NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Peshawar Campus

Operating Systems

Lab Task #08

Submitted By

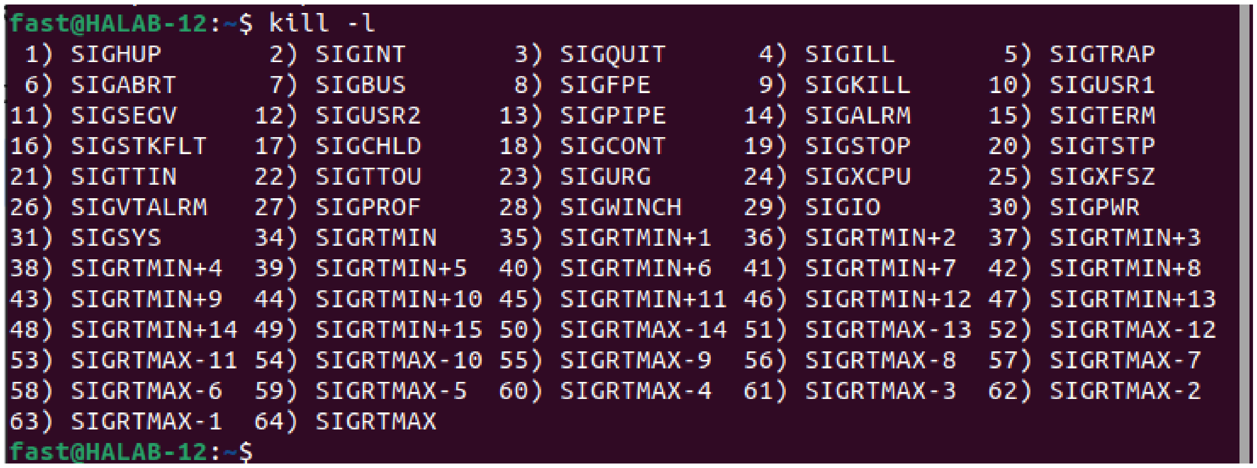
Name: Khushal Das

Roll No: 22P-9341
Program: BSE-5B

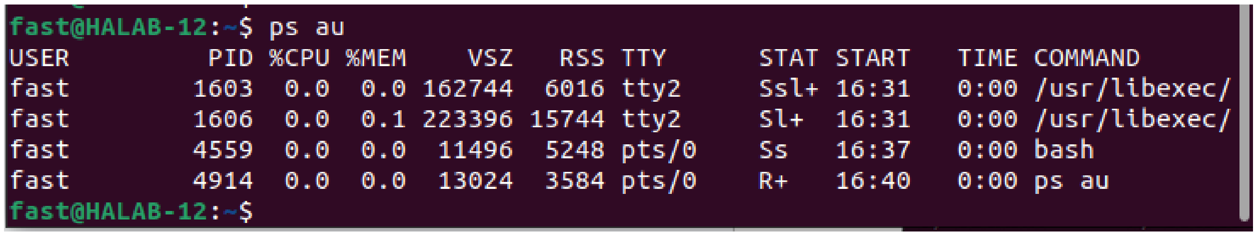
Submitted To
Sir Saad Ahmad

Department of Computer Science
FAST-NUCES

The `kill -l` command in Linux lists all the available signal names

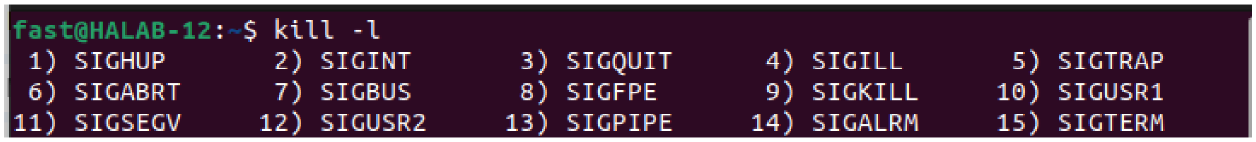

```
fast@HALAB-12:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
fast@HALAB-12:~$
```

The `ps au` command displays information about running processes.

