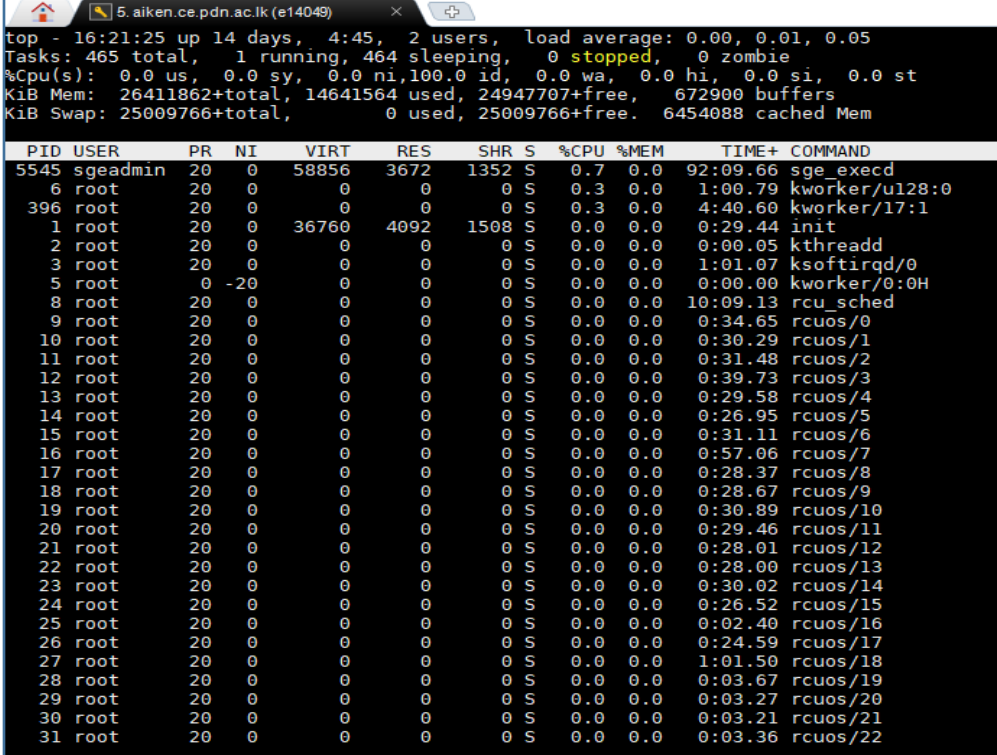


EXERCISE 01 :

i). **top** command: list all the processes. which is sorted by cpu usage by default.

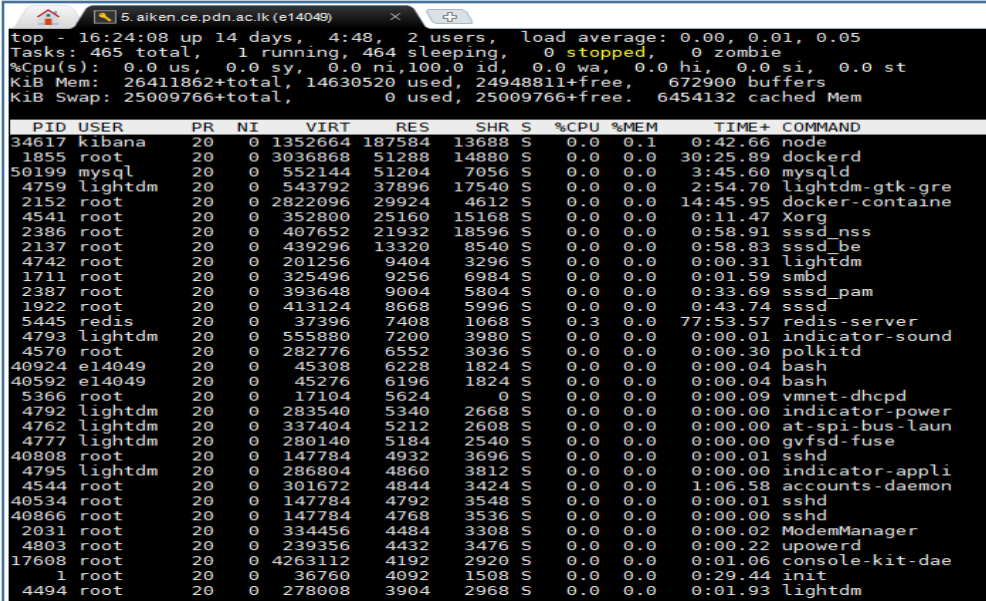


```

top - 16:21:25 up 14 days, 4:45, 2 users, load average: 0.00, 0.01, 0.05
Tasks: 465 total, 1 running, 464 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 26411862+total, 14641564 used, 24947707+free, 672900 buffers
KiB Swap: 25009766+total, 0 used, 25009766+free, 6454088 cached Mem
  
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5545	sgeadmin	20	0	58856	3672	1352	S	0.7	0.0	92:09.66	sge_execd
6	root	20	0	0	0	0	S	0.3	0.0	1:00.79	kworker/u128:0
396	root	20	0	0	0	0	S	0.3	0.0	4:40.60	kworker/17:1
1	root	20	0	36760	4092	1508	S	0.0	0.0	0:29.44	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.05	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	1:01.07	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
8	root	20	0	0	0	0	S	0.0	0.0	10:09.13	rcu_sched
9	root	20	0	0	0	0	S	0.0	0.0	0:34.65	rcuos/0
10	root	20	0	0	0	0	S	0.0	0.0	0:30.29	rcuos/1
11	root	20	0	0	0	0	S	0.0	0.0	0:31.48	rcuos/2
12	root	20	0	0	0	0	S	0.0	0.0	0:39.73	rcuos/3
13	root	20	0	0	0	0	S	0.0	0.0	0:29.58	rcuos/4
14	root	20	0	0	0	0	S	0.0	0.0	0:26.95	rcuos/5
15	root	20	0	0	0	0	S	0.0	0.0	0:31.11	rcuos/6
16	root	20	0	0	0	0	S	0.0	0.0	0:57.06	rcuos/7
17	root	20	0	0	0	0	S	0.0	0.0	0:28.37	rcuos/8
18	root	20	0	0	0	0	S	0.0	0.0	0:28.67	rcuos/9
19	root	20	0	0	0	0	S	0.0	0.0	0:30.89	rcuos/10
20	root	20	0	0	0	0	S	0.0	0.0	0:29.46	rcuos/11
21	root	20	0	0	0	0	S	0.0	0.0	0:28.01	rcuos/12
22	root	20	0	0	0	0	S	0.0	0.0	0:28.00	rcuos/13
23	root	20	0	0	0	0	S	0.0	0.0	0:30.02	rcuos/14
24	root	20	0	0	0	0	S	0.0	0.0	0:26.52	rcuos/15
25	root	20	0	0	0	0	S	0.0	0.0	0:02.40	rcuos/16
26	root	20	0	0	0	0	S	0.0	0.0	0:24.59	rcuos/17
27	root	20	0	0	0	0	S	0.0	0.0	1:01.50	rcuos/18
28	root	20	0	0	0	0	S	0.0	0.0	0:03.67	rcuos/19
29	root	20	0	0	0	0	S	0.0	0.0	0:03.27	rcuos/20
30	root	20	0	0	0	0	S	0.0	0.0	0:03.21	rcuos/21
31	root	20	0	0	0	0	S	0.0	0.0	0:03.36	rcuos/22

In order to sort them by memory usage **top -o %MEM**



