PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for

assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.

Name: J SAI NISHITH

Date: 02-12-2019

Signature: JSN

1. Write a program that forks 2 child processes. Child 1 prints a message every 1 sec. Child 2 sleeps for 10 secs, then kills child 1, then sleeps for 10 secs and terminates. The parent waits for both child processes to terminate then exits. Each process should print a 1-line message including its pid before/after each significant action such as sleep, kill, terminate, etc.

In the main function, I have created a new child process(say child 1) using fork(); within child 1(fork()==0), I have printed a message from child 1 with it's PID with gap of 1 second between successive messages using sleep() function.

In the parent process, I have created one more child process(say child 2) using fork(); I made this process to sleep for 10 seconds using sleep() function and then printed a message from child 2 with it's PID . Then using child 2 i have killed process using kill(PidOfChild1,SIGKILL) command and printed a message from child 2 after killing the child 1 and made child 2 to sleep for 10 seconds using sleep() function and terminated it with exit(0).

Then in parent process, I made it wait till both the processes complete(used wait(NULL)) and then i have printed a message from parent and then terminated it.

SAMPLE OUTPUT:

```
I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child1 with process id 5918

I am Child2 with process id 5919,slept for 10 seconds and about to kill child1

I am Child2 with process id 5919 and killed child1 and about to sleep for 10 seconds

Parent process with process id 5917 about to exit
```

4. Measure the time taken for context switching. Create two processes that switch between themselves by sending signals to one another (use kill() and signal()). Measure the time for a large number of switches. Note the system details such as OS type and version, CPU type and speed, amount of cache/RAM.

In the main function, I have created a new child process(say child 1) using fork(). IN the parent process i.e. fork()!=0; i have created a pipe for sharing time variable and to schedule the process immediately, so that we get the scheduled time.In the parent process i have started the clock, printed a message in parent, sent a signal using

```
                    kill (child1,SIGINT);
```

Then the signal goes to child and then i have printed a message if signal is successfully sent to the child. I have measured the time to go to the child and back to the parent,essentially this is 2 context switches, so I have divided the obtained time by 2 and then printed this time.

Here I have used the signal SIGINT and used the header #include <signal.h> for functional support for signals.

For OS details:

I have used a header file #include <sys/utsname.h> to get the OS details.I created a pre defined utsname name structure info.

For OS name : We have attribute of sysname in utsname struct;

For OS Version : We have attribute of version in utsname struct;

So i have accessed the OS name,Kernel version,OS version with info.syname,info.release,info.version respectively.

For CPU Details :

I created a file pointer pf;

I used the help of command

```
        $ inxi -C _Which fetches the CPU details
```

Please Note : If inxi isn't present we should do

```
                sudo apt get inix
```

For functioning of inix.

I have used the popen and stored the output buffer to string and then the printed the string to get details of the CPU which includes the Cache.

SAMPLE OUTPUT:

Parent is about to send the signal SIGINT

Child has received a signal SIGINT from parent

Time taken for a switch is 0.000017 seconds

System details:

Operating System : Linux

Operating System Version: #39~18.04.1-Ubuntu SMP Tue Nov 12 11:09:50 UTC 2019

Kernel Version: 5.0.0-36-generic

CPU details:

Architecture of processor: x86_64

Other details :

CPU: 6 core Intel Core i7-8750H (-MT-MCP-) cache: 9216 KB