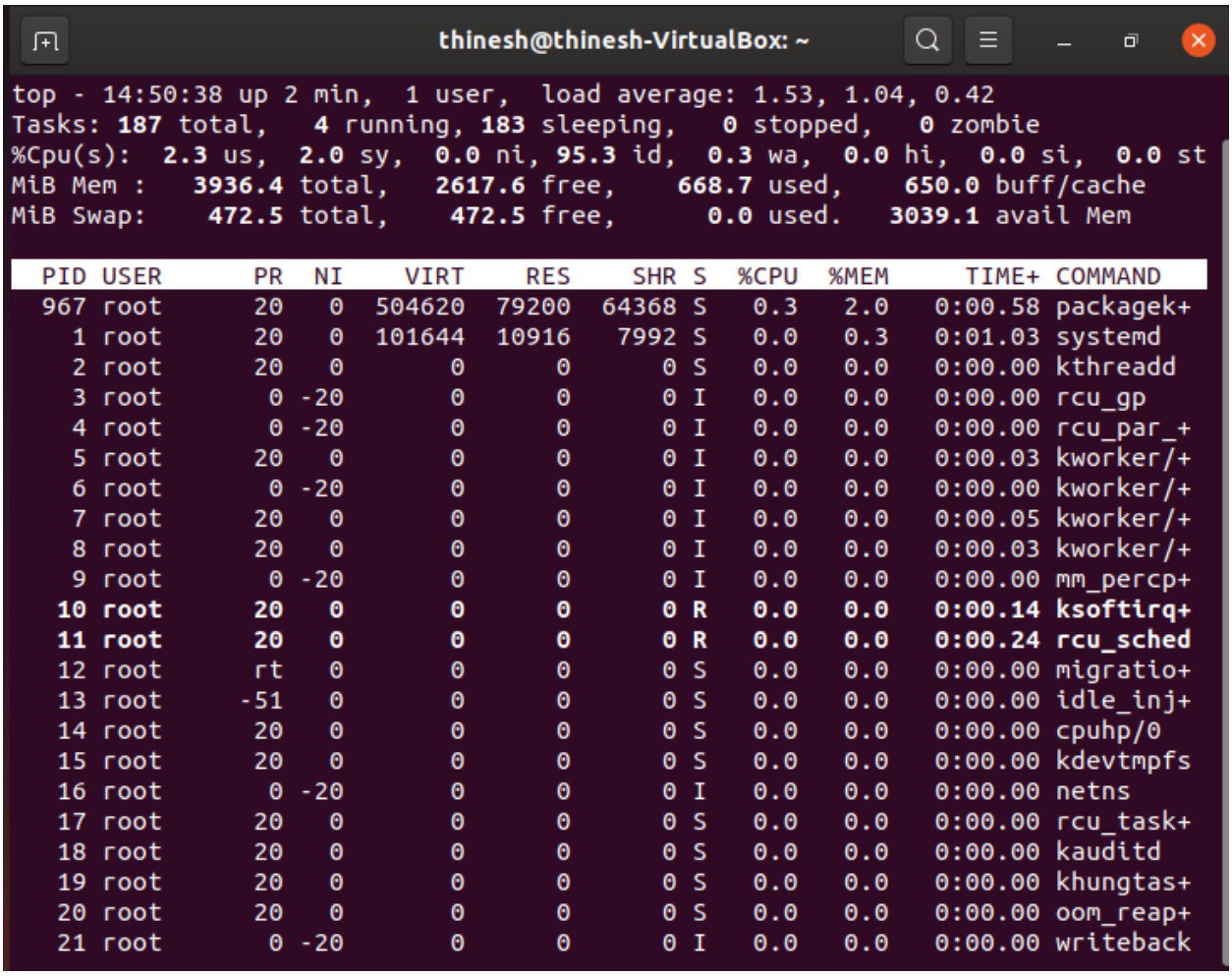
S.THINESH

E/15/366

Multiprocessing

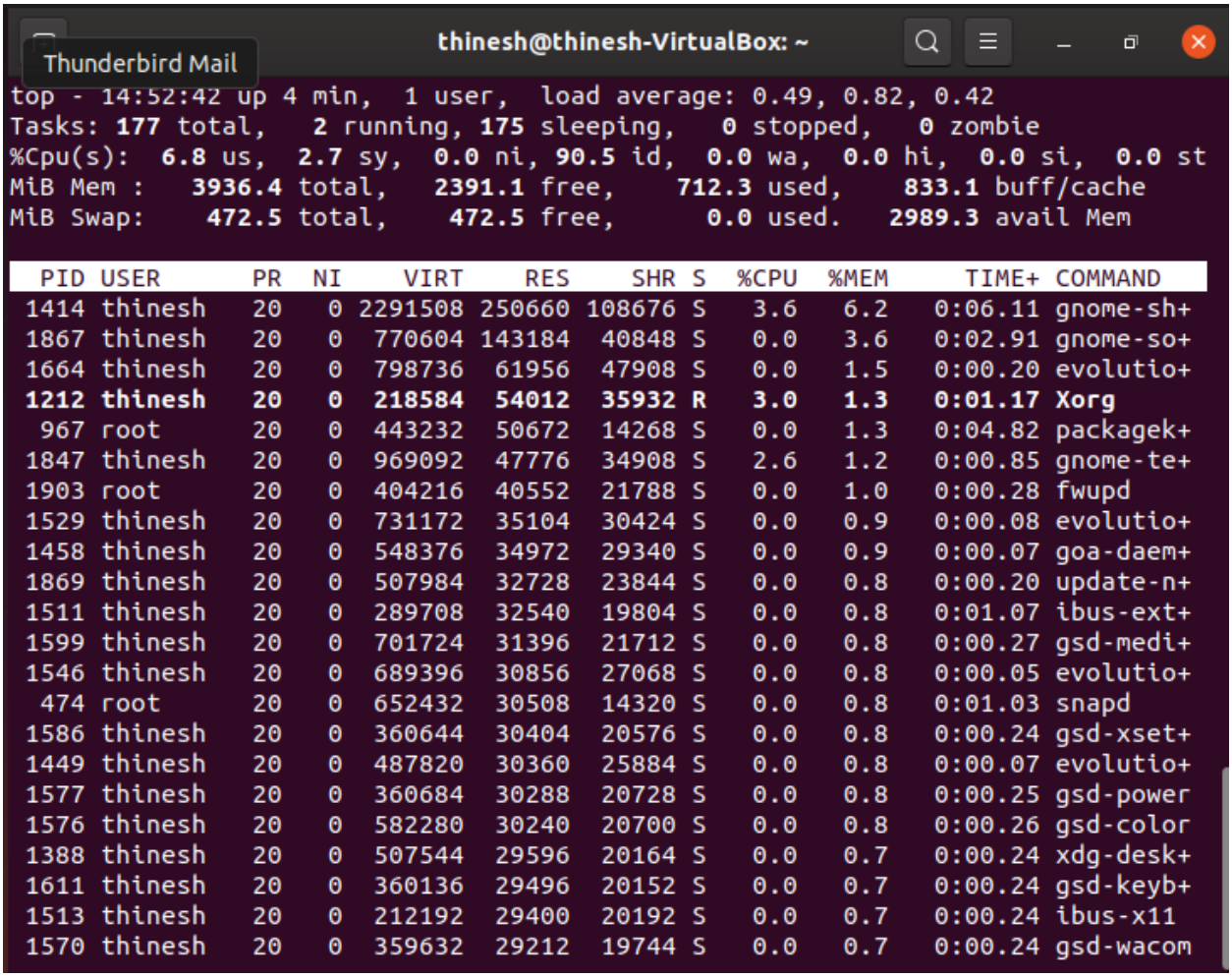
1. Processes

i. top shows you details of active processes. The processes are sorted by CPU usage by default. Sort them by memory usage.



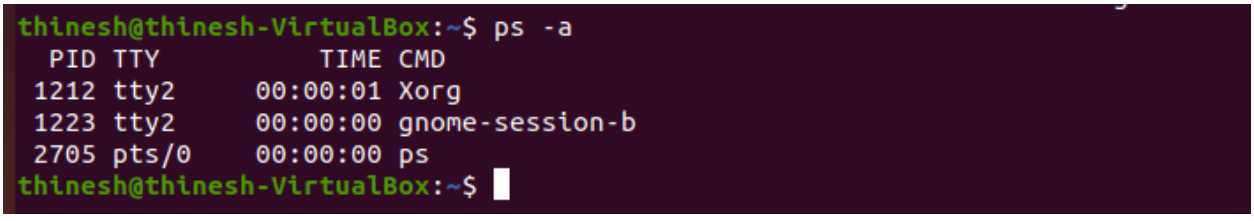
Sort them by memory usage.

