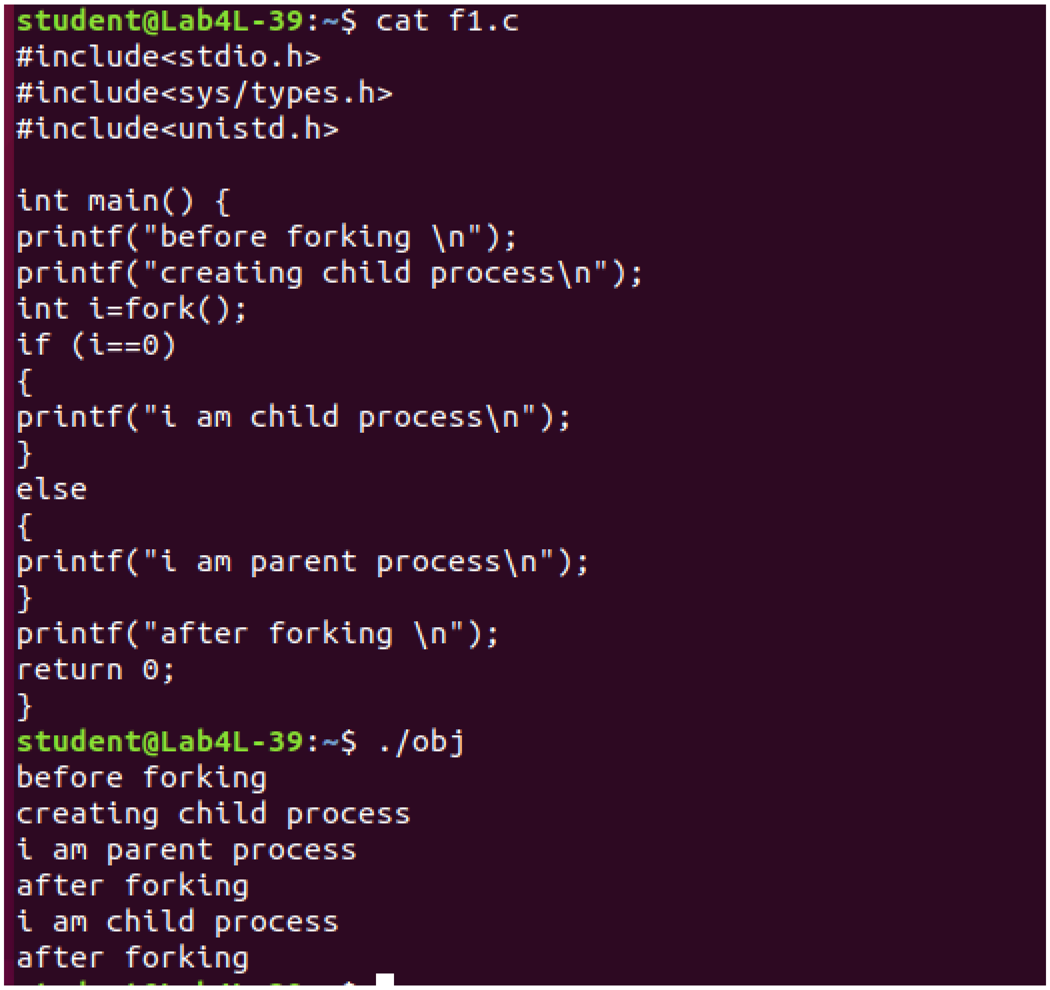
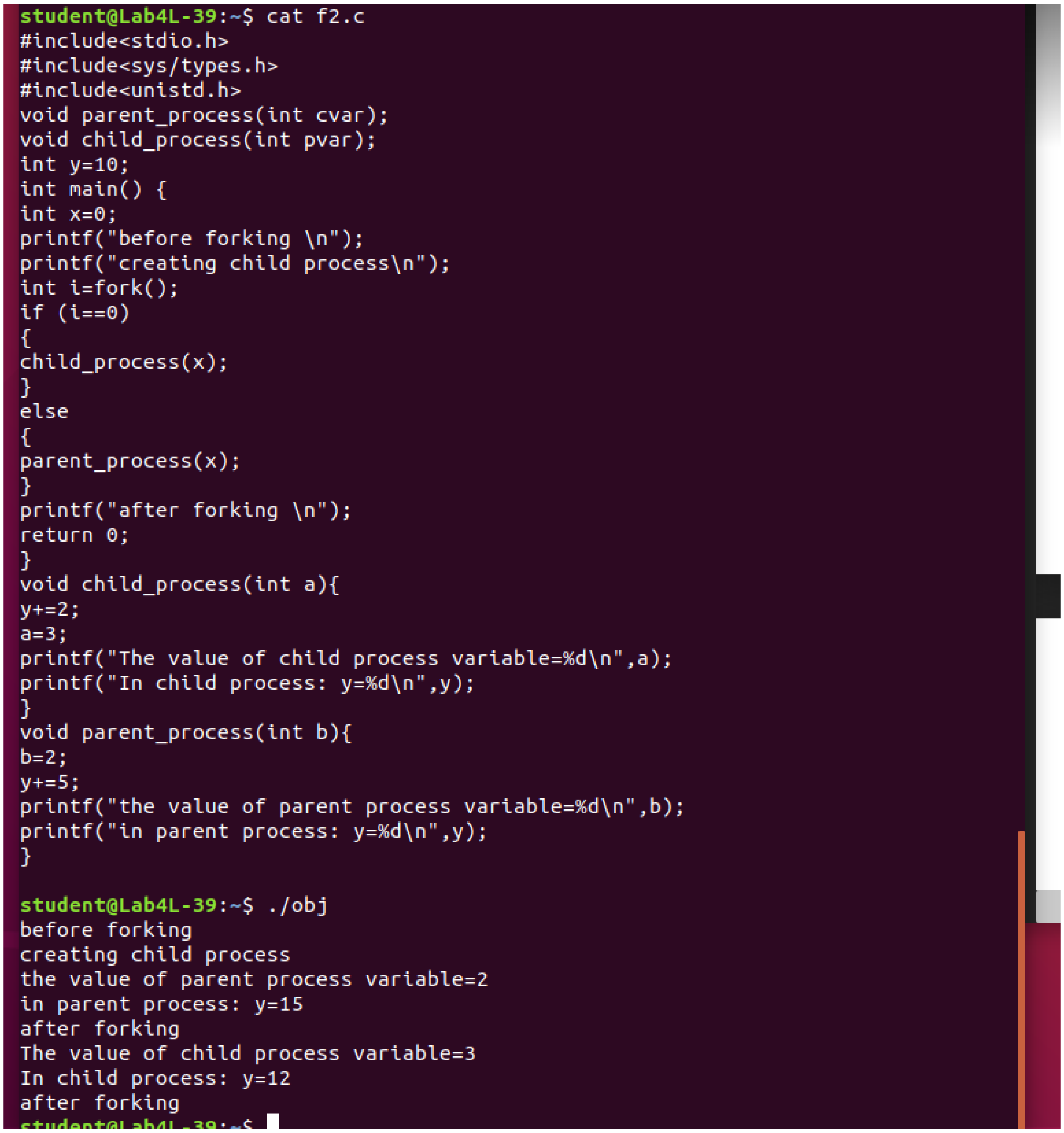
MUHAMMAD AYAZ HASAN
20K-1044
BS-SE (5A)

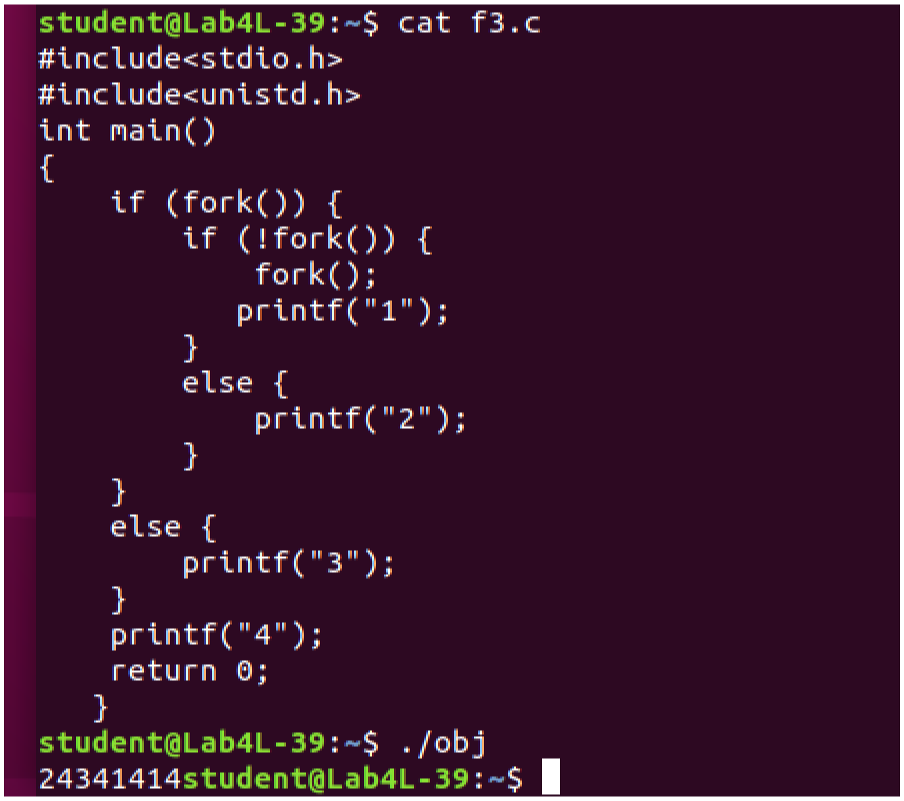
Lab 6

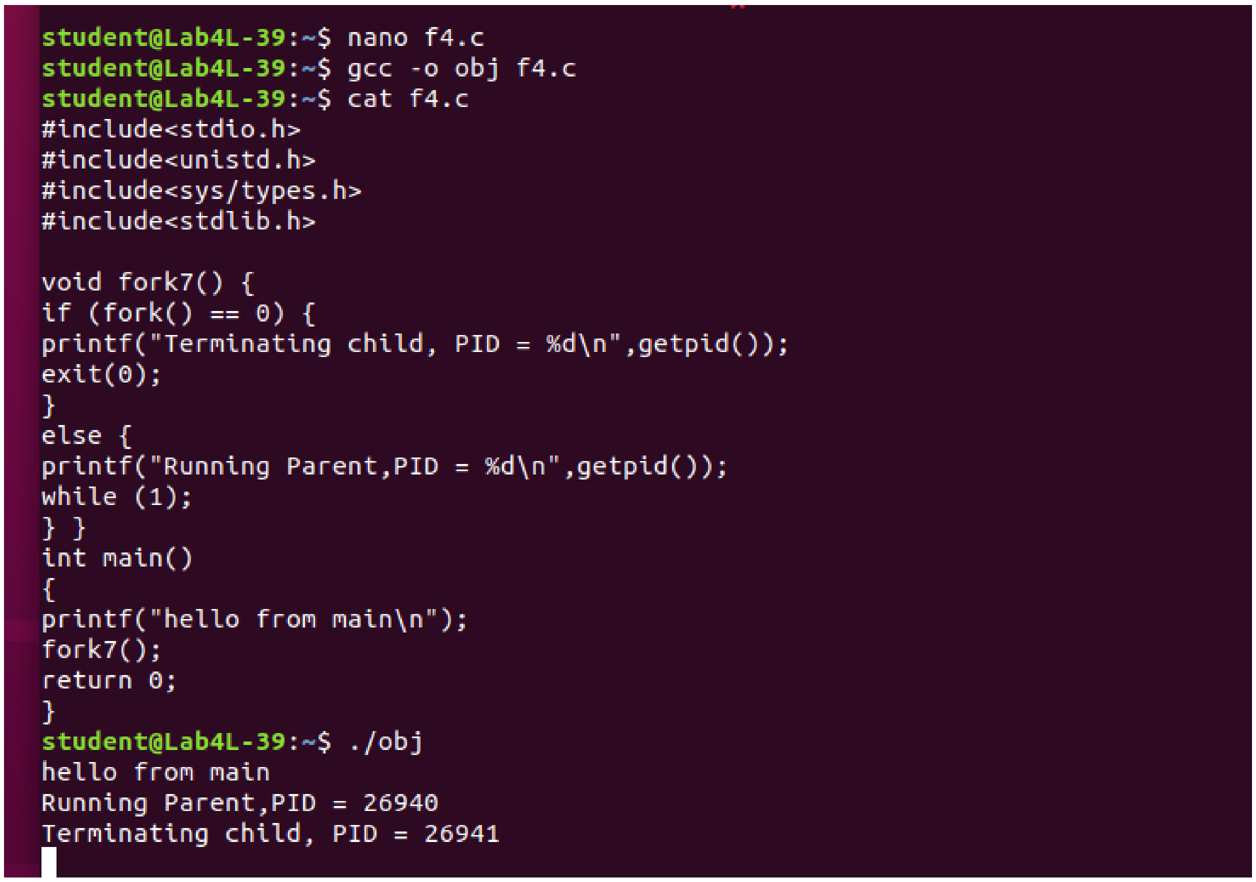


```
student@Lab4L-39:~$ cat f1.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main() {
printf("before forking \n");
printf("creating child process\n");
int i=fork();
if (i==0)
{
printf("i am child process\n");
}
else
{
printf("i am parent process\n");
