CS342 Operating Systems, Spring 2015
Project 3
Synchronization: Mutex and Condition Variables

Assigned: March 23, 2015
Due date: April 6, 2015

This time you will write a multithreaded program that will do what you program in Project 2 was doing, but will use bounded-buffers instead of files to store intermediate results. What to be done is the same as in Project 2. That means you will again develop a word-count program (called this time wcountb) using multiple threads. There will be N mapper threads and R reducer threads. There will be a single merger thread. Each mapper will get the input from its respective input file and write the output to R bounded-buffers (no longer to intermediate files). At a mapper, a word read from an input file will be put into one of the R buffers using the rule in Project 2. Each reducer will read the input from N bounded-buffers and will put the output to another bounded-buffer that can be accessed by the merger thread as well. The merger thread will read input from R bounded-buffers and will write the output to the final output file. There will be N input files. The buffer-size will be a parameter of the program.

At runtime, there will be N*R buffers between mappers and reducers and R buffers between reducers and the merger thread. There will be a total of N+R+1 threads. The program name will be wcountb. and it will be invoked as follows:

wcountb <N> <R> <infile1> … <infileN> <finalfile> <bufsize>

An example invocation is:

wcountb 3 2 infile1.txt infile2.txt infile3.txt final.txt 100

The buffer size in this case is 100. That means it can hold at most 100 words. Then it is considered full. A mapper thread, for example, can not put a new word into a full buffer. Similarly, a reducer thread can not retrieve a word from an empty buffer. If it can not put anything, a mapper will sleep; if it can not retrieve anything, a reducer will sleep. When the conditions of the buffers change, a sleeping mapper or reducer may be waken up. Similarly, the merger will sleep if there is nothing to retrieve from any of the R buffers that it can read from.

You will use POSIX Pthreads mutex and condition variables to solve the concurrency (synchronization) problems that you will face (mutual exclusion, sleeping, waking up, etc.). Make sure you study the related POSIX Pthreads API very well and write some small exercise programs to make sure that you learned them. Especially, the following API functions must be learned very well: functions related to creation and initialization of mutex and condition variables, functions to lock and unlock mutex variables, functions to wait, signal and broadcast on a condition variables. Study the design patterns that use mutex/ condition variables for solving some synchronization problems. Try to use these patterns in your solution.

The minimum value of buffer size can be 10. The maximum value can be 10000. You can implement the buffer (i.e., queue) as a linked list or as an array. The maximum size of a word can be 255 characters. N can be at most 20 (minimum is 1) and R can be at most 10 (minimum is 1) in this project. A filename can be at most 255 characters long. There is no limit in the number of words that an input file can hold. The limit is the file size that the system can support.

Experiments: Do some experiments. Change for example buffersize and re-run your program. Do this for various buffer sizes. Also change the input sizes (file sizes; small or big files). Change the number of mappers and reducers, and son on. Measure the completion time of your program. Plot your results. Try to interpret them. Write your comments. Obtain your report at the end in PDF form. And upload this PDF file as your report.pdf.

Submission: Submission rules are as usual. Files to upload in a folder: README.txt, wcountb.c, Makefile, report.pdf.

Clarifications will be put in course website, besides the project spec (PDF).