CSE2005- Operating Systems

Lab Ex. 5 Signal Handling

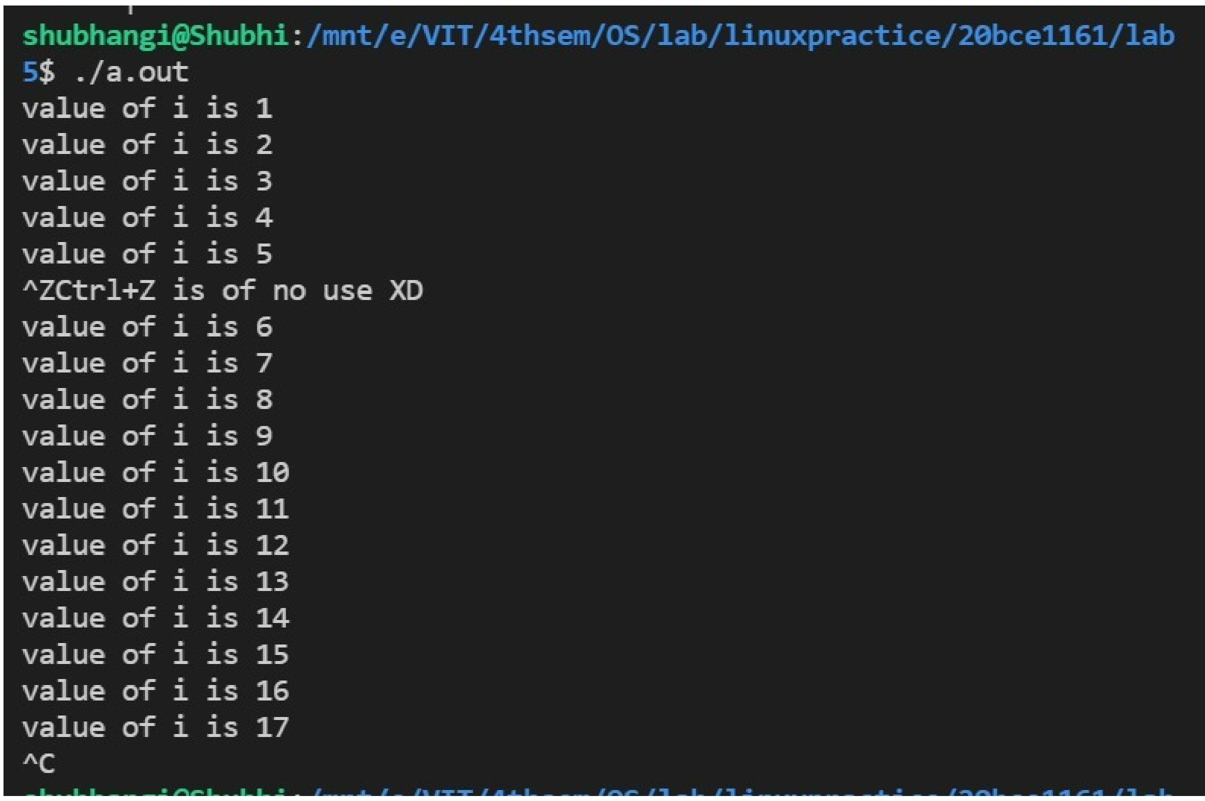
1. **Write your own C handlers to handle the following signals**

   a. **Send a stop signal using Ctrl-Z**

Code:

```
#include <stdlib.h>

#include <stdio.h>

#include <signal.h>


void myhandler(int signum){

  printf("Ctrl+Z is of no use XD\n");

}


int main(){

 int i=1;

signal(SIGTSTP, myhandler);

 while(i)

{

printf("value of i is %d\n",i);

sleep(2);

i++;

}

 }
```

Output:



```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab
5$ ./a.out
value of i is 1
value of i is 2
value of i is 3
value of i is 4
value of i is 5
^ZCtrl+Z is of no use XD
value of i is 6
value of i is 7
value of i is 8
value of i is 9
value of i is 10
value of i is 11
value of i is 12
value of i is 13
value of i is 14
value of i is 15
value of i is 16
value of i is 17
^C
```

