CSE 344 System Programming - HW4
Deadline: 24.05.2024 (sunday) 21.00 PM

Objective: Develop a directory copying utility called "MWCp" that copies files and sub-directories in parallel. Use a worker-manager approach to synchronize thread activity. Use POSIX and Standard C libraries.

Main Program:

- Accepts buffer size, number of workers, and source/destination directories as command-line arguments.
- Starts worker threads and waits for completion.
- Measures execution time to copy files in the directory. Keep statistics about the number and types of files copied.

Manager:

- You should have only one manager thread.
- Reads source & destination directory paths.
- Opens files for reading and creates corresponding files in the destination directory.
- 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. Then, two open file descriptors and names are passed into a buffer.
- Buffer structure: The manager waits to fill the reserved buffer, and worker waits for the buffer to empty.
- You manage the buffer (is it empty or full, is it okay to access the buffer or should the execution wait until it is available) so that the threads can be terminated gracefully.
- Notifies program completion when the producer finishes filling the buffer with file names for the given directories, it should set a done flag and exits.

Worker:

- Reads file information from the buffer.
- Copies files from source to destination.
- Writes completion status to standard output.
- Critical section: the producers and the multiple consumers write the standard output.
- Terminates when signaled.
- Worker thread pool: to regulate the number of active threads at any time. A fixed number of threads are made available to handle the load.

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 10 ../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, just describe the general structure with pseudocode.
- 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_HW4.7z
|→ Makefile
|→ StudentID_main.c
|→ ... (Any other source **files**, not test directories!)
|→ StudentID_report.pdf (include SS with test cases)

Grading:

- 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.