Name: Vaishnavi Vivekanand Kulkarni

PRN: **2020BTEIT00031** **Class: TY**

2e: IPC: Interrupts and Signals: signal(any fives type of signal), alarm, kill, raise, killpg, signal , sigaction

Q.2 e) Write a program to send signal by five different signal sending system calls

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.

Description:

TYPES OF SIGNALS:

1. SIGHUP:-Hang up detected on controlling terminal or death of controlling process
   SIGINT:-Issued if the user sends an interrupt signal (Ctrl + C)
2. SIGQUIT:-Issued if the user sends a quit signal (Ctrl + D)
3. SIGFPE:-Issued if an illegal mathematical operation is attempted
4. SIGKILL:-If a process gets this signal it must quit immediately and will not perform any cleanup operations
5. SIGALRM:-Alarm clock signal (used for timers)

Types of System Call:

1. **Alarm System Call**:

1. alarm - set an alarm clock for delivery of a signal.alarm() arranges for a SIGALRM signal to be delivered to the process in seconds seconds.If seconds is zero, no new alarm() is scheduled.In any event any previously set alarm() is cancelled.

2. **kill System Call**:

1. The kill() system call can be used to send any signal to any process group or process. If pid is positive, then signal sig is sent to the process with the ID specified by pid.
2. If pid equals 0, then sig is sent to every process in the process group of the calling process.If pid equals -1, then sig is sentto every process for which the calling process has permission to send signals, except for process 1 (init).If pid is less than -1, then sig is sent to every process in the process group whose ID is -pid.

3. **Raise System call :**

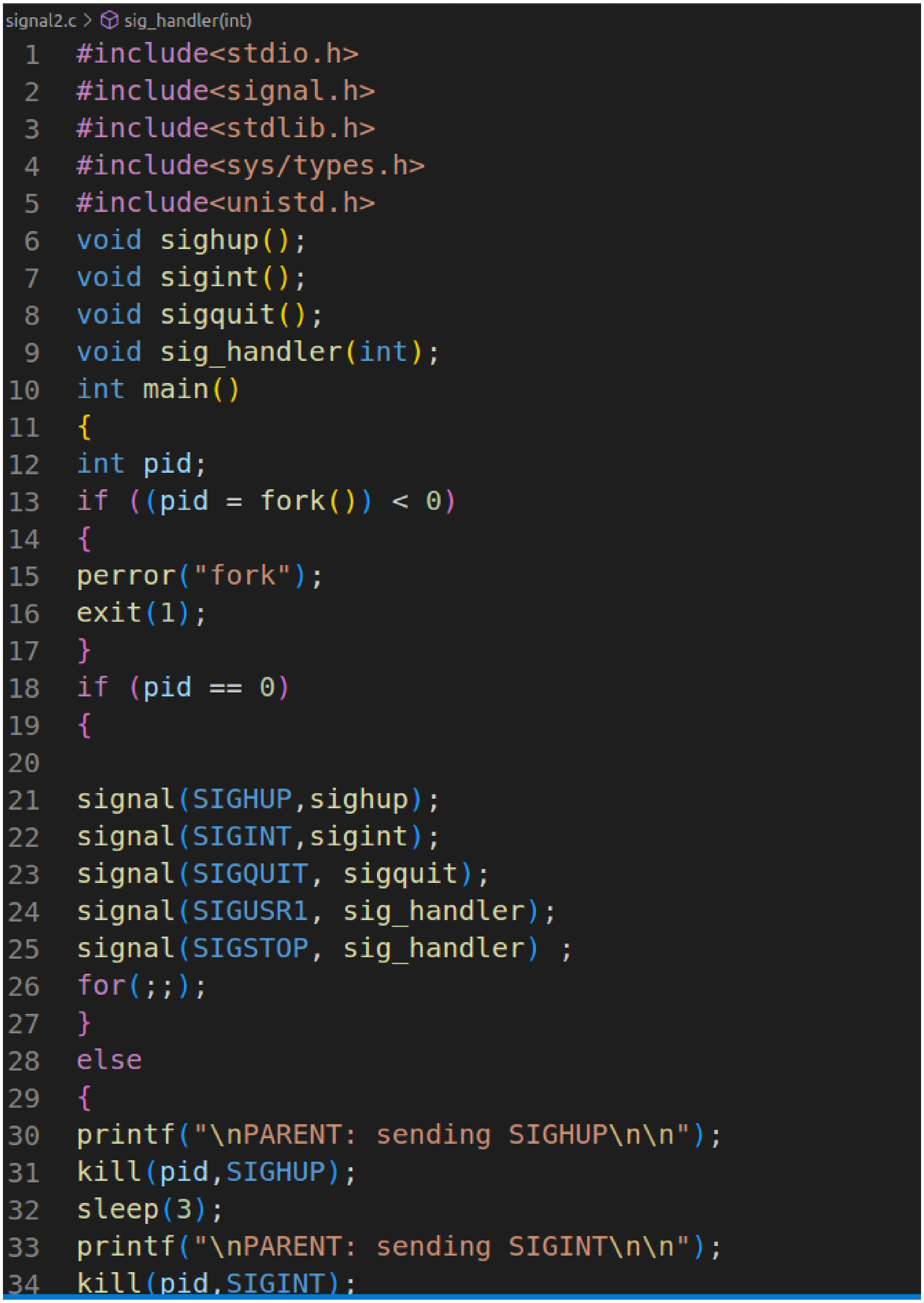
1. Send a signal to the caller The raise() function sends a signal to the calling process or threaasingle- threaded program it is equivalent to kill(getpid(), sig);

