $ mkdir OS

The above command creates a directory named “OS” under your current working directory.

rmdir- this command is used to remove directories

Syntax: $ rmdir dir_name

Example: $ rmdir dir1

chmod - stands for "change mode", and it is used to define the way a file can be accessed.

Syntax: chmod who +/- /= permissions filename

Who – user, group, others

Permissions: r/w/x

Permissions defines the permissions for the owner of the file (the "**user**"), members of the group who owns the file (the "**group**"), and anyone else ("**others**"). There are two ways to represent these permissions: with symbols (alphanumeric characters), or with octal numbers (the digits 0 through 7; r =4; w=2; x=1).

Syntax: chmod ugo+x filename (to assign execute permission to user, group and others)

Chmod ugo – x filename (to take away execute permission from user, group and others)

ps - The ps command produces a list of the currently running processes on your computer.

To invoke ps simply type the following: ps

The output will show rows of data containing the following information:

PID : The PID is the process ID which identifies the running process.

TTY : The TTY is the terminal type.

Time and Command

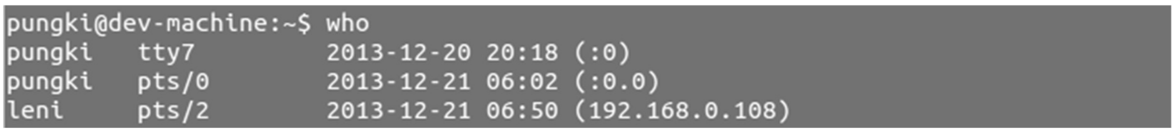
To view all the running processes use either of the following commands: ps -A , ps -e

d) **Misc commands**

who command: Linux maintains an account of all users who are currently logged in to the system.

Syntax:- $ who

Output:

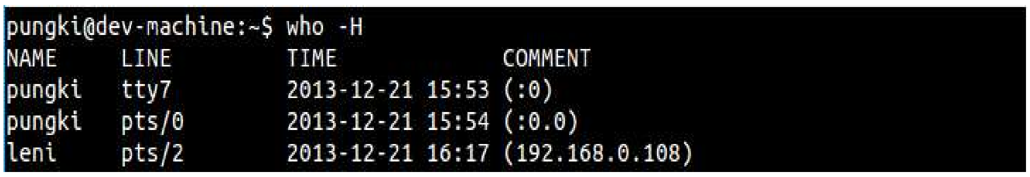


```
pungki@dev-machine:~$ who
pungki   tty7         2013-12-20 20:18 (:0)
pungki   pts/0        2013-12-21 06:02 (:0.0)
leni     pts/2        2013-12-21 06:50 (192.168.0.108)
```

- 1st column show the user name
- 2nd column show how the user connected. Tty means the user is connected directly to the computer, while pts means the user is connected from remote
- 3rd and 4th columns show the date and time
- 5th column show the IP Address where the users are connected

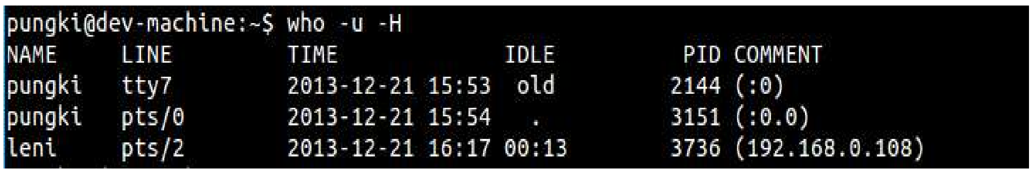
OPTIONS

-H:- Prints the column headings



```
pungki@dev-machine:~$ who -H
NAME     LINE         TIME             COMMENT
pungki   tty7         2013-12-21 15:53 (:0)
pungki   pts/0        2013-12-21 15:54 (:0.0)
leni     pts/2        2013-12-21 16:17 (192.168.0.108)
```

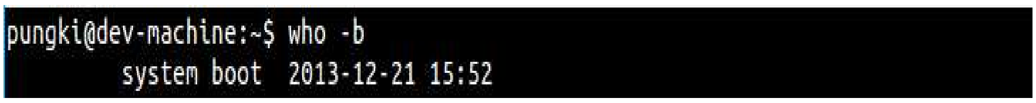
-u:- Gives the detailed output



```
pungki@dev-machine:~$ who -u -H
NAME     LINE         TIME             IDLE          PID COMMENT
pungki   tty7         2013-12-21 15:53  old         2144 (:0)
pungki   pts/0        2013-12-21 15:54   .          3151 (:0.0)
leni     pts/2        2013-12-21 16:17 00:13        3736 (192.168.0.108)
```

The idle time contains the number of hours and minutes since last activity occurred. So 00:13 means that user leni has been idle for 13 minutes. The dot (.) sign tell us that the terminal has seen activity on the last minute. During that time, we can call it “current”. The PID is a process ID of the user’s shell.

-b:- Indicate the time and date of the last reboot



```
pungki@dev-machine:~$ who -b
         system boot  2013-12-21 15:52
```

e) **who am i:** Displays the details of the user who invoked the command.

f) **cal command:** The cal command is used to see the calendar of any specific month or a complete year (from 1 – 9999). This facility is total accurate and takes into account the leap year adjustments that took place in the year 1752.

Syntax: $ cal

OPTIONS

-1	Display a single month. This is the default.
-3	Display three months: last month, this month, and next month.
-s	Display the calendar using Sunday as the first day of the week.
-m	Display Monday as the first day of the week.
-j	Display dates of the Julian calendar.
-y	Display a calendar for the entire current year.

$ cal yyyy = Displays calendar for the specified year.

g) **date command:** The Linux system maintains an internal clock meant to run perpetually. When the system is shut down a battery backup keeps the clock ticking.

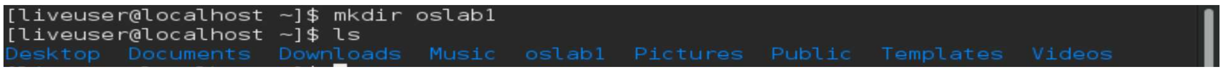
Syntax: $ date

Output: Sat Nov 7 22:44:59 IST 2009

$ date +”%m” current month name

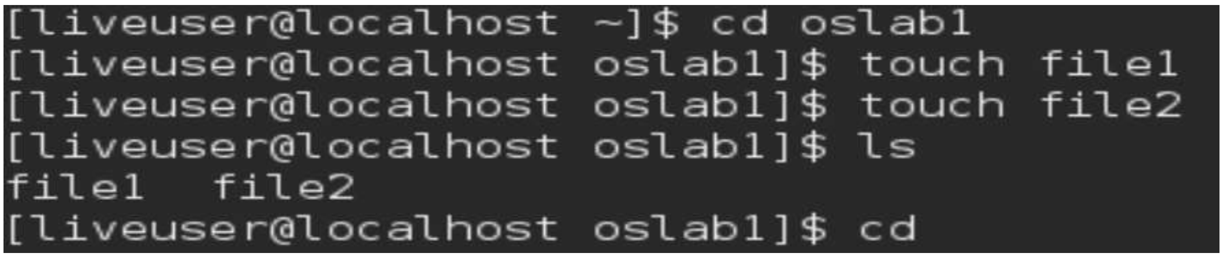
OUTPUTS

mkdir:



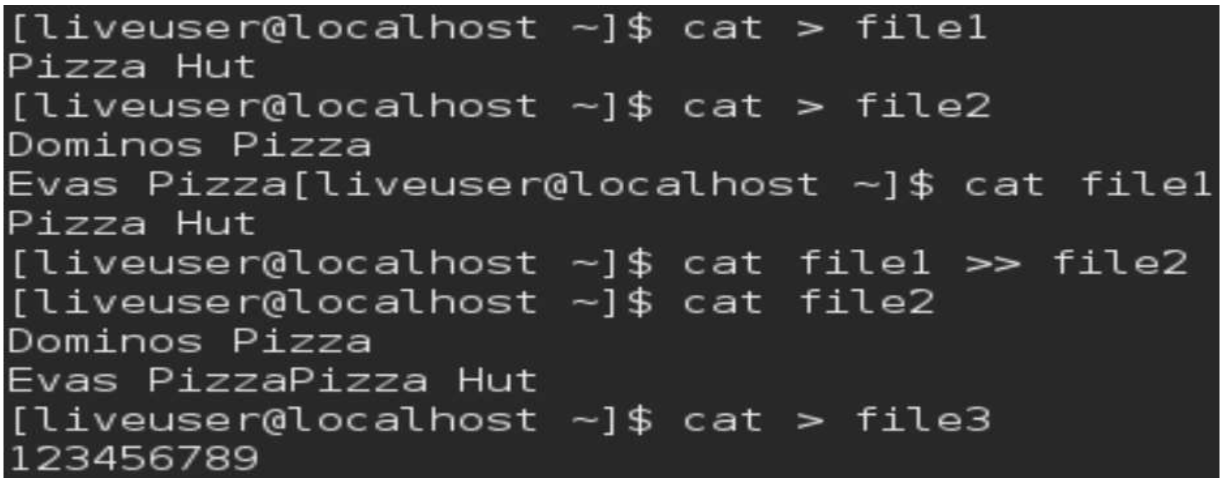
```
[liveuser@localhost ~]$ mkdir oslab1
[liveuser@localhost ~]$ ls
Desktop  Documents  Downloads  Music  oslab1  Pictures  Public  Templates  Videos
```

touch:



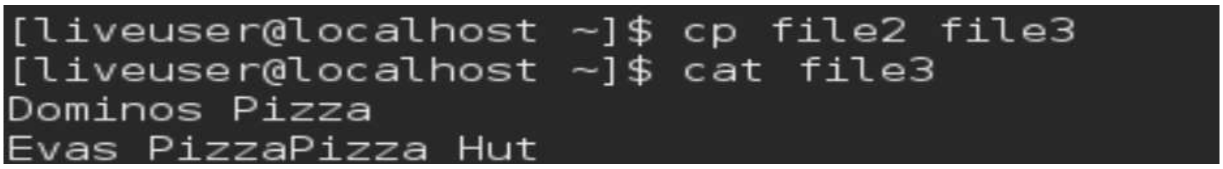
```
[liveuser@localhost ~]$ cd oslab1
[liveuser@localhost oslab1]$ touch file1
[liveuser@localhost oslab1]$ touch file2
[liveuser@localhost oslab1]$ ls
file1  file2
[liveuser@localhost oslab1]$ cd
```

cat:



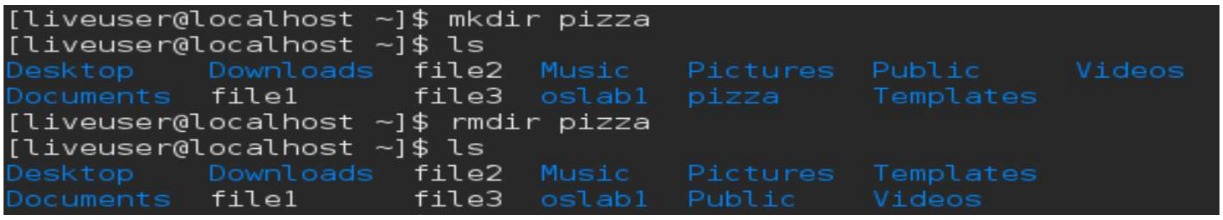
```
[liveuser@localhost ~]$ cat > file1
Pizza Hut
[liveuser@localhost ~]$ cat > file2
Dominos Pizza
Evas Pizza[liveuser@localhost ~]$ cat file1
Pizza Hut
[liveuser@localhost ~]$ cat file1 >> file2
[liveuser@localhost ~]$ cat file2
Dominos Pizza
Evas PizzaPizza Hut
[liveuser@localhost ~]$ cat > file3
123456789
```

cp:



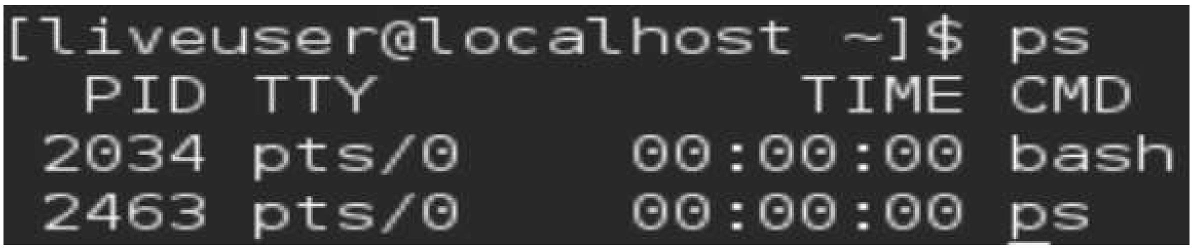
```
[liveuser@localhost ~]$ cp file2 file3
[liveuser@localhost ~]$ cat file3
Dominos Pizza
Evas PizzaPizza Hut
```

rmdir:



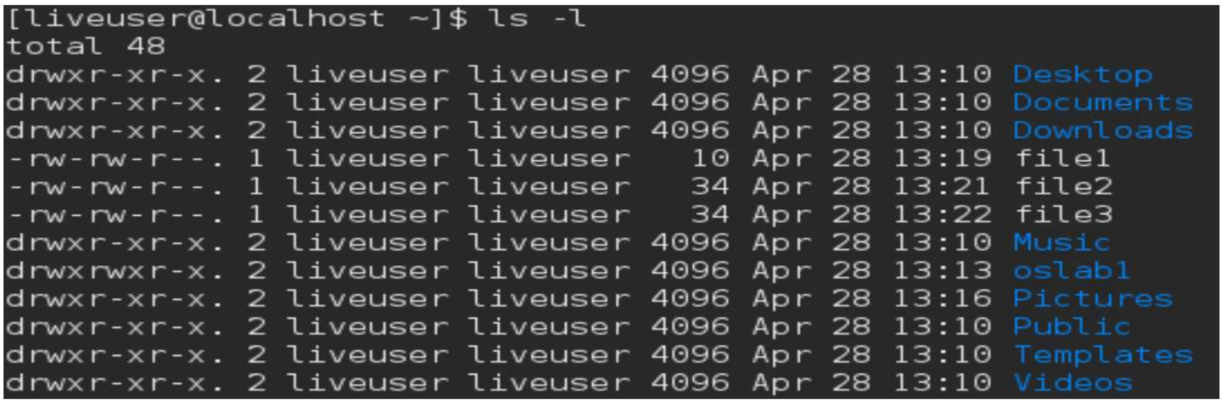
```
[liveuser@localhost ~]$ mkdir pizza
[liveuser@localhost ~]$ ls
Desktop    Downloads  file2  Music    Pictures  Public     Videos
Documents  file1      file3  oslab1   pizza     Templates
[liveuser@localhost ~]$ rmdir pizza
[liveuser@localhost ~]$ ls
Desktop    Downloads  file2  Music    Pictures  Templates
Documents  file1      file3  oslab1   Public    Videos
```

