Ayesha Zubair

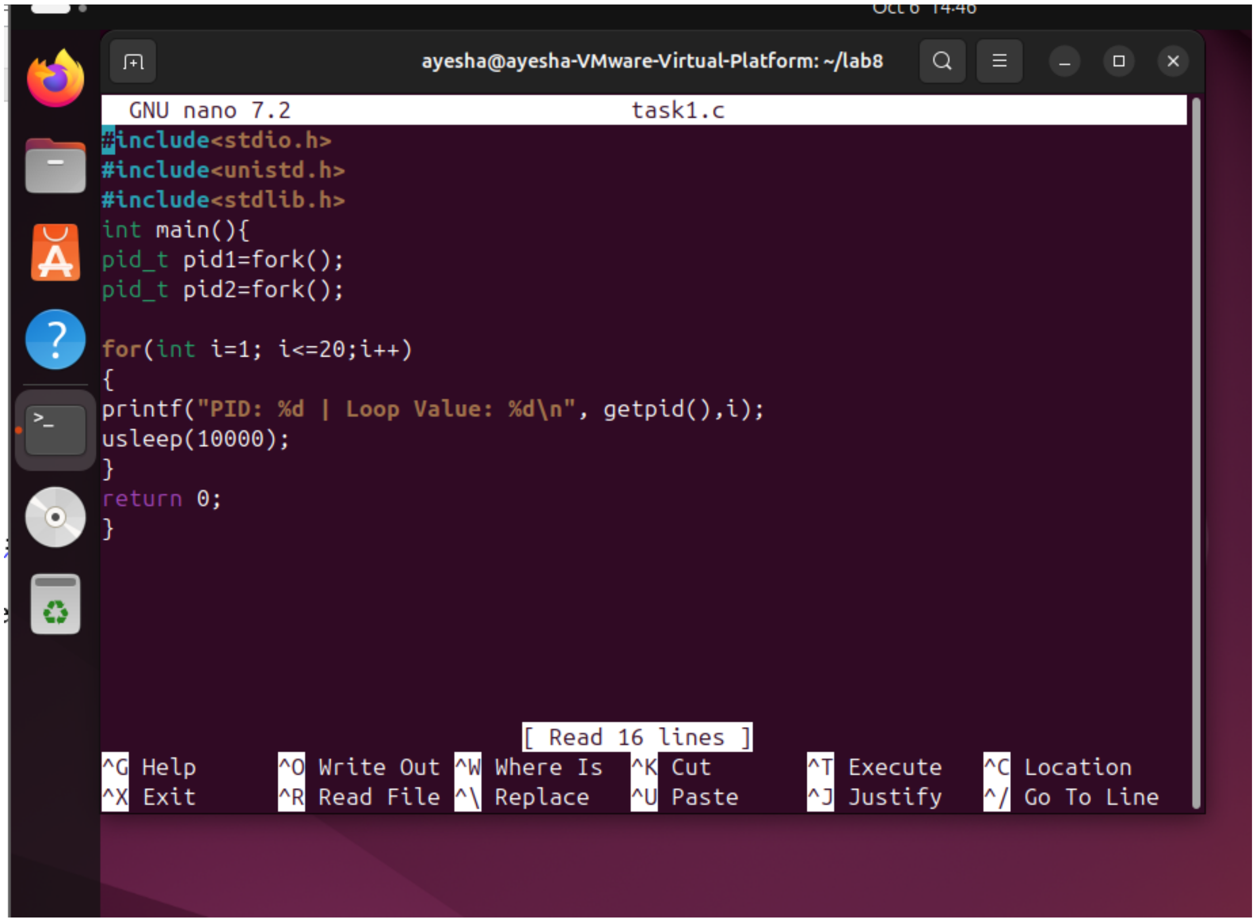
52916

Lab 8

Task 1: 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.

