CSCI 4210 — Operating Systems
Exam 2 Question 10 — April 14, 2021 (v1.1)

Overview

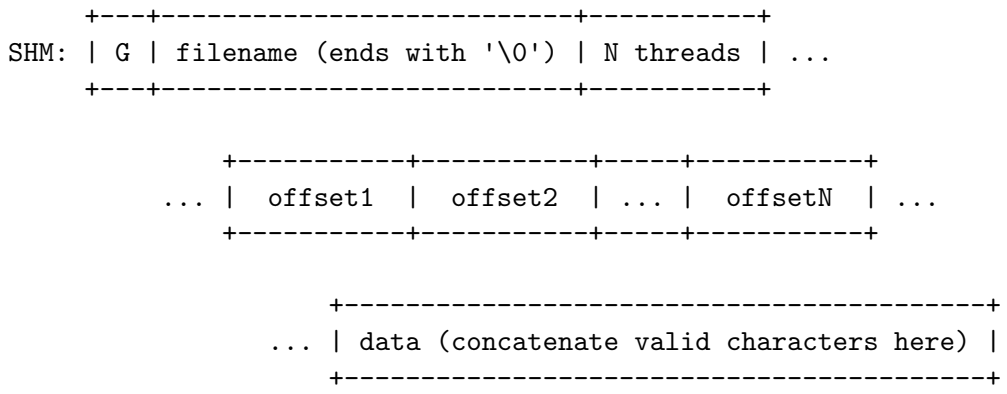
- You can continue to work on Exam 2 Question 10 and submit your code by 11:59PM ET on Monday, April 19, 2021
- Late days may **not** be used for this extended window
- Submit your `Q10.c` code on the separate "Exam 2 Question 10" gradeable; do **not** submit anything on the original "Exam 2" gradeable
- You will have 20 new submissions on the new gradeable before points start to be deducted
- Exam 2 "rules" still apply, i.e., do **not** copy or communicate with others about this question; further, do **not** post anything related to the exam on the Discussion Forum; that said, you may ask for assistance from a TA, mentor, or instructor during office hours
- Exam 2 Question 10 is auto-graded out of 35 points; your grade will be incorporated into your Exam 2 grade by replacing your current grade there, though we will use the higher of the two grades (so submitting further `Q10.c` attempts cannot hurt your previous grade)
- You will keep the 1 point of credit for submitting a PDF called `upload.pdf` on the original gradeable; please do **not** re-submit this PDF
- If you are taking a make-up exam, submit all of your work to the original "Exam 2" gradeable

10. **(35 POINTS)** Place all code for this problem in a `Q10.c` source file.

    For this problem, you are to write a multi-threaded program in C that gets its instructions from a shared memory segment. The shared memory key is given as the first command-line argument; assume the size of the shared memory to be 32,768 bytes.

    In your solution, each child thread reads from a given input file, concatenating valid characters to the "data" portion of the shared memory segment. A valid character is an alphanumeric character as identified via the `isalnum()` library function.

    The shared memory segment is structured as follows:



    The first byte of the shared memory segment is not used until all child threads have completed their work. Once your main thread knows all child threads are done, set the first byte of the shared memory segment to `'G'`. This will let the Submitty test code to evaluate your shared memory segment.

    The `filename` is a variable-length string ending with `'\0'`, e.g., `"inputfile.txt"`.

    The `N threads` field is a four-byte `int` variable indicating how many child threads you must create. It also tells you how many offset values are present (i.e., `offset1`, `offset2`, etc.). Each child thread is assigned an offset value, which indicates the byte offset to start reading from in the given file.

    The `data` portion of the shared memory segment is initially empty (i.e., set to `'\0'` or zero bytes). Each child thread reads from its given offset in the given file, copying exactly 111 valid characters from the file into the shared memory segment. If there are less than 111 valid characters through to the end of the file, copy only the valid characters encountered.

    The order in which these characters are added to the shared memory segment does not matter. Submitty will sort this data before validating it.

    Be sure to parallelize the child threads above to the extent possible.

    You are required to use `pthread_create()`, `pthread_join()`, and `pthread_exit()`. Further, each child thread must at least call `lseek()` and `read()`.

    Perform basic error-checking, returning `EXIT_SUCCESS` or `EXIT_FAILURE` from the main thread accordingly; however, note that the hidden test cases are all successful program executions.