CSE 344 System Programming - HW4

Objective: Develop an enhanced version of the directory copying utility "MWCp" from HW4. The new version will include a worker-manager approach to synchronize thread activity using both condition variables and barriers. Use POSIX and Standard C libraries.

Condition variables will be used to signal when the buffer is not empty (so workers can start processing) and when the buffer is not full (so the manager can add more items). When the buffer is full, the manager thread will wait on a condition variable. When a worker thread removes an item from the buffer, it will signal the condition variable to wake up the manager thread. Similarly, when the buffer is empty, worker threads will wait on a different condition variable. When the manager thread adds an item to the buffer, it will signal this condition variable to wake up the worker threads.

Use **barriers** to ensure that all worker threads wait at a certain point before proceeding. This can be useful to ensure that all threads have completed a phase of processing before moving on to the next phase.

Experimentation:

- Test different buffer sizes and worker numbers. We want you to try your program with at least tree different test scenarios like below.
    - Test1: valgrind ./MWCp 10 10 ../testdir/src/libvterm ../tocopy, #memory leak checking, buffer size=10, number of workers=10, sourceFile, destinationFile
    - Test2: ./MWCp 10 4 ../testdir/src/ libvterm/src ../toCopy #buffer size=10, number of workers=4, sourceFile, destinationFile
    - Test3: ./MWCp 10 100 ../testdir ../toCopy #buffer size=10, number of workers=10, sourceFile, destinationFile
- You should put the uncut screenshots for these scenarios at the end of the report.
- Your report should be short, you must summarize the changes made on top of HW4.
- If the command line arguments are missing/invalid your program must print usage information and exit.
- Observe behavior when exceeding file descriptor limits.
- Check for memory leaks (valgrind usage) and any stuck.
- Properly handle signals (ctrl+c).
- Don't use busy waiting of any kind, don't use timed waiting.
- Each thread should free allocated resources explicitly.

Submission:

StudentID_Name_Surname_HW5.7z
|→ Makefile
|→ StudentID_main.c
|→ … (Any other source **files**, not test directories!)
|→ StudentID_report.pdf (include SS with test cases)

Grading:

- You should report the changes made to hw4. Explain clearly and simply where the added code parts are and what changes they cause.
- If you used condition variable or barrier in hw4 and submitted again hw4 show it at the top of the report.
- If you used neither condition variable nor barrier in hw4 and submitted the exact same hw4, you will get -100 points.
- Compilation errors: -100 points.
- Make sure to include a make file with the "make clean" command.
- Provide a make file to compile your homework. Do not run your program inside make file. Just compile it.
- Failure to submit your PDF report will result in a deduction of 100 points.
- Late submissions will not be accepted.
- Memory leaks, segmentation faults and stuck controls are important, another main part is worker-manager structure with directory copying.