New York University
Tandon School of Engineering
Department of Electrical & Computer Engineering

Introduction to Operating Systems (CS-GY6233)
Spring 2021

Assignment 4
(15 points)

1. (7 points)
   a. Write a program that accepts a single parameter (`n`) that is passed to your program from the command line/shell. Your program shall create a shared memory for an integer (you may name it `counter`) and then spawn a child process using fork(), i.e. both the parent and child processes can access the shared memory containing that integer. The parent process shall have a loop that increments the `counter` `n` times, whereas the child process shall implement a loop that decrements the `counter` `n` times and then exits.
   After the parent process has incremented the `counter` `n` times, it shall wait for the child process to exit, and then print the value of `counter` to the screen.

   You shall run your program multiple times (let's say 10) for different value of `n` (which you pass from the command-line), and `n` should perhaps change from 10 to 100,000,000 in a logarithmic manner (i.e. multiply by 10 after each experiment).

   b. Tabulate your results by recording the values of `n` and `counter` in each experiment, and then answer the following questions:
   c. Does the final printed value of the counter change from one experiment to the other, or is it constant? If it is constant, what is the value? Explain why if not constant.
   d. Is there a range of values for `n` where the behavior is different from the behavior in other ranges? Explain why if any.

   Hint: You may want to search for the term "atomic instructions".

2. (8 points)
   a. Modify the Fibonacci function you created in your previous assignment such that it takes NO parameters and returns the value of the next number in the sequence, i.e. when you call it the first time it returns 1, when you call it the second time it also returns 1 (second element), you call it again and it returns 2 (third element), then 3, then 5. Then 8 and so on.

   Create the main routine of your program, in which a length parameter `n` is passed to your program as an argument from the command line/shell. The program shall then create a shared memory of size `5*sizeof(int)` (i.e. 5 elements) and then spawn a child process using fork().

   The child process shall then wait for a random time (between 0 and 1 seconds) and then call the Fibonacci function to obtain the next number in the sequence. It needs to pass that number to the parent process via the shared memory (created earlier by the parent). The child process repeats this `n` times and then exits (`n` obtained from the user as previously stated).

After spawning the child process, the parent waits till an entry (i.e. a number from the sequence) is produced in the shared memory, and prints that entry to the screen **as soon as it becomes available, not after the child exits**. When `n` entries are obtained and printed, it then waits for the child process to exit, destroys the shared buffer and then exits.

Note that the shared buffer (or memory) shall be administered without using any OS supported synchronization primitives (i.e. mutexes, semaphores, monitors, etc.). This can be done as explained in the lecture 5, slides 5-8.

b. Using you implementation (as described in slides 5-8), what's the maximum number of elements the shared buffer can actually hold? Why?

Submission file structure:

Please submit a **single .zip file** named **[Your Netid]_lab#.zip**. It shall have the following structure (replace # with the actual assignment number):

```
└── [Your Netid] hw# (Single folder includes all your submissions)
    ├── lab#_1.c (Source code for problem 1)
    ├── lab#_2a.c (Source code for problem 2a, and so on)
    ├── lab#_1.h (Source code header file, if any)
    ├── Makefile (makefile used to build your program, if any)
    ├── lab#.pdf (images + Report/answers to short-answer questions)
```

What to hand in (using NYU Classes):

- Source file(s) named as described above.
- A .pdf file named **"lab3.pdf"**, containing:
  - Screen shot(**s**) of your terminal window showing the current directory, the command used to compile your program, the command used to run your program and the output of your program.
  - Answers to H/W questions

RULES:

- You shall **use kernel version 4.x.x or above**. You shall not use kernel version 3.x.x.
- You may consult with other students about GENERAL concepts or methods but copying code (or code fragments) or algorithms is NOT ALLOWED and is considered cheating (whether copied form other students, the internet or any other source).
- If you are having trouble, please ask your teaching assistant for help.
- You must submit your assignment prior to the deadline.