COSC 3360 - 15772 - Fundamentals of Operating Systems

Programming Assignment 3

Available from: Saturday, 29 October 2022, 12:00 AM
Due date: Saturday, 3 December 2022, 12:00 AM
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.

Given a message (string or char array) as input, you need to implement a multithreaded version of the Shannon-Fano-Elias code generator based on the following steps:

- Read the input from STDIN (the Moodle server will implement input redirection to send the information from a file to STDIN). The input has a single line with the message (string or char array).

Example Input File: AAAAAABBBBBCCCCDDDEE

- Create n POSIX threads (where n is the number of symbols to encode). Each child thread executes the following tasks:
  - Receives the probabilities of the symbols from the main thread.
  - Implements the Shannon-Fano-Elias encoding algorithm to determine the code for the assigned symbol.
  - Print the Shannon-Fano-Elias code for the assigned symbol.

Given the previous input, the expected output is:

```
SHANNON-FANO-ELIAS Codes:

Symbol A, Code: 001
Symbol B, Code: 011
Symbol C, Code: 1010
Symbol D, Code: 1101
Symbol E, Code: 11110
```

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 that we 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.**
- **You can only use named POSIX semaphores, pthreads mutex semaphores, or pthreads condition variables to achieve synchronization. Using pthread_join or sleep to synchronize your threads is not allowed (you must use pthread_join to guarantee that the parent thread waits for all its child threads to end before ending its execution). A penalty of 100% will be applied to submissions using the previous system calls to synchronize the child threads.**
- **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.**

Requested files

main.cpp

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

VPL