}
printf("after forking \n");
return 0;
}
student@Lab4L-39:~$ ./obj
before forking
creating child process
i am parent process
after forking
i am child process
after forking
```

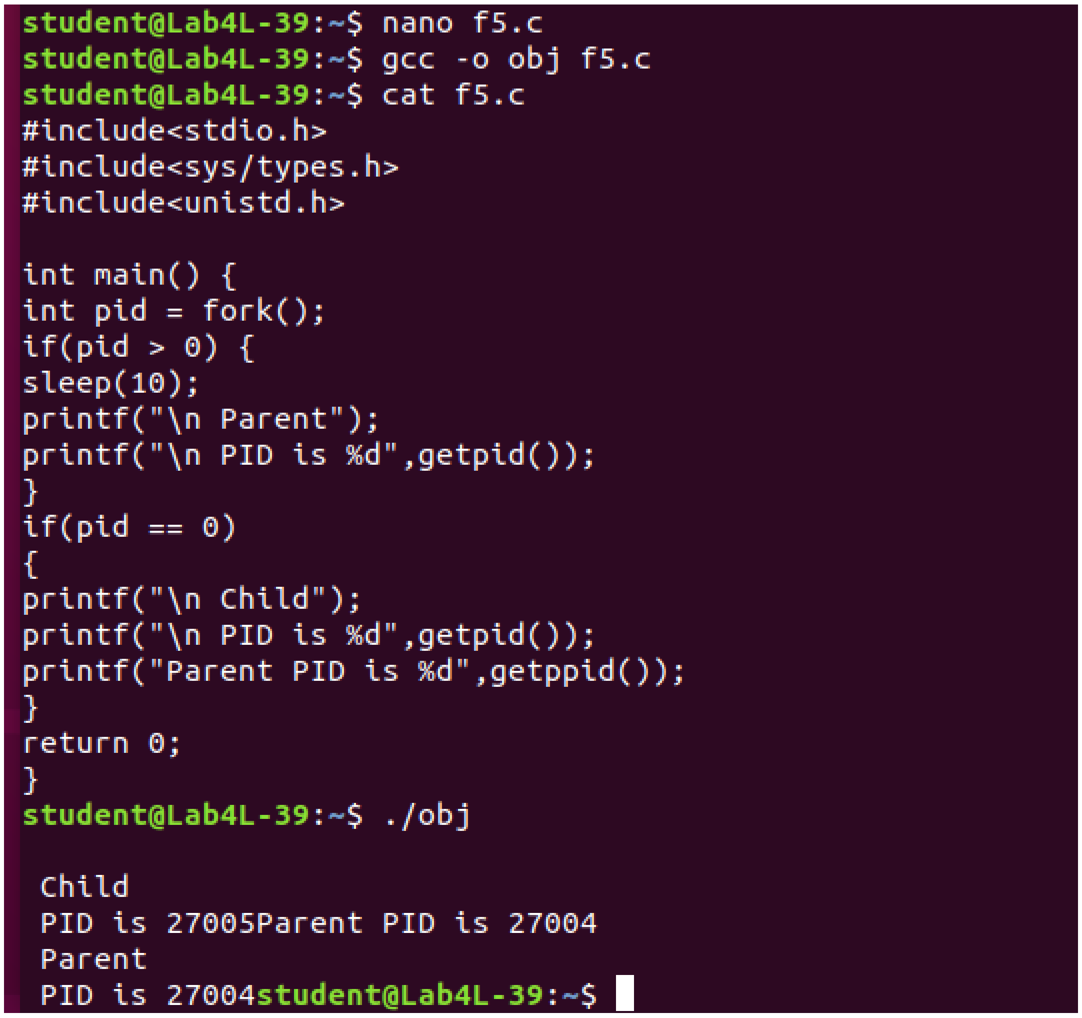


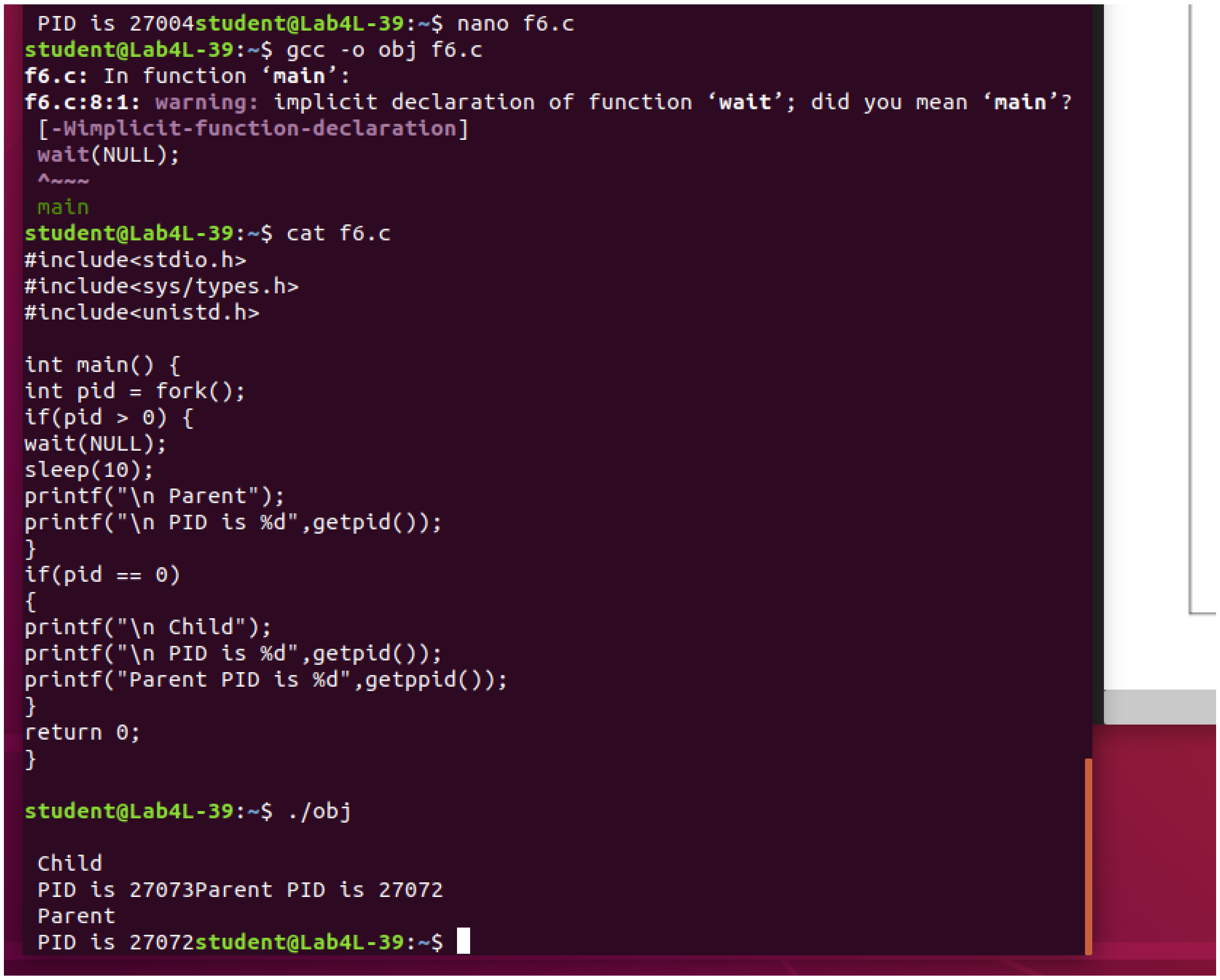
```
student@Lab4L-39:~$ cat f2.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
void parent_process(int cvar);
void child_process(int pvar);
int y=10;
int main() {
int x=0;
printf("before forking \n");