ps-



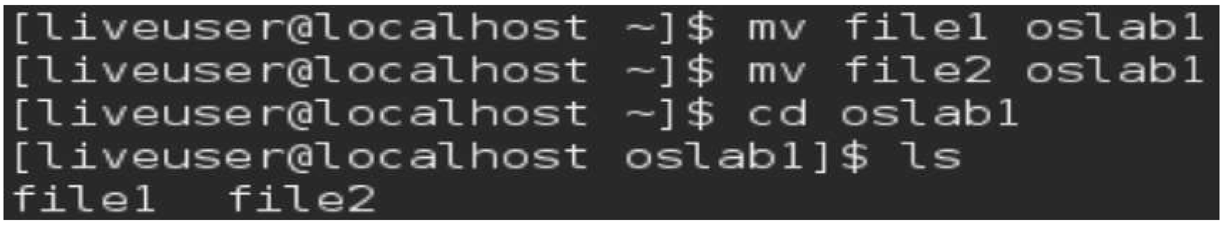
```
[liveuser@localhost ~]$ ps
  PID TTY          TIME CMD
 2034 pts/0    00:00:00 bash
 2463 pts/0    00:00:00 ps
```

ls -l



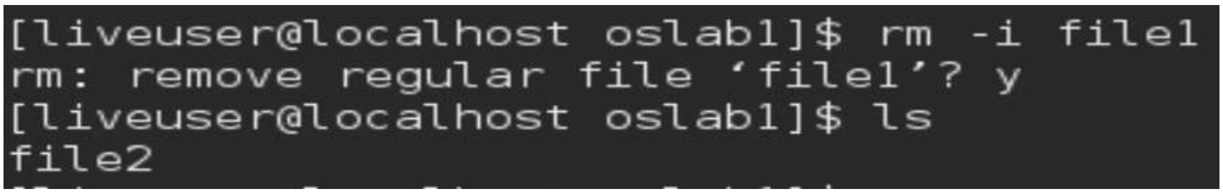
```
[liveuser@localhost ~]$ ls -l
total 48
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:10 Desktop
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:10 Documents
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:10 Downloads
-rw-rw-r--. 1 liveuser liveuser   10 Apr 28 13:19 file1
-rw-rw-r--. 1 liveuser liveuser   34 Apr 28 13:21 file2
-rw-rw-r--. 1 liveuser liveuser   34 Apr 28 13:22 file3
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:10 Music
drwxrwxr-x. 2 liveuser liveuser 4096 Apr 28 13:13 oslab1
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:16 Pictures
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:10 Public
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:10 Templates
drwxr-xr-x. 2 liveuser liveuser 4096 Apr 28 13:10 Videos
```

mv:



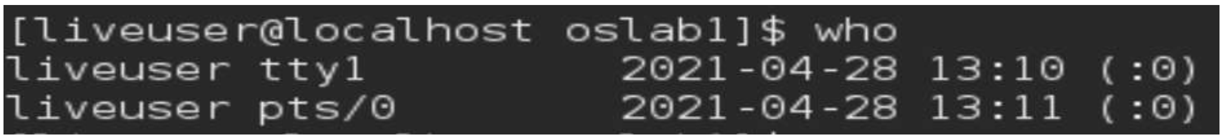
```
[liveuser@localhost ~]$ mv file1 oslab1
[liveuser@localhost ~]$ mv file2 oslab1
[liveuser@localhost ~]$ cd oslab1
[liveuser@localhost oslab1]$ ls
file1  file2
```

rm:



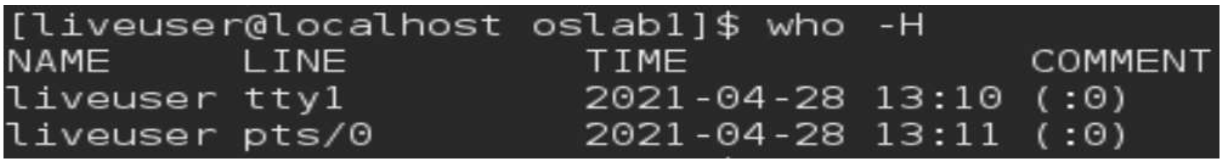
```
[liveuser@localhost oslab1]$ rm -i file1
rm: remove regular file ‘file1’? y
[liveuser@localhost oslab1]$ ls
file2
```

who:



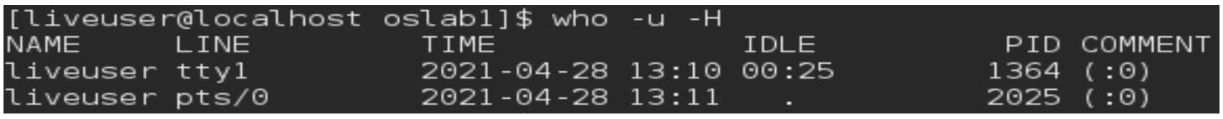
```
[liveuser@localhost oslab1]$ who
liveuser tty1         2021-04-28 13:10 (:0)
liveuser pts/0        2021-04-28 13:11 (:0)
```

who -H



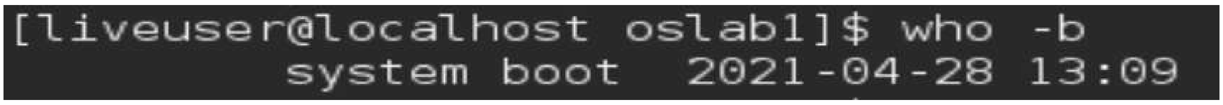
```
[liveuser@localhost oslab1]$ who -H
NAME     LINE         TIME             COMMENT
liveuser tty1         2021-04-28 13:10 (:0)
liveuser pts/0        2021-04-28 13:11 (:0)
```

who -u -H



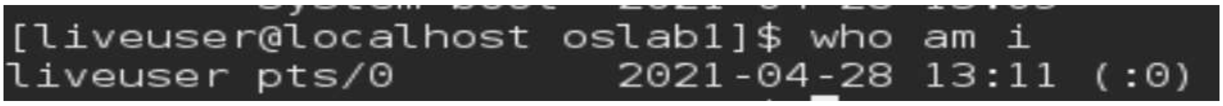
```
[liveuser@localhost oslab1]$ who -u -H
NAME     LINE         TIME             IDLE          PID COMMENT
liveuser tty1         2021-04-28 13:10 00:25        1364 (:0)
liveuser pts/0        2021-04-28 13:11   .          2025 (:0)
```

who -b



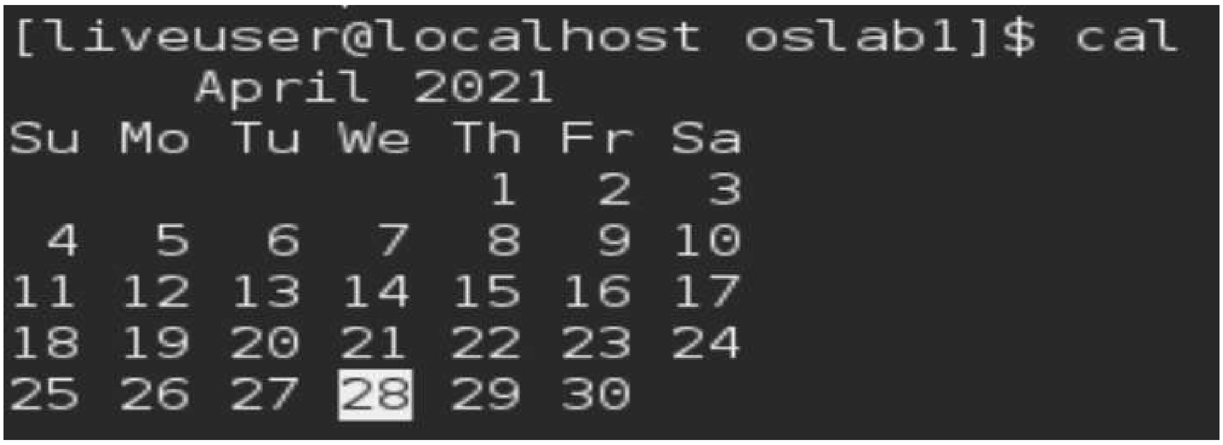
```
[liveuser@localhost oslab1]$ who -b
         system boot  2021-04-28 13:09
```