```

top - 16:24:08 up 14 days, 4:48, 2 users, load average: 0.00, 0.01, 0.05
Tasks: 465 total, 1 running, 464 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 26411862+total, 14630520 used, 24948811+free, 672900 buffers
KiB Swap: 25009766+total, 0 used, 25009766+free, 6454132 cached Mem
  
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
34617	kibana	20	0	1352664	187584	13688	S	0.0	0.1	0:42.66	node
1855	root	20	0	3036868	51288	14880	S	0.0	0.0	30:25.89	dockerd
50199	mysqld	20	0	552144	51204	7056	S	0.0	0.0	3:45.60	mysqld
4759	lightdm	20	0	543792	37896	17540	S	0.0	0.0	2:54.70	lightdm-gtk-gre
2152	root	20	0	2822096	29924	4612	S	0.0	0.0	14:45.95	docker-containe
4541	root	20	0	352800	25160	15168	S	0.0	0.0	0:11.47	Xorg
2386	root	20	0	407652	21932	18596	S	0.0	0.0	0:58.91	sssd_nss
2137	root	20	0	439296	13320	8540	S	0.0	0.0	0:58.83	sssd_be
4742	root	20	0	201256	9404	3296	S	0.0	0.0	0:00.31	lightdm
1711	root	20	0	325496	9256	6984	S	0.0	0.0	0:01.59	smbd
2387	root	20	0	393648	9004	5804	S	0.0	0.0	0:33.69	sssd_pam
1922	root	20	0	413124	8668	5906	S	0.0	0.0	0:43.74	sssd
5445	redis	20	0	37396	7408	1088	S	0.3	0.0	77:53.57	redis-server
4793	lightdm	20	0	555880	7200	3980	S	0.0	0.0	0:00.01	indicator-sound
4570	root	20	0	282776	6552	3036	S	0.0	0.0	0:00.30	polkitd
40924	e14049	20	0	45308	6228	1824	S	0.0	0.0	0:00.04	bash
40592	e14049	20	0	45276	6196	1824	S	0.0	0.0	0:00.04	bash
5366	root	20	0	17104	5624	0	S	0.0	0.0	0:00.09	vmnet-dhcpd
4792	lightdm	20	0	283540	5340	2668	S	0.0	0.0	0:00.00	indicator-power
4762	lightdm	20	0	337404	5212	2608	S	0.0	0.0	0:00.00	at-spi-bus-laun
4777	lightdm	20	0	280140	5184	2540	S	0.0	0.0	0:00.00	gvfsd-fuse
40808	root	20	0	147784	4932	3696	S	0.0	0.0	0:00.01	sshd
4795	lightdm	20	0	286804	4860	3812	S	0.0	0.0	0:00.00	indicator-appli
4544	root	20	0	301672	4844	3424	S	0.0	0.0	1:06.58	accounts-daemon
40534	root	20	0	147784	4792	3548	S	0.0	0.0	0:00.01	sshd
40866	root	20	0	147784	4768	3536	S	0.0	0.0	0:00.00	sshd
2031	root	20	0	334456	4484	3308	S	0.0	0.0	0:00.02	ModemManager
4803	root	20	0	239356	4432	3476	S	0.0	0.0	0:00.22	upowerd
17608	root	20	0	4263112	4192	2920	S	0.0	0.0	0:01.06	console-kit-dae
1	root	20	0	36760	4092	1508	S	0.0	0.0	0:29.44	init
4494	root	20	0	278008	3904	2968	S	0.0	0.0	0:01.93	lightdm

ii). **ps-a**: select all processors except session leaders/get it in man

ps-x : process own by you

ps-u: Select by effective user ID (EUID) or name. This selects the processes who's effective User list.

ps-w: window size will be adjust.

