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

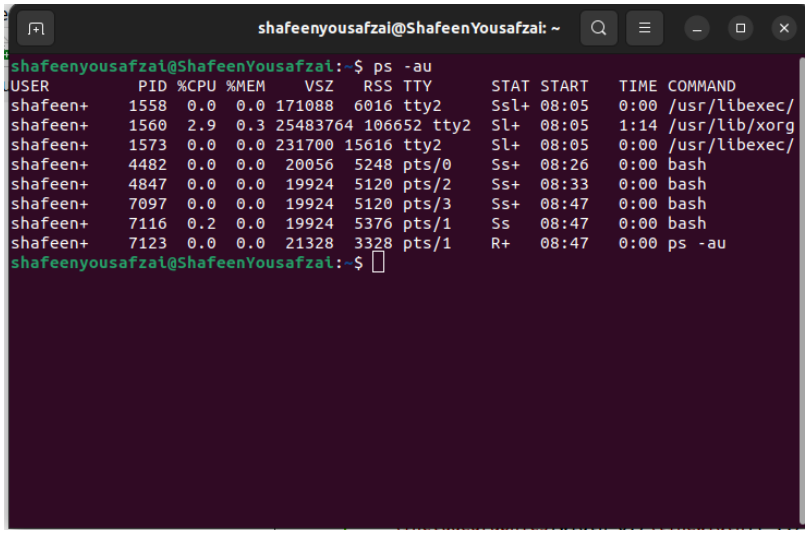


Figure 1: Showing processes that are runnning

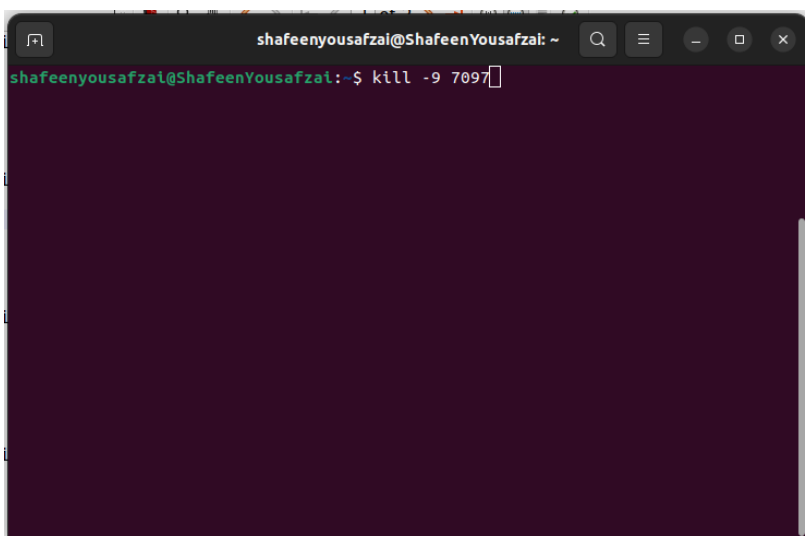


Figure 2: killing the bash with process id 7097

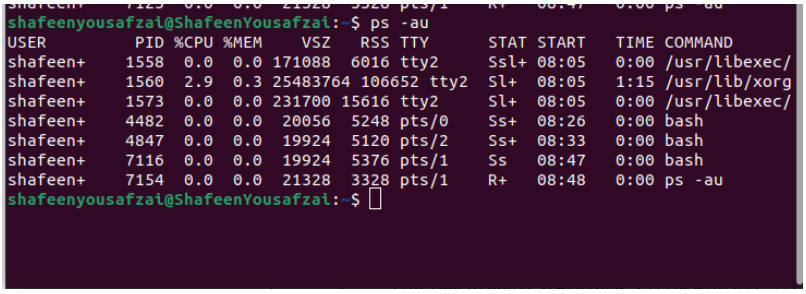


Figure 3: As you can see the process we just killed has been terminated

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

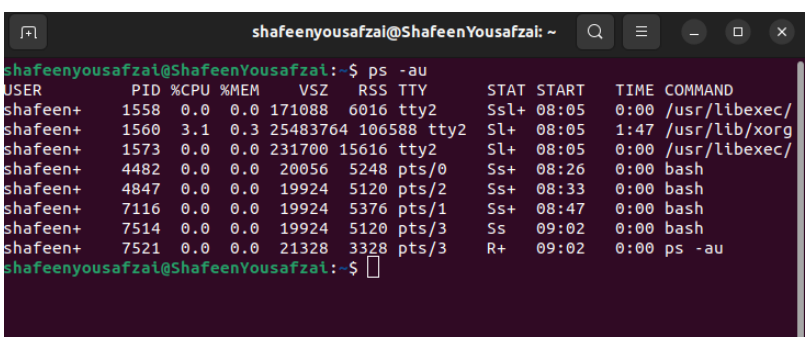


Figure 4: This show the current processes

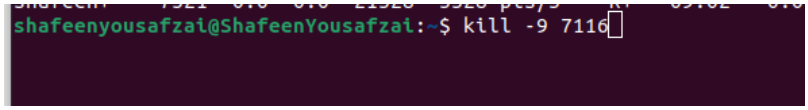
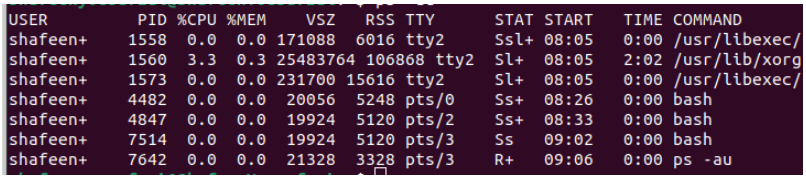


Figure 5: Killing the terminal process



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)

```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;
}
```

Listing 1: Mini-Shell using execvp

3.1 Screenshots of C code