top -o %MEM

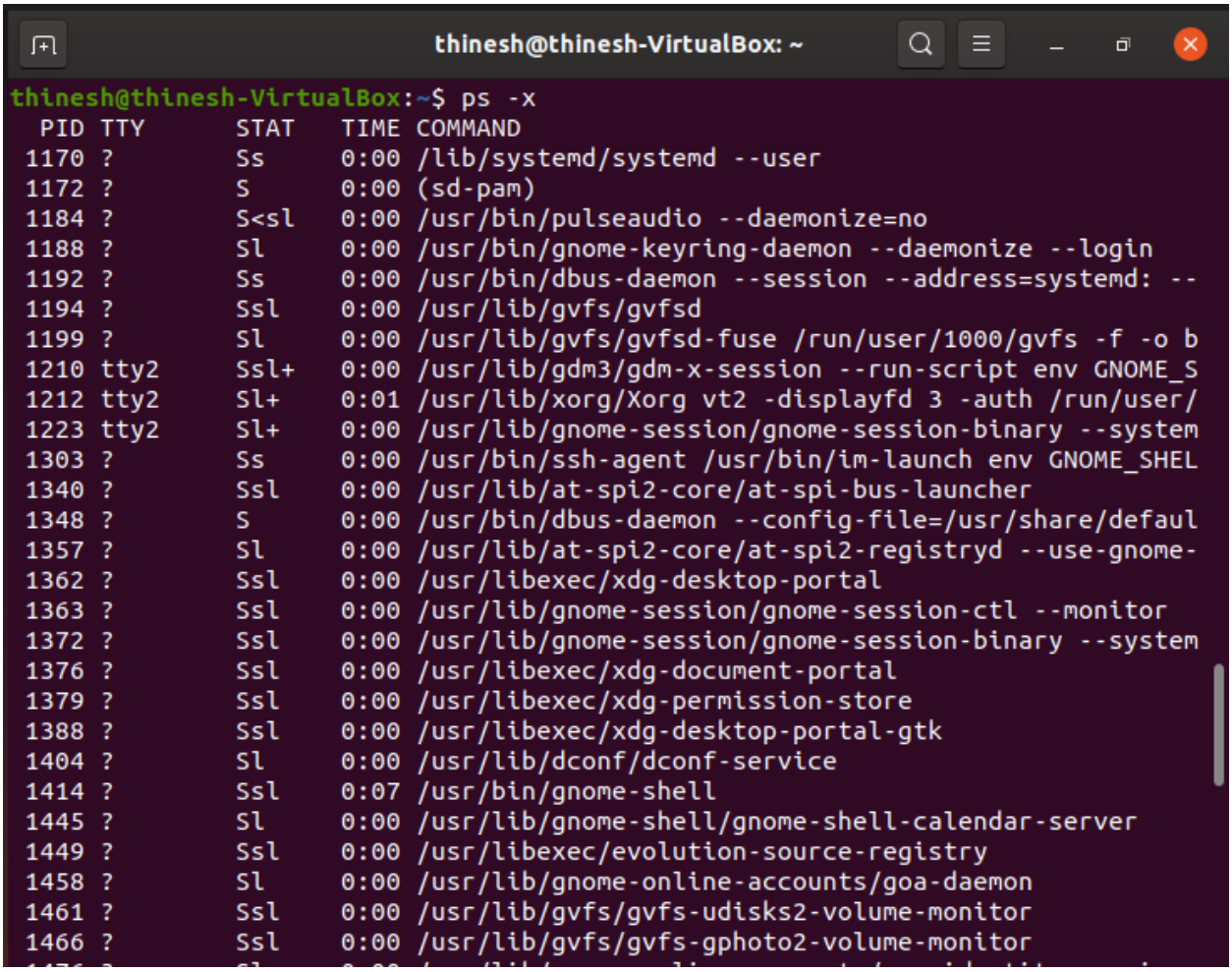


ii. Run ps with the following options: -a, -x, -u, -w. What is the name of the process with PID 1?

