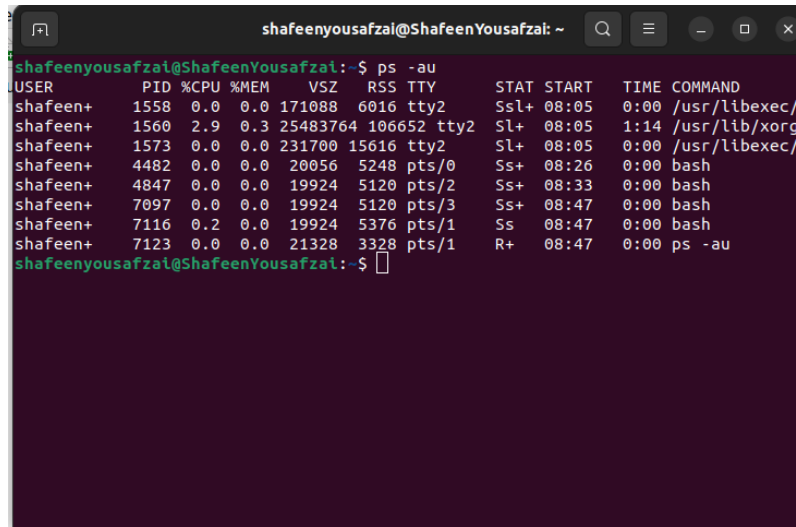


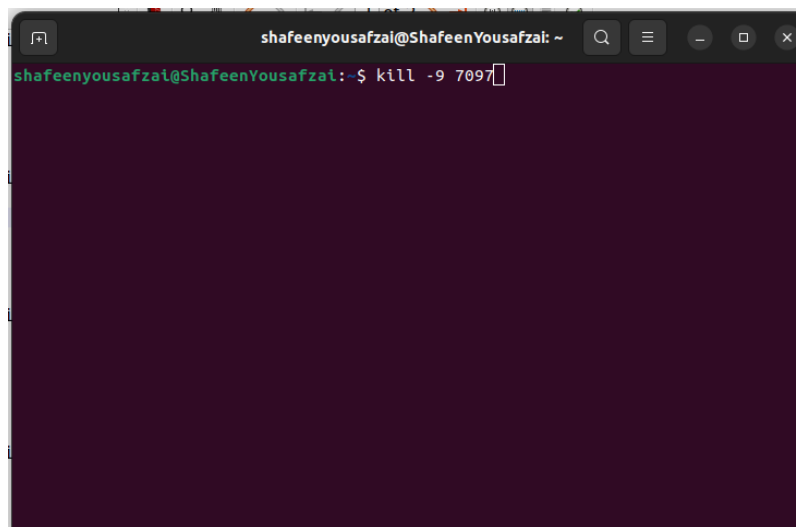
# 1 Kill Command

This command is used to politely kill the running process , we have different sub commands or kill processes under the kill command



```
shafeenyousafzal@ShafeenYousafzal: ~  
shafeenyousafzal@ShafeenYousafzal:~$ ps -au  
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND  
shafeen+  1558  0.0  0.0 171088  6016 tty2    Ssl+  08:05   0:00 /usr/libexec/  
shafeen+  1560  2.9  0.3 25483764 106652 tty2    Sl+   08:05   1:14 /usr/lib/xorg  
shafeen+  1573  0.0  0.0 231700  15616 tty2    Sl+   08:05   0:00 /usr/libexec/  
shafeen+  4482  0.0  0.0 20056   5248 pts/0    Ss+   08:26   0:00 bash  
shafeen+  4847  0.0  0.0 19924   5120 pts/2    Ss+   08:33   0:00 bash  
shafeen+  7097  0.0  0.0 19924   5120 pts/3    Ss+   08:47   0:00 bash  
shafeen+  7116  0.2  0.0 19924   5376 pts/1    Ss+   08:47   0:00 bash  
shafeen+  7123  0.0  0.0 21328  3328 pts/1    R+    08:47   0:00 ps -au  
shafeenyousafzal@ShafeenYousafzal:~$
```

