

- : OS-LAB-3 :-

1. write a c program to block a parent process untill the child completes using a wait system call.

pgm1.c

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

int main()
{
    int status;
    pid_t newPid;
    newPid = fork();
    if (newPid == -1)
    {
        printf("Error while creating child process\n");
    }
    else if (newPid == 0)
    {
        printf("I am child process\n");
        exit(0);
    }
    else
    {
        wait(&status);
        printf("I am the parent process\n");
        printf("The returned status of child is : %d\n", status);
    }
    return 0;
}
```

OUTPUT

```
student@V310Z-000: ~/190905514/FIFTH-SEM/OS-LAB/LAB3
File Edit View Search Terminal Help
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ls
190905514_MOHAMMAD_TOFIK_OS_LAB3.odt  pgm1.c  pgm1.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ gcc pgm1.c -o pgm1.o
pgm1.c: In function 'main':
pgm1.c:11:14: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
    newPid = fork();
               ^~~~~
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ./pgm1.o
I am child process
I am the parent process
The returned status of child is : 0
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$
```

2. write a c program to load the binary executable of the previous program in a child process using exec system call.

pgm2.c

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

```
int main()
{
    pid_t newPid;
    newPid = fork();

    if (newPid == -1)
    {
        fprintf(stderr, "Error while creating child process.\n");
        exit(-1);
    }

    else if (newPid == 0)
    {
        execlp("./pgm1.o", "pgm1", NULL);
    }

    else
    {
        wait(NULL);
        printf("Child process complete\n");
        exit(0);
    }

    return 0;
}
```

OUTPUT

```
student@V310Z-000: ~/190905514/FIFTH-SEM/OS-LAB/LAB3
File Edit View Search Terminal Help
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ls
190905514_MOHAMMAD_TOFIK_OS_LAB3.odt  pgm1.c  pgm1.o  pgm1.png  pgm2.c  pgm2.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ gcc pgm2.c -o pgm2.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ./pgm2.o
I am child process
I am the parent process
The returned status of child is : 0
Child process complete
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$
```

3.write a program to create a child process.Display the process Ids of the process,parent and child(if any)in both the parent and child processes.

pgm3.c

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

int main()
{
```

```
pid_t newPid;
newPid = fork();

if (newPid == -1)
{
printf("Error creating child process\n");
exit(-1);
}

else if (newPid == 0)
{
printf("In the child process\n");
printf("Process ID is = %d\n", getpid());
printf("Parent's Process ID is = %d\n", getppid());
printf("Child's Process ID is = %d\n", newPid);
}

else
{
wait(NULL);
printf("\nIn the parent process\n");
printf("Process ID is = %d\n", getpid());
printf("Parent's Process ID is = %d\n", getppid());
printf("Child's Process ID is = %d\n", newPid);
}

return 0;
}
```

OUTPUT

```
student@V310Z-000: ~/190905514/FIFTH-SEM/OS-LAB/LAB3
File Edit View Search Terminal Help
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ls
190905514_MOHAMMAD_TOFIK_OS_LAB3.odt  pgm1.o    pgm2.c  pgm2.png  pgm3.o
pgm1.c                                pgm1.png  pgm2.o   pgm3.c
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ gcc pgm3.c -o pgm3.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ./pgm3.o
In the child process
Process ID is = 5546
Parent's Process ID is = 5545
Child's Process ID is = 0

In the parent process
Process ID is = 5545
Parent's Process ID is = 3899
Child's Process ID is = 5546
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$
```

4.create a zombie(defunct)child process(a child with exit() call,but no corresponding wait()) in the sleeping parent) and allow init process to adopt it(after parent terminates).Run the process as a background process and run the “PS” command.

pgm4.c

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
int main(int argc, char const *argv[])
{
    pid_t pid;
    pid = fork();
    if(pid == -1)
    {
```

```

printf("Error while creating fork\n");
exit(-1);
}
if(pid == 0)
{
printf("child process\n");
exit(0);
}
else
{
sleep(2);
printf("parent process\n");
}
return 0;
}

```

OUTPUT

```

student@V310Z-000: ~/190905514/FIFTH-SEM/OS-LAB/LAB3
File Edit View Search Terminal Help
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ls
190905514_MOHAMMAD_TOFIK_OS_LAB3.odt  pgm1.c  pgm2  pgm2.png  pgm3.png  pgm4.png
pgm1.o  pgm1.o  pgm2.c  pgm3.c  pgm4.c
pgm11.png  pgm1.png  pgm2.o  pgm3.o  pgm4.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ gcc pgm4.c -o pgm4.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ./pgm4.o
child process
^Z
[2]+  Stopped                  ./pgm4.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ps
  PID TTY          TIME CMD
 3899 pts/0        00:00:00 bash
 6969 pts/0        00:00:00 pgm4.o
 6970 pts/0        00:00:00 pgm4.o <defunct>
 6983 pts/0        00:00:00 pgm4.o
 6984 pts/0        00:00:00 pgm4.o <defunct>
 6985 pts/0        00:00:00 ps
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ bg
[2]+  ./pgm4.o &
parent process
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$ ps
  PID TTY          TIME CMD
 3899 pts/0        00:00:00 bash
 6969 pts/0        00:00:00 pgm4.o
 6970 pts/0        00:00:00 pgm4.o <defunct>
 6991 pts/0        00:00:00 ps
[2]-  Done                  ./pgm4.o
student@V310Z-000:~/190905514/FIFTH-SEM/OS-LAB/LAB3$

```