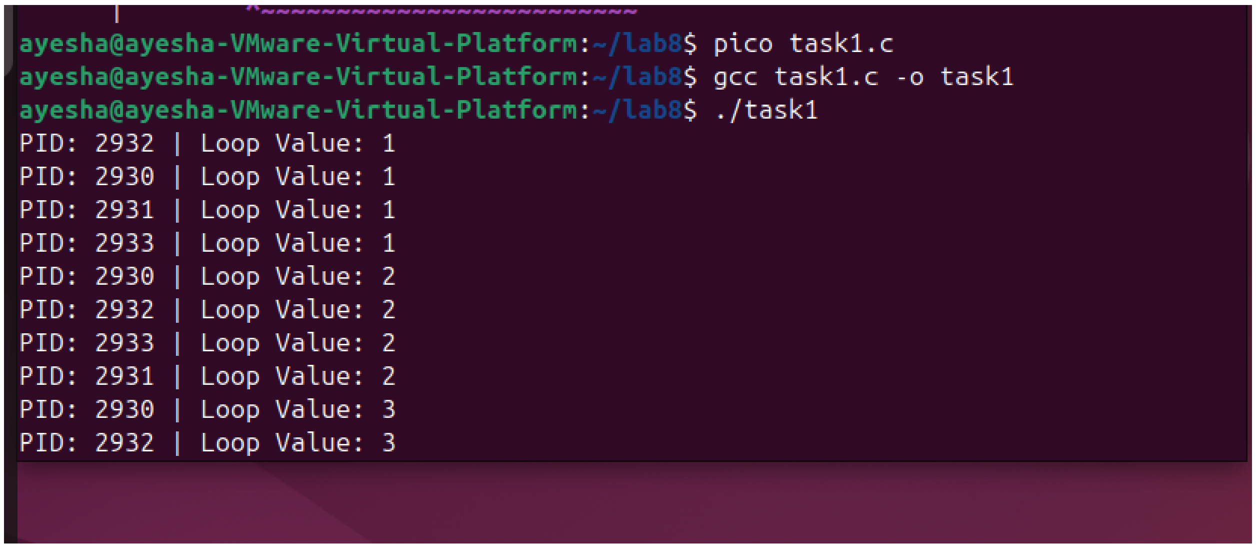
Code:



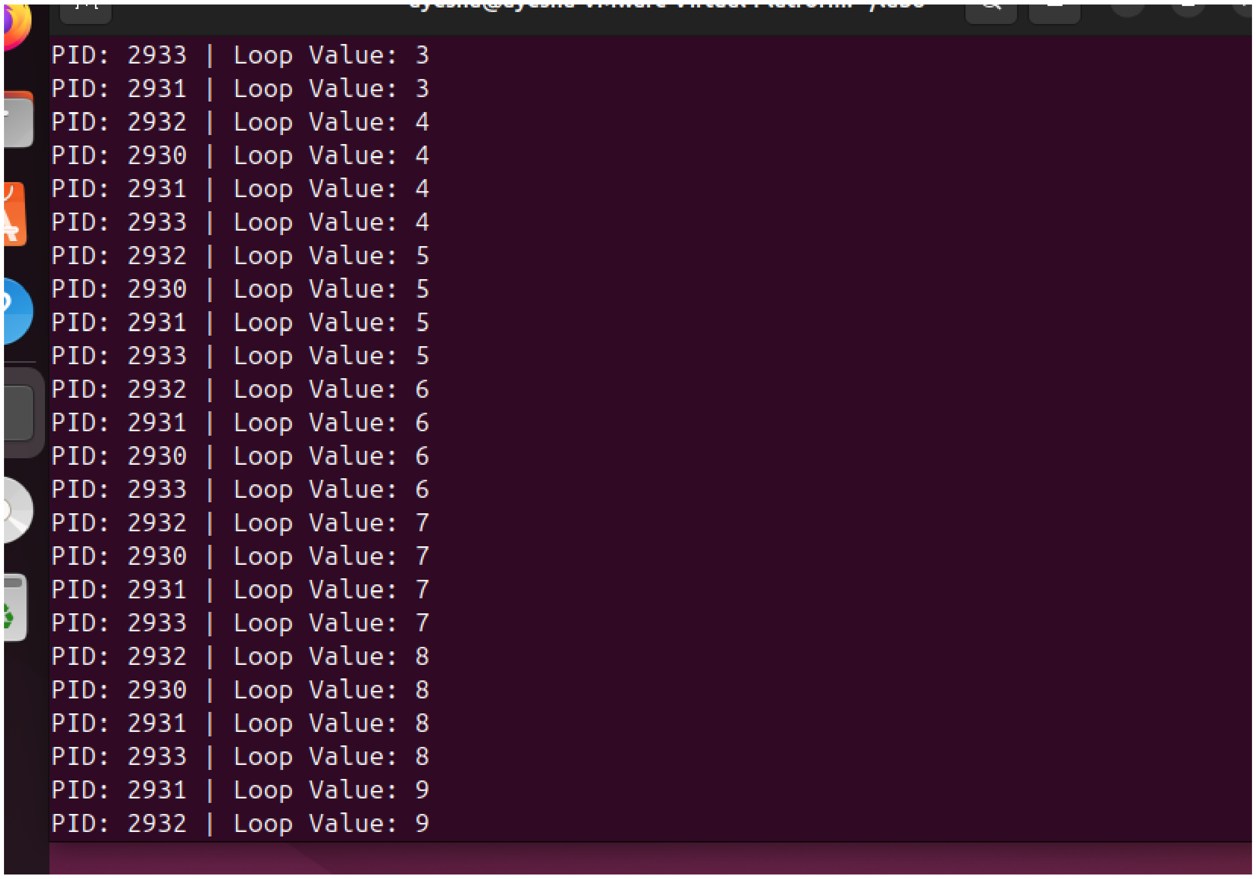
```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(){
pid_t pid1=fork();
pid_t pid2=fork();

for(int i=1; i<=20;i++)
{
printf("PID: %d | Loop Value: %d\n", getpid(),i);
usleep(10000);
}
return 0;
}
```

```
                              [ Read 16 lines ]
^G Help          ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit          ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

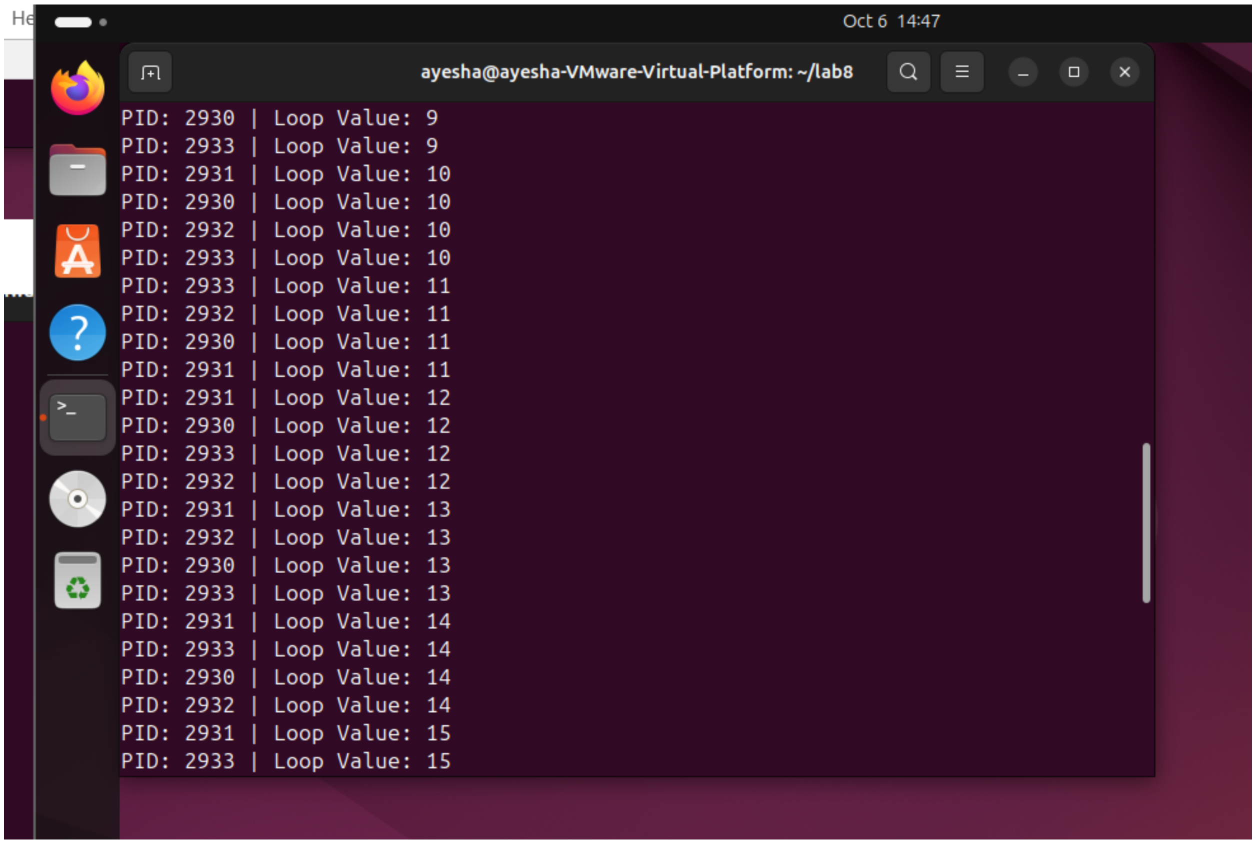
Output:



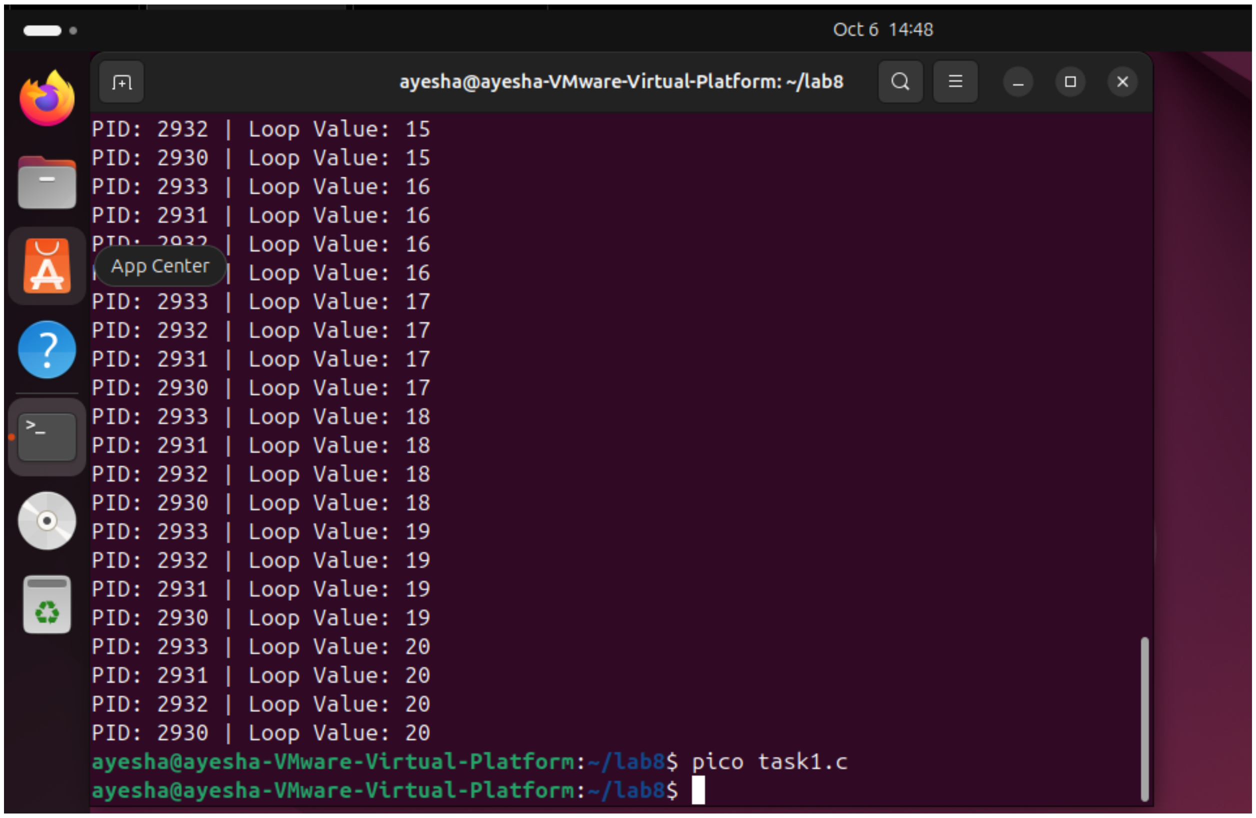
```
ayesha@ayesha-VMware-Virtual-Platform:~/lab8$ pico task1.c
ayesha@ayesha-VMware-Virtual-Platform:~/lab8$ gcc task1.c -o task1
ayesha@ayesha-VMware-Virtual-Platform:~/lab8$ ./task1
PID: 2932 | Loop Value: 1
PID: 2930 | Loop Value: 1
PID: 2931 | Loop Value: 1
PID: 2933 | Loop Value: 1
PID: 2930 | Loop Value: 2
PID: 2932 | Loop Value: 2
PID: 2933 | Loop Value: 2
PID: 2931 | Loop Value: 2
PID: 2930 | Loop Value: 3
PID: 2932 | Loop Value: 3
```