printf("creating child process\n");
int i=fork();
if (i==0)
{
child_process(x);
}
else
{
parent_process(x);
}
printf("after forking \n");
return 0;
}
void child_process(int a){
y+=2;
a=3;
printf("The value of child process variable=%d\n",a);
printf("In child process: y=%d\n",y);
}
void parent_process(int b){
b=2;
y+=5;
printf("the value of parent process variable=%d\n",b);
printf("in parent process: y=%d\n",y);
}

student@Lab4L-39:~$ ./obj
before forking
creating child process
the value of parent process variable=2
in parent process: y=15
after forking
The value of child process variable=3
In child process: y=12
after forking
student@Lab4L-39:~$
```


```
student@Lab4L-39:~$ cat f3.c
#include<stdio.h>
#include<unistd.h>
int main()
{
    if (fork()) {
        if (!fork()) {
            fork();
           printf("1");
        }
        else {
            printf("2");
        }
    }
    else {
        printf("3");
    }
    printf("4");
    return 0;
   }
student@Lab4L-39:~$ ./obj
24341414student@Lab4L-39:~$
```

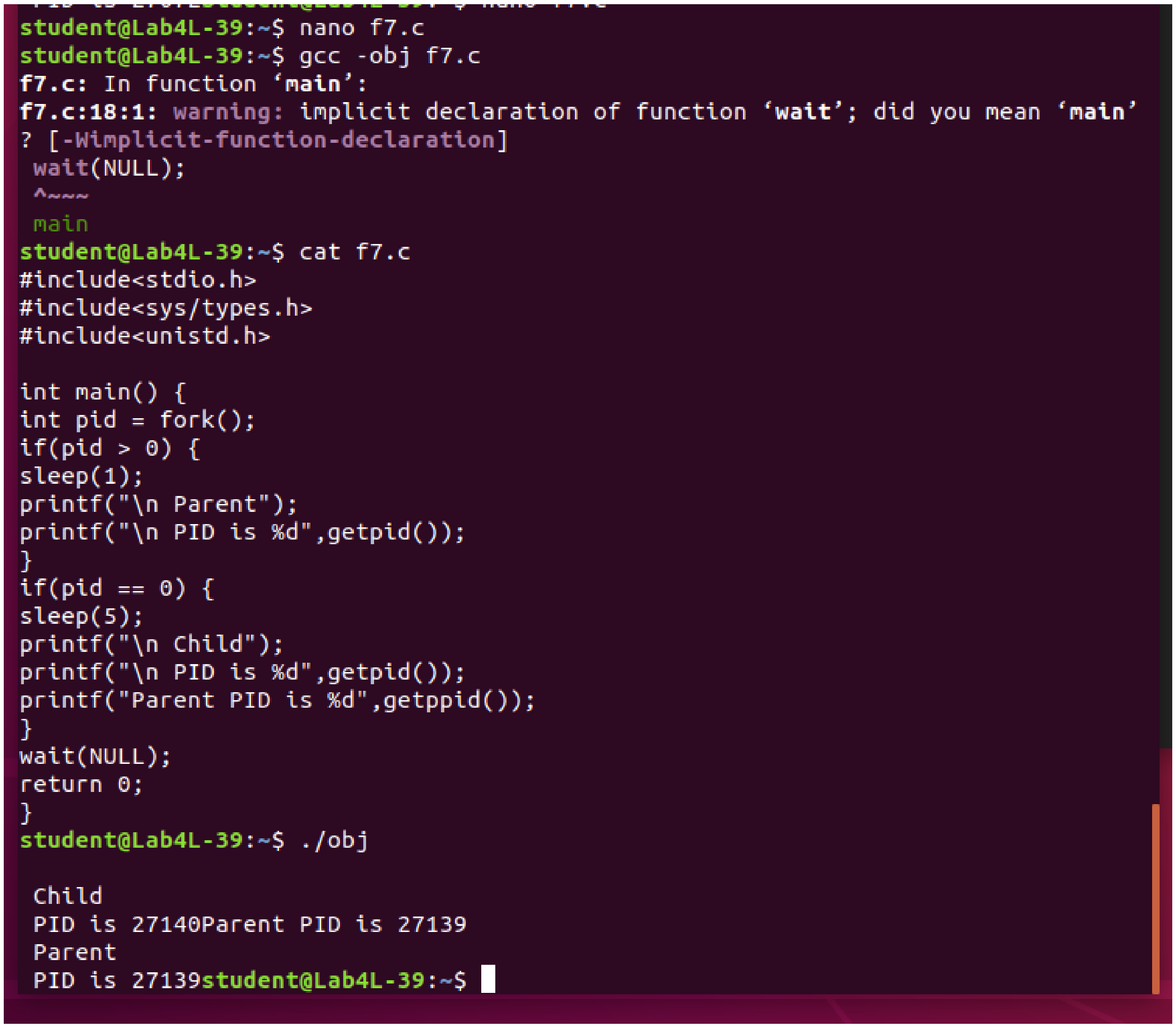


```
student@Lab4L-39:~$ nano f4.c
student@Lab4L-39:~$ gcc -o obj f4.c
student@Lab4L-39:~$ cat f4.c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<stdlib.h>

void fork7() {
if (fork() == 0) {
printf("Terminating child, PID = %d\n",getpid());
exit(0);
}
else {
printf("Running Parent,PID = %d\n",getpid());
while (1);
} }