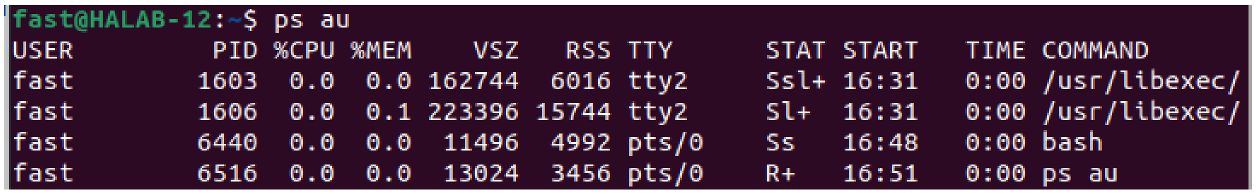

```
fast@HALAB-12:~$ ps au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
fast        1603  0.0  0.0 162744  6016 tty2     Ssl+ 16:31   0:00 /usr/libexec/
fast        1606  0.0  0.1 223396 15744 tty2     Sl+  16:31   0:00 /usr/libexec/
fast        4559  0.0  0.0  11496  5248 pts/0    Ss   16:37   0:00 bash
fast        4914  0.0  0.0  13024  3584 pts/0    R+   16:40   0:00 ps au
fast@HALAB-12:~$
```

5.1.1.1 Exercise

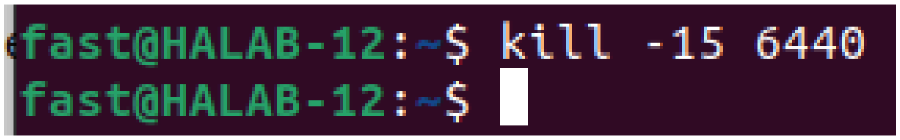
The integer representation of `SIGTERM` is **15**.


```
fast@HALAB-12:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
```

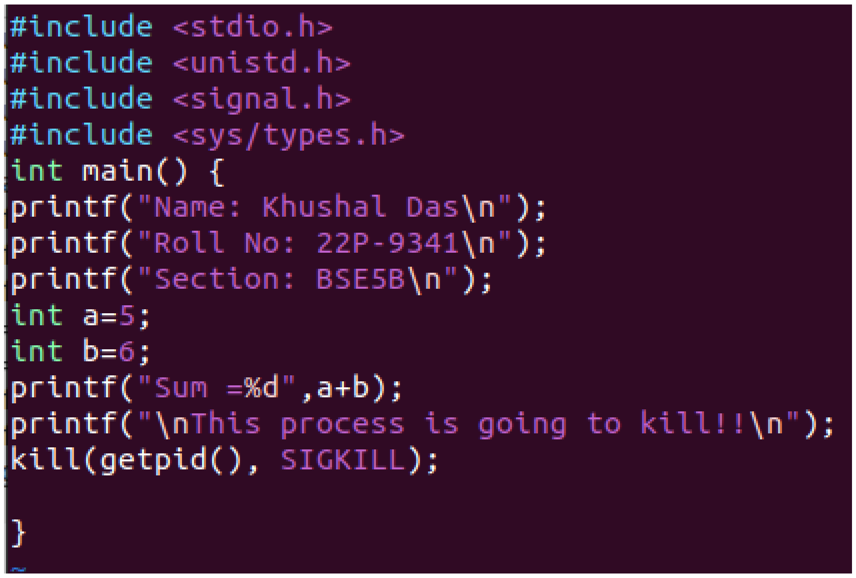
PID of the current active bash shell: 6440


```
fast@HALAB-12:~$ ps au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
fast        1603  0.0  0.0 162744  6016 tty2     Ssl+ 16:31   0:00 /usr/libexec/
fast        1606  0.0  0.1 223396 15744 tty2     Sl+  16:31   0:00 /usr/libexec/
fast        6440  0.0  0.0  11496  4992 pts/0    Ss   16:48   0:00 bash
fast        6516  0.0  0.0  13024  3456 pts/0    R+   16:51   0:00 ps au
```