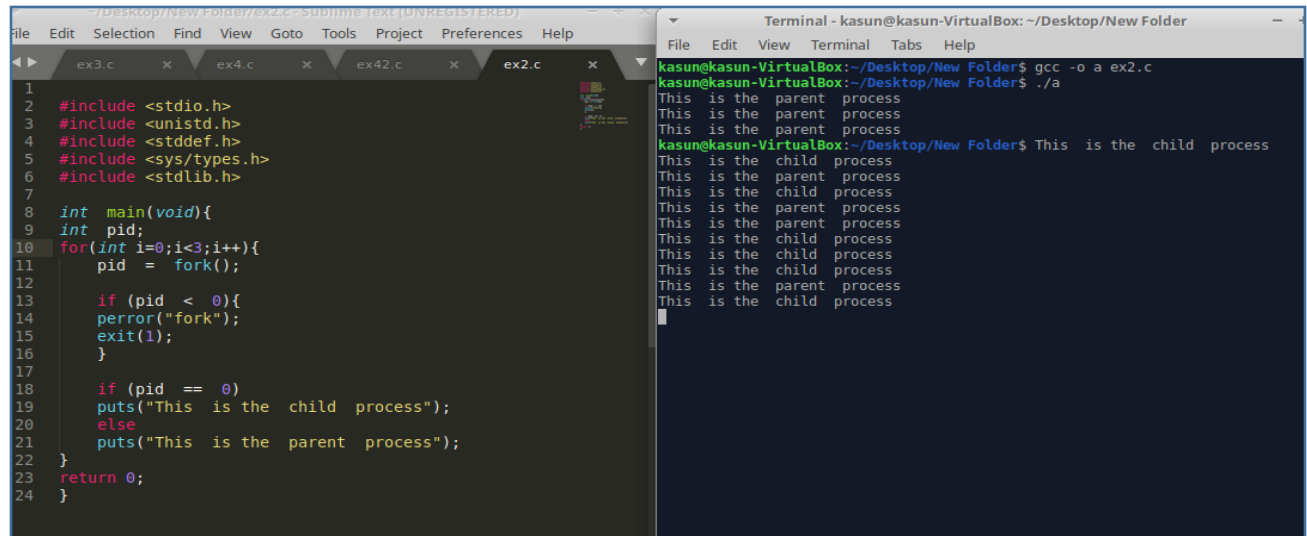
```
e14049@aiken:~$ ps -a
  PID TTY          TIME CMD
 40961 pts/1    00:00:00 ps
e14049@aiken:~$ ps -x
  PID TTY          STAT TIME   COMMAND
 40591 ?            S      0:00   sshd: e14049@pts/3
 40592 pts/3      Ss+    0:00   -bash
 40611 ?            Ss     0:00   /usr/lib/openssh/sftp-server
 40864 ?            S      0:00   sshd: e14049@pts/1
 40922 ?            S      0:00   sshd: e14049@notty
 40923 ?            Ss     0:00   /usr/lib/openssh/sftp-server
 40924 pts/1      Ss     0:00   -bash
 40962 pts/1      R+     0:00   ps -x
e14049@aiken:~$ ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
e14049    40592  0.0  0.0  45276  6196 pts/3    Ss+   16:01   0:00 -bash
e14049    40924  0.0  0.0  45308  6228 pts/1    Ss    16:21   0:00 -bash
e14049    40963  0.0  0.0  38796  1420 pts/1    R+    16:34   0:00 ps -u
e14049@aiken:~$ ps -w
  PID TTY          TIME CMD
 40924 pts/1    00:00:00 bash
 40967 pts/1    00:00:00 ps
e14049@aiken:~$
```

To find the PID =1 process “ **ps -p 1 -o comm=** ” or “**ps -eaf**” commands.

```
e14049@aiken:~$ ps -p 1 -o comm=
init
e14049@aiken:~$ clear
e14049@aiken:~$ ps -eaf
UID          PID    PPID  C STIME TTY          TIME CMD
root         1        0  0 0ct22 ?           00:00:29 /sbin/init
root         2        0  0 0ct22 ?           00:00:00 [kthreadd]
root         3        2  0 0ct22 ?           00:01:01 [ksoftirqd/0]
root         5        2  0 0ct22 ?           00:00:00 [kworker/0:0H]
root         6        2  0 0ct22 ?           00:01:00 [kworker/u128:0]
root         8        2  0 0ct22 ?           00:10:09 [rcu_sched]
root         9        2  0 0ct22 ?           00:00:34 [rcuos/0]
root        10        2  0 0ct22 ?           00:00:30 [rcuos/1]
root        11        2  0 0ct22 ?           00:00:31 [rcuos/2]
root        12        2  0 0ct22 ?           00:00:39 [rcuos/3]
root        13        2  0 0ct22 ?           00:00:29 [rcuos/4]
root        14        2  0 0ct22 ?           00:00:26 [rcuos/5]
root        15        2  0 0ct22 ?           00:00:31 [rcuos/6]
root        16        2  0 0ct22 ?           00:00:57 [rcuos/7]
root        17        2  0 0ct22 ?           00:00:28 [rcuos/8]
root        18        2  0 0ct22 ?           00:00:28 [rcuos/9]
root        19        2  0 0ct22 ?           00:00:30 [rcuos/10]
root        20        2  0 0ct22 ?           00:00:29 [rcuos/11]
root        21        2  0 0ct22 ?           00:00:28 [rcuos/12]
root        22        2  0 0ct22 ?           00:00:28 [rcuos/13]
root        23        2  0 0ct22 ?           00:00:30 [rcuos/14]
root        24        2  0 0ct22 ?           00:00:26 [rcuos/15]
root        25        2  0 0ct22 ?           00:00:02 [rcuos/16]
root        26        2  0 0ct22 ?           00:00:24 [rcuos/17]
root        27        2  0 0ct22 ?           00:01:01 [rcuos/18]
root        28        2  0 0ct22 ?           00:00:03 [rcuos/19]
root        29        2  0 0ct22 ?           00:00:03 [rcuos/20]
root        30        2  0 0ct22 ?           00:00:03 [rcuos/21]
root        31        2  0 0ct22 ?           00:00:03 [rcuos/22]
```

