**Name: Rohan Nivas Chavan**

**PRN:2019BTEIT00084**

**UOS LAB**

## IPC: Interrupts and Signals: signal(any fives type of signal ), alarm, kill, raise, killpg, signal , sigaction

**2.4** Write aapplication or program to trap a ctrl-c but not quit onthis signal.

**Objectives:**

1. To learn about IPC through signal.
2. To know the process management of Unix/Linux OS
3. Use of system call to write effective application programs.

**Theory:**

1.We can use signal handling in C for this. When Ctrl+C is pressed, SIGINT signal is generated, we can catch this signal and run our defined signal handler.

2.C standard defines following 6 signals in signal.h header file.

SIGABRT – abnormal termination.

SIGFPE – floating point exception.

SIGILL – invalid instruction.

SIGINT – interactive attention request sent to the program.

SIGSEGV – invalid memory access.

SIGTERM – termination request sent to the program.

**Data Dictionary:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr Number | Variable/Function | Datatype | Use |
|  |  |  |  |
| 1 | sigproc | void | Function to disable Ctrl-c |
|  |  |  |  |
| 2 | quitproc | void | Function to quit using Ctrl-backslash. |
|  |  |  |  |
| 3 | quitprocess | void | Function to quit using Ctrl-z. |
|  |  |  |  |

**Program:**

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

int ctrl\_c\_count = 0;

void (\* old\_handler)(int);

void ctrl\_c(int);

int main()

{

int c;

old\_handler = signal(SIGINT, ctrl\_c);

while ((c = getchar())!='\n');

printf("ctrl-c count = %d\n", ctrl\_c\_count); (void) signal(SIGINT, ctrl\_c); for (;;);

return 0;

}

void ctrl\_c(int signum)

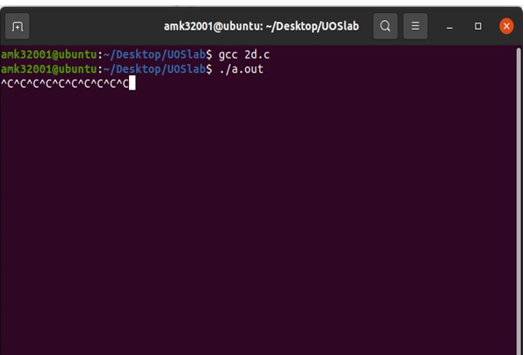
{

//(void) signal(SIGINT, ctrl\_c);

++ctrl\_c\_count;

}

**Output:**



**Conclusion:**

Signals like SIGINT which is generated by ctrl+c can be handled by replacing its old handler with the new behavior.

**References:**

<https://www.geeksforgeeks.org/write-a-c-program-that-doesnt-terminate-when-ctrlc-is-pressed/>