Send the SIGTERM signal to the bash shell

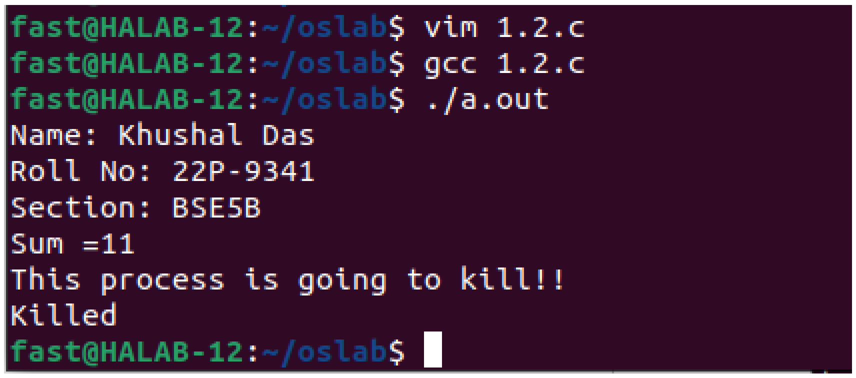

```
fast@HALAB-12:~$ kill -15 6440
fast@HALAB-12:~$ 
```

Code for killed.



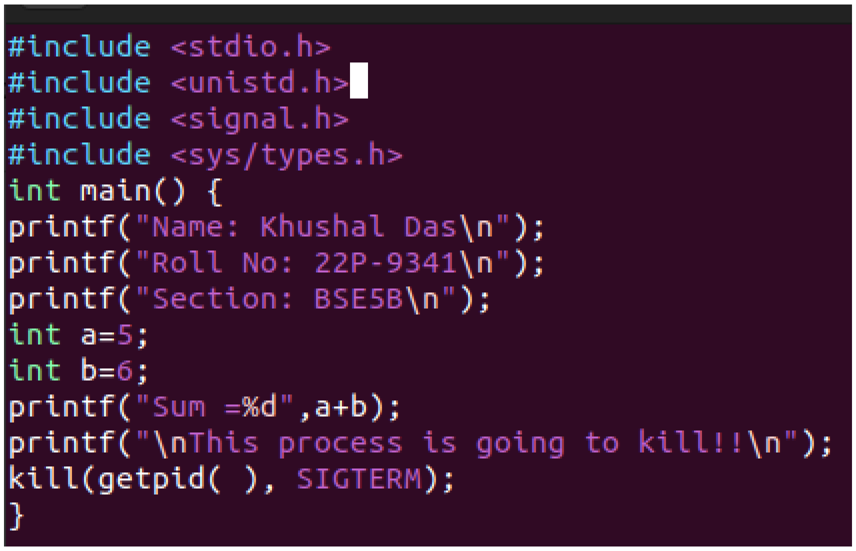
```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
int main() {
printf("Name: Khushal Das\n");
printf("Roll No: 22P-9341\n");
printf("Section: BSE5B\n");
int a=5;
int b=6;
printf("Sum =%d",a+b);
printf("\nThis process is going to kill!!\n");
kill(getpid(), SIGKILL);

}
```

Output of the Killed.

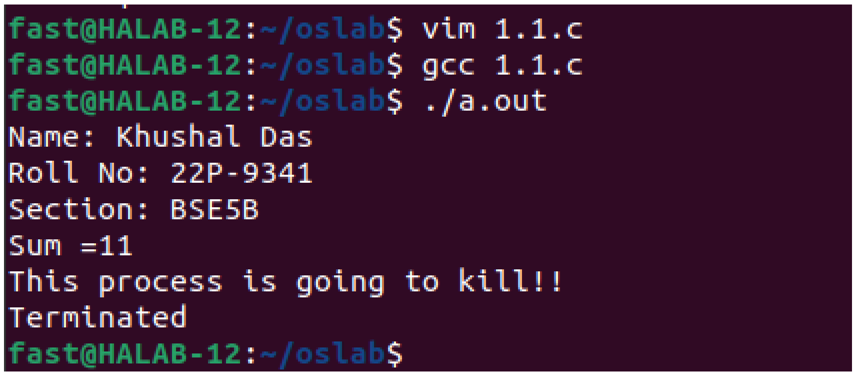


```
fast@HALAB-12:~/oslab$ vim 1.2.c
fast@HALAB-12:~/oslab$ gcc 1.2.c
fast@HALAB-12:~/oslab$ ./a.out
Name: Khushal Das
Roll No: 22P-9341
Section: BSE5B
Sum =11
This process is going to kill!!
Killed
fast@HALAB-12:~/oslab$
```

