CSC 4320: Operating Systems

60 Points

Homework 2

© 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 "HW2" 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_HW2_FisrtnameLastname (.docx/.pdf)
- Deadline: Submit by February 15, 2025, 11:59 pm

1. (10 points) Using the program shown below, explain what the output will be at LINE A.

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main() {
    pid t pid;

    pid = fork();
    if (pid == 0) { /* child process */
       value += 15;
       return 0;
    }
    else if (pid > 0) { /* parent process */
       wait(NULL);
       printf("PARENT: value = %d",value); /* LINE A */
       return 0;
    }
}
```

2. (10 points) Including the initial parent process, how many processes are created by the program shown below. How many processes will be there which have at least one children? Explain your answer.

```c
#include <stdio.h>
#include <unistd.h>

int main()
{
    /* fork a child process */
    fork();

    /* fork another child process */
    fork();

    /* and fork another */
    fork();
    return 0;
}
```

3. (10 points) Consider that we are using the shared memory model. The shared buffer is implemented as a circular array of size `N` with two logical pointers: `in` and `out`. The variable `in` points to the next free position in the buffer; `out` points to the first full position in the buffer. The buffer is empty when `in == out` and full when `(in + 1)% buffer_size == out`. Explain why there will be no race condition in this solution (where at most N – 1 buffer locations can be filled).

4. (10 points) Using the program below, identify the values of pid at lines A, B, C,and D. Provide explanation of your answer. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main() {
    pid t pid, pid1;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }

    else if (pid == 0) { /* child process */
        pid1 = getpid();
        printf("child: pid = %d",pid); /* A */
        printf("child: pid1 = %d",pid1); /* B */
    }

    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d",pid); /* C */
        printf("parent: pid1 = %d",pid1); /* D */
        wait(NULL);
    }

    return 0;
```

```
}
```

5. (10 points) Implement a system of three processes which read and write numbers to a file. Assume that you have a text file `f.txt` that contains 0 initially. Each of the three processes P1, P2, and P3 must obtain 200 times an integer from the file. The file only holds one integer at any given time. Given a file F, containing a single integer N, each process must perform the following steps:

```
1.  Open F
2.  Read the integer N from the file
3.  Close F
4.  Output N and the process' PID (either on screen or test file)
5.  Increment N by 1
6.  Open F
7.  Write N to F (overwriting the current value in F)
8.  Close F
```

   Use the programming language `C` for this.

6. (10 points) Using either a UNIX or a Linux system, write a C program `zombie_creator.c` that forks a child process that ultimately becomes a zombie process. This zombie process must remain in the system for at least 10 seconds. Run your program with &, the shell starts the process in the background. It will immediately show you the process ID (PID) of the background process. Process states can be obtained from the command `ps -l`

   Use the `ps -l` command to determine whether the child is a zombie process. Terminate the parent process using the `kill` command. For example, if the `pid` of the parent is 4884, you would enter

   ```
   kill 4884
   ```

   Again issue `ps -l` command to make sure your zombie process no longer exists in the system.

   Attach your `zombie_creator.c` program and screenshots of your terminal.

Question:	1	2	3	4	5	6	Total
Points:	10	10	10	10	10	10	60
Bonus Points:	0	0	0	0	0	0	0
Score:							