**Name:** Shangirne Kharbanda

**Registration Number:** 20BAI1154

# OS LAB-8

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

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

### b. Segmentation fault

### c. Divide by zero error

## Code:

### stop_signal.c:

```c
#include<stdio.h>
#include<signal.h>
#include<unistd.h>
#include<stdlib.h>

void myhandler(int mysignal)
{
printf("Exiting\n");
exit(0);
}

int main(int argc, char * argv[])
{
signal(SIGTSTP, myhandler);

while(1)
{
printf("pid = %d\n", getpid());
sleep(1);
}
return 0;
}
```

```
alaric@alaric-virtual-machine:~/Desktop$ gcc stop_signal.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
pid = 3227
pid = 3227
pid = 3227
pid = 3227
pid = 3227
pid = 3227
pid = 3227
pid = 3227
pid = 3227
^ZExiting
```

## segmentation.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void myhandle(int signum)
{
printf("There is a segmentation fault!\nPlease check!");
exit(2);
}

int main()
{
int *p = NULL;
signal(SIGSEGV, myhandle);
*p = 1;

return 0;
}

```

```
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
There is a segmentation fault!
Please check!alaric@alaric-virtual-machine:~/Desktop$
```

## divide_by_zero.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void myhandle(int signum)
{
printf("There is a divide by zero error\n");
exit(2);
}

int main()
{
int result;
int v1, v2;
v1 = 100;
v2 = 0;
printf("dividing: %d/%d\n", v1, v2);
signal(SIGFPE, myhandle);
result = v1 / v2;

return 0;
}

```

```
alaric@alaric-virtual-machine:~/Desktop$ gcc divide_by_zero.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
dividing: 100/0
There is a divide by zero error
```

## 2. Child process that kills Parent process.

## Code:

### child_kills_parent.c:

```c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

void myhandle(int signo)
{
if (signo == SIGINT)
printf("Signal received by parent and got terminated...\n");
return;
}

int main()
{
pid_t pid, ppid;
ppid = getpid();
int x = 7;
printf("ppid = %d\n", ppid);

if ((pid = fork()) == 0)
{
x++;
printf("x:%d\n", x);
sleep(1);
printf("Signal sent to parent..\n");
kill(ppid, SIGINT);
}
else
{
x = x + 5;
printf("x:%d\n", x);
signal(SIGINT, myhandle);
sleep(5);
}

return 0;
}
```

```
alaric@alaric-virtual-machine:~/Desktop$ gcc child_kills_parent.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
ppid = 2820
x:12
x:8
Signal sent to parent..
Signal received by parent and got terminated...
```

**3. Write two c programs: One displaying the PID infinitely and the other program sending a signal to terminate the first program.**

## Code:

**victim.c:**

```c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
while(1)
{
printf("pid = %d\n", getpid());
sleep(1);
}

return(0);
}
```

**killer.c:**

```c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
signal(SIGKILL, SIG_DFL);
int pid;
printf("Enter process ID: ");
scanf("%d",&pid);
kill(pid, SIGKILL);

return(0);
}
```

```
alaric@alaric-virtual-machine:~/Desktop$ gcc victim.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
pid = 4958
Killed
```

```
alaric@alaric-virtual-machine:~/Desktop$ gcc killer.c
alaric@alaric-virtual-machine:~/Desktop$ ./a.out
Enter process ID: 4958
```