Code for Terminated

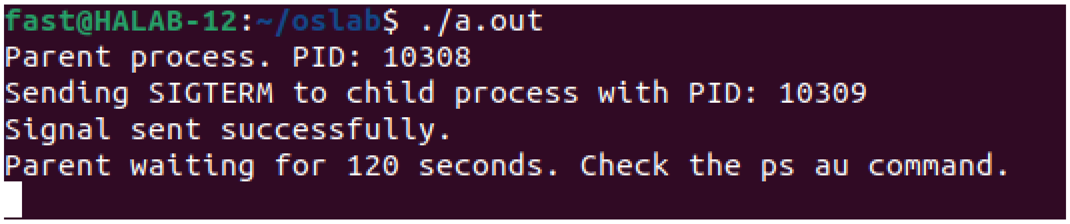

```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
int main() {
printf("Name: Khushal Das\n");
printf("Roll No: 22P-9341\n");
printf("Section: BSE5B\n");
int a=5;
int b=6;
printf("Sum =%d",a+b);
printf("\nThis process is going to kill!!\n");
kill(getpid( ), SIGTERM);
}
```

Output of the Terminated.

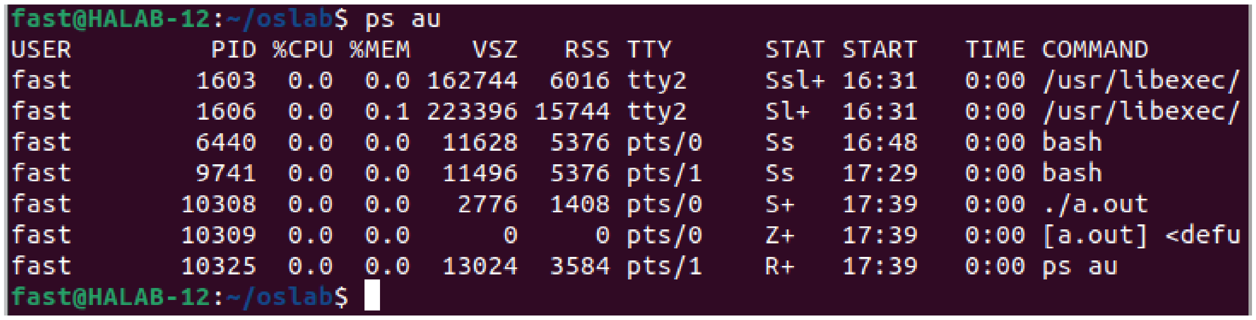

```
fast@HALAB-12:~/oslab$ vim 1.1.c
fast@HALAB-12:~/oslab$ gcc 1.1.c
fast@HALAB-12:~/oslab$ ./a.out
Name: Khushal Das
Roll No: 22P-9341
Section: BSE5B
Sum =11
This process is going to kill!!
Terminated
fast@HALAB-12:~/oslab$
```

5.1.2.1 Exercise

Output:


```
fast@HALAB-12:~/oslab$ ./a.out
Parent process. PID: 10308
Sending SIGTERM to child process with PID: 10309
Signal sent successfully.
Parent waiting for 120 seconds. Check the ps au command.
```