```
PID: 2933 | Loop Value: 3
PID: 2931 | Loop Value: 3
PID: 2932 | Loop Value: 4
PID: 2930 | Loop Value: 4
PID: 2931 | Loop Value: 4
PID: 2933 | Loop Value: 4
PID: 2932 | Loop Value: 5
PID: 2930 | Loop Value: 5
PID: 2931 | Loop Value: 5
PID: 2933 | Loop Value: 5
PID: 2932 | Loop Value: 6
PID: 2931 | Loop Value: 6
PID: 2930 | Loop Value: 6
PID: 2933 | Loop Value: 6
PID: 2932 | Loop Value: 7
PID: 2930 | Loop Value: 7
PID: 2931 | Loop Value: 7
PID: 2933 | Loop Value: 7
PID: 2932 | Loop Value: 8
PID: 2930 | Loop Value: 8
PID: 2931 | Loop Value: 8
PID: 2933 | Loop Value: 8
PID: 2931 | Loop Value: 9
PID: 2932 | Loop Value: 9
```



```
PID: 2930 | Loop Value: 9
PID: 2933 | Loop Value: 9
PID: 2931 | Loop Value: 10
PID: 2930 | Loop Value: 10
PID: 2932 | Loop Value: 10
PID: 2933 | Loop Value: 10
PID: 2933 | Loop Value: 11
PID: 2932 | Loop Value: 11
PID: 2930 | Loop Value: 11
PID: 2931 | Loop Value: 11
PID: 2931 | Loop Value: 12
PID: 2930 | Loop Value: 12
PID: 2933 | Loop Value: 12
PID: 2932 | Loop Value: 12
PID: 2931 | Loop Value: 13
PID: 2932 | Loop Value: 13
PID: 2930 | Loop Value: 13
PID: 2933 | Loop Value: 13
PID: 2931 | Loop Value: 14
PID: 2933 | Loop Value: 14
PID: 2930 | Loop Value: 14
PID: 2932 | Loop Value: 14
PID: 2931 | Loop Value: 15
PID: 2933 | Loop Value: 15
```

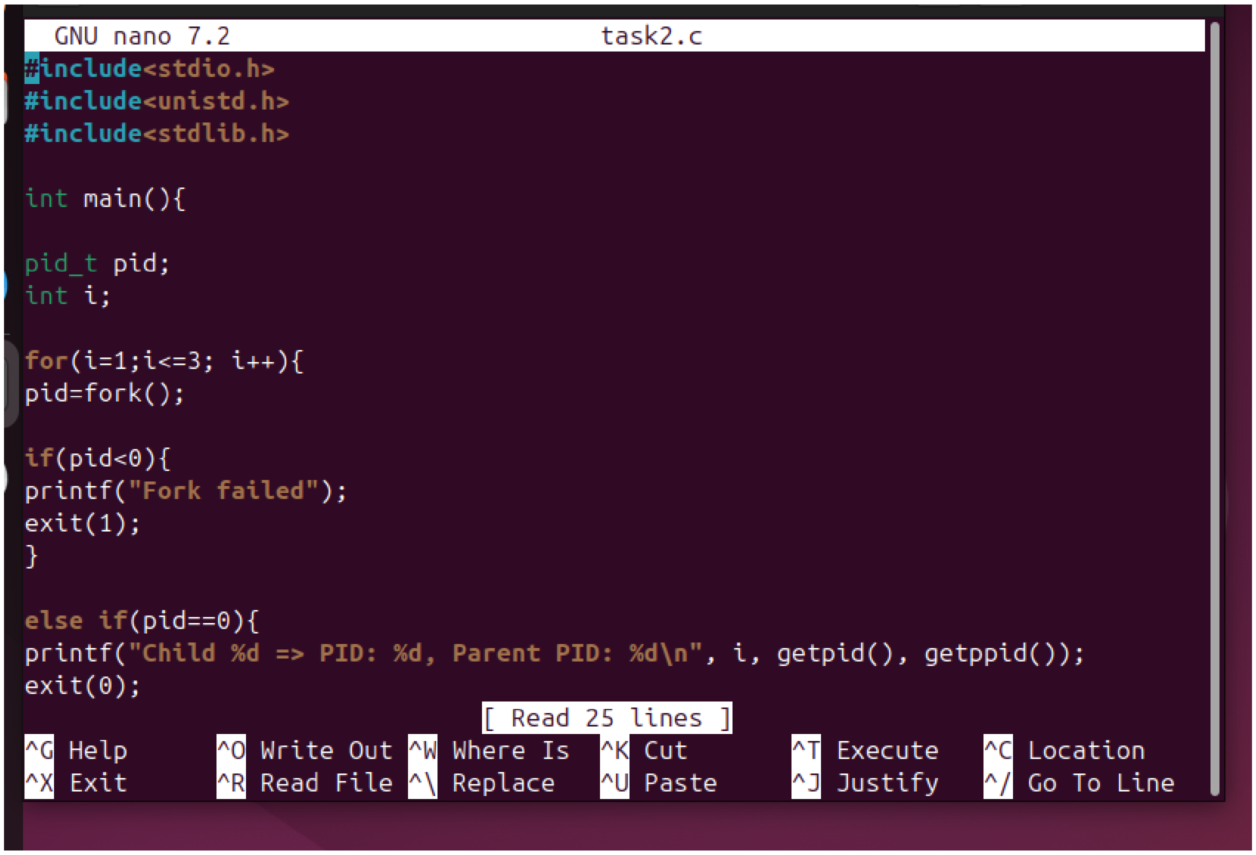


