**Kathmandu University**

**Department of Computer Science and Engineering**

**Dhulikhel, Kavre**



**A Lab Report #1**

**[ Course title: COMP 307]**

**Submitted by:**

**Bisheshwor Neupane(35)**

**Submitted to:**

**Mr. Dhiraj Shrestha**

**Department of Computer Science and Engineering**

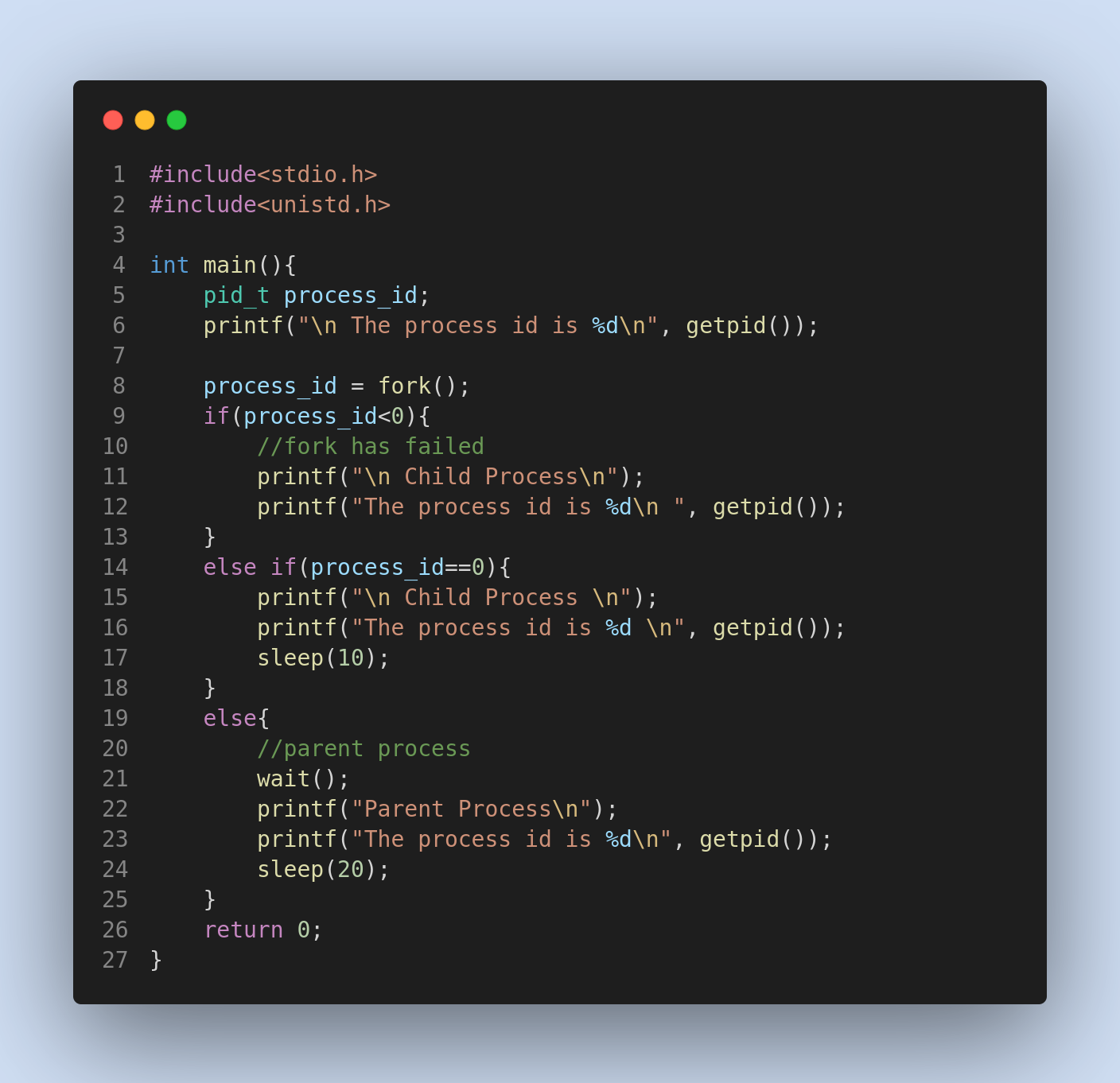
**Submission Date:**

**August 20, 2021**

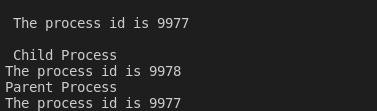
**(Understanding Process Concepts)**

1. **Write a C program to implement process System calls ? [Hint use fork())]**

**Code:**



Output:



After getting an output, we assure that a child process is called first. Only after it’s termination, parent process is called.