int main()
{
printf("hello from main\n");
fork7();
return 0;
}
student@Lab4L-39:~$ ./obj
hello from main
Running Parent,PID = 26940
Terminating child, PID = 26941
```

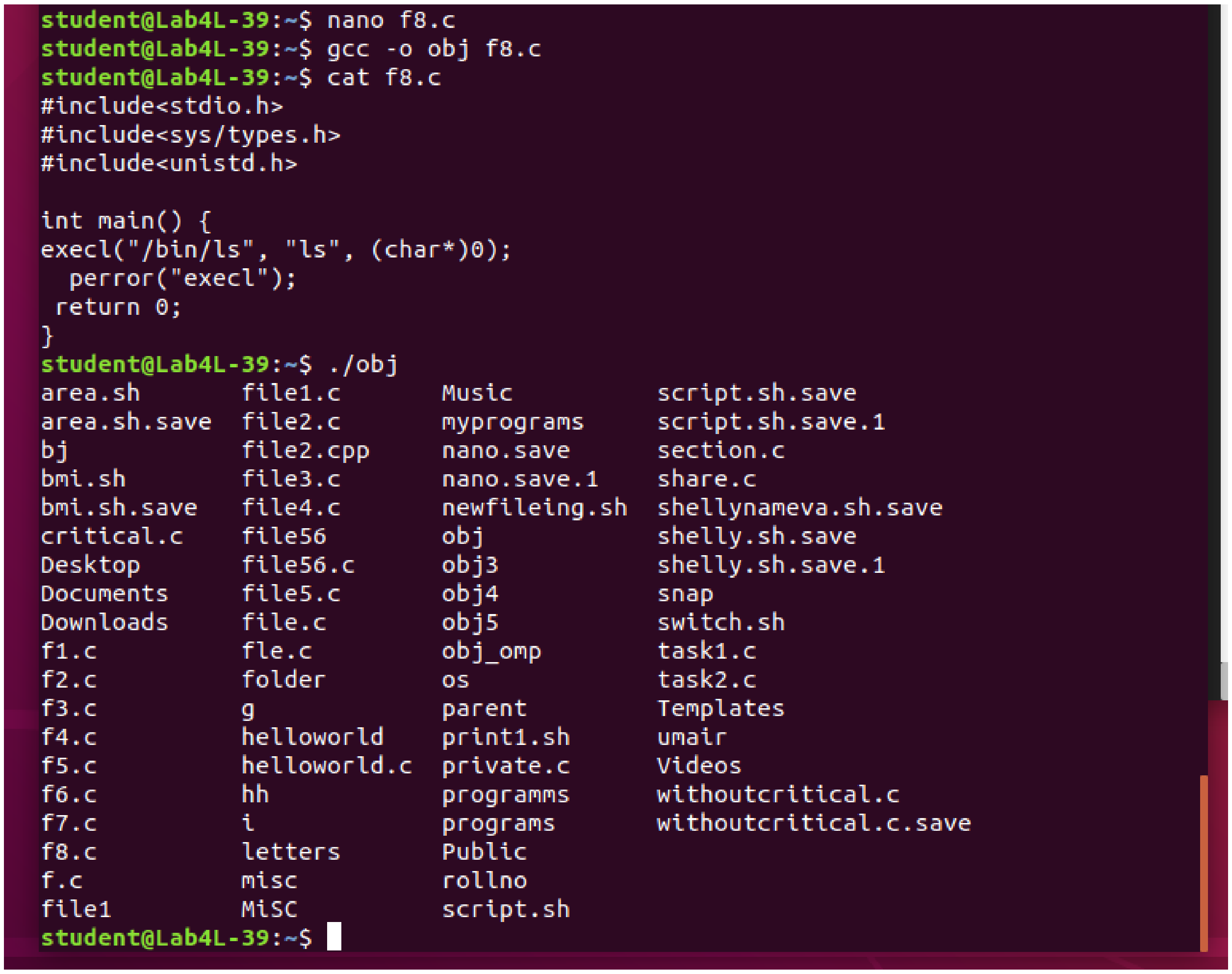

```
student@Lab4L-39:~$ nano f5.c
student@Lab4L-39:~$ gcc -o obj f5.c
student@Lab4L-39:~$ cat f5.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main() {
int pid = fork();
if(pid > 0) {
sleep(10);
printf("\n Parent");
printf("\n PID is %d",getpid());
}
