BGL312E - Assignment 1

Ömer Malik Kalembası - 150180112

02/04/2023

1 Introduction

In this assignment, you are tasked to use the "fork()" system call to create a specific process hierarchy given in the figure below. Getting better at handling fork system calls provides a better understanding of process management in operating systems which is quite useful for parallel programming and resource management.

You can run the program using "make all" command on the current directory.

2 Question 1

Program takes input N from user and creates N depth process tree which descripted in assginment using process management function fork().

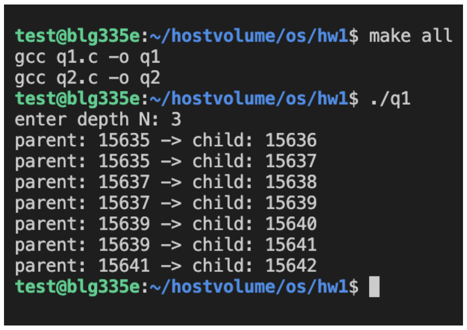
The processTree() function takes an integer parameter called depth, which represents the depth of the process tree. If depth equals 0, the function immediately exits, stopping the creation of child processes. If depth equals 1, the function creates a left child process using the fork() system call, and prints the parent process ID and the child process ID to the console. If depth is greater than 1, the function creates both a left and a right child process using fork(). The left child process immediately exits, while the right child process calls processTree() recursively with depth - 1 as the new parameter. Finally, the parent process prints its process ID, along with the process IDs of its left and right child processes.

Number of fork() processes can be consiedered as number of parent processes. In order to this view:

Each parent creates two child except last parent. Last parent creates just left child and call 1 fork() function. Number of parents can expressed as N+1,

So number of parent processes should be 2*(N+1) - 1.

Here is the output of the program 1 and visualized process tree 2.



```
test@blg335e:~/hostvolume/os/hw1$ make all
gcc q1.c -o q1
gcc q2.c -o q2
test@blg335e:~/hostvolume/os/hw1$ ./q1
enter depth N: 3
parent: 15635 -> child: 15636
parent: 15635 -> child: 15637
parent: 15637 -> child: 15638
parent: 15637 -> child: 15639
parent: 15639 -> child: 15640
parent: 15639 -> child: 15641
parent: 15641 -> child: 15642
test@blg335e:~/hostvolume/os/hw1$ 
```

Figure 1: q1 code

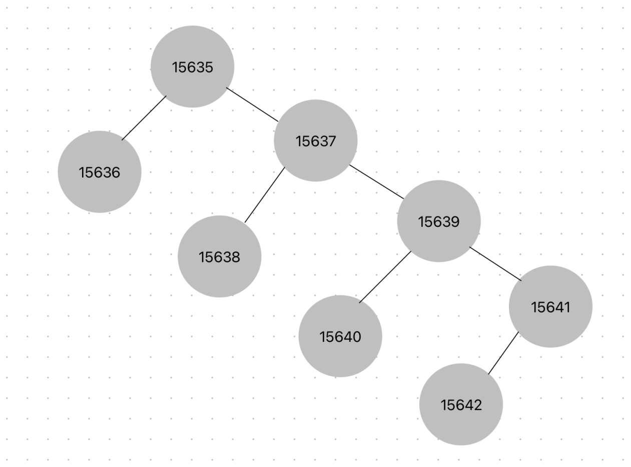


Figure 2: q1 process tree

3 Question 2

Program takes two inputs N and M from user and creates N depth - M subtree depth process tree which descripted in assginment using process management function fork().

The code begins by defining two functions: leftTree() and processTree(). The leftTree() function creates the left subtree of the process tree recursively. The processTree() function creates the right subtree of the process tree, and for each node in the right subtree, it also creates the left subtree using the leftTree() function.

The leftTree() function takes an integer depth as input, which represents the depth of the left subtree. If the depth is 1, then the function terminates the process by calling exit(0). Otherwise, the function creates a child process using fork(), which becomes the left child of the current node. The child process then recursively calls leftTree() with the depth decreased by 1. The parent process waits for the child process to terminate before it itself exits.

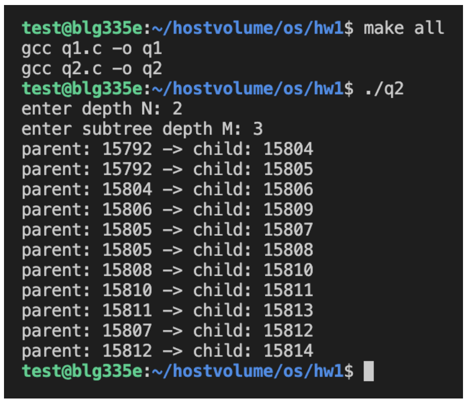
The processTree() function takes two integers depth and M as input, which represent the depth of the subtree and the depth of the left subtrees, respectively. If the depth is 0, then the function terminates the process by calling exit(0). Otherwise, the function creates a child process using fork(), which becomes the left child of the current node. The child process then calls the leftTree() function with the depth M to create the left subtree. The parent process then waits for the left child process to terminate before it continues to create the right child process.

Number of fork() processes can be consiedered as number of parent processes. In order to this view:

Each main tree parent creates 1 right child except last parent. Last parent creates just left subtree. Each subtree call fork() function M times. Number of subtrees can be evaluate as number of parents which is N+1. In a subtree, there will be M times fork() function calls.

So number of parent processes should be (N+1)*M + N.

Here is the output of the program 3 and visualized process tree 4.



```
test@blg335e:~/hostvolume/os/hw1$ make all
gcc q1.c -o q1
gcc q2.c -o q2
test@blg335e:~/hostvolume/os/hw1$ ./q2
enter depth N: 2
enter subtree depth M: 3
parent: 15792 -> child: 15804
parent: 15792 -> child: 15805
parent: 15804 -> child: 15806
parent: 15806 -> child: 15809
parent: 15805 -> child: 15807
parent: 15805 -> child: 15808
parent: 15808 -> child: 15810
parent: 15810 -> child: 15811
parent: 15811 -> child: 15813
parent: 15807 -> child: 15812
parent: 15812 -> child: 15814
test@blg335e:~/hostvolume/os/hw1$
```

Figure 3: q2 code

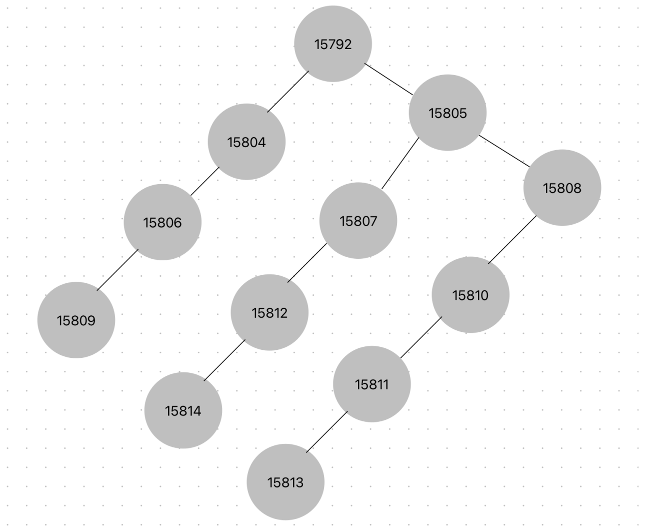


Figure 4: q2 process tree