Figure 1: Showing processes that are running



```
shafeenyousafzal@ShafeenYousafzal: ~  
shafeenyousafzal@ShafeenYousafzal:~$ kill -9 7097
```

Figure 2: killing the bash with process id 7097

```

shafeenyousafzal@ShafeenYousafzal:~$ ps -au
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
shafeen+    1558  0.0  0.0 171088  6016 tty2      Ssl+  08:05   0:00 /usr/libexec/
shafeen+    1560  2.9  0.3 25483764 106652 tty2      Sl+   08:05   1:15 /usr/lib/xorg
shafeen+    1573  0.0  0.0 231700 15616 tty2      Sl+   08:05   0:00 /usr/libexec/
shafeen+    4482  0.0  0.0 20056  5248 pts/0      Ss+   08:26   0:00 bash
shafeen+    4847  0.0  0.0 19924  5120 pts/2      Ss+   08:33   0:00 bash
shafeen+    7116  0.0  0.0 19924  5376 pts/1      Ss    08:47   0:00 bash
shafeen+    7154  0.0  0.0 21328  3328 pts/1      R+    08:48   0:00 ps -au
shafeenyousafzal@ShafeenYousafzal:~$ 

```

Figure 3: As you can see the process we just killed has been terminated

## 2 5.1.1.1 Exercise

### 2.1 Question :

The integer representation for the SIGTERM signal

### 2.2 Answer :

The integer representation for the SIGTERM is 15

### 2.3 Question :

The PID of your current active bash shell cess, we will use using the ps command

### 2.4 Answer :

The integer representation for bash shell on my pc : 4482 , 4847 , 7116 , 7514

```

shafeenyousafzal@ShafeenYousafzal:~$ ps -au
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
shafeen+    1558  0.0  0.0 171088  6016 tty2      Ssl+  08:05   0:00 /usr/libexec/
shafeen+    1560  3.1  0.3 25483764 106588 tty2      Sl+   08:05   1:47 /usr/lib/xorg
shafeen+    1573  0.0  0.0 231700 15616 tty2      Sl+   08:05   0:00 /usr/libexec/
shafeen+    4482  0.0  0.0 20056  5248 pts/0      Ss+   08:26   0:00 bash
shafeen+    4847  0.0  0.0 19924  5120 pts/2      Ss+   08:33   0:00 bash
shafeen+    7116  0.0  0.0 19924  5376 pts/1      Ss+   08:47   0:00 bash
shafeen+    7514  0.0  0.0 19924  5120 pts/3      Ss    09:02   0:00 bash
shafeen+    7521  0.0  0.0 21328  3328 pts/3      R+    09:02   0:00 ps -au
shafeenyousafzal@ShafeenYousafzal:~$ 

```

Figure 4: This show the current processes

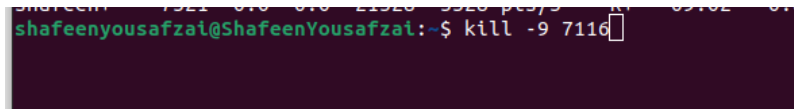


Figure 5: Killing the terminal process

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
shafeen+	1558	0.0	0.0	171088	6016	tty2	Ssl+	08:05	0:00	/usr/libexec/
shafeen+	1560	3.3	0.3	25483764	106868	tty2	Sl+	08:05	2:02	/usr/lib/xorg
shafeen+	1573	0.0	0.0	231700	15616	tty2	Sl+	08:05	0:00	/usr/libexec/
shafeen+	4482	0.0	0.0	20056	5248	pts/0	Ss+	08:26	0:00	bash
shafeen+	4847	0.0	0.0	19924	5120	pts/2	Ss+	08:33	0:00	bash
shafeen+	7514	0.0	0.0	19924	5120	pts/3	Ss	09:02	0:00	bash
shafeen+	7642	0.0	0.0	21328	3328	pts/3	R+	09:06	0:00	ps -au