```
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
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
```

Figure 8: Output of the code

```
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

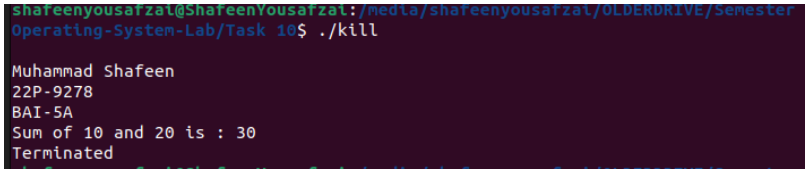


Figure 10: Output of the code

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

```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=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;
  }
}
```

Listing 2: Mini-Shell using execvp

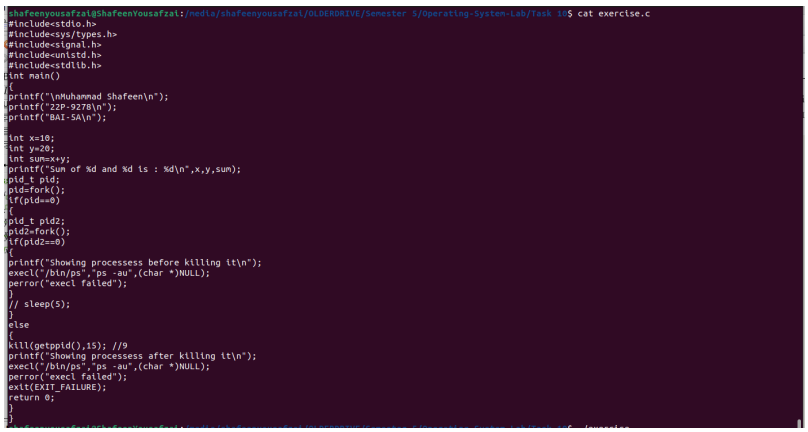


Figure 11: Code for kill() using fork()

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

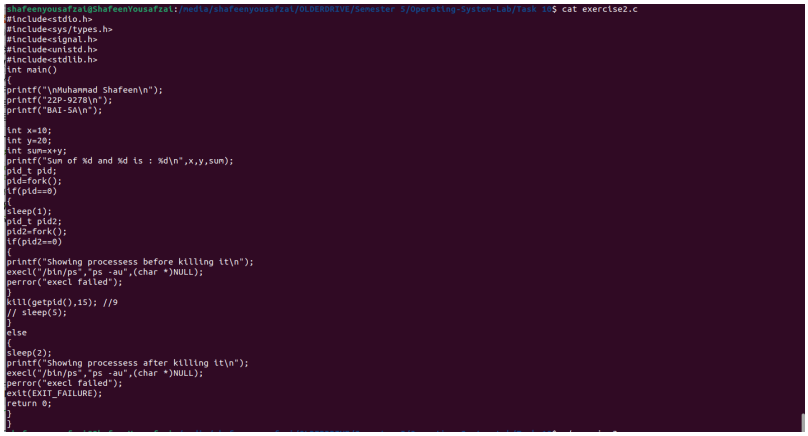
4.2 Code using fork , parent signal-ing child to kill

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