EXERCISE 02 :

1).parent process is the one who creating the child processes. Printed order may be different with the CPU scheduling criteria. But most of the time parents will print before their respective Childs.



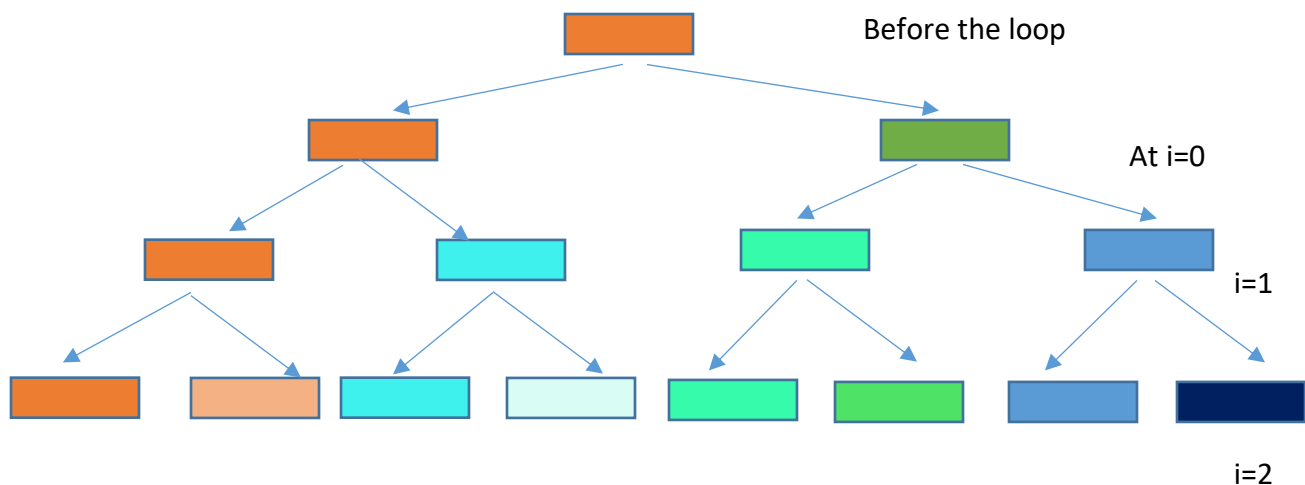
The screenshot shows a code editor on the left with a C program and a terminal on the right showing its output. The C program is as follows:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6 #include <unistd.h>
7
8 int main(void){
9     int pid;
10    for(int i=0;i<3;i++){
11        pid = fork();
12
13        if (pid < 0){
14            perror("fork");
15            exit(1);
16        }
17
18        if (pid == 0)
19            puts("This is the child process");
20        else
21            puts("This is the parent process");
22    }
23    return 0;
24 }
```

The terminal output shows the execution of the program, where the parent process prints "This is the parent process" and the child processes print "This is the child process". The output is as follows:

```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ gcc -o a ex2.c
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
This is the parent process
This is the parent process
This is the parent process
kasun@kasun-VirtualBox:~/Desktop/New Folder$ This is the child process
This is the child process
This is the parent process
This is the child process
This is the parent process
This is the child process
This is the child process
This is the parent process
This is the child process
This is the parent process
This is the child process
```

2).when you have multiple fork(); in a loops there will be 7 children's finally with the parent. All together there will be 8 processes.



As you can see in the figure there will be 8 process will be create in the loop. One parent with 7 children's. Using "getppid();" we can get the parent pid but "getcpid()" in not available. Because when fork(); execute it return the pid of the child to the parent. Therefore no need to have a function called "getcpid()".

wait() and waitpid()

The `wait()` system call suspends execution of the calling process until one of its children terminates. The call `wait(&wstatus)` is equivalent to `waitpid(-1, &wstatus, 0)`;

The value of pid can be:

- < -1 meaning wait for a child process whose process group ID is equal to the absolute value of pid.
- 1 meaning wait for any child process.
- 0 meaning wait for any child process whose process group ID is equal to that of the calling process.
- > 0 meaning wait for the child whose process ID is equal to the value of pid.

