**ASSIGNMENT-9**

**Aim:**

This program should count the number of times the user has sent the SIGINT signal to the running process. Pressing Ctl-C from the keyboard send this signal. When the process receives the SIGTSTP signal (Ctl-Z), it should print to the terminal the number of SIGINT signals it has received. After it receives 5 SIGINT signals, the program should prompt the user to exit. If the user does not respond within 10 seconds, an SIGALRM signal should force the program to exit.

**Code:**

#include<stdio.h>

#include<signal.h>

#include<unistd.h>

#include <stdlib.h>

int ctrl\_c\_count = 0;

int got\_response = 0;

#define THRESHOLD 5

void sig\_handler(int signo)

{

ctrl\_c\_count++;

if (ctrl\_c\_count >= THRESHOLD) {

char answer[30];

printf("\nReally exit? [Y/n]: ");

alarm(10);

fflush(stdout);

fgets(answer, sizeof(answer), stdin);

if (answer[0] == 'n' || answer[0] == 'N') {

printf("\nContinuing\n");

fflush(stdout);

ctrl\_c\_count = 0;

}

else {

printf("\nExiting...\n");

fflush(stdout);

exit(0);

}

}

}

void catch\_ctrlz(int sig\_num)

{

printf("\n\nSo far, '%d' Ctrl-C presses were counted\n\n", ctrl\_c\_count);

fflush(stdout);

signal(SIGTSTP, catch\_ctrlz);

}

int main(void)

{

if (signal(SIGINT, sig\_handler) == SIG\_ERR)

printf("\ncan't catch SIGINT\n");

signal(SIGTSTP, catch\_ctrlz) ;

while(1){

}

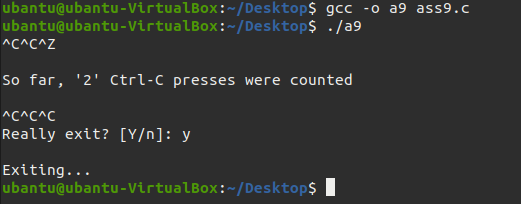
while(1)

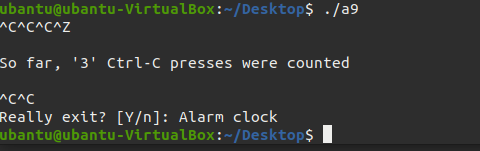
sleep(1);

return 0;

}

**Output:**





**PRACTICAL-11**

**Aim:**

**a)** Catch the terminal interrupts by using appropriate signals.

**Code:**

#include<stdio.h>

#include<signal.h>

#include<unistd.h>

void sig\_handler(int signo){

if (signo == SIGINT)

printf("received SIGINT\n");

}

int main(void) {

if (signal(SIGINT, sig\_handler) == SIG\_ERR)

printf("\ncan't catch SIGINT\n");

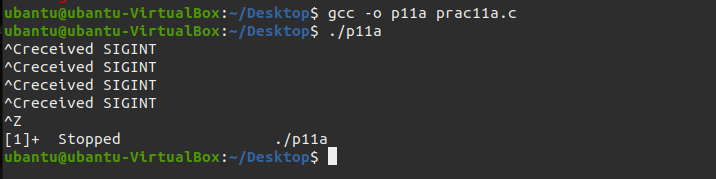
while(1)

sleep(1);

return 0;

}

**Output:**



**b)** Schedule a signal for a process.

**Code:**

#include<stdio.h>

#include<unistd.h>

#include<signal.h>

void sig\_handler (int signum) {

printf("Inside handler function\n");

}

int main(){

signal (SIGALRM, sig\_handler);

alarm(2);

for (int i=1;;i++){

printf("%d: Inside main function\n", i);

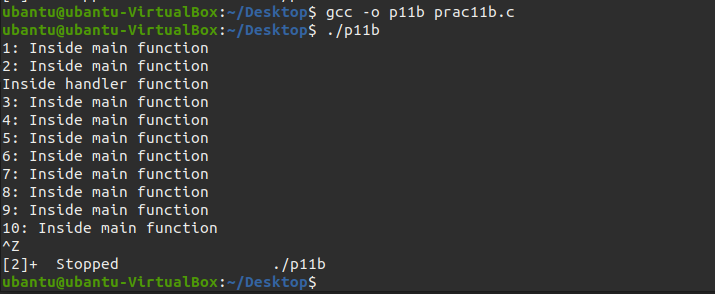
sleep (1);

}

return 0;

}

**Output:**



**C)** Kill the process using signals.

**Code:**

#include<stdio.h>

#include<signal.h>

void myHandler(int signum){

printf("the signal caught %d",signum);

}

int main(){

signal(SIGINT,myHandler);

signal(SIGTSTP,myHandler);

printf(" infinite loop starts...");

for (;;) {

printf("hello\n");

}

return 0;

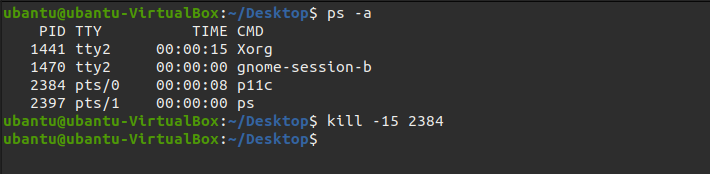
}

**Output:**

Process goes into infinite loop



Here, we have checked running process id and then kill that process by SIGTERM(15).



The process has been terminated.



**ASSIGNMENT-10 & PRACTICAL 12**

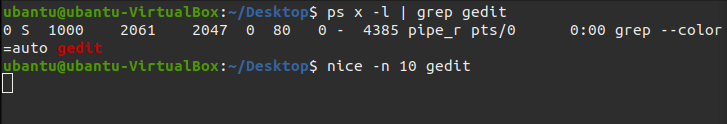
**Aim:**

Create an infinite running process. Initialize the process with 10 as nice value and after the process starts change it by 15.

Change the Priority of a Running Process

**Command with output:**

Here, we have initialized process with nice value 10. After that process has been started.



Here, we have change nice value as 15.