```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=fork();
  if(pid==0)
  {
    sleep(1);
    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");
    }
    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;
  }
}
```

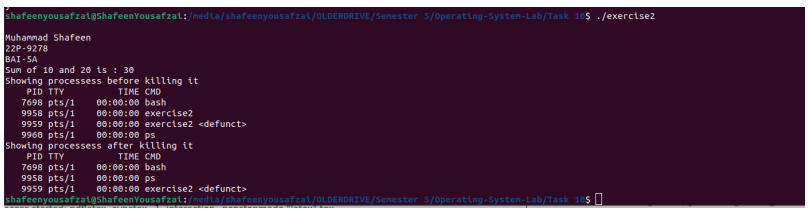
Listing 3: Mini-Shell using execvp



```
shafeenyousafzai@ShafeenYousafzai:/media/shafeenyousafzai/OLDERDRIVE/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=fork();
if(pid==0)
{
sleep(1);
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");
}
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:



```
shafeenyousafzai@ShafeenYousafzai:/media/shafeenyousafzai/OLDERDRIVE/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>
shafeenyousafzai@ShafeenYousafzai:/media/shafeenyousafzai/OLDERDRIVE/Semester 5/Operating-System-Lab/Task 10$
```

Figure 14:

5 Signal Handling Exercise

```
#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;
}

```

Listing 4: Mini-Shell using execvp

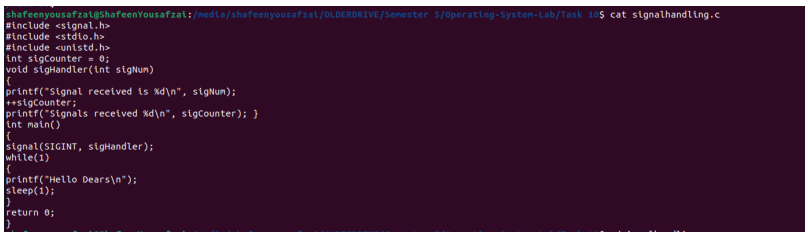


Figure 15: The code for infinite loop

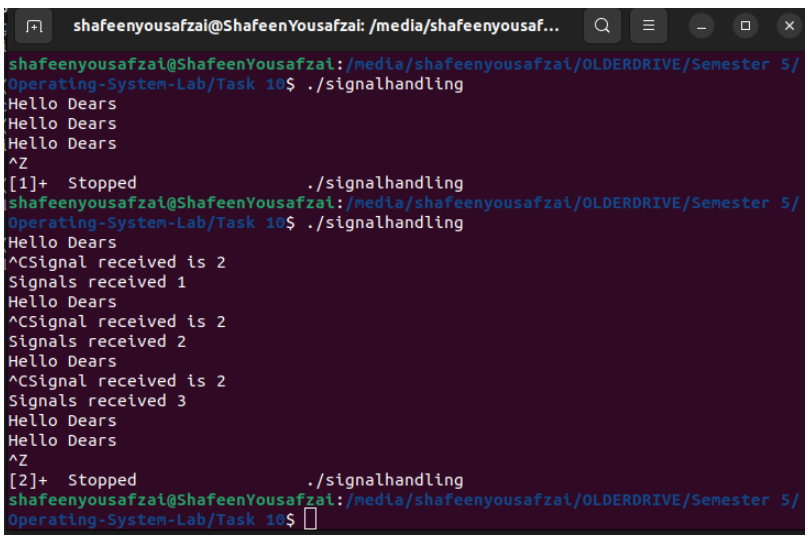


Figure 16: Execution of infinite loop