```

Edit Selection Find View Goto Tools Project Preferences Help
ex3.c x ex4.c x ex42.c x ex2.c x
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
#include <stddef.h>
#include <sys/types.h>
#include <stdlib.h>

int main(void){
int pid,wstatus;
for(int i=0;i<3;i++){
    pid = fork();

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

    if (pid == 0)
        puts("This is the child process");
    else{
        waitpid(-1, &wstatus, 0);
        puts("This is the parent process");
    }
}
return 0;
}

Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
File Edit View Terminal Tabs Help
kasun@kasun-VirtualBox:~/Desktop/New Folder$ gcc -o a ex3.c
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
This is the child process
This is the child process
This is the child process
This is the parent process
This is the parent process
This is the child process
This is the parent process
This is the parent process
This is the child process
This is the child process
This is the parent process
This is the parent process
This is the child process
This is the child process
This is the parent process
kasun@kasun-VirtualBox:~/Desktop/New Folder$
```

As you can see in the output when we use the “waitpid(-1, &wstatus, 0);” it wait for child process. After that the parent process will execute.

EXERCISE 04 :

i). The `exec()` family of functions replaces the current process image with a new process image. It loads the program into the current process space and runs it from the entry point. Therefore nothing will be printed after executing the `execl()`.

The screenshot shows a code editor with a C program and a terminal window. The C program in the editor is as follows:

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(char argc, char *argv[]){
    execl("/bin/ls", "-", argv[1], NULL);
    puts("Program ls has terminated");
    return 0;
}
```

The terminal window shows the compilation and execution of the program:

```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ gcc -o a ex4.c
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
a
ex2.c ex3.png ex4.c fork1.png
ex22.c ex3.c ex42.c ex4.png forkloop.png
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a -n
total 544
-rwxrwxr-x 1 1000 1000 7212 2020-06-15 15:19 a
-rw-rw-r-- 1 1000 1000 346 2020-06-11 15:59 ex22.c
-rw-rw-r-- 1 1000 1000 302 2020-06-10 10:56 ex2.c
-rw-rw-r-- 1 1000 1000 442 2020-06-12 12:30 ex3.c
-rw-rw-r-- 1 1000 1000 129109 2020-06-12 12:28 ex3.png
-rw-rw-r-- 1 1000 1000 543 2020-06-15 03:15 ex42.c
-rw-rw-r-- 1 1000 1000 278 2020-06-13 13:43 ex4.c
-rw-rw-r-- 1 1000 1000 203077 2020-06-13 13:30 ex4.png
-rw-rw-r-- 1 1000 1000 79143 2020-06-10 10:59 fork1.png
-rw-rw-r-- 1 1000 1000 109361 2020-06-12 12:00 forkloop.png
kasun@kasun-VirtualBox:~/Desktop/New Folder$
```

ii).

The screenshot shows a code editor with a C program and a terminal window. The C program in the editor is as follows:

```
#include <stdlib.h>

int main(char argc, char *argv[]){
    int pid, wstatus;
    char string[30];

    pid = fork();

    if (pid == 0){
        puts("This is the child process");
        scanf("%s", string);
        execl("/bin/ls", "-", string, NULL);
    }
    else{
        waitpid(-1, &wstatus, 0);
        puts("This is the parent process\n");
        execl("./a", "-", string, NULL);
    }

    return 0;
}
```

The terminal window shows the execution of the program:

```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
This is the child process
./
a
ex2.c ex3.png ex4.c fork1.png
ex22.c ex3.c ex42.c ex4.png forkloop.png
This is the parent process

This is the child process
-d
.
This is the parent process

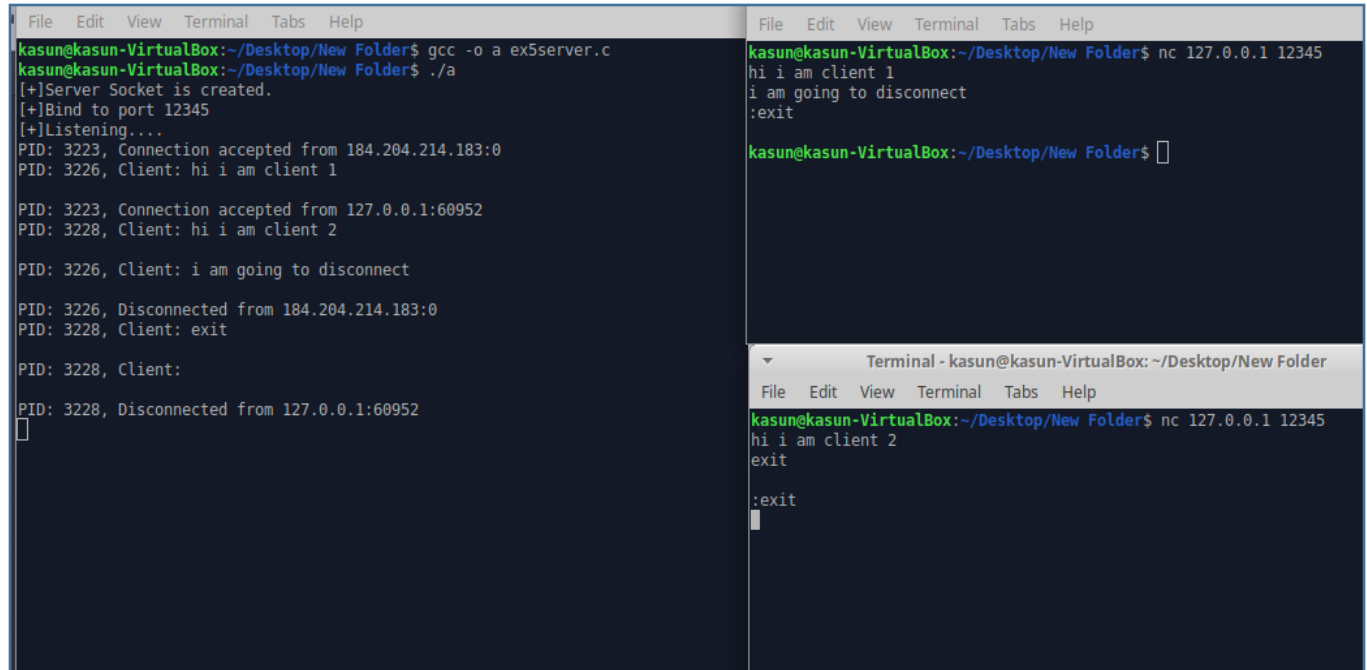
This is the child process
-n
total 544
-rwxrwxr-x 1 1000 1000 7404 2020-06-15 03:15 a
-rw-rw-r-- 1 1000 1000 346 2020-06-11 15:59 ex22.c
-rw-rw-r-- 1 1000 1000 302 2020-06-10 10:56 ex2.c
-rw-rw-r-- 1 1000 1000 442 2020-06-12 12:30 ex3.c
-rw-rw-r-- 1 1000 1000 129109 2020-06-12 12:28 ex3.png
-rw-rw-r-- 1 1000 1000 543 2020-06-15 03:15 ex42.c
-rw-rw-r-- 1 1000 1000 278 2020-06-13 13:43 ex4.c
-rw-rw-r-- 1 1000 1000 203077 2020-06-13 13:30 ex4.png
-rw-rw-r-- 1 1000 1000 79143 2020-06-10 10:59 fork1.png
-rw-rw-r-- 1 1000 1000 109361 2020-06-12 12:00 forkloop.png
This is the parent process

This is the child process

