2023-2024 Spring Term

CS 307 Operating Systems Course

-

Programming Assignment 1 Report

Berke Ayyıldızlı

31018

Program Outline

In this assignment, we are trying to implement a tree structure using the fork() function in unix. Since a binary search tree structure also has pointers that help the algorithm to move between nodes, here, we have the properties of the fork() function. Starting from the root, when the fork() function is called on the parent nodes, the current operation is split into 2, a new child operation and the copy of the parent (at the start the root). After this, we fork again in this newly created parent to create another child, namely the right child. This whole forking operation creates the tree in a recursive way, by making the necessary checks such as looking to pids and calling the execvp() with the needed arguments. Here, because of the aim of the homework, a number called num1 is requested from the user in the root node, and then this num1 is being carried to the bottom-left leaf, where the num2 value is taken from the leaves as 1. After this part, the program, using the lr parameter, knows if they are moving towards the left children or right children, and executes the correct operation function called “left” or “right” using execvp(). If the numbers moving from a left child to the parent, an addition operation is done with the numbers num1 and num2, and similarly, if the numbers are moving from a right child to the parent, a multiplication is done. This whole moving the numbers parts are being done with the usage of pipe(). 2 pipes are implemented in the code to first, move the numbers from root to the leaves, and second, move the numbers from leaves to the root. When the numbers traverse the whole tree and do the necessary operations, the program ends with printing the result at the root node.

The Program Hierarchy

I first started code by writing the necessary imports such as “stdio.h”, “sys/wait.h” and “string.h”, for the usage of pipes and execvps. Then, i wrote an if statement to correctly show the wanted parameters of the argc. This is for the first test case. Argv parameter inputs are correctly converted using atoi here. Before continuing with the main part, I initialized the numbers here and also wrote the important fprintf statements here before the forking process. These statements include how many “---” characters and the prompt for writing the num1 to the terminal. After

successfully taking num1 from the user, the program continues with the main part. Here I check if the current depth curDepth is less than maximum depth maxDepth, if so, the program continues recursively. For the execvp() function to be called correctly here with the parameters, I opened character arrays here and converted them using sprintf(). After this assignment, I created 2 pypes, named firstPipe, for the movement from root to the leaves and secondPipe, for the movement form leaves to the root. I also checked if they returned 0. After that I used my first fork() statement, and checked with an if statement that we are on the child, and namely left child here.

On the left child, first, I closed the write end of the firstPipe, because we are going to do reading operations with it here, then using dup2() function, I wired the read end of the firstPipe to the STDIN. For the vice versa, I also closed the read end of the secondPipe and used dup2 to wire secondPipe's write end to the STDOUT. After this operation, I called treePipe again with the arguments using execvp.

On the parent, I closed the read end of the firstPipe and used dup2 to wire write end with the STDOUT, then closed the write end. After this, to get the num1, I used printf.fflush() is also used here to ensure a clean data stream. After waiting the parent to finish, I printed necessary information such as curDepth and num1, with a result statement.

After the parent, using again fork() function, we create the right child. On this child, also the treePipe function is called recursively using execvp. In its parent, after waiting it to finish its processes, I check if the current depth is equal to 0. This ensures that the "final result is" part is written at only when we came back to the root.

Also on the end of the recursion (the last else), I print the result part, this means after the movement from one node to another.

This concludes my code and thus my homework, as I, using the requested commands, execvp(),fork(),pipe(),open() and close(), tried to create a recursive tree-like structure, which I tried to traverse in an inorder fashion.