Description

The purpose of this project will be to build understanding of threading and process synchronization.

You will build a program to implement a single producer/multiple consumer queue. The program will:

a) Take as a parameter the number of consumer tasks to run.
b) The main program will start up the number of tasks specified on the command line. Each task will be assigned a task number that is passed to the task as a parameter.
c) Read lines from `stdin`. The reason for using `stdin` is that you can either type in lines to test with or produce a text file that you redirect into the program. To grade the program, I will be using a large text file (I'm thinking the text for one of Shakespeare's plays). You will need to figure out how to detect end of file on `stdin`.
d) Each line from the file will be pushed to a queue by the main task.
e) The consumer tasks will pop lines off of the queue when available. The order of the lines popped is not important.
f) For each line a consumer task pops from the queue it will:
    a. Print the task number and the line.
    b. Count the words on the line (don't worry about punctuation, my test file will have all non-alphabetic character removed) and accumulate the sum of the number of words.
g) When all lines of the input have been read:
    a. Signal all consumer tasks to terminate
    b. Accumulate the word counts
    c. Print the total word count

There are a variety of different ways to solve this problem. I will be looking for the program to be

1. Thread-safe: I will be testing your program on a multi-core machine with a large file that will show any issues with thread safety.
2. Efficient: Obviously, you can just lock everything. This isn't efficient and we've been discussing more efficient alternatives in class.

Choice of Languages

I will accept the program being written in the following languages ONLY: C, Go, Rust. There will be up to a 20-point bonus for using one of the languages other than C. Go and Rust are both modern, up and coming languages that are being used for systems programming.

Due Date and Deliverables

The project is due by midnight, Tuesday, March 31. Late work will be accepted according to the schedule shown in the syllabus.

By Thursday, March 24, I will need to know the names of students that are working together (DM in slack preferably). The project is to be worked in groups of two students, not one, not three. If there are an odd number of students participating, it will be at the instructor's discretion to allow either one individual submission or one group of three. You are free to use the same partner as for the first project or choose a different person.

I will be expecting, at a minimum, a source file, a `Makefile`, and a `README` file describing how to build the project. My preference would be a link to a GitHub (Bitlocker, GitLab, etc.) repository that I can clone to see the code, but I will accept the aforementioned files submitted into the Oaks dropbox.

Rubric

The project will be worth 50 points with a possible 20 point bonus allocated as follows:

1. 35 Points – Operation

    - (5 points) Does the program build correctly?
    - (5 points) Does the program run and produce useful, readable, output?
    - (15 points) Is the program thread-safe?
    - (10 points) Does the program use the most efficient methods for synchronization (that we've discussed)?

2. 15 Points – Programming Style

    - (5 points) Is the code well commented and readable?
    - (5 points) Are all needed files/libraries provided?
    - (5 points) Does the coding demonstrate an understanding of the features of the language used?

3. 20 Points – Bonus for use of the Go or Rust programming languages.