Lab # 3
File/Directory Permission and Process Management

I. File/Directory Permission

Type			
d (directory)	4 2 1 R w x (read) (write) (execute)	4 2 1 r w x (read) (write) (execute)	4 2 1 r w x (read) (write) (execute)
	4+2+1	4+0+1	000
	rwx	r-x	---
	owner	group	other

Linux (and almost all other Unixish systems) have three user classes as follows:

- User (u): The owner of file
- Group (g): Other user who are in group (to access files)
- Other (o): Everyone else

You can setup following mode on each files. In a Linux and UNIX set of permissions is called as mode:

- Read (r)
- Write (w)
- Execute (x)

You can use octal number to represent mode/permission:
- r: 4
- w: 2
- x: 1

For example, for file owner you can use octal mode as follows. Read, write and execute (full) permission on a file in octal is

0+r+w+x = 0+4+2+1 = 7

Only Read and write permission on a file in octal is

0+r+w+x = 0+4+2+0 = 6

Only read and execute permission on a file in octal is

0+r+w+x = 0+4+0+1 = 5

Permissions	Symbolic	Binary	Octal
read, write, and execute	rwx	111	7
read and write	rw-	110	6
read and execute	r-x	101	5
Read	r--	100	4
write and execute	-wx	011	3
Write	-w-	010	2
Execute	--x	001	1
no permissions	---	000	0

A) Working with Users

i) <u>Adding user:</u>

Login as root.

```
$ useradd <username>
Example: $ useradd omer
         $ tail -5 /etc/passwd
         $ id omer
```

ii) <u>Adding user with specific full name:</u>

```
$ useradd –c <userfullname> <username>
Example: $ useradd –c ‘Hadi Khan’ Hadi
         $ tail -5 /etc/passwd
         $ id Hadi
```

iii) <u>Creating user with account expiration date</u>

```
$ useradd -e ‘YYY-MM-DD’ <username>
Example: $ useradd -e ‘2024-12-31’ faisal
         $ tail -5 /etc/passwd
         $ id faisal
```

iv) Adding user in a group:

```
$ useradd –g <groupname> <username>
Example: $ useradd –g Students Hadi
         $ tail -5 /etc/passwd
         $ id Hadi
```

iv) Changing password:

```
$ passwd <username>
  Example: $ passwd Hadi
```

v) Loacking/unlocking password:

```
$ passwd –l <username>
  Example: $ passwd –l Hadi

$ passwd –u <username>
  Example: $ passwd –u Hadi
```

vi) Deleting user:

```
$ userdel <username>
  Example: $ userdel faisal
                $ tail -5 /etc/passwd
                $ id faisal
```

B) Working with Group

i) <u>Adding Group:</u>

Login as root.

```
$ groupadd <groupname>
Example: $ groupadd students
              $ tail -5 /etc/group
```

ii) <u>Adding Group with specific Group ID:</u>

```
$ groupadd –g <groupID(Numeric)> <groupname>
Example: $ groupadd –g 1009 OSstudents
              $ tail -5 /etc/group
```

iii) <u>Changing Group ID/Group Name:</u>

```
$ groupmod –g <newgroupID(Numeric)> <groupname>
Example: $ groupmod –g 1008 OSstudents
              $ tail -5 /etc/group

$ groupmod –n <newgroupname)> <groupname>
Example: $ groupmod –n OSLabStudents OSstudents
              $ tail -5 /etc/group
```

iv) <u>Adding users to Group</u>

```
$ usermod –aG <groupname> <username>
Example: $ usermod –aG OSLabStudents omer
              $ id omer
```

v) Removing users from Group

```
$ gpasswd --delete <username> <groupname>
Example: $ gpasswd –-delete omer OSLabStudents
               $ id omer
```

vi) Removing Group

```
$ groupdel <groupname>
Example: $ groupdel OSLabStudents
               $ tail -5 /etc/group
```

C) File/Directory permissions

i) Changing owner

```
$ chown <newowner>:<newgroup> <directoryname/filename>
Example:  $mkdir testdir
               $ chown bilal:students testdir
               $ ls –l
```

ii) Changing rights

```
$ chmod <u+rwx, g+rwx, o+rwx> <directoryname/filename>
Example:  chmod 770 testdir
               $ ls -lh
```

For example, if a text file has 666 permissions, it grants read and write permission to everyone. Similarly a directory with 777 permissions, grants read, write, and execute permission to everyone.

D) Process Management:

How to see processes
How to kill processes

i) Reviewing process

```
$ ps
```

What processes are running

```
$ ps ax
```

Show all-extended processes running

```
$ ps aux
```

Show processes running and user information

$ pstree
Show processes in tree structure.

ii) Killing Process

$ kill <pid>
Example: $ kill 2598
Kill Process immediately
$ kill -9 <pid>
Example: $ kill -9 2598

Kill Process with name
$ pkill <processname>
Example: $ pkill gcalctool

To kill processes with name.
$ pkillall <processname>
Example: $ pkillall gcalctool

iii) System Monitoring Commands

To show system time when it is open , Open from how many hours, User login and Load average of processes
$ uptime

System monitoring command, Uptime info, Processes refresh after every 2 minute
$ top
System monitoring command

Processes update after every 2 second.
$ watch uptime

III. System Call:

Fork()

fork() creates a new child process. If we call fork() in the parent program, it creates a child process which shares an exact copy of the address space but a different one. Both parent and child processes have different address spaces, but they share the same memory segment.

```
#include <stdio.h>
#include <unistd.h>

int main(int argc,char *argv[])
{
	//forkFunction usage
	int i = 0;
	printf("before fork\n");
	pid_t pid = fork();
```

```
        printf("after fork\n");
        if (pid < 0){
                printf("error\n");
                return 1;
        }
        else if (pid == 0)
        {
                printf("fork success,this is son process\n");
                while (i<10)
                {
                        i += 1;
                        printf("this is son process,i=%d\n",i);
                        sleep(1);
                }
        }
        else
        {
                printf("fork success,this is father process,son process id is %d \n",pid);
                while (i<10)
                {
                        i += 2;
                        printf("this is father process,i=%d\n",i);
                        sleep(2);
                }
        }
  return 0;
}
```

OUTPUT

```
before fork
after fork
fork success,
this is father process, son process id is 11054
this is father process,i=2
after fork fork success,
this is son process this is son process, i=1
this is son process,i=2
this is father process,i=4
this is son process,i=3
this is son process,i=4
this is father process,i=6
this is son process,i=5
this is son process,i=6
this is father process,i=8
this is son process,i=7
this is son process,i=8
this is father process,i=10
this is son process,i=9
this is son process,i=10
```

Lab Tasks

1. Create two users user1 and user2 and add them in a group OSLab

2. Rename OSLab group to OSLabStudents

3. Create a filename OSLabFile1.txt and change its owner and group to user1 and OSLabStudents respectively.

4. Delete user1

5. Calculate number of times hello is printed:

```
#include <stdio.h>
#include <sys/types.h>
int main()
{
        fork();
        fork();
        fork();
        printf("hello\n");
   return 0;
}
```

6. Predict output of below program.

```
#include <stdio.h>
#include <unistd.h>
int main()
{
        fork();
        fork() && fork() || fork();
        fork();
        printf("forked\n");
  return 0;
```