2. In a multithreaded program it is equivalent to pthread_kill (pthread_self(), sig); If the signal causes a handler to be called,raise() will return only after the signal handler has returned.

4. **Killpg System Call**: send signal to a process group .

5. **Sigaction System Call**: examine and change a signal action.

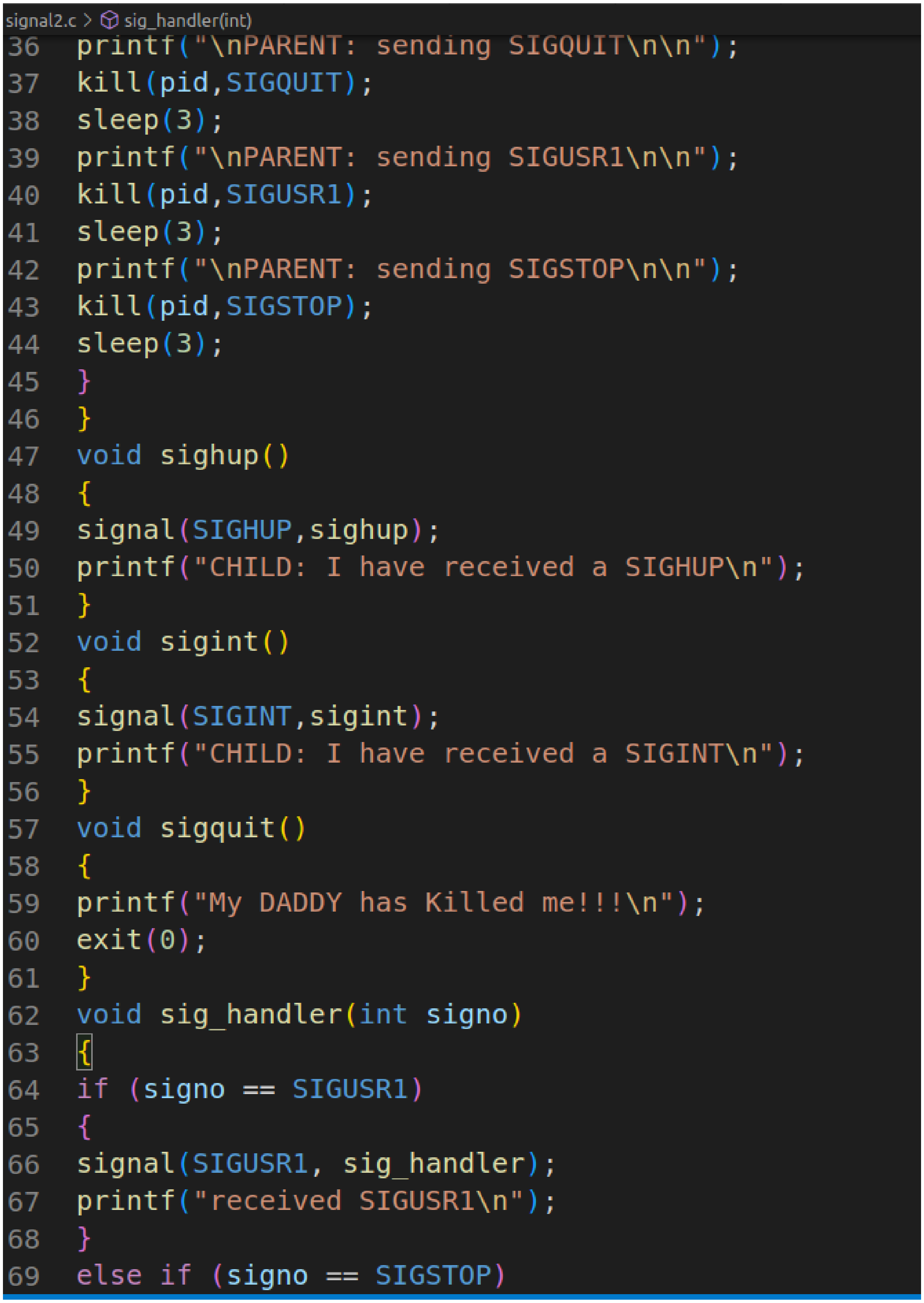
Program:



signal2.c > sig_handler(int)