```

EXERCISE 05 :

i).



```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ gcc -o a ex5server.c
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
[+]Server Socket is created.
[+]Bind to port 12345
[+]Listening....
PID: 3223, Connection accepted from 184.204.214.183:0
PID: 3226, Client: hi i am client 1

PID: 3223, Connection accepted from 127.0.0.1:60952
PID: 3228, Client: hi i am client 2

PID: 3226, Client: i am going to disconnect

PID: 3226, Disconnected from 184.204.214.183:0
PID: 3228, Client: exit

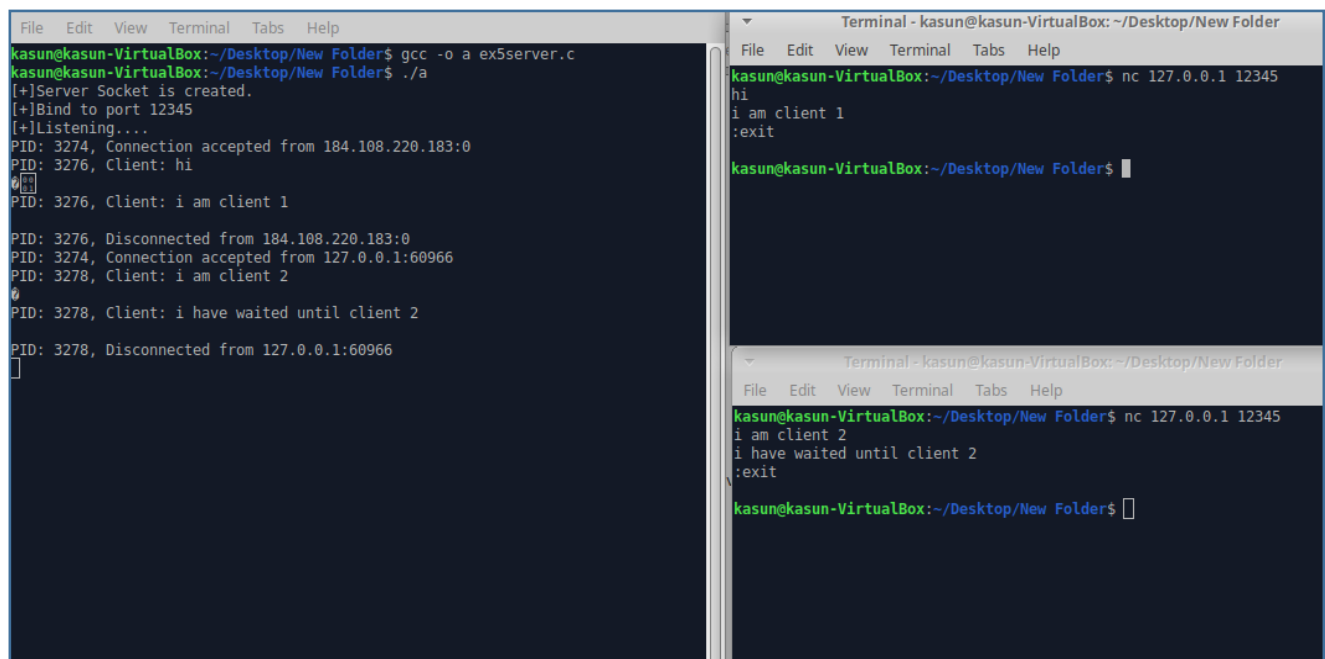
PID: 3228, Client:

PID: 3228, Disconnected from 127.0.0.1:60952
█
```

```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
hi i am client 1
i am going to disconnect
:exit

kasun@kasun-VirtualBox:~/Desktop/New Folder$ █
```

ii). Since the parent process wait until the child process finished then the server only able to handle one client at a time. Other clients has to wait until the serving client finishes.



```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ gcc -o a ex5server.c
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
[+]Server Socket is created.
[+]Bind to port 12345
[+]Listening....
PID: 3274, Connection accepted from 184.108.220.183:0
PID: 3276, Client: hi
0
PID: 3276, Client: i am client 1

PID: 3276, Disconnected from 184.108.220.183:0
PID: 3274, Connection accepted from 127.0.0.1:60966
PID: 3278, Client: i am client 2
0
PID: 3278, Client: i have waited until client 2

PID: 3278, Disconnected from 127.0.0.1:60966
█
```

