K. N. Toosi University of Technology

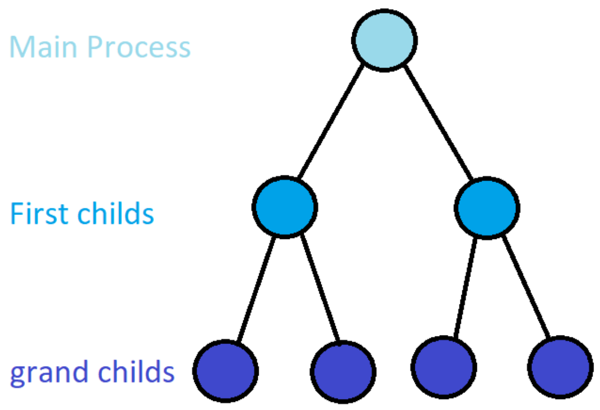
Operating Systems Course
Instructor : Dr.Shafiei
Written by: Amirreza Sokhankhosh

Processes

October 27, 2021

Overview

In chapter 3, you were introduced to processes. Your task in this project is to practice creating processes and practically handling them. You are expected to get 2 numbers as inputs, and then create 2 generations of processes with respect to those numbers. The first number is the number of child processes that their parent is the process which is running your program. The other number is given for the children of those child processes, or you may say “grandchildren” of the main process. Here is a graph for illustration:



Goals

1. **Create these 2 generations** with the instruction you’ve learned from chapter 3. For simplicity, use the **fork()** function in creation. Remember that you should create them with respect to the numbers you have in your input. In the figure shown above, input was ***2, 3.*** **Remember, each process should wait for its children to finish if it wants to end.**

2. **Create a pipe that can connect a parent to each of its children in each generation.** Transmit a simple number (really doesn't matter what) through your pipe from the parent to the children and reverse. As you know, you should use the **pipe()** function. Remember, you have 2 parent and child relationships, one between the main process and its children, and the other between the children and the grandchildren of the main process. So you should have 2 **types (not number)** of pipes. **(Edges of the figure shown above are the relationships).**

3. **What's the efficient way of creating (a * b) grandchildren processes?** After using the pipe of each grandchild, add a sleep time (amount is your choice), so you could get a better sense of processing time, according to your input. Hopefully, your program will take a little time to finish, but what would that time be if you changed the number of processes in each of these 2 generations?

   For better understanding, consider the example above with ***2, 3*** as inputs. We have 6 grandchildren in total, but let's make these 6 processes with another input like (***1, 6),*** or (***6, 1)***, and so on. How much time will it take to end your program in these settings? What's the best way (**lesser time**) to arrange these inputs? To understand that, compute the processing time of each of these possible settings and plot a figure if you can (or at least show it in your terminal).

Specifications

1. Use **Linux** operating system and **C** to complete this project. Other programming languages will not be accepted at all.
2. Write a PDF, explaining your code, its output, along with your explanation of the output.
3. Explain your code part by part; put the part in the PDF, along with its output.
4. Remember to write your document pretty convincing, because that's all we look at to give you your score. We also run the program to check that it's runnable.
5. All of the projects will be tested for similarities by a coded script, So if we find an obvious similarity between 2 or more projects all of them will get **0** points.