if(pid == 0)
{
printf("\n Child");
printf("\n PID is %d",getpid());
printf("Parent PID is %d",getppid());
}
return 0;
}
student@Lab4L-39:~$ ./obj

 Child
 PID is 27005Parent PID is 27004
 Parent
 PID is 27004student@Lab4L-39:~$ 
```



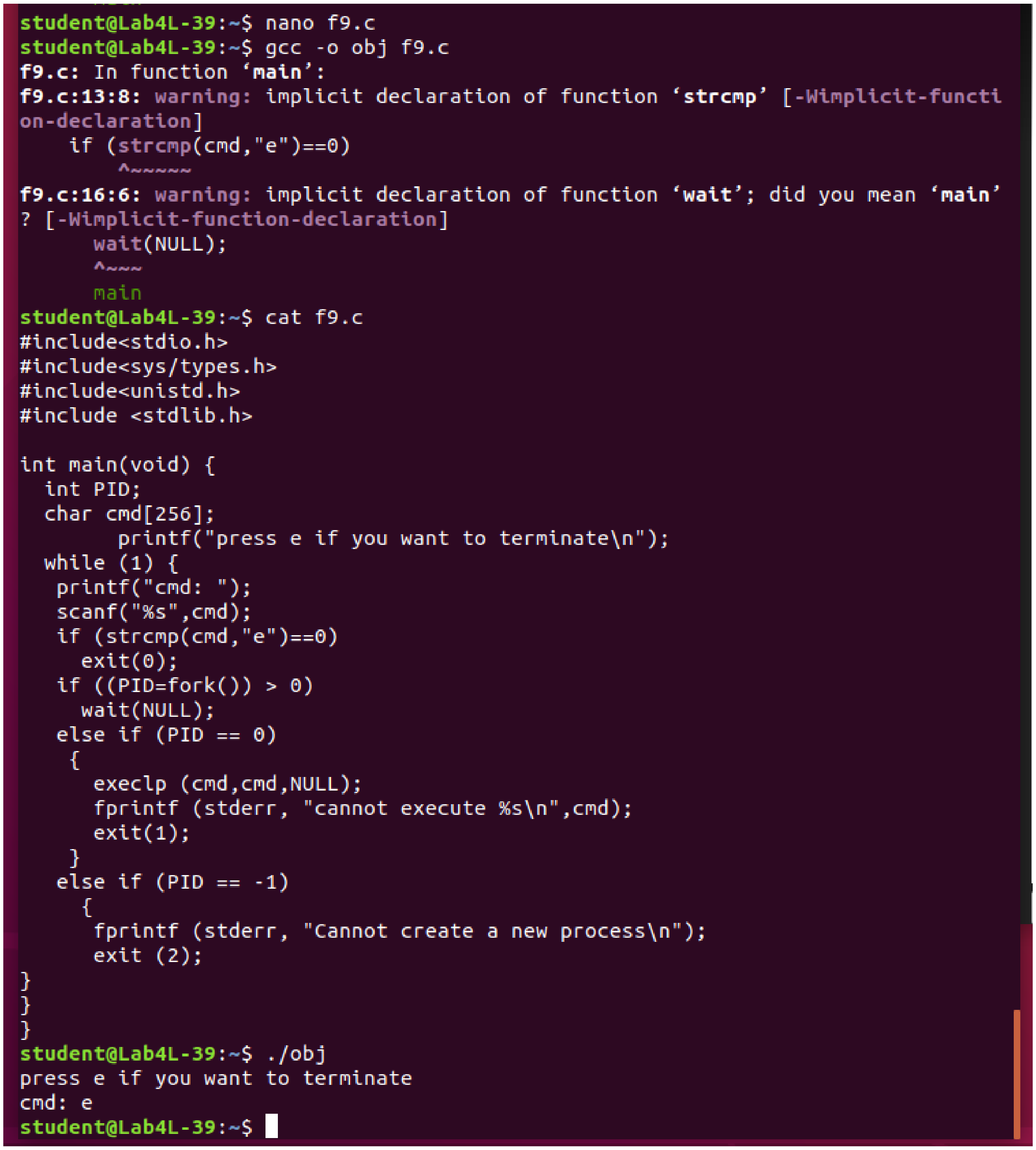
```
 PID is 27004student@Lab4L-39:~$ nano f6.c
student@Lab4L-39:~$ gcc -o obj f6.c
f6.c: In function ‘main’:
f6.c:8:1: warning: implicit declaration of function ‘wait’; did you mean ‘main’?