```
PID: 2932 | Loop Value: 15
PID: 2930 | Loop Value: 15
PID: 2933 | Loop Value: 16
PID: 2931 | Loop Value: 16
PID: 2932 | Loop Value: 16
[illegible] | Loop Value: 16
PID: 2933 | Loop Value: 17
PID: 2932 | Loop Value: 17
PID: 2931 | Loop Value: 17
PID: 2930 | Loop Value: 17
PID: 2933 | Loop Value: 18
PID: 2931 | Loop Value: 18
PID: 2932 | Loop Value: 18
PID: 2930 | Loop Value: 18
PID: 2933 | Loop Value: 19
PID: 2932 | Loop Value: 19
PID: 2931 | Loop Value: 19
PID: 2930 | Loop Value: 19
PID: 2933 | Loop Value: 20
PID: 2931 | Loop Value: 20
PID: 2932 | Loop Value: 20
PID: 2930 | Loop Value: 20
ayesha@ayesha-VMware-Virtual-Platform:~/lab8$ pico task1.c
ayesha@ayesha-VMware-Virtual-Platform:~/lab8$
```

Task 2: Write a C++ program that creates three child processes using the `fork()` system call. Each child process should:

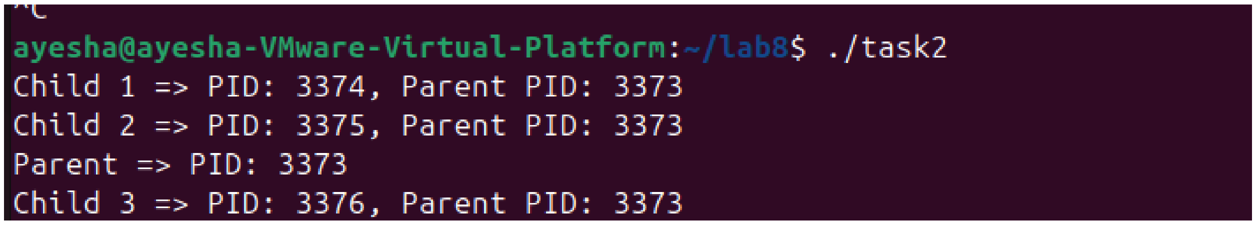
1. Print its own process ID (PID) and its parent process ID (PPID).
2. Terminate using `exit()`.
3. After creating the child processes, the parent process should print its own PID.

Code:



```
  GNU nano 7.2                          task2.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>

int main(){

pid_t pid;
int i;

for(i=1;i<=3; i++){
pid=fork();

if(pid<0){
printf("Fork failed");
exit(1);
}

else if(pid==0){
printf("Child %d => PID: %d, Parent PID: %d\n", i, getpid(), getppid());
exit(0);
                              [ Read 25 lines ]
^G Help        ^O Write Out ^W Where Is  ^K Cut        ^T Execute   ^C Location
^X Exit        ^R Read File ^\ Replace   ^U Paste      ^J Justify   ^/ Go To Line
```

Output:



```
^C
ayesha@ayesha-VMware-Virtual-Platform:~/lab8$ ./task2
Child 1 => PID: 3374, Parent PID: 3373
Child 2 => PID: 3375, Parent PID: 3373
Parent => PID: 3373
Child 3 => PID: 3376, Parent PID: 3373
```

Task3: Explain the working of system calls with its types and examples according to your understanding.

System Calls: These are the methods by which users request services from the kernel.

Fork(): This creates a copy of the current process with different process ids.

exec(): Replaces the current process with a new program but the id remains the same.

Wait(): The parent process waits for the child process execution to finish and then continues its own execution.

Exit(): Exit system calls ends a process.

Getpid(): Returns the process id of the current process.

Getppid(): Returns process id of the parent process.