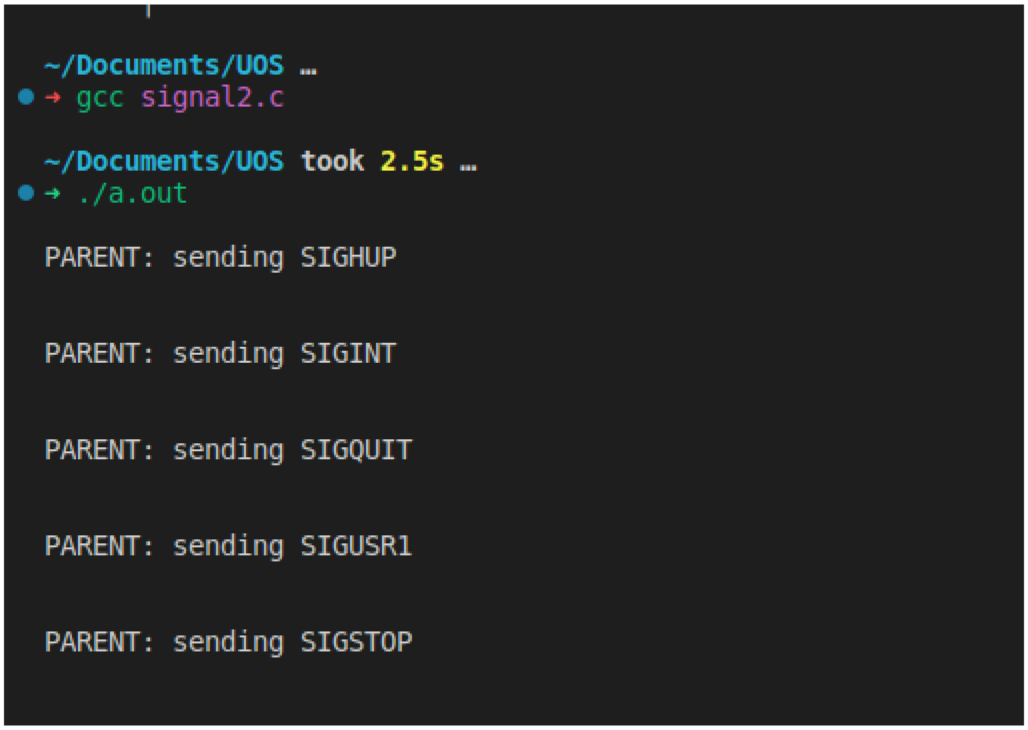
```c
#include<stdio.h>
#include<signal.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
void sighup();
void sigint();
void sigquit();
void sig_handler(int);
int main()
{
int pid;
if ((pid = fork()) < 0)
{
perror("fork");
exit(1);
}
if (pid == 0)
{

signal(SIGHUP,sighup);
signal(SIGINT,sigint);
signal(SIGQUIT, sigquit);
signal(SIGUSR1, sig_handler);
signal(SIGSTOP, sig_handler) ;
for(;;);
}
else
{
printf("\nPARENT: sending SIGHUP\n\n");
kill(pid,SIGHUP);
sleep(3);
printf("\nPARENT: sending SIGINT\n\n");
kill(pid,SIGINT);
```


signal2.c > sig_handler(int)

```c
printf("\nPARENT: sending SIGQUIT\n\n");
kill(pid,SIGQUIT);
sleep(3);
printf("\nPARENT: sending SIGUSR1\n\n");
kill(pid,SIGUSR1);
sleep(3);
printf("\nPARENT: sending SIGSTOP\n\n");
kill(pid,SIGSTOP);
sleep(3);
}
}
void sighup()
{
signal(SIGHUP,sighup);
printf("CHILD: I have received a SIGHUP\n");
}
void sigint()
{
signal(SIGINT,sigint);
printf("CHILD: I have received a SIGINT\n");
}
void sigquit()
{
printf("My DADDY has Killed me!!!\n");
exit(0);
}
void sig_handler(int signo)
{
if (signo == SIGUSR1)
{
signal(SIGUSR1, sig_handler);
printf("received SIGUSR1\n");
}
else if (signo == SIGSTOP)
```

Output:



```
~/Documents/UOS …
● → gcc signal2.c

~/Documents/UOS took 2.5s …
● → ./a.out

PARENT: sending SIGHUP


PARENT: sending SIGINT


PARENT: sending SIGQUIT


PARENT: sending SIGUSR1


PARENT: sending SIGSTOP
```

Conclusion:

1. The following assignment deals with providing the guideline about the various types of system call and their usage in IPC and management about process in windows and linux operating system and written the application using the signal and alarm system call.

References:

1) **www.cs.cf.ac.uk/Dave/C/CE.html**

2) **ftp://10.1013.3/pub/UOS..../**