Figure 6: The process has been killed

### 3 Kill () command C code)

```

1  #include<stdio.h>
2  #include<sys/types.h>
3  #include<signal.h>
4  #include<unistd.h>
5  int main()
6  {
7      printf("\nMuhammad Shafeen\n");
8      printf("22P-9278\n");
9      printf("BAI-5A\n");
10
11     int x=10;
12     int y=20;
13     int sum=x+y;
14     printf("Sum of %d and %d is : %d\n",x,y,sum);
15
16     kill(getpid(),9);
17
18     printf("The program has been killed\n");
19     return 0;
20 }
```

Listing 1: Mini-Shell using execvp

### 3.1 Screenshots of C code

```
shafeenyousafzai@shafeenyousafzai:/media/shafeenyousafzai/OLDERDRIVE/Semester-2/Operating-System-Lab/Task 10$ cat kill.c
#include<stdio.h>
#include<sys/types.h>
#include<signal.h>
#include<unistd.h>
int main()
{
printf("\nMuhammad Shafeen\n");
printf("22P-9278\n");
printf("BAI-5A\n");

int x=10;
int y=20;
int sum=x+y;
printf("Sum of %d and %d is : %d\n",x,y,sum);

kill(getpid(),9);

printf("The program has been killed\n");
return 0;
}
```

Figure 7: The code for killing a process using C code

```
shafeenyousafzai@shafeenyousafzai:/media/shafeenyousafzai/OLDERDRIVE/Semester-2/Operating-System-Lab/Task 10$ ./kill
Muhammad Shafeen
22P-9278
BAI-5A
Sum of 10 and 20 is : 30
Killed
shafeenyousafzai@shafeenyousafzai:/media/shafeenyousafzai/OLDERDRIVE/Semester-2/Operating-System-Lab/Task 10$
```

Figure 8: Output of the code

```
shafeenyousafzai@shafeenyousafzai:/media/shafeenyousafzai/OLDERDRIVE/Semester-2/Operating-System-Lab/Task 10$ cat kill.c
#include<stdio.h>
#include<sys/types.h>
#include<signal.h>
#include<unistd.h>
int main()
{
printf("\nMuhammad Shafeen\n");
printf("22P-9278\n");
printf("BAI-5A\n");

int x=10;
int y=20;
int sum=x+y;
printf("Sum of %d and %d is : %d\n",x,y,sum);

kill(getpid(),15); //9

printf("The program has been killed\n");
return 0;
}
```

Figure 9: The code to use 15 as kill

```
shafeenyousarzat@ShafeenYousarzat:/media/shafeenyousarzat/OLDERDRIVE/Semester
Operating-System-Lab/Task 10$ ./kill

Muhammad Shafeen
22P-9278
BAI-5A
Sum of 10 and 20 is : 30
Terminated
```

Figure 10: Output of the code

## 4 5.1.5.1 Exercise

### 4.1 Code using fork , Child signal-ing parent to kill

I have used the `execl` command to show the processes and then killed the process with `SIGTERM` and then display the processes after killing it

```
1 #include<stdio.h>
2 #include<sys/types.h>
3 #include<signal.h>
4 #include<unistd.h>
5 #include<stdlib.h>
6 int main()
7 {
8     printf("\nMuhammad Shafeen\n");
9     printf("22P-9278\n");
10    printf("BAI-5A\n");
11
12    int x=10;
13    int y=20;
14    int sum=x+y;
15    printf("Sum of %d and %d is : %d\n",x,y,sum);
16    pid_t pid;
17    pid=fork();
18    if(pid==0)
19    {
20        pid_t pid2;
21        pid2=fork();
22        if(pid2==0)
23        {
24            printf("Showing processess before killing it\n");
25            execl("/bin/ps","ps -au",(char *)NULL);
26            perror("execl failed");
27        }
28        // sleep(5);
29    }
30    else
```

```

31 {
32     kill(getppid(),15); //9
33     printf("Showing processess after killing it\n");
34     execl("/bin/ps","ps -au",(char *)NULL);
35     perror("execl failed");
36     exit(EXIT_FAILURE);
37     return 0;
38 }
39 }

