RIPHAH INTERNATIONAL UNIVERSITY

Name: Deher Zainab

Sap ID: 49710

BSCS-5th Semester

OS Lab # 8

Submitted to:

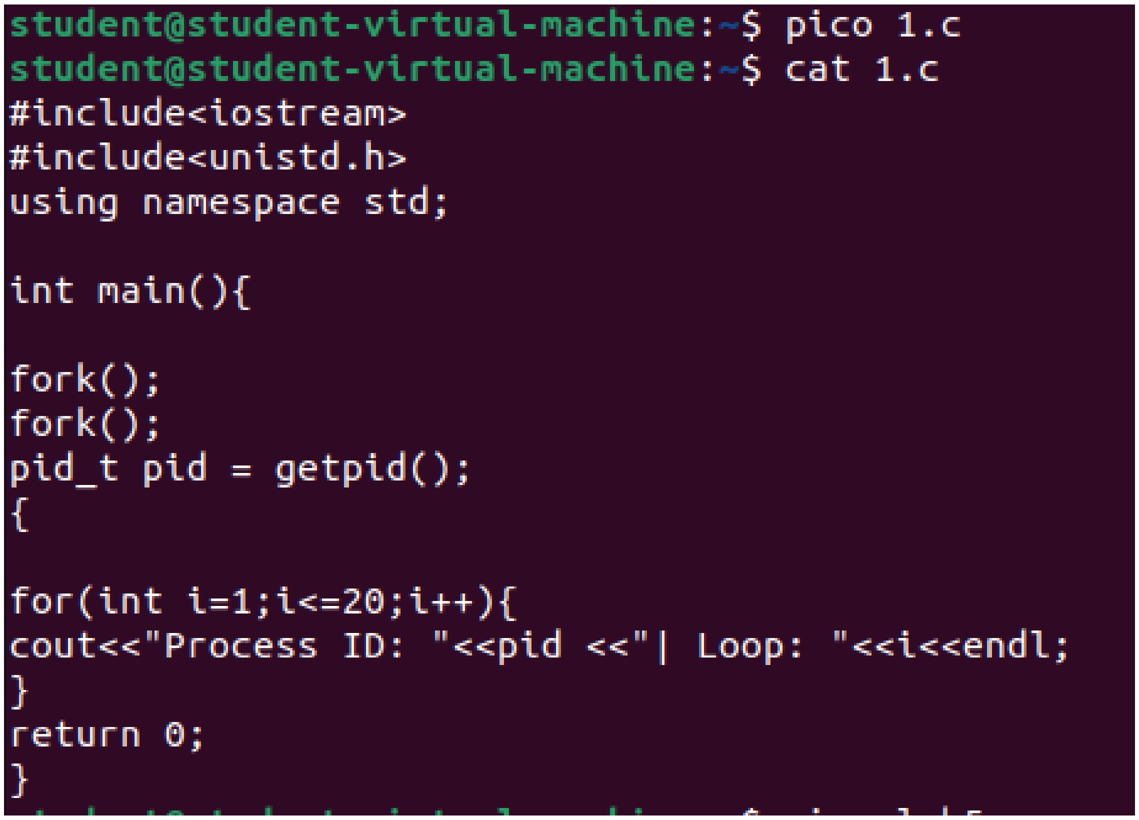
Mam Ayesha Akram

Lab Task # 1:

File name **1.c**

Write a C++ program that uses two fork() calls . Each process should:

1. Print its process ID (PID) and a loop value from 1 to 20**.**



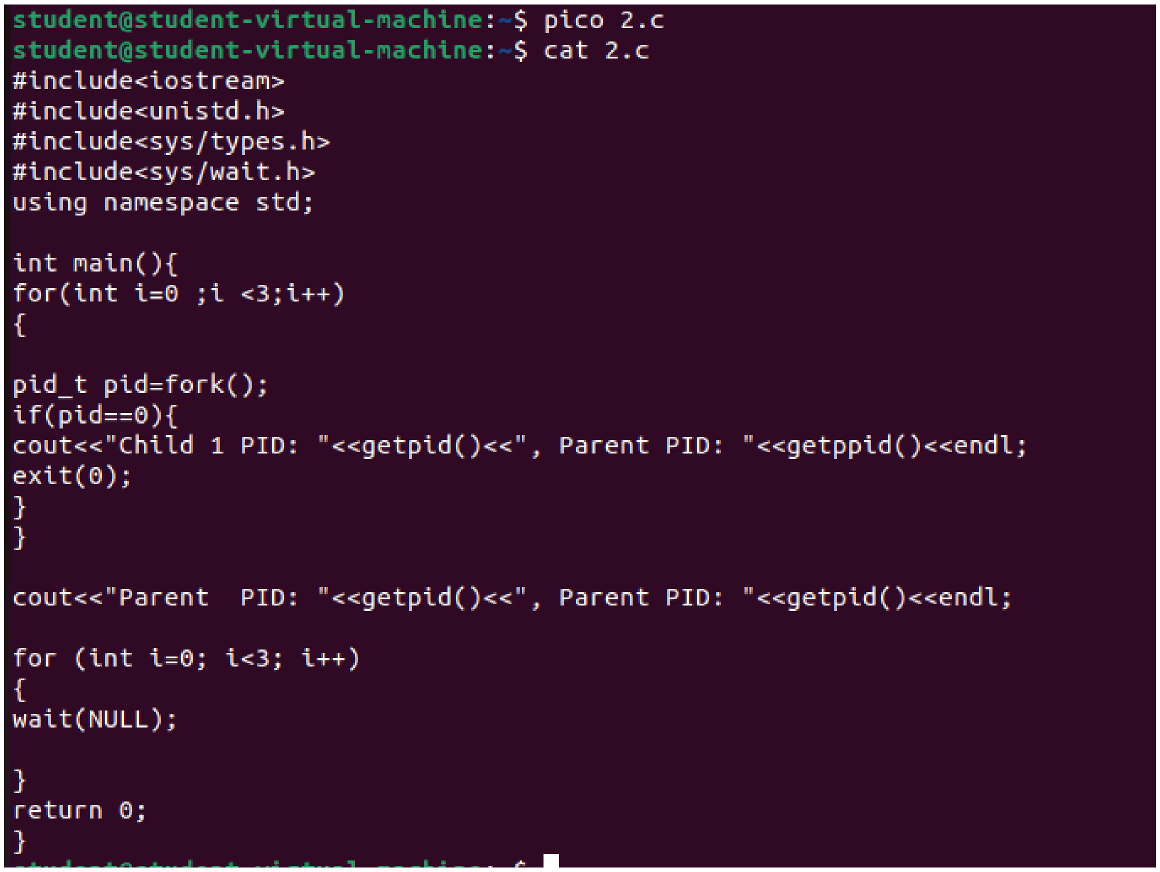
```
student@student-virtual-machine:~$ pico 1.c
student@student-virtual-machine:~$ cat 1.c
#include<iostream>
#include<unistd.h>
using namespace std;

int main(){

fork();
fork();
pid_t pid = getpid();
{

for(int i=1;i<=20;i++){
cout<<"Process ID: "<<pid <<"| Loop: "<<i<<endl;
}
return 0;
}
```

Lab Task # 2:

File name 2.c

Write a C++ program that creates three child processes using the fork() system call. Each child process should:

1. **Print its own process ID (PID) and its parent process ID (PPID).**
2. **Terminate using exit().**
3. **After creating the child processes, the parent process should print its own PID.**



```
student@student-virtual-machine:~$ pico 2.c
student@student-virtual-machine:~$ cat 2.c
#include<iostream>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
using namespace std;

int main(){
for(int i=0 ;i <3;i++)
{

pid_t pid=fork();
if(pid==0){
cout<<"Child 1 PID: "<<getpid()<<", Parent PID: "<<getppid()<<endl;
exit(0);
}
}

cout<<"Parent  PID: "<<getpid()<<", Parent PID: "<<getpid()<<endl;

for (int i=0; i<3; i++)
{
wait(NULL);

}
return 0;
}
```

Lab Task # 3:

Explain the working of system calls with its types and examples according to your understanding.

System calls are the way programs (user-level processes) interact with the operating system (kernel-level).

How System Calls Work:

1. **User Program Requests a Service**
   A program needs to do something like read a file, write to disk, or create a process — things only the OS can do.
2. **System Call Invoked**
   The program uses a predefined function (like read(), write(), fork(), etc.) which triggers a **system call**.
3. **Mode Switch: User to Kernel**
   The CPU switches from **user mode** to **kernel mode** to allow access to protected system resources.
4. **Execution in Kernel**
   The OS performs the requested operation (e.g., reading a file from disk).
5. **Return to User Mode**
   Once done, the OS switches back to user mode and returns the result to the program.

Example:

When a program call read (), it doesn't read the file directly. It triggers a system call, and the OS reads the file and gives the data back.

Types of System Calls:

- **Process Control**

Create, execute, terminate processes.

Examples: fork(), exec(), exit(), wait()

- **File Management**

Open, read, write, close files.

Examples: open(), read(), write(), close()

- **Device Management**

Request or release devices.

Examples: ioctl(), read(), write()

- **Information Maintenance**

Get or set system data/time, process information.

Examples: getpid(), alarm(), sleep()

- **Communication**

For inter-process communication (IPC).

Examples: pipe(), shmget(), mmap(), msgsnd()