CSE 344 System Programming

Homework #5

Due June 3

In this project you are expected to implement a directory coping utility “ pCp ”, that creates a new thread to copy each file and sub-directory to perform the overall task in parallel. Similar utilities may quickly exceed system resources when called under a large directory tree. Therefore, in order to regulate the number of active threads at any time, you are expected to implement a worker thread pool. In a worker thread pool implementation, a fixed number of threads are made available to handle the load. The workers block on a synchronization point (in our case, maybe an empty buffer) and one worker unblocks when a request arrives (an item is put in the buffer). The overall implementation of the utility should use a producer-consumer based synchronization with the details explained below.

Producer: Start by creating a producer thread function that takes as a parameter an array of at least 2 entries (for the pathnames of the two directories). For each file in the first directory, the producer opens the file for reading and creates a file of the same name in the second directory for writing. If a file already exists in the destination directory with the same name, the file should be opened and truncated. If an error occurs in opening either file, both files are closed, and an informative message is sent to standard output. The two open file descriptors and the names of the files are then passed into a buffer. You are also expected to manage the buffer (is it empty or full, is it okay to access the buffer or should the execution wait until it is available) properly so that the threads can be terminated gracefully. Note that this is a producer-driven bounded buffer problem. When the producer finishes filling the buffer with file names for the given directories, it should inform the rest of the program (by setting a done flag) and exit.

Consumers: Each consumer thread reads an item from the buffer, copies the file from the source file descriptor to the destination file descriptor, closes the files and writes a message to standard output giving the file name and the completion status of the operation. Note that the producers and the multiple consumers are writing the standard output, and this is a critical section that has to be protected. The consumers should terminate when they detect that a done flag has been set and no more entries remain in the buffer.

Main program: should take the buffer size, number of consumers and the source and destination directories as command-line arguments. (Your utility should have only one producer thread). The main program should start the threads and use **pthread_join** to wait for the threads to complete. Use **gettimeofday** to get the time before the first thread is created and after the last join. Display the total time to copy files in the directory. Your utility should copy both regular files and FIFOs. Should be able to recursively copy subdirectories. Keep statistics about the number and types of files copied. Keep track of the total number of bytes copied.
Experiment with different buffer sizes and different number of consumer threads. Write a report and show your results and comment on the combinations of buffer/number of consumer threads that produce the best results. What happens when you exceed the per-process limit on the

number of open file descriptors? Your utility needs to check if this produces an error or not. Check tour program for memory leaks. Signals should also be handled properly in order to obtain full credit.

Grading:

1) Compilation errors will result in a deduction of 100 points.
2) Make sure to include a makefile with the "make clean" command. Failure to do so will result in a deduction of 30 points.
3) Your report is essential, as it will serve as the primary basis for evaluation. Failure to submit your PDF report will result in a deduction of 100 points.
4) Poor synchronization, race conditions, crashes, and freezing will result in a deduction of 100 points.
5) Late submissions will not be accepted.
6) Failure to clean up after child processes or leaving zombies will result in a deduction of 30 points.
7) The presence of memory leaks (which will be verified with valgrind) will result in a deduction of 30 points.

Submission rules:

1) Your source files, your makefile and a report; place them all in a directory named “lastname_firstname_studentnumber”, and zip the directory.
2) Your makefile should only compile the program, not run it!
3) Your report must be in PDF format and should include a detailed explanation of how you solved the problem, which requirements you met, and which commands you tested
4) You should do your homework on your own. Your homework will be compared against online sources as well as each other’s homework. Proven cases of plagiarism will result in –100 grade.