Burak Bozdağ
150170110

BLG312E – Computer Operating Systems

2019-2020 Spring Term

Assignment 3 Report

Introduction

I coded a multi-processing program that simulates a moneybox. There are two types of processes. First type of process increases the amount of money in the moneybox, whereas second type of process decreases this amount. Initially, only first type of processes work until there is a specific amount of money in the box. After the money in the box becomes equal to or greater that this specific number, second type of processes start to work also. These two process groups work in turns. Eventually, decreaser processes work faster and when the money in the box becomes zero, the program terminates.

Program Input

My program accepts five command line arguments as input:

- N: The amount of money that needs to be stored in the moneybox before decreaser processes start working.
- n_i: Number of increaser processes.
- n_d: Number of decreaser processes.
- t_i: Amount of turns the increaser processes will run before decreasers start working.
- t_d: Amount of turns the decreaser processes will run before increasers start working.

Processes

- **Master Process**: Reads the command line arguments, creates increaser and decreaser (even though these will not work at the start, they will exist in the memory waiting their turn) processor at the start of the program, and keeps the program working until all children (increaser and decreaser) processes finish their work. This process is also responsible for initializing a shared memory section for amount of money in the box. This process does not perform any operations between initialization stage and termination stage.
- **Increaser Process**: These processes will increase the amount of money in the box. Half of these processes increase the money by 10, and the other half increase the money by 15. Number of increaser processes are always even.
- **Decreaser Process**: These processes will decrease the amount of money in the box. Each decreaser process works with its own fibonacci numbers independent of other decreasers. First decrement starts with the first fibonacci number (starts from 1, not 0), and the next decrement will be performed with next number, and so on (an example will be given in the next section below). Half of these processes will only work if the current amount of money in the box is an even number, and other half only works when the amount of money is an odd number. Again, number of decreaser processes are always even. If the amount of money being decreased is greater than or equal to the amount of money in the box, decreaser process will signal master process to stop and kill all children.

Working Order of the Processes

- Initially, only increaser processes will change amount of money in the moneybox.
- After enough money is stored in the box, increaser and decreaser processes will work in turns. Increaser process will work for t_i turns consecutively, then decreaser processes will work for t_d turns consecutively, and they will continue working this way until program terminates.
- During each of the turns, each process in that group can only work at most once. If the current amount of money is an even number and there is not an even decreaser process left, that decreaser turn will be finished, and vice versa.

Compilation and Running of the Program

To compile: *gcc “Source Code.c” -o moneybox -lpthread -lm*

To run: *./moneybox 150 4 2 2 4*