

COSC 3360 (16770) - COSC 6310 (11209) - Operating Systems


[Dashboard](#) / [My courses](#) / [COSC3360SP202502](#) / [PROGRAMMING ASSIGNMENTS](#) / [Programming Assignment 3](#)

 Description



 [Submission](#)

 [Edit](#)

 [Submission view](#)

 **Available from:** Tuesday, 1 April 2025, 12:00 AM

 **Due date:** Sunday, 27 April 2025, 11:59 PM

 **Requested files:** main.cpp ( [Download](#))

Type of work:  Individual work

Similarity Threshold: 90%

Objective:

This assignment will introduce you to interprocess synchronization mechanisms in UNIX using named POSIX semaphores, pthread mutex semaphores, and pthread condition variables.

Problem:

For this assignment, you will modify your solution for [programming assignment 1](#) to comply with the restrictions explained below.

Your multithreaded program using the Encoding Method for Sparse Binary Data presented by Weiwei Duan et al. (2017) must execute the following steps:

- Read the input from STDIN (the Moodle server will implement input redirection to send the information from a file to STDIN).
- Create n POSIX threads (where n is the number of rows in the image). Each child thread executes the following tasks:
 - Receives the row number to print, the headPos and dataPos arrays, the information about the symbols to print from the main thread, and the memory location to store the results of the decoding process.
 - Decodes the assigned row using the Encoding Method for Sparse Binary Data.
 - Prints the assigned row.

Given the following input:

26 7

U 0 10,H 15 25

0 4 8 12 25 29 33

0 10 15 25 0 10 15 25 0 10 15 25 0 10 15 16 17 18 19 20 21 22 23 24 25 0 10 15 25 1 9 15 25 2 3 4 5 6 7 8 15 25

The expected output is:

U	U	H	H
U	U	H	H
U	U	H	H
U	U	HHHHHHHHHH	
U	U	H	H
U	U	H	H
UUUUUUU		H	H

NOTES:

- You can safely assume that the input files will always be in the proper format.
- You cannot use global variables. A 100% penalty will be applied to submissions using global variables.
- You must define the critical sections following the guidelines discussed in class.
- You must use POSIX threads. A penalty of 100% will be applied to submissions using a thread library other than the pthread library.
- To achieve synchronization, you can only use named POSIX semaphores, pthreads mutex semaphores, or pthreads condition variables. You are not allowed to use pthread_join or sleep to synchronize your threads (you must use pthread_join to guarantee that the parent thread waits for all its child threads to end before ending its execution). Submissions using the previous system calls to synchronize the child threads will be penalized 100%.
- You cannot use different memory addresses to pass the information from the parent thread to the child threads.
- You must use the output statement format based on the example above. The child threads must print the encoded messages in the order in which they were created.

ASSIGNMENT RUBRIC:

- **15 points for each test correct output (max 30 points).**
- **25 points for using a single memory location for thread argument passing.**
- **5 points for code readability (modularity + comments).**
- **10 points for taking full advantage of multithreading.**
- **30 points for solving the two synchronization problems correctly (correct usage of critical sections, semaphores, pthreads mutex semaphores, pthreads condition variables, pthread_join).**

Penalties:

1. **100% for using global variables.**
2. **100% for not using pthread library.**
3. **100% for presenting a solution that does not compile.**

Requested files

main.cpp

```
1 // Write your program here
```