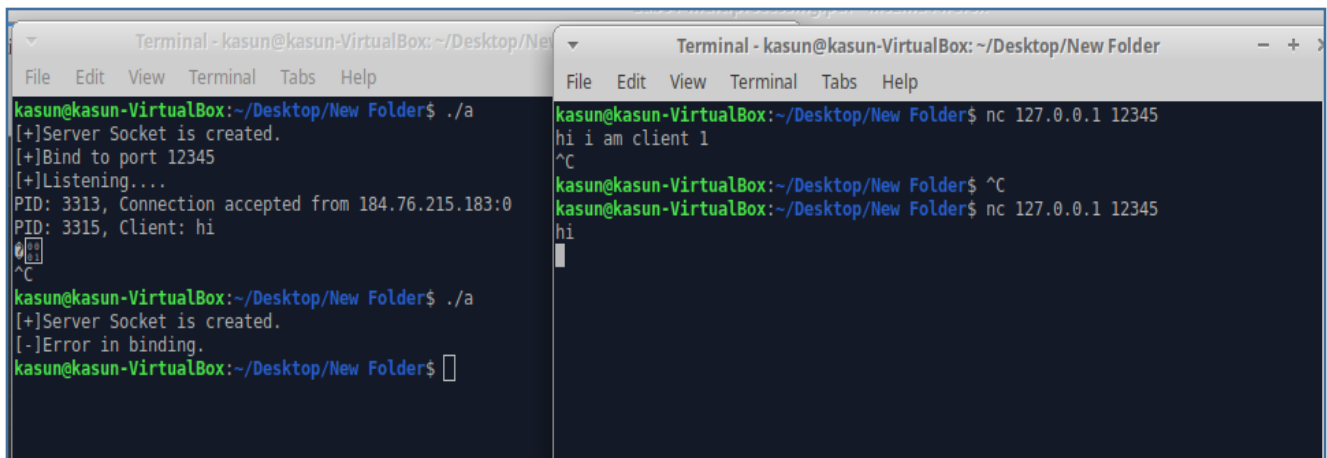
```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
hi
i am client 1
:exit

kasun@kasun-VirtualBox:~/Desktop/New Folder$ █
```

```
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
i am client 2
i have waited until client 2
:exit

kasun@kasun-VirtualBox:~/Desktop/New Folder$ █
```

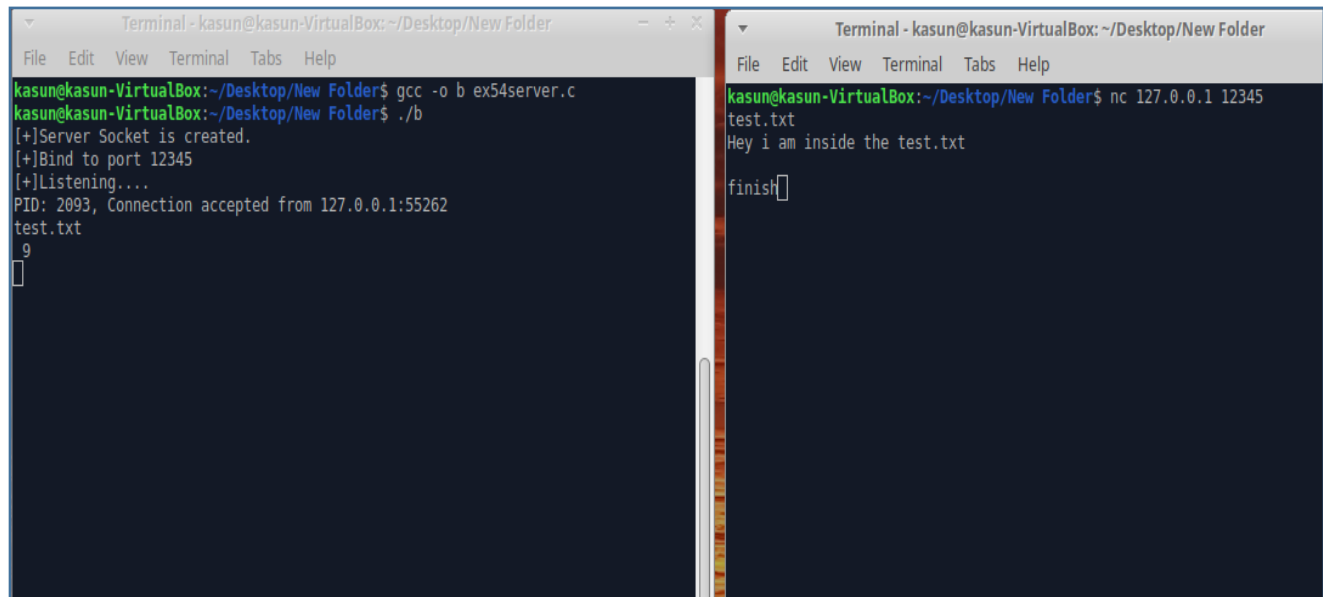
iii). When you terminate the server while a client is connected, and then try to restart it the port will be busy. Because the client is using that port.



```
Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
[+]Server Socket is created.
[+]Bind to port 12345
[+]Listening....
PID: 3313, Connection accepted from 184.76.215.183:0
PID: 3315, Client: hi
^C
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./a
[+]Server Socket is created.
[-]Error in binding.
kasun@kasun-VirtualBox:~/Desktop/New Folder$

Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
hi i am client 1
^C
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
hi
```

