BLG312E – Computer Operating Systems
2019-2020 Spring Term

Assignment 3

Introduction

You will code a multi-processing program that simulates a moneybox. There will be two types of processes. First type of process will increase the amount of money in the moneybox, whereas second type of process will decrease this amount. Initially, only first type of processes will work until there is a specific amount of money in the box. After the money in the box becomes equal to or greater than this specific number, second type of processes will start to work also. These two process groups will work in turns. Eventually, decreaser processes will work faster and when the money in the box becomes zero, the program will terminate. Details will be explained below.

Program Input

Your program will accept 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.
- ti: Amount of turns the increaser processes will run before decreasers start working.
- td: Amount of turns the decreaser processes will run before increasers start working.

Processes

Your program should have the following functionality for 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 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 ti turns consecutively, then decreaser processes will work for td 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.

Example:
./yourprogram 150 4 2 2 4
Master Process: Current money is 0
Increaser Process 0: Current money is 10
Increaser Process 1: Current money is 20
Increaser Process 3: Current money is 35
Increaser Process 2: Current money is 50, increaser processes finished their turn 1
Increaser Process 1: Current money is 60
Increaser Process 3: Current money is 75
Increaser Process 0: Current money is 85
Increaser Process 2: Current money is 100, increaser processes finished their turn 2

// increaser processes worked for ti turns, but still not enough money accumulated in the box.

Increaser Process 1: Current money is 110
Increaser Process 0: Current money is 120
Increaser Process 2: Current money is 135
Increaser Process 3: Current money is 150, increaser processes finished their turn 3

// even though there is enough money in the box, increaser processes must still run 1 more turn to complete ti turns.

Increaser Process 2: Current money is 165
Increaser Process 1: Current money is 175
Increaser Process 0: Current money is 185
Increaser Process 3: Current money is 200, increaser processes finished their turn 4

// increaser processes finished ti turns, and conditions for decreaser processes to start working are met.

Decreaser Process 0: Current money is 199 (1st fibonacci number for decreaser 0)
Decreaser Process 1: Current money is 198 (1st fibonacci number for decreaser 1), decreaser processes finished their turn 1
Decreaser Process 0: Current money is 197 (2nd fibonacci number for decreaser 0),
Decreaser Process 1: Current money is 196 (2nd fibonacci number for decreaser 1), decreaser processes finished their turn 2
Decreaser Process 0: Current money is 194 (3rd fibonacci number for decreaser 1), decreaser processes finished their turn 3
// turn is finished because current money is even, and there is not an even decreaser process left

Decreaser Process 0: Current money is 191 (4th fibonacci number for decreaser 0),
Decreaser Process 1: Current money is 189 (3rd fibonacci number for decreaser 0), decreaser processes finished their turn 4

// decreaser processes worked for td turns, increaser processes will start now

Increaser Process 3: Current money is 204
Increaser Process 1: Current money is 214
Increaser Process 0: Current money is 224
Increaser Process 2: Current money is 239, increaser processes finished their turn 5
…

...
Decreaser Process 0: Current money is less than 377, signaling master to finish (13th fibonacci number for decreaser 0)
Master Process: Killing all children and terminating the program

Note: When you run the program with same parameters, you may get different results!

Submission Deadline: 03.06.2020

Important notes:

- You are expected to work individually on this homework. All forms of collaboration are discouraged and will be treated as plagiarism. This includes actions such as, but not limited to, submitting the work of others as one's own (even if in part and even with modifications) and copy/pasting from other resources (including Internet resources) even with proper reference. Such offenses are reported to the administration for disciplinary measures. All parties involved in the act will be treated equally.

- You should submit your homework through Ninova system. Late submissions are not accepted.

- Your code must be written in C, and should be compiled and run on ITU's Linux Server (you can access it through SSH) using gcc. Your code must compile without any errors; otherwise, you may get a grade of zero on the assignment.

Submission Details:

- You should submit two files through Ninova: (i) C source file of your implementation, (ii) report (pdf file) containing **pseudo code of the increaser and decreaser processes** and **necessary instructions** for compiling and running your program.

- In your implementation, you should avoid solutions that include busy-waiting or polling(e.g. checking if conditions are met with regular intervals). Solutions that involve these will not be accepted as correct.

- In addition to above, required synchronizations (turns, oddness/evenness, waiting for initial N money...) must only be done via semaphore usage in icreaser/decreaser processes. Do not use signal mechanisms for synchronization. Master process may create semaphores and assign their initial value, but after that it will not play any role in the program other than waiting for termination signal.

- If you have further questions about the assignment, you may contact the course assistant (unlut@itu.edu.tr)