```

Listing 2: Mini-Shell using execvp

```

shafeenyousafzal@shafeenyousafzal:/media/shafeenyousafzal/04B8B810/Semester 5/Operating-System-Lab/Task 10$ cat exercise.c
#include<stdio.h>
#include<sys/types.h>
#include<signal.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
    printf("\nMuhammad Shafeen\n");
    printf("22P-9278\n");
    printf("BAI-SA\n");

    int x=10;
    int y=20;
    int sum=x+y;
    printf("Sum of %d and %d is : %d\n",x,y,sum);
    pid_t pid;
    pid=fork();
    if(pid==0)
    {
        pid_t pid2;
        pid2=fork();
        if(pid2==0)
        {
            printf("Showing processess before killing it\n");
            execl("/bin/ps","ps -au",(char *)NULL);
            perror("execl failed");
            // sleep(5);
        }
        else
        {
            kill(getppid(),15); //9
            printf("Showing processess after killing it\n");
            execl("/bin/ps","ps -au",(char *)NULL);
            perror("execl failed");
            exit(EXIT_FAILURE);
            return 0;
        }
    }
}

```

Figure 11: Code for kill() using fork()

```

shafeenyousafzal@shafeenyousafzal:/media/shafeenyousafzal/04B8B810/Semester 5/Operating-System-Lab/Task 10$ ./exercise
Muhammad Shafeen
22P-9278
BAI-SA
Sum of 10 and 20 is : 30
Showing processess before killing it
Showing processess after killing it
PID TTY TIME CMD
7698 pts/1 00:00:00 bash
7698 pts/1 00:00:00 bash
9612 pts/1 00:00:00 ps
9612 pts/1 00:00:00 ps
9613 pts/1 00:00:00 exercise <defunct>
9613 pts/1 00:00:00 exercise <defunct>
9614 pts/1 00:00:00 ps
9614 pts/1 00:00:00 ps

```

Figure 12: Output of the code for kill() and fork() )

## 4.2 Code using fork , parent signal-ing child to kill

I have used the execl command to show the processes and then killed the process with SIGTERM and then display the processes after killing it

```
1 #include<stdio.h>
2 #include<sys/types.h>
3 #include<signal.h>
4 #include<unistd.h>
5 #include<stdlib.h>
6 int main()
7 {
8     printf("\nMuhammad Shafeen\n");
9     printf("22P-9278\n");
10    printf("BAI-5A\n");
11
12    int x=10;
13    int y=20;
14    int sum=x+y;
15    printf("Sum of %d and %d is : %d\n",x,y,sum);
16    pid_t pid;
17    pid=fork();
18    if(pid==0)
19    {
20        sleep(1);
21        pid_t pid2;
22        pid2=fork();
23        if(pid2==0)
24        {
25            printf("Showing processess before killing it\n");
26            execl("/bin/ps","ps -au",(char *)NULL);
27            perror("execl failed");
28        }
29        kill(getpid(),15); //9
30        // sleep(5);
31    }
32    else
33    {
34        sleep(2);
35        printf("Showing processess after killing it\n");
36        execl("/bin/ps","ps -au",(char *)NULL);
37        perror("execl failed");
38        exit(EXIT_FAILURE);
39        return 0;
40    }
41 }
```

