COSC 3360 - Fundamentals of Operating Systems

Programming Question 1

Available from: Thursday, 31 March 2022, 4:00 PM
Due date: Thursday, 31 March 2022, 6:40 PM
Requested files: main.cpp (Download)
Type of work: Individual work

This question closes at 5:20 PM (unless you have testing accommodations approved by the University). You must save your work before this time, as Moodle will not save the work for you.

Write a C++ program that creates nchildthreads (number of child threads from the input). Each child thread must print a message into STDOUT based on the following rules:

1. You will assign a thread number to each child thread based on the order they are created. The value for the thread number must start from zero and be increased by one before assigning it to the next child thread.
2. Each child thread will print a predefined message (available in the template file). The print statement executed by the child thread cannot be inside of a critical section.
3. Your program must guarantee that messages are printed in increasing order, starting with the child threads with even thread numbers and ending with the child threads with odd thread numbers.

For nchildthreads = 5, the expected output is:

```
I am Thread 0
I am Thread 2
I am Thread 4
I am Thread 1
I am Thread 3
```

NOTES

1. **You must create n child threads (where n is the value that your program receives from STDIN).**
2. **You can only use POSIX semaphores, pthreads mutex semaphore, or pthreads condition variables to achieve synchronization (A penalty of 100% will be applied to solutions using mechanisms other than the synchronization mechanisms listed before).**
3. **To simplify your solution, the template file defines several global variables that you are allowed to use in your program for this programming question. However, remember that you should avoid using global variables when writing programs.**

Requested files

main.cpp

```
#include <pthread.h>
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>

static pthread_mutex_t bsem;
static pthread_cond_t waitTurn = PTHREAD_COND_INITIALIZER;
static int turn;
static int nthreads;

void *print_in_increasing_order_even_odd(void *void_ptr_argv)
{
    // std::cout << "I am Thread " << /*variable identifier*/ << std::endl;
    return NULL;
}

int main()
{
    std::cin >> nthreads;
    pthread_mutex_init(&bsem, NULL); // Initialize access to 1
    pthread_t *tid= new pthread_t[nthreads];
    int *threadNumber=new int[nthreads];

    for(int i=0;i<nthreads;i++)
    {
        // Call pthread_create
    }
    // Call pthread_join

    delete [] threadNumber;
    delete [] tid;
    return 0;
}
```

VPL

◄ Banker's Algorithm Question

Jump to...

Theory Part - Exam 3 (30 points / 1 attempt / 45 minutes) ►