Check the processes using ps au command



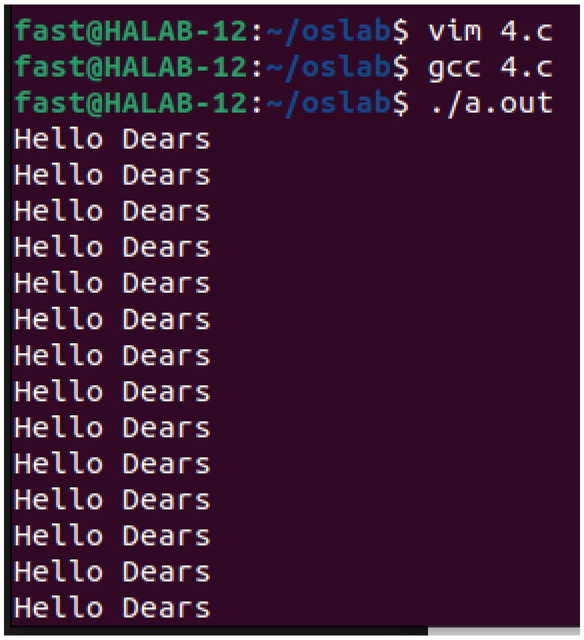
```
fast@HALAB-12:~/oslab$ ps au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
fast        1603  0.0  0.0 162744  6016 tty2     Ssl+ 16:31   0:00 /usr/libexec/
fast        1606  0.0  0.1 223396 15744 tty2     Sl+  16:31   0:00 /usr/libexec/
fast        6440  0.0  0.0  11628  5376 pts/0    Ss   16:48   0:00 bash
fast        9741  0.0  0.0  11496  5376 pts/1    Ss   17:29   0:00 bash
fast       10308  0.0  0.0   2776  1408 pts/0    S+   17:39   0:00 ./a.out
fast       10309  0.0  0.0      0     0 pts/0    Z+   17:39   0:00 [a.out] <defu
fast       10325  0.0  0.0  13024  3584 pts/1    R+   17:39   0:00 ps au
fast@HALAB-12:~/oslab$ 
```

5.1.2.2 Exercise

While running the program the system is shutting down b/c the zombie process is created

5.2 Signal Handling Using Signal

Run the code.
Output.

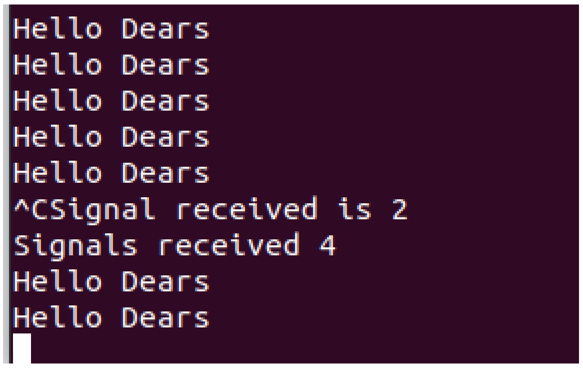


```
fast@HALAB-12:~/oslab$ vim 4.c
fast@HALAB-12:~/oslab$ gcc 4.c
fast@HALAB-12:~/oslab$ ./a.out
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
```

TASK

Pressing CTRL+C with the signal()

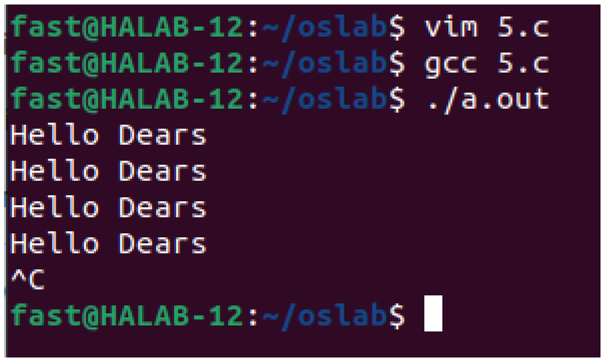
When CTRL+C is pressed, the SIGINT signal is sent to the process. Since signal(SIGINT, sigHandler) has been used to set a custom signal handler (sigHandler), the handler function will be executed when the signal is received. The handler prints the signal number and counts how many times the signal has been received, incrementing sigCounter each time. Afterward, the process continues to run in an infinite loop.



```
Hello Dears
Hello Dears
Hello Dears
Hello Dears
Hello Dears
^CSignal received is 2
Signals received 4
Hello Dears
Hello Dears
```

Pressing CTRL+C without the signal()

If no custom signal handler is set using signal(), the default behavior of the process is to terminate upon receiving the SIGINT signal when CTRL+C is pressed. The program would immediately stop execution, and no signal-related information would be printed.



```
fast@HALAB-12:~/oslab$ vim 5.c
fast@HALAB-12:~/oslab$ gcc 5.c
fast@HALAB-12:~/oslab$ ./a.out
Hello Dears
Hello Dears
Hello Dears
Hello Dears
^C
fast@HALAB-12:~/oslab$
```