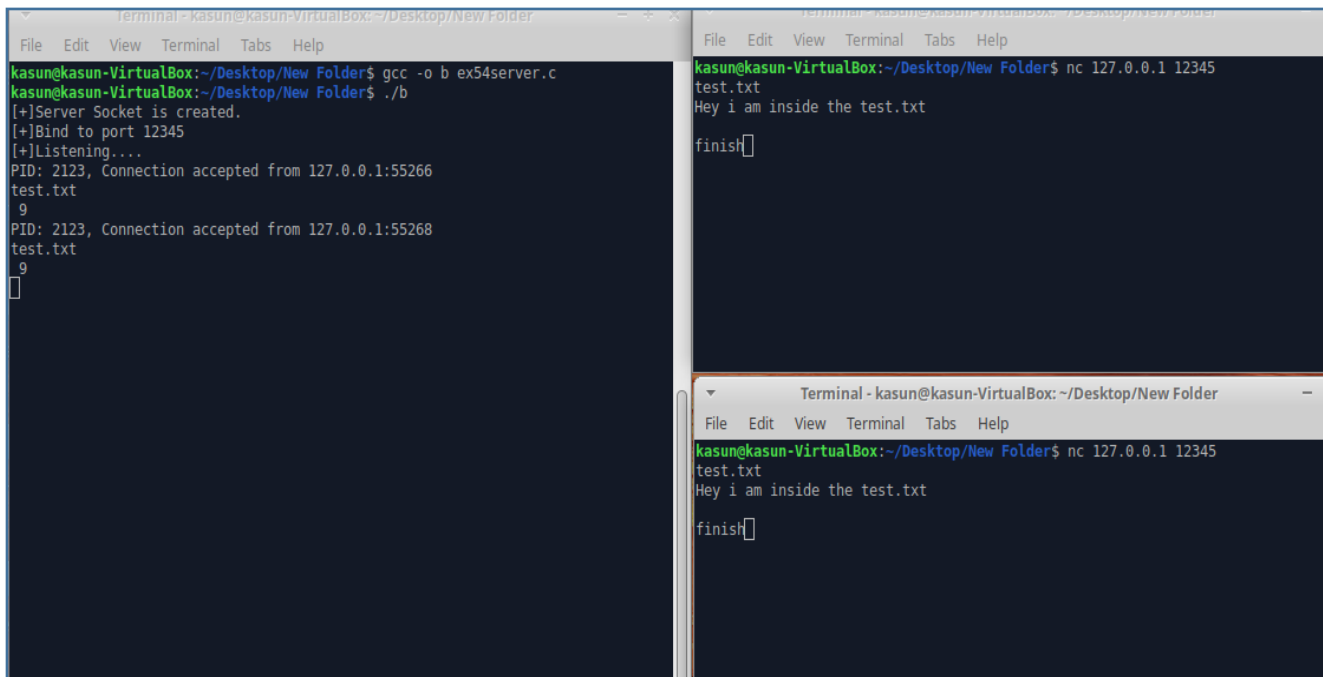
iv).



```
Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
kasun@kasun-VirtualBox:~/Desktop/New Folder$ gcc -o b ex54server.c
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./b
[+]Server Socket is created.
[+]Bind to port 12345
[+]Listening....
PID: 2093, Connection accepted from 127.0.0.1:55262
test.txt
9
[]

Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
test.txt
Hey i am inside the test.txt
finish[]
```

It is possible to two client to access the same file concurrently.



The screenshot shows three terminal windows in a VirtualBox environment. The leftmost window runs a C program (ex54server.c) that acts as a server, listening on port 12345. It receives two concurrent connections from 127.0.0.1. The middle and right windows are Netcat (nc) clients connected to the same host and port. Both clients send the content of 'test.txt' (which is 'Hey i am inside the test.txt') to the server and then type 'finish'.

```
Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
File Edit View Terminal Tabs Help
kasun@kasun-VirtualBox:~/Desktop/New Folder$ gcc -o b ex54server.c
kasun@kasun-VirtualBox:~/Desktop/New Folder$ ./b
[+]Server Socket is created.
[+]Bind to port 12345
[+]Listening....
PID: 2123, Connection accepted from 127.0.0.1:55266
test.txt
9
PID: 2123, Connection accepted from 127.0.0.1:55268
test.txt
9

```

```
Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
File Edit View Terminal Tabs Help
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
test.txt
Hey i am inside the test.txt
finish
```

```
Terminal - kasun@kasun-VirtualBox: ~/Desktop/New Folder
File Edit View Terminal Tabs Help
kasun@kasun-VirtualBox:~/Desktop/New Folder$ nc 127.0.0.1 12345
test.txt
Hey i am inside the test.txt
finish
```