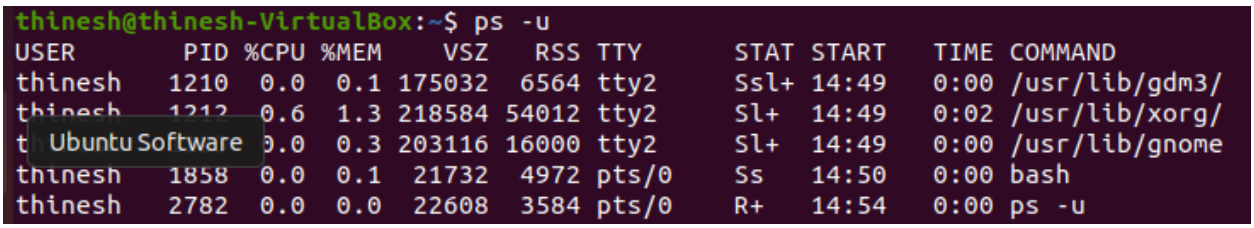
Ps –a → list all the processes



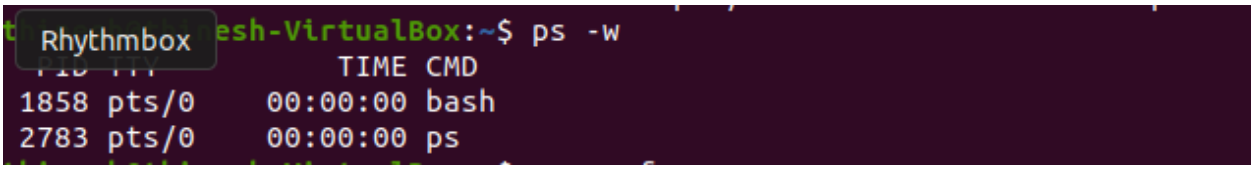
Ps -x


thinesh@thinesh-VirtualBox: ~
thinesh@thinesh-VirtualBox:~$ ps -x
PID TTY STAT TIME COMMAND
1170 ? Ss 0:00 /lib/systemd/systemd --user
1172 ? S 0:00 (sd-pam)
1184 ? S<sl 0:00 /usr/bin/pulseaudio --daemonize=no
1188 ? Sl 0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
1192 ? Ss 0:00 /usr/bin/dbus-daemon --session --address=systemd: --
1194 ? Ssl 0:00 /usr/lib/gvfs/gvfsd
1199 ? Sl 0:00 /usr/lib/gvfs/gvfsd-fuse /run/user/1000/gvfs -f -o b
1210 tty2 Ssl+ 0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_S
1212 tty2 Sl+ 0:01 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/
1223 tty2 Sl+ 0:00 /usr/lib/gnome-session/gnome-session-binary --system
1303 ? Ss 0:00 /usr/bin/ssh-agent /usr/bin/im-launch env GNOME_SHEL
1340 ? Ssl 0:00 /usr/lib/at-spi2-core/at-spi-bus-launcher
1348 ? S 0:00 /usr/bin/dbus-daemon --config-file=/usr/share/defaul
1357 ? Sl 0:00 /usr/lib/at-spi2-core/at-spi2-registryd --use-gnome-
1362 ? Ssl 0:00 /usr/libexec/xdg-desktop-portal
1363 ? Ssl 0:00 /usr/lib/gnome-session/gnome-session-ctl --monitor
1372 ? Ssl 0:00 /usr/lib/gnome-session/gnome-session-binary --system
1376 ? Ssl 0:00 /usr/libexec/xdg-document-portal
1379 ? Ssl 0:00 /usr/libexec/xdg-permission-store
1388 ? Ssl 0:00 /usr/libexec/xdg-desktop-portal-gtk
1404 ? Sl 0:00 /usr/lib/dconf/dconf-service
1414 ? Ssl 0:07 /usr/bin/gnome-shell
1445 ? Sl 0:00 /usr/lib/gnome-shell/gnome-shell-calendar-server
1449 ? Ssl 0:00 /usr/libexec/evolution-source-registry
1458 ? Sl 0:00 /usr/lib/gnome-online-accounts/goa-daemon
1461 ? Ssl 0:00 /usr/lib/gvfs/gvfs-udisks2-volume-monitor
1466 ? Ssl 0:00 /usr/lib/gvfs/gvfs-gphoto2-volume-monitor

Ps –u


thinesh@thinesh-VirtualBox:~$ ps -u
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
thinesh 1210 0.0 0.1 175032 6564 tty2 Ssl+ 14:49 0:00 /usr/lib/gdm3/
thinesh 1212 0.6 1.3 218584 54012 tty2 Sl+ 14:49 0:02 /usr/lib/xorg/
Ubuntu Software
thinesh 1858 0.0 0.1 21732 4972 pts/0 Ss 14:50 0:00 bash
thinesh 2782 0.0 0.0 22608 3584 pts/0 R+ 14:54 0:00 ps -u

Ps –w


Rhythmbox
esh-VirtualBox:~$ ps -w
TIME CMD
1858 pts/0 00:00:00 bash
2783 pts/0 00:00:00 ps

What is the name of the process with PID 1 :

Ps –eaf

PID 1 will be always the init process

1.1. Creating a new process

Exercise 2:

1. In what order are the messages from parent and child printed? Is the order always the same? (Code attached ex2_1.c)

I got parent process first and then the child process printed.