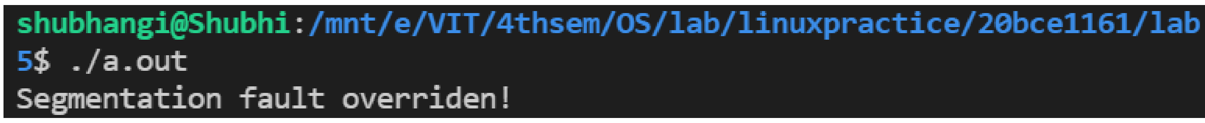
b. Segmentation fault

Code:

```c
#include <stdio.h>

#include <signal.h>

#include <stdlib.h>



void myhandler(){

    printf("Segmentation fault overriden!\n");

    exit(0);

}

int fnc(){

    float *a, *b;
```

```
     a = (float*)malloc(1000);

     b[0] = 1.0;

     return 0;



}

int main(){

    signal(SIGSEGV,myhandler);

    int x=fnc();



}
```

Output:



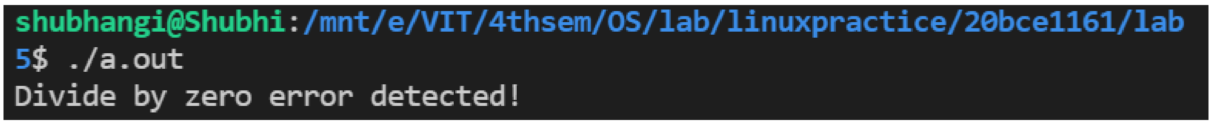
```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab
5$ ./a.out
Segmentation fault overriden!
```

c. Divide by zero error

```
#include <stdio.h>

#include <signal.h>

#include <stdlib.h>



void myhandler(){

    printf("Divide by zero error detected!\n");

    exit(0);

}

int fnc(){

    int a, b=0;
```

```c
     a = a/b;

     return 0;



}

int main(){

    signal(SIGFPE,myhandler);

    int x=fnc();



}
```

Output:



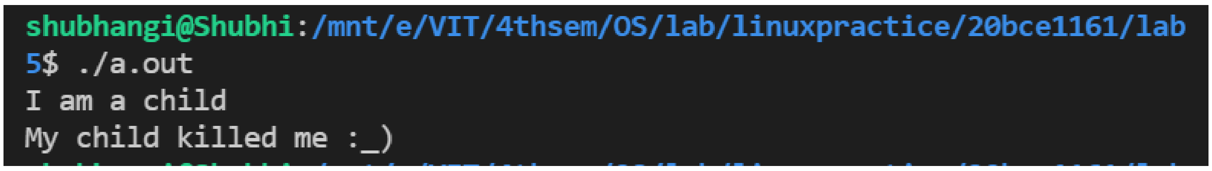
```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab
5$ ./a.out
Divide by zero error detected!
```

2. **Write a program which creates a child process and continues to run along with its child (choose any small task of your own). Once the child completes its task, it should send a signal to the parent which in turn terminates the parent. (Expected output: output of the task carried out by the child process, termination of parent)**

Code:

```c
#include <stdio.h>

#include <signal.h>

#include <stdlib.h>

#include <unistd.h>



void myhandler(){

    printf("My child killed me :_)\n");
    exit(0);

}


int main(){


  pid_t cpid;

  pid_t ppid;


  signal(SIGQUIT, myhandler);


  if ( (cpid = fork()) == 0){

      printf("I am a child\n");

    ppid = getppid();

    kill(ppid, SIGQUIT);

//  exit(0);

   }

else{

wait(NULL);

    printf("I won't be printed");

  }



}
```

Output:



```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab
5$ ./a.out
I am a child
My child killed me :_)
```

3. **Write two c programs: One displaying the PID infinitely and the other program sending a signal to terminate the first program.(Note: Execute the programs in separate terminals)**

Code (1st program):

