Sam Dressler

CSCI 451

10/4/2020

HW 6

The input file was modified to include a “&” symbol that, when encountered, would cause a certain event to happen. During the first case, the thread reading “HW6-2.in” will read in a character from that file, as usual, and return control to the main thread. The main thread then checks if it should print out the character through a “print” Boolean variable and calls the “write\_char()” function that will cause the thread to exit if the character being printed is a “&.”

The result is the main thread ends by calling “pthread\_exit()” and the other three threads will remain locked. The remaining threads will remain locked because the architecture of the program uses mutex locks and conditions that will lock and then unlock each of the input threads one at a time until the end of each file is reached. Once the end of a file is met in one of the three threads, a boolean variable “finished#” will be set to true for each one of the three spawned threads. These are checked at the beginning of the loop in main and will continue reading characters until three of the condition variables are true.

The second case is similar, but when it encounters a “&”, it will call “kill -s 9 TID,” where TID is the thread that is executing when the command is called. I thought that when I would execute that command, the main process would end, and the program would quit but that is not what I saw. In my program with this modification, it lets the processing of the input files continue but the program will not finish successfully. This is because the input files are not all of equal length with the addition of the termination character. Since the program was not designed to handle files of various sizes, the second thread will get locked up and unable to reach the end of the file.