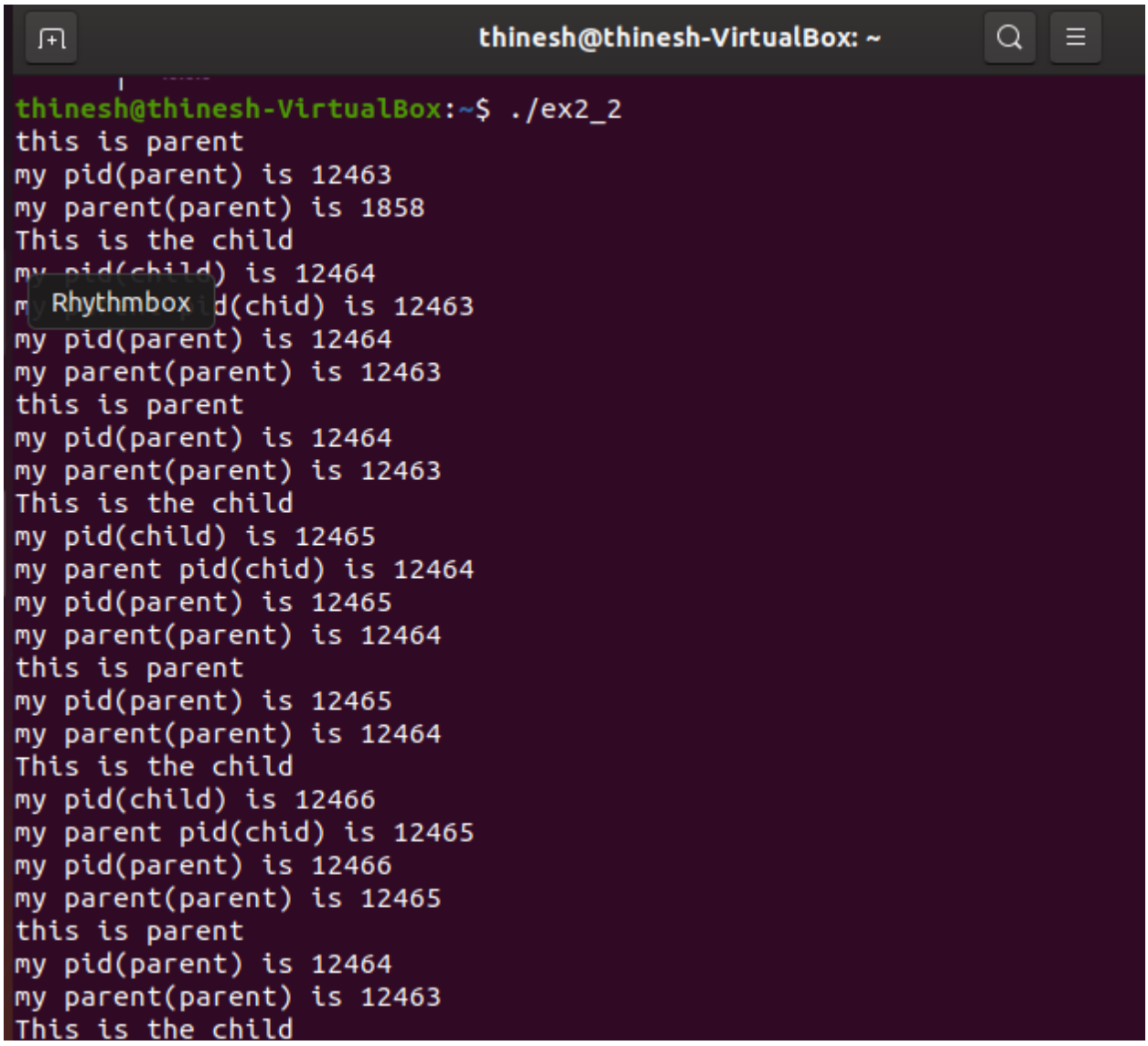
But always parent process first and child process printed.
This is parent
This is the child
OR
This is the child
This is parent
Here, two outputs are possible because the parent process and child process are running concurrently. So we don't know whether the OS will first give control to the parent process or the child process.

