Assignment-4

CS342: Operating System Lab

General Instruction

• Assignments should be submitted through shared link.
• Assignments will not be accepted after the due time.
• Markings will be based on the correctness and soundness of the outputs. In case of plagiarism zero marks will be awarded.
• The assignments must be written in C language. Proper indentation & appropriate comments (if necessary) are mandatory in the code.

In today's lab session, the goal is to demonstrate the creation of orphan and zombie process. Further, you will learn about multithreaded programming.

You may refer to given link for multithreading assignment:

1. https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html

----

Problem1: Write a program **orphan.c** to demonstrate the state of process as an orphan. The program forks a process. Further,
The parent process prints the following messages.
Parent: I am the parent process and my ID is <the id of the parent>
Parent: I have created a child process whose id is <the id of the child>
The child process prints message
Child: I am a child process and my id is <the id of the child>
Child: I have been created by my parent process whose id is <the id of the parent>
Post printing this message the child sleeps for 30sec meanwhile the parent process should have exited. Finally, the child process prints the following message
Child: I am a child process and my id is <the id of the child>
Child: My current parent id is <the id of the parent>

Problem2: Write a program **zombie.c** to demonstrate the presence of zombie processes. The programs forks a process, and the parent process prints the message
Parent : My process ID is: <the id of the parent>
Parent : The child process ID is: <the id of the child>
and the child process prints the message
Child : My process ID is: <the id of the child>
Child : The parent process ID is: <the id of the parent>
After printing the messages, the parent process sleeps for 1 minute, and then waits for the child process to exit. The child process waits for some keyboard input from user after displaying the messages, and then exits. Display the process state of child process while it was waiting for input and after the input using given ps command.
ps -o pid,stat --pid <child's PID>.

Problem3: Write a C program **mthread.c** to compute the sum for the first N natural numbers. The user provides the N as a command line argument. In the program, create two threads, namely t1 and t2. Both the threads should execute the same function add(). The t1 thread should compute the sum of the first half of natural numbers and update the global variable **result**; similarly, t2 should compute the sum of the second half of natural numbers and update the global variable **result.** The parent process should wait for the completion of both the threads and print the final result as output**. Note that there may be overwriting issue among the threads. It would be best to use MUTEX (see the shared link)** to handle this.