 [-Wimplicit-function-declaration]
 wait(NULL);
 ^~~~
 main
student@Lab4L-39:~$ cat f6.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main() {
int pid = fork();
if(pid > 0) {
wait(NULL);
sleep(10);
printf("\n Parent");
printf("\n PID is %d",getpid());
}
if(pid == 0)
{
printf("\n Child");
printf("\n PID is %d",getpid());
printf("Parent PID is %d",getppid());
}
return 0;
}

student@Lab4L-39:~$ ./obj

 Child
 PID is 27073Parent PID is 27072
 Parent
 PID is 27072student@Lab4L-39:~$
```



```
student@Lab4L-39:~$ nano f7.c
student@Lab4L-39:~$ gcc -obj f7.c
f7.c: In function ‘main’:
f7.c:18:1: warning: implicit declaration of function ‘wait’; did you mean ‘main’
? [-Wimplicit-function-declaration]
 wait(NULL);
 ^~~~
 main
student@Lab4L-39:~$ cat f7.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main() {
int pid = fork();
if(pid > 0) {
sleep(1);
printf("\n Parent");
printf("\n PID is %d",getpid());
}
if(pid == 0) {
sleep(5);
printf("\n Child");
printf("\n PID is %d",getpid());
printf("Parent PID is %d",getppid());
}
wait(NULL);
return 0;
}
student@Lab4L-39:~$ ./obj

 Child
 PID is 27140Parent PID is 27139
 Parent
 PID is 27139student@Lab4L-39:~$
```



```
student@Lab4L-39:~$ nano f8.c
student@Lab4L-39:~$ gcc -o obj f8.c
student@Lab4L-39:~$ cat f8.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main() {
execl("/bin/ls", "ls", (char*)0);
  perror("execl");
 return 0;
}
student@Lab4L-39:~$ ./obj
area.sh       file1.c       Music         script.sh.save
area.sh.save  file2.c       myprograms    script.sh.save.1
bj            file2.cpp     nano.save     section.c
bmi.sh        file3.c       nano.save.1   share.c
bmi.sh.save   file4.c       newfileing.sh shellynameva.sh.save
critical.c    file56        obj           shelly.sh.save
Desktop       file56.c      obj3          shelly.sh.save.1
Documents     file5.c       obj4          snap
Downloads     file.c        obj5          switch.sh
f1.c          fle.c         obj_omp       task1.c
f2.c          folder        os            task2.c
f3.c          g             parent        Templates
f4.c          helloworld    print1.sh     umair
f5.c          helloworld.c  private.c     Videos
f6.c          hh            programms     withoutcritical.c
f7.c          i             programs      withoutcritical.c.save
f8.c          letters       Public
f.c           misc          rollno
file1         MiSC          script.sh
student@Lab4L-39:~$ 
```



```
student@Lab4L-39:~$ nano f9.c
student@Lab4L-39:~$ gcc -o obj f9.c
f9.c: In function ‘main’:
f9.c:13:8: warning: implicit declaration of function ‘strcmp’ [-Wimplicit-functi
on-declaration]
    if (strcmp(cmd,"e")==0)
        ^~~~~~
f9.c:16:6: warning: implicit declaration of function ‘wait’; did you mean ‘main’
? [-Wimplicit-function-declaration]
      wait(NULL);
      ^~~~
      main
student@Lab4L-39:~$ cat f9.c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include <stdlib.h>

int main(void) {
  int PID;
  char cmd[256];
        printf("press e if you want to terminate\n");
  while (1) {
   printf("cmd: ");
   scanf("%s",cmd);
   if (strcmp(cmd,"e")==0)
     exit(0);
   if ((PID=fork()) > 0)
     wait(NULL);
   else if (PID == 0)
    {
      execlp (cmd,cmd,NULL);
      fprintf (stderr, "cannot execute %s\n",cmd);
      exit(1);
    }
   else if (PID == -1)
     {
      fprintf (stderr, "Cannot create a new process\n");
      exit (2);
}
}
}
student@Lab4L-39:~$ ./obj
press e if you want to terminate
cmd: e
student@Lab4L-39:~$ 
```