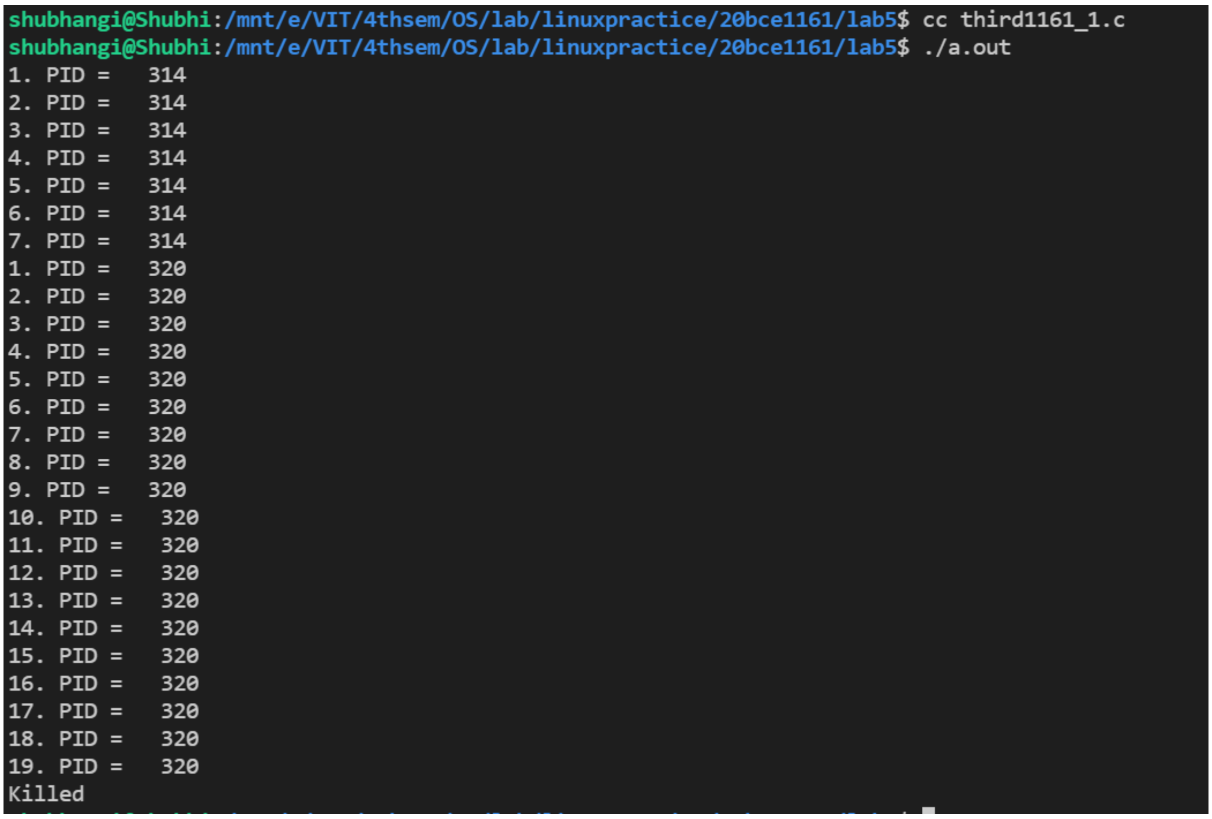
```c
#include <stdio.h>

#include <signal.h>

#include <stdlib.h>

#include <unistd.h>

int main(){

    pid_t x=getpid();

    int i=1;

    while(i){

    printf("%d. PID =    %d\n",i,x);

    i++;

    sleep(1);

    }

}
```

2nd program:

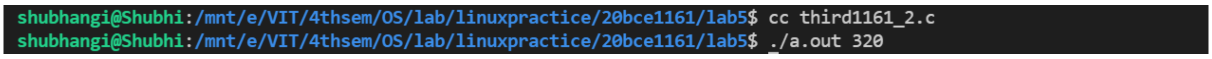
```c
#include <stdio.h>
```

```c
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
void myhandler(){
   /**/


}
int main(int argc,char* argv[]){
    int x=strtol(argv[1],NULL,10);
    signal(SIGKILL, myhandler);
    kill(x,SIGKILL);



}
```

Output:



```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab5$ cc third1161_1.c
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab5$ ./a.out
1. PID =   314
2. PID =   314
3. PID =   314
4. PID =   314
5. PID =   314
6. PID =   314
7. PID =   314
1. PID =   320
2. PID =   320
3. PID =   320
4. PID =   320
5. PID =   320
6. PID =   320
7. PID =   320
8. PID =   320
9. PID =   320
10. PID =   320
11. PID =   320
12. PID =   320
13. PID =   320
14. PID =   320
15. PID =   320
16. PID =   320
17. PID =   320
18. PID =   320
19. PID =   320
Killed
```



```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab5$ cc third1161_2.c
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab5$ ./a.out 320
```