CSC 4320: Operating Systems — 50 Points

Homework 4

© Instructor: Dr. Md. Mahfuzur Rahman **Spring 2025**

Work on the following problems and submit your own answers. You are allowed to discuss with other students. However, do not copy the solutions from peers or other sources. If the assignment has programming component, your program(s) must compile with **gcc** and execute on **snowball.cs.gsu.edu**! Please see https://cscit.cs.gsu.edu/sp/guide/snowball for more details.

Instructions:

- Upload an electronic copy (MS word or pdf) of your answer sheet to the folder named "HW4" in iCollege.
- Please add the course number, homework number, and your name at the top of your answer sheet.
- Please write down your answers with the question number only in the answer sheet.
- Also submit your .c file (c program), if you are asked to write a program.
- Name your file in the format of CSC4320_HW4_FisrtnameLastname (.docx/.pdf)
- Deadline: Submit by March 22, 2025, 11:59 pm

1. (25 points) Create five child processes (use `fork`) which will read the last number from a file. Then it will increase that value by 1 and write/append to the same file. Assume that you have a text file `f.txt` that contains 0 initially (only one integer). The file holds all the integers written/appended so far, including the initial 0. Given a file F, containing a sequence of integers (initially only 0), each process must perform the following steps:

```
1.  Open F
2.  Read the last integer N from the file
3.  Close F
4.  Output N and the process's PID (either on screen or test file)
5.  Increment N by 1
6.  Open F in append mode
7.  Append N to F (no overwriting, add to the next line)
8.  Close F
```

   Use C language to solve this problem. To ensure **mutual exclusion** among the competing child processes while accessing the shared file (critical section in this case), use/create a named `semaphore` (include `semaphore.h` among other required libraries).

   Compilation: `gcc sema_solution.c -o sema_solution` (on `snowball.cs.gsu.edu` server)

2. (25 points) Write a multi-threaded (parent thread and a child thread) C program to generate first 20 numbers of the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, ... Recall that the first two Fibonacci numbers are 0 and 1. Then the next number in the sequence is always the sum of the preceding two Fibonacci numbers. The child thread will generate the Fibonacci sequence, while the parent thread will display those numbers. The parent thread will not wait for the child thread to finish its execution before printing out the computed values. Instead, we let the parent thread access the Fibonacci numbers as soon as they were computed by the child thread. Maintain a global variable `computed` to check on the required condition. You will need to implement the POSIX condition variable for active communication between the parent thread and the child thread. To help you get started, I am giving you a basic structure below:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define MAX 20

int fib[MAX];
int computed = 0;
pthread_mutex_t mutex;
pthread_cond_t cond;

void *generate_fibonacci(void *param) {

    /* write your code */

    pthread_exit(0);
}

int main() {

    pthread_mutex_init(&mutex, NULL);
    pthread_cond_init(&cond, NULL);

    /* write your code */

    return 0;
}
```

Now, complete your program, compile and run it on Snowball server. Submit .c file and attach screenshots of your output.

Question:	1	2	Total
Points:	25	25	50
Bonus Points:	0	0	0
Score:			