who am i



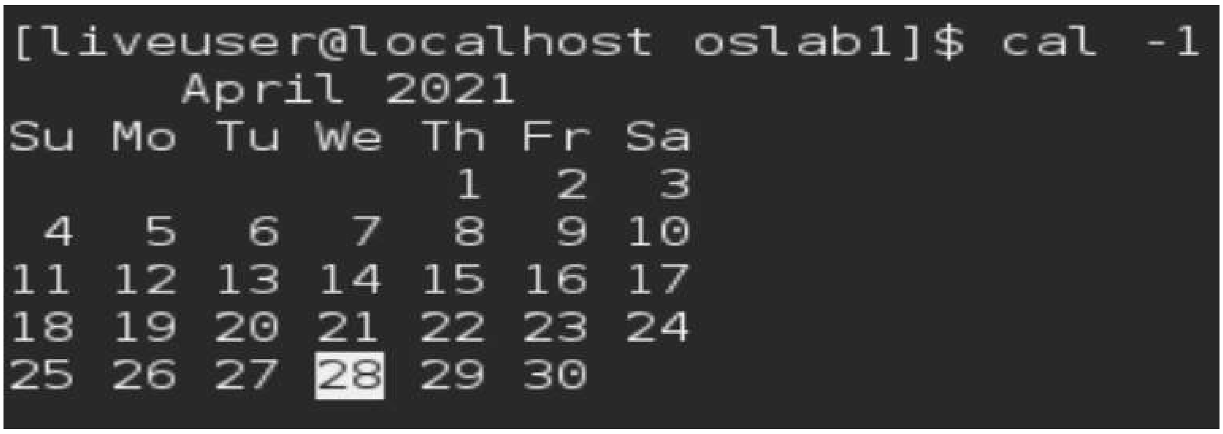
```
[liveuser@localhost oslab1]$ who am i
liveuser pts/0        2021-04-28 13:11 (:0)
```

cal:



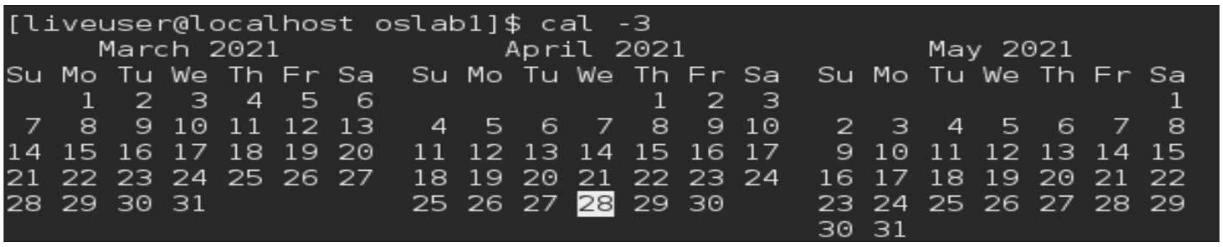
```
[liveuser@localhost oslab1]$ cal
     April 2021
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

cal -1



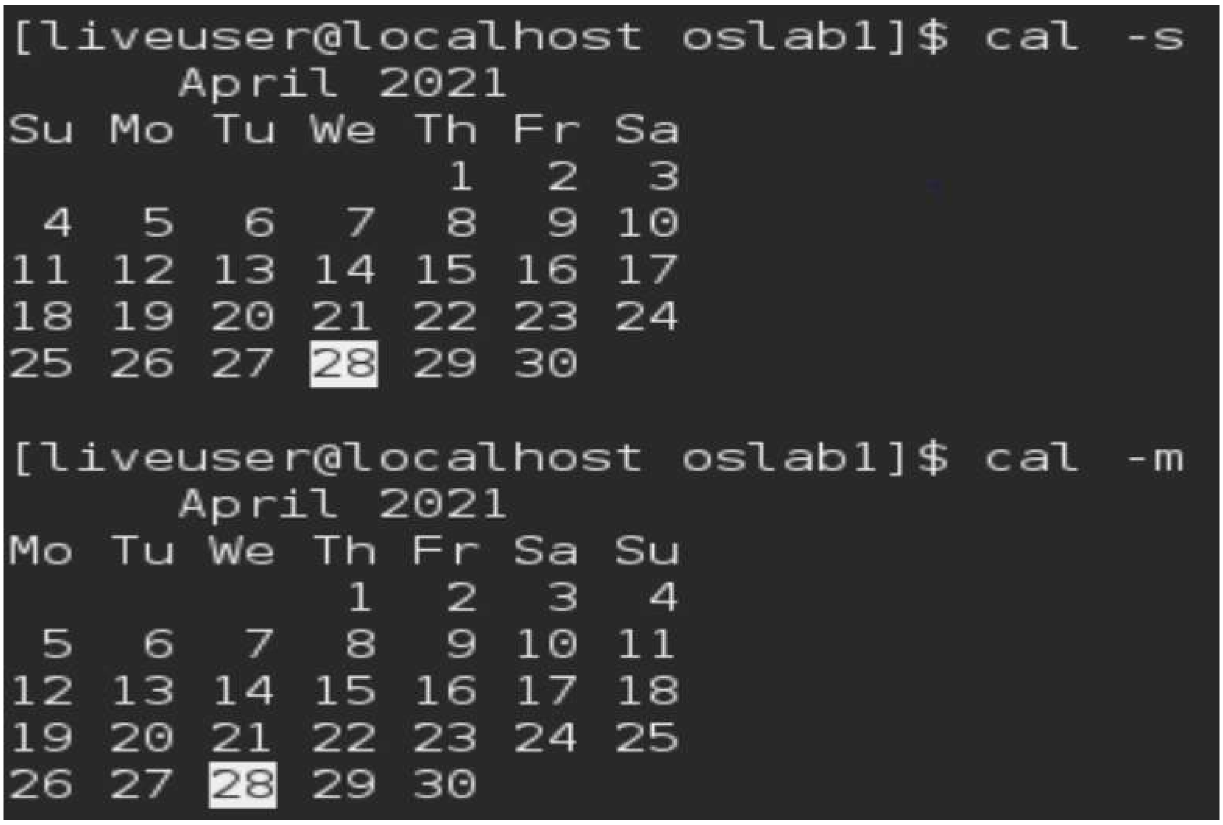
```
[liveuser@localhost oslab1]$ cal -1
     April 2021
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