ii. How many children will the following program spawn? Draw a diagram illustrating the parent-child relationships between processes. (Code attached to ex2_2.c)

```
  GNU nano 4.3                          ex2_2.c
int pid;
for(int i=0;i<3;i++){
pid=fork();

if (pid<0){
 perror("fork");
 exit(1);
}

//while(pid!=0){
//wait();
//}

if (pid==0){
puts("This is the child");
printf("my pid(child) is %d\n",getpid());
printf("my parent pid(chid) is %d\n",getppid());
wait(2);
}
else
puts("this is parent");
printf("my pid(parent) is %d\n",getpid());
printf("my parent(parent) is %d\n",getppid());
wait(2);
```

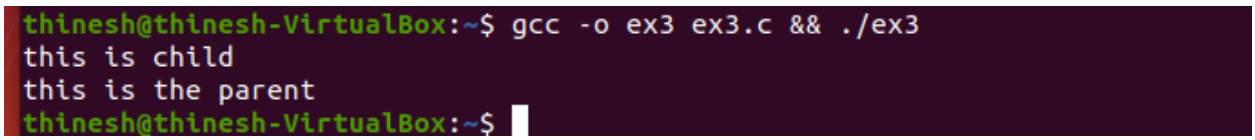
There are eight processes. The program changed for identify how many child and parent processes are created.

```
fork ();   // Line 1
fork ();   // Line 2
fork ();   // Line 3

       L1       // There will be 1 child process
    /     \     // created by line 1.
  L2      L2    // There will be 2 child processes
 /  \    /  \   //  created by line 2
L3  L3  L3  L3  // There will be 4 child processes
                // created by line 3
```



1.2 Waiting for children

Modify the program in section 1.1 so that the parent always prints its message after the child. (Code attached to ex3.c)



1.3 Replacing the process image

1. Compile and run the above code giving it a path as an argument. How many times is the message “Program ls has terminated” printed?

(Code attached to ex4_1.c)

```
thinesh@thinesh-VirtualBox:~$ gcc -o ex3 ex4_1.c && ./ex3
Desktop    Downloads  ex2_1.c  ex2_2.c  ex3.c    Music      Public     Videos
Documents  ex2_1      ex2_2    ex3      ex4_1.c  Pictures   Templates
```

Message was not printed even once.

2. Write a very simple shell that repeatedly prompts the user for a command and runs it with any arguments given. Make sure your shell waits until the command has completed before prompting the user for the next command. (Code attached to ex4_2.c)

```
thinesh@thinesh-VirtualBox:~$ ./ex4
e15366_simple_shell>> ls
ls

Desktop    ex2_1.c  ex3.c    ex5      exnew      Music      Videos
Documents  ex2_2    ex4      ex5_1.c  exnew.c    Pictures
Downloads  ex2_2.c  ex4_1.c  ex5_2.c  lab01      Public
ex2_1      ex3      ex4_2.c  ex5_3.c  lab01.zip  Templates
Child exited
e15366_simple_shell>> dir
dir

Desktop    ex2_1.c  ex3.c    ex5      exnew      Music      Videos
Documents  ex2_2    ex4      ex5_1.c  exnew.c    Pictures
Downloads  ex2_2.c  ex4_1.c  ex5_2.c  lab01      Public
ex2_1      ex3      ex4_2.c  ex5_3.c  lab01.zip  Templates
Child exited
e15366_simple_shell>> cd Lab01
cd
Lab01

Child process could not do execvp
e15366_simple_shell>>
```

2. Multiprocess servers

Exercise 5:

1. Open three terminals and run the server in one. Use nc() to connect as two clients concurrently on port 12345. Type some text in both clients and examine the client and server outputs.

Code attached ex5_1.c

2. Suppose we modify the server parent process to call wait() on the last line above (highlited) to wait until the child serving a client terminates. What would happen?

If we modify the server parent process to call wait() on the last line above ,at the same time multiple clients can be connected but can not send message from one client to another client at the same time.

3. What happens if you terminate the the server while a client is connected, and then try to restart it? (Resolving this issue requires a signal handler.)

When client connected after the termination of the server, the client also terminated.

4. Modify this server to do the following: The client sends the path to a file whose contents the server will send back to the client (if the file exists.) Verify that your new server can handle multiple concurrent connections by using nc(). Can two concurrent clients request the same file?