1. **How many processes are created in a given program?**

#include <stdio.h>

#include <unistd.h>

int main()

{

int i;

for (i = 0; i < 4; i++)

fork();

return 0;

}

**Solution:**

As fork() method is called 4 times, calculated number of child processes is 15. i.e. 2^n - 1 = 2^4 - 1 = 15 processes. Including parent process, the total number of processes is 16.

When fork gets executed, a new process is created whereas 3 forks are left to be executed. The first fork call creates a child process p1 and remaining 3 more fork calls have to be executed. When the second fork calls gets executed, the parent process creates 3 more processes p2, p3, p4. The child process also creates another 3 processes p5, p6, p7. Subsequently, the total fork calls left is two. To sum up, 15 forks are created.

1. **When will line J be reached?**

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

int main()

{

pid\_t pid;

/\* fork a child process \*/

pid = fork();

if (pid < 0) { /\* error occurred \*/

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0) { /\* child process \*/

execlp("/bin/ls","ls",NULL);

printf("LINE J");

}

else { /\* parent process \*/

/\* parent will wait for the child to complete \*/

wait(NULL);

printf("Child Complete");

}

return 0;

}

**Solution:**

The fork system call creates a new child process and runs parallel with parent process. The process waits for the child to completely finish its processes. The child process, pid = 0 executes the statement,

*execlp("/bin/ls","ls",NULL);*

When this statement is called, the child processes are completely replaced with a new program that displays the directory contents. So, new statements in the child process are not executed. Thus, line J will not be reached.

1. **What are the pid values?**

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

int main()

{

pid\_t pid, pid1;

/\* fork a child process \*/

pid = fork();

if (pid < 0) { /\* error occurred \*/

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0) { /\* child process \*/

pid1 = getpid();

printf("child: pid = %d",pid); /\* A \*/

printf("child: pid1 = %d",pid1); /\* B \*/

}

else { /\* parent process \*/

pid1 = getpid();

printf("parent: pid = %d",pid); /\* C \*/

printf("parent: pid1 = %d",pid1); /\* D \*/

wait(NULL);

}

return 0;

}

**Solution:**

The pid values for child processes are: pid = 0 and pid1 = 13062.

The pid values for parent processes are: pid = 13062 and pid1 = 13061

1. parent :pid = 13062

It is returned by the fork system call to recognize it as a parent process with a positive value which is equal to the process id of child process.

1. parent :pid = 13061

It is less than the process id of the child process.

1. child : pid = 0

It is returned by the fork system call to represent it as the child process.

1. child: pid = 13062

It is returned by the fork system call to represent it as the child process.

1. **What will be at Line X and Line Y?**

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

#define SIZE 5

int nums[SIZE] = {0,1,2,3,4};

int main()

{

int i;

pid t pid;

pid = fork();

if (pid == 0) {

for (i = 0; i < SIZE; i++) {

nums[i] \*= -i;

printf("CHILD: %d ",nums[i]); /\* LINE X \*/

}

}

else if (pid > 0) {

wait(NULL);

for (i = 0; i < SIZE; i++)

printf("PARENT: %d ",nums[i]); /\* LINE Y \*/

}

return 0;

}

**Solution:**

At line X,CHILD: 0

CHILD: -1

CHILD: -4

CHILD: -9

CHILD: -16

At line Y,PARENT: 0

PARENT: 1

PARENT: 2

PARENT: 3

PARENT: 4

At first, line X was executed where fork executed child processes; therein ran print statement 5 times in a loop. In the loop, ith index of nums array was updated to a new value in each iteration of the loop. Then, the line X was displayed on the screen.

Similarly, parent processes were executed by the fork and obsequently, output is displayed.