Listing 3: Mini-Shell using execvp

```

shafeenyousafzal@shafeenyousafzal:~/media/shafeenyousafzal/OLDENRIVE/Semester 5/Operating-System-Lab/Task 10$ cat exercise2.c
#include<stdio.h>
#include<sys/types.h>
#include<signal.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
    printf("\nMuhammad Shafeen\n");
    printf("22P-9278\n");
    printf("BAI-5A\n");

    int x=10;
    int y=20;
    int sum=x+y;
    printf("Sum of %d and %d is : %d\n",x,y,sum);
    pid_t pid;
    pid_t pid2;
    pidfork1();
    if(pid==0)
    {
        sleep(1);
        pid_t pid2;
        pid2=fork1();
        if(pid2==0)
        {
            printf("Showing processess before killing it\n");
            execl("/bin/ps","ps","-au", (char *)NULL);
            perror("execl failed");
            kill(getpid(),15); //9
            // sleep(5);
        }
        else
        {
            sleep(2);
            printf("Showing processess after killing it\n");
            execl("/bin/ps","ps","-au", (char *)NULL);
            perror("execl failed");
            exit(EXIT_FAILURE);
            return 0;
        }
    }
}

```

Figure 13:

```

shafeenyousafzal@shafeenyousafzal:~/media/shafeenyousafzal/OLDENRIVE/Semester 5/Operating-System-Lab/Task 10$ ./exercise2
Muhammad Shafeen
22P-9278
BAI-5A
Sum of 10 and 20 is : 30
Showing processess before killing it
  PID TTY          TIME CMD
 7698 pts/1    00:00:00 bash
 9958 pts/1    00:00:00 exercise2
 9959 pts/1    00:00:00 exercise2 <defunct>
 9960 pts/1    00:00:00 ps
Showing processess after killing it
  PID TTY          TIME CMD
 7698 pts/1    00:00:00 bash
 9958 pts/1    00:00:00 ps
 9959 pts/1    00:00:00 exercise2 <defunct>
shafeenyousafzal@shafeenyousafzal:~/media/shafeenyousafzal/OLDENRIVE/Semester 5/Operating-System-Lab/Task 10$

```

Figure 14:

## 5 Signal Handling Exercise

```

1 #include <signal.h>
2 #include <stdio.h>
3 #include <unistd.h>
4 int sigCounter = 0;
5 void sigHandler(int sigNum)
6 {
7     printf("Signal received is %d\n", sigNum);
8     ++sigCounter;
9     printf("Signals received %d\n", sigCounter); }
10 int main()
11 {
12     signal(SIGINT, sigHandler);
13     while(1)
14     {
15         printf("Hello Dears\n");

```



```
16     sleep(1);  
17 }  
18 return 0;  
19 }  
20
```

Listing 4: Mini-Shell using execvp

```
shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 10$ cat signalhandling.c  
#include <signal.h>  
#include <stdio.h>  
#include <unistd.h>  
int sigCounter = 0;  
void sigHandler(int sigNum)  
{  
    printf("Signal received is %d\n", sigNum);  
    ++sigCounter;  
    printf("Signals received %d\n", sigCounter);  
}  
int main()  
{  
    signal(SIGINT, sigHandler);  
    while(1)  
    {  
        printf("Hello Dears\n");  
        sleep(1);  
    }  
    return 0;  
}
```

Figure 15: The code for infinite loop

```
shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 10$ ./signalhandling  
Hello Dears  
Hello Dears  
Hello Dears  
^Z  
[1]+  Stopped                  ./signalhandling  
shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 10$ ./signalhandling  
Hello Dears  
^CSignal received is 2  
Signals received 1  
Hello Dears  
^CSignal received is 2  
Signals received 2  
Hello Dears  
^CSignal received is 2  
Signals received 3  
Hello Dears  
Hello Dears  
^Z  
[2]+  Stopped                  ./signalhandling  
shafeenyousafzal@ShafeenYousafzal: /media/shafeenyousafzal/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 10$
```

Figure 16: Execution of infinite loop