cal -3



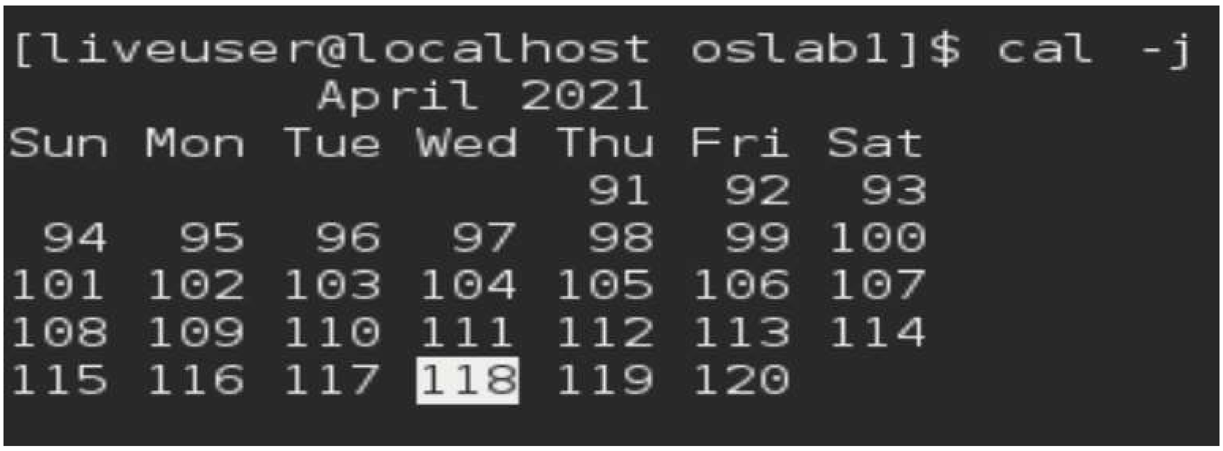
```
[liveuser@localhost oslab1]$ cal -3
     March 2021             April 2021              May 2021
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6               1  2  3                     1
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   2  3  4  5  6  7  8
14 15 16 17 18 19 20  11 12 13 14 15 16 17   9 10 11 12 13 14 15
21 22 23 24 25 26 27  18 19 20 21 22 23 24  16 17 18 19 20 21 22
28 29 30 31           25 26 27 28 29 30     23 24 25 26 27 28 29
                                            30 31
```

cal -s / cal -m



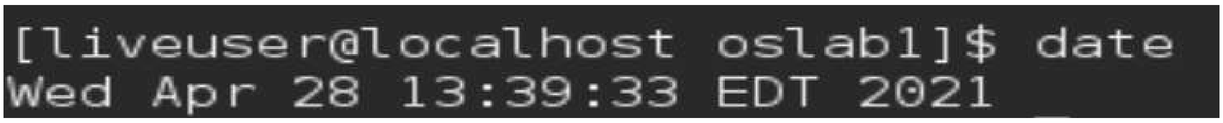
```
[liveuser@localhost oslab1]$ cal -s
     April 2021
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30

[liveuser@localhost oslab1]$ cal -m
     April 2021
Mo Tu We Th Fr Sa Su
          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
```

cal -j

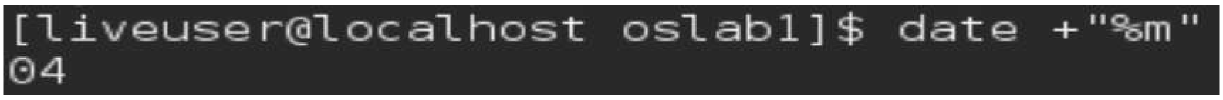


```
[liveuser@localhost oslab1]$ cal -j
          April 2021
Sun Mon Tue Wed Thu Fri Sat
                 91  92  93
 94  95  96  97  98  99 100
101 102 103 104 105 106 107
108 109 110 111 112 113 114
115 116 117 118 119 120
```

date

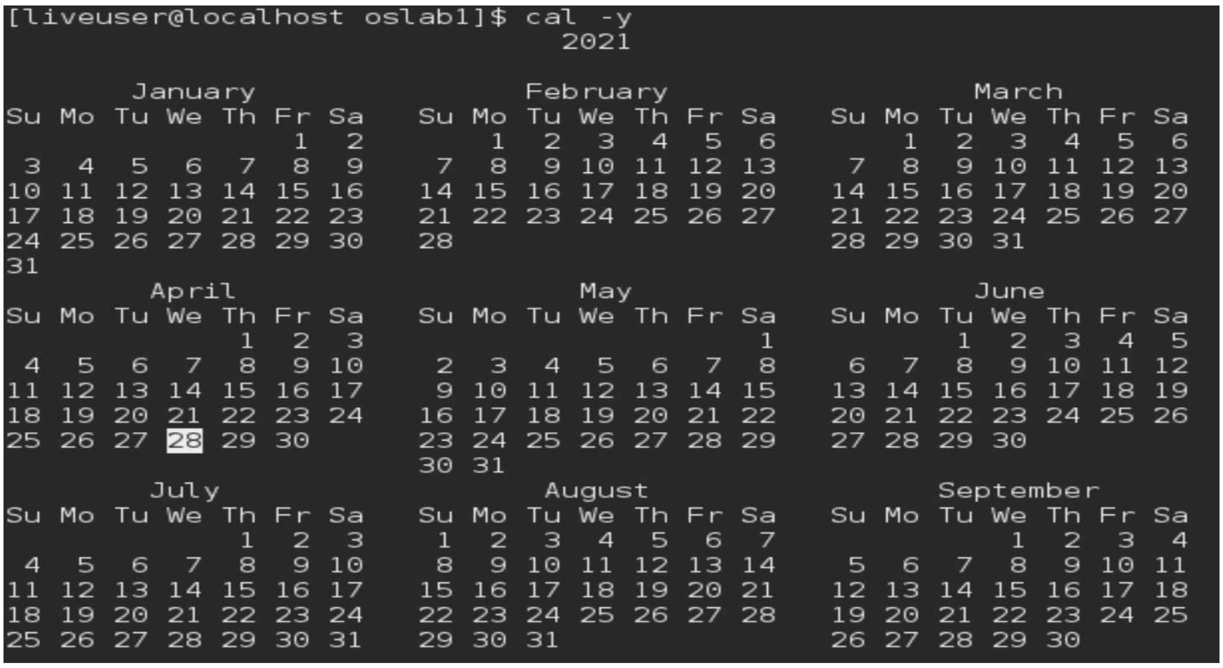


```
[liveuser@localhost oslab1]$ date
Wed Apr 28 13:39:33 EDT 2021
```

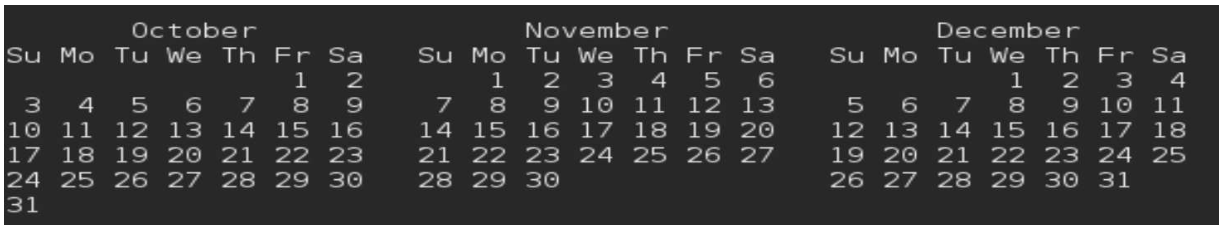


```
[liveuser@localhost oslab1]$ date +"%m"
04
```

cal -y



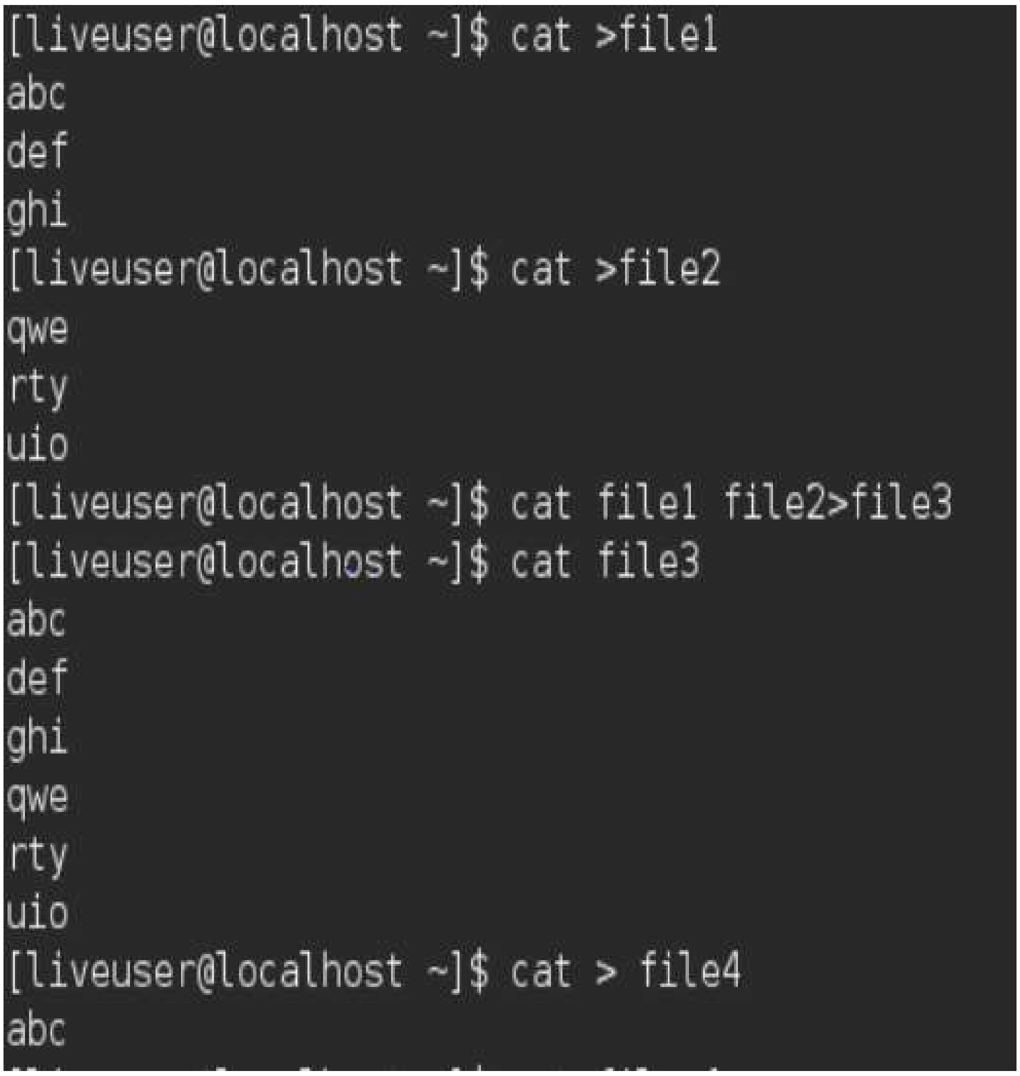
```
[liveuser@localhost oslab1]$ cal -y
                               2021

       January                February                 March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2       1  2  3  4  5  6       1  2  3  4  5  6
 3  4  5  6  7  8  9    7  8  9 10 11 12 13    7  8  9 10 11 12 13
10 11 12 13 14 15 16   14 15 16 17 18 19 20   14 15 16 17 18 19 20
17 18 19 20 21 22 23   21 22 23 24 25 26 27   21 22 23 24 25 26 27
24 25 26 27 28 29 30   28                     28 29 30 31
31
        April                   May                   June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
             1  2  3                      1          1  2  3  4  5
 4  5  6  7  8  9 10    2  3  4  5  6  7  8    6  7  8  9 10 11 12
11 12 13 14 15 16 17    9 10 11 12 13 14 15   13 14 15 16 17 18 19
18 19 20 21 22 23 24   16 17 18 19 20 21 22   20 21 22 23 24 25 26
25 26 27 28 29 30      23 24 25 26 27 28 29   27 28 29 30
                       30 31
        July                  August               September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
             1  2  3    1  2  3  4  5  6  7             1  2  3  4
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    5  6  7  8  9 10 11
11 12 13 14 15 16 17   15 16 17 18 19 20 21   12 13 14 15 16 17 18
18 19 20 21 22 23 24   22 23 24 25 26 27 28   19 20 21 22 23 24 25
25 26 27 28 29 30 31   29 30 31               26 27 28 29 30
```



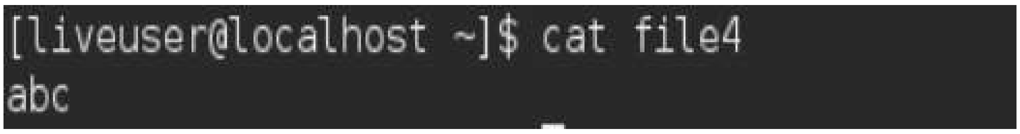
```
       October                November               December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2       1  2  3  4  5  6             1  2  3  4
 3  4  5  6  7  8  9    7  8  9 10 11 12 13    5  6  7  8  9 10 11
10 11 12 13 14 15 16   14 15 16 17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

POST EXPERIMENT EXERCISE

Q1: Create two files, file1 and file2 in dir1

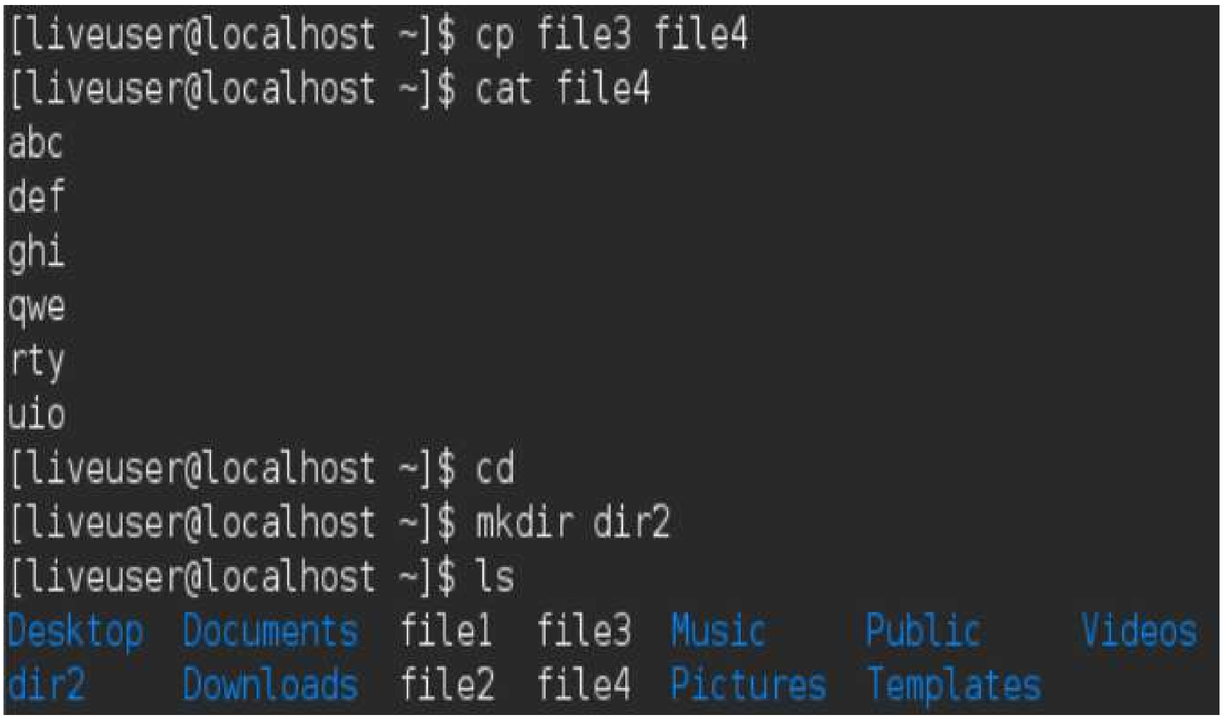


```
[liveuser@localhost ~]$ cat >file1
abc
def
ghi
[liveuser@localhost ~]$ cat >file2
qwe
rty
uio
[liveuser@localhost ~]$ cat file1 file2>file3
[liveuser@localhost ~]$ cat file3
abc
def
ghi
qwe
rty
uio
[liveuser@localhost ~]$ cat > file4
abc
```

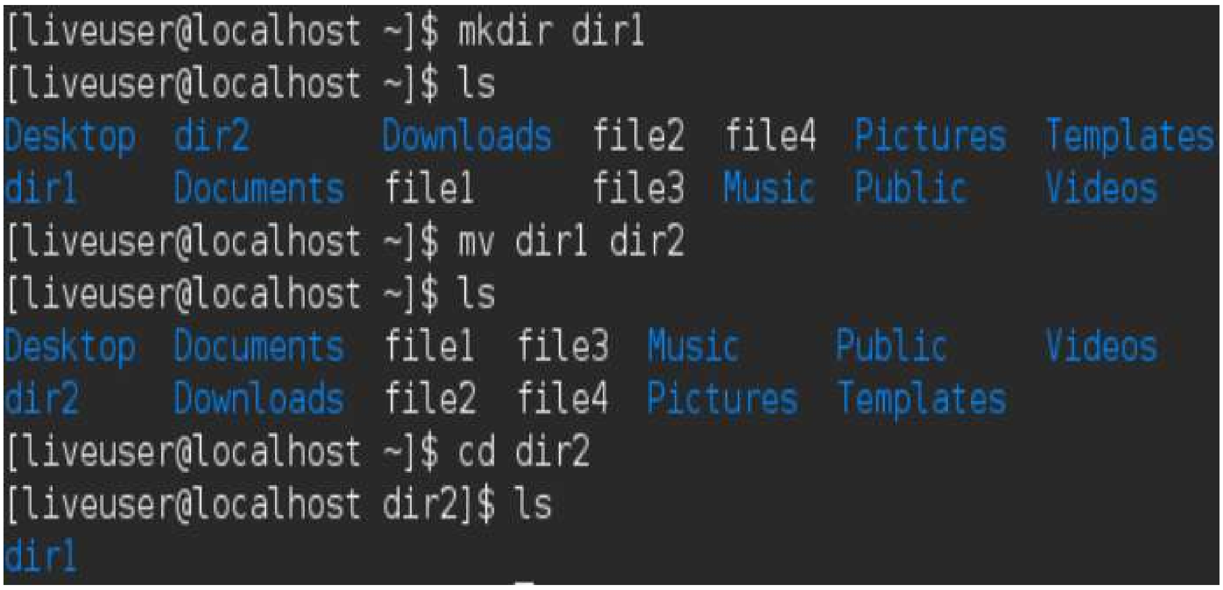


```
[liveuser@localhost ~]$ cat file4
abc
```

- Append content of both files to file3.
- Concatenate content of both files to file4. Make note of difference between output of two commands.
- Move content of dir1 to dir2
- Display dir1. Make note of output.



```
[liveuser@localhost ~]$ cp file3 file4
[liveuser@localhost ~]$ cat file4
abc
def
ghi
qwe
rty
uio
[liveuser@localhost ~]$ cd
[liveuser@localhost ~]$ mkdir dir2
[liveuser@localhost ~]$ ls
Desktop  Documents  file1  file3  Music     Public     Videos
dir2     Downloads  file2  file4  Pictures  Templates
```



