File Watch Program

This homework will require you to write an application in C or C++ that monitors a single file for access. The program should list all the files in a directory and allow the user to select one by name. Then, the program should continually monitor that file for read or write access. The program does not terminate (unless errors are encountered). The directory to list can be passed in on the command line. If it is not, the program should list files in the current working directory for the process. (Do not use C/C++ library I/O for printing the contents of the directory, other than `printf`.)

1. The program **shall** list each file in a supplied directory. The supplied directory will either be an argument or the current working directory as detailed below.
   (a) The program **shall** accept an optional path to a directory as a command-