```
[liveuser@localhost ~]$ mkdir dir1
[liveuser@localhost ~]$ ls
Desktop  dir2       Downloads  file2  file4  Pictures  Templates
dir1     Documents  file1      file3  Music  Public    Videos
[liveuser@localhost ~]$ mv dir1 dir2
[liveuser@localhost ~]$ ls
Desktop  Documents  file1  file3  Music     Public     Videos
dir2     Downloads  file2  file4  Pictures  Templates
[liveuser@localhost ~]$ cd dir2
[liveuser@localhost dir2]$ ls
dir1
```

While appending the files the contents of file1 and file2 are seen in file3 but when we concat the contents from file1 and file2 to file4 and then when we display the contents of file4 only the contents of file2 can be seen. When the contents from dir1 are moved to dir2 then dir1 gets empty and dir2 gets the contents of dir1.

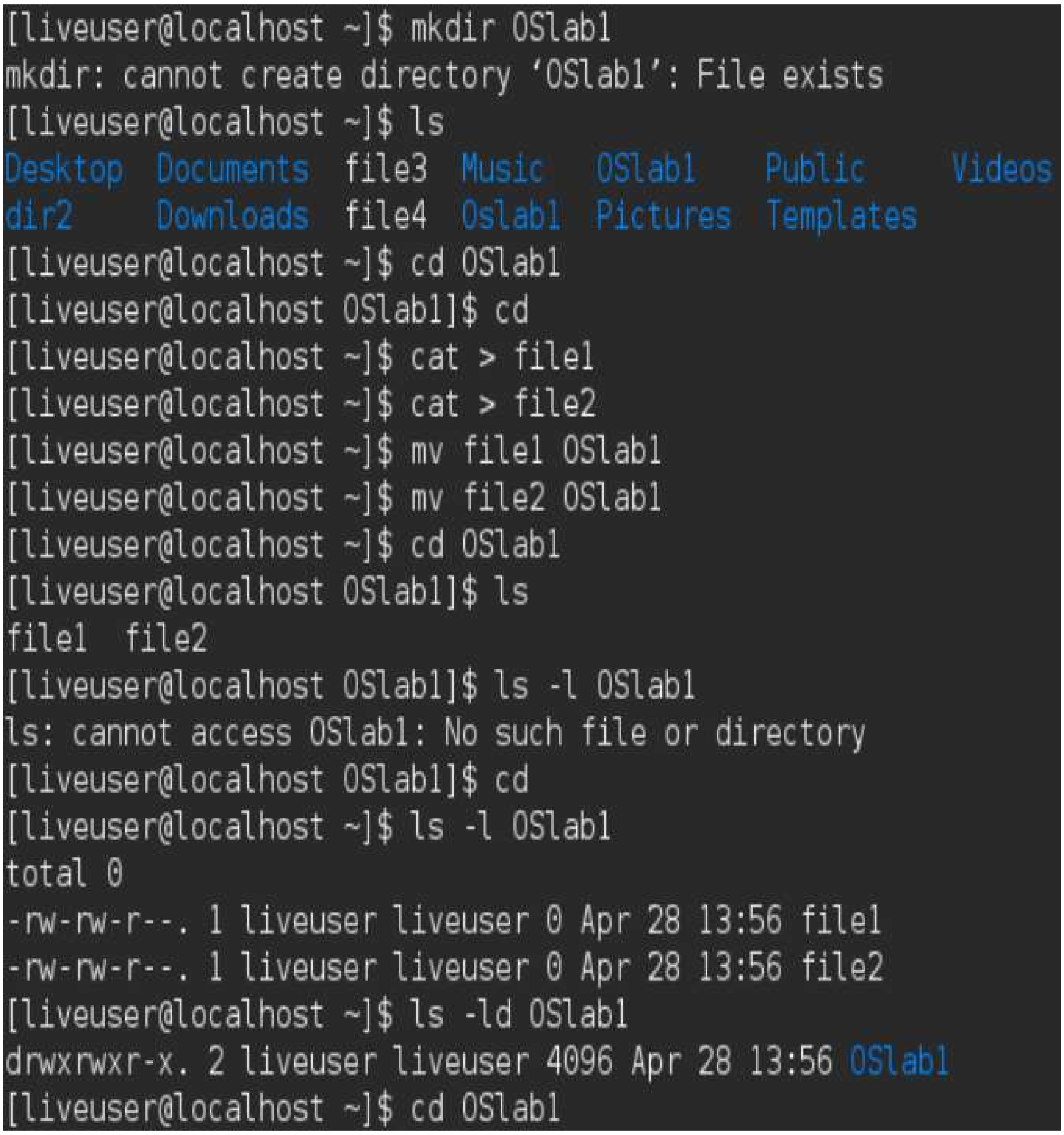
Q 2 : Create a directory called OSlab1. Note that once a directory is created, you cannot create it again - try creating the directory again and note the error message you get.

a) Create two empty files file1 and file2 and move to OSlab1

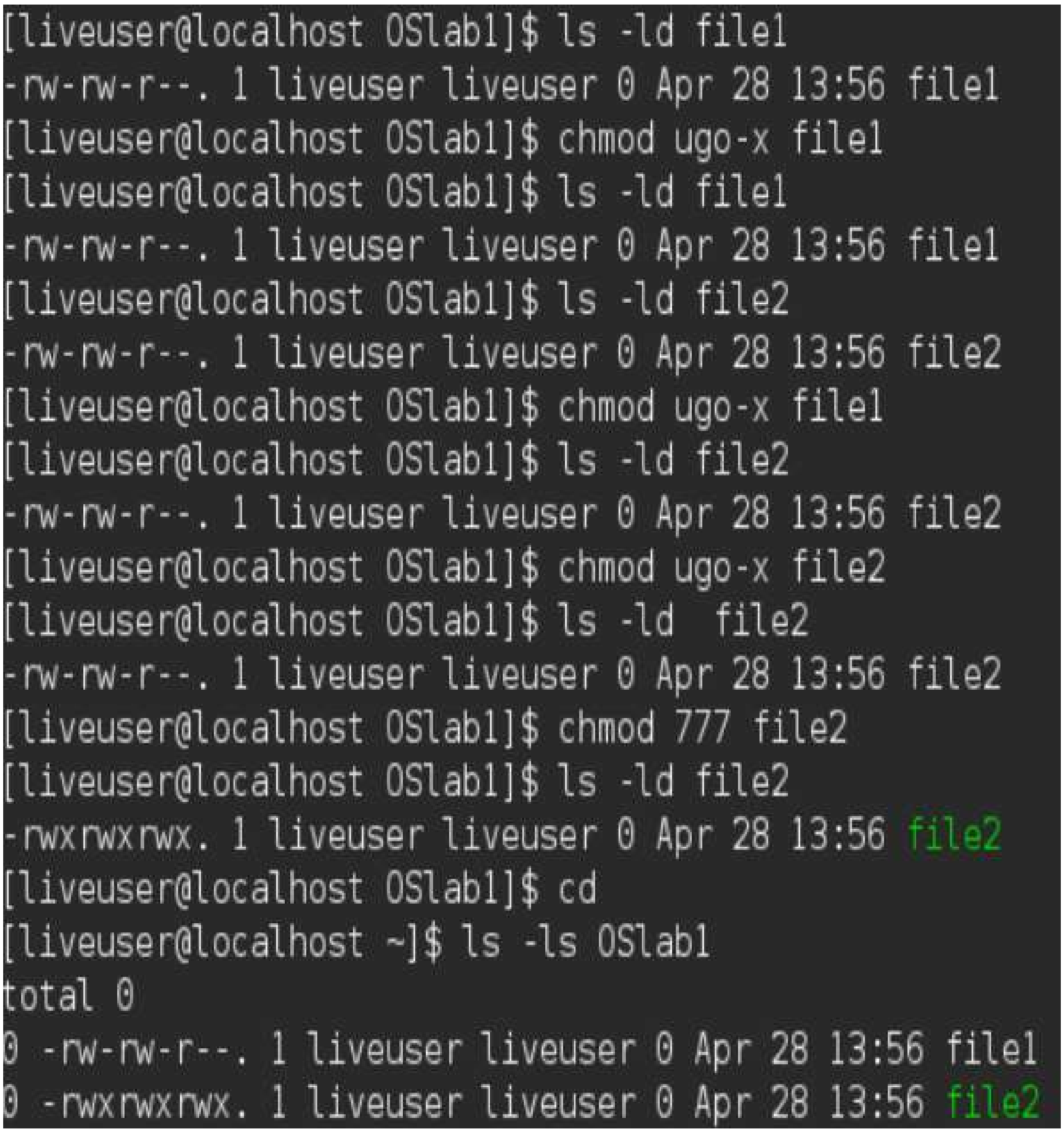
b) Also try the two commands: ls -l OSlab1 and ls -ld OSlab1 and note the differences.

c) Display the contents of the current directory. Make note of the output.

d) Change permissions of file1 so as all type of users get execute permission without specifying type of user in command.



```
[liveuser@localhost ~]$ mkdir OSlab1
mkdir: cannot create directory ‘OSlab1’: File exists
[liveuser@localhost ~]$ ls
Desktop  Documents  file3  Music   OSlab1    Public     Videos
dir2     Downloads  file4  Oslab1  Pictures  Templates
[liveuser@localhost ~]$ cd OSlab1
[liveuser@localhost OSlab1]$ cd
[liveuser@localhost ~]$ cat > file1
[liveuser@localhost ~]$ cat > file2
[liveuser@localhost ~]$ mv file1 OSlab1
[liveuser@localhost ~]$ mv file2 OSlab1
[liveuser@localhost ~]$ cd OSlab1
[liveuser@localhost OSlab1]$ ls
file1  file2
[liveuser@localhost OSlab1]$ ls -l OSlab1
ls: cannot access OSlab1: No such file or directory
[liveuser@localhost OSlab1]$ cd
[liveuser@localhost ~]$ ls -l OSlab1
total 0
-rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file1
-rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file2
[liveuser@localhost ~]$ ls -ld OSlab1
drwxrwxr-x. 2 liveuser liveuser 4096 Apr 28 13:56 OSlab1
[liveuser@localhost ~]$ cd OSlab1
```



```
[liveuser@localhost OSlab1]$ ls -ld file1
-rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file1
[liveuser@localhost OSlab1]$ chmod ugo-x file1
[liveuser@localhost OSlab1]$ ls -ld file1
-rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file1
[liveuser@localhost OSlab1]$ ls -ld file2
-rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file2
[liveuser@localhost OSlab1]$ chmod ugo-x file1
[liveuser@localhost OSlab1]$ ls -ld file2
-rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file2
[liveuser@localhost OSlab1]$ chmod ugo-x file2
[liveuser@localhost OSlab1]$ ls -ld  file2
-rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file2
[liveuser@localhost OSlab1]$ chmod 777 file2
[liveuser@localhost OSlab1]$ ls -ld file2
-rwxrwxrwx. 1 liveuser liveuser 0 Apr 28 13:56 file2
[liveuser@localhost OSlab1]$ cd
[liveuser@localhost ~]$ ls -ls OSlab1
total 0
0 -rw-rw-r--. 1 liveuser liveuser 0 Apr 28 13:56 file1
0 -rwxrwxrwx. 1 liveuser liveuser 0 Apr 28 13:56 file2
```

When we try creating the same directory, it says that it cannot create the directory because it exists already. After entering the empty files into OSLab1, when we list the contents of the directory the two files are seen. To change the permissions, we need to put the necessary commands and then the permissions are granted to the user. The difference between ls -l and ls -ld is that in ls -l we get to know all the details of the files in the listed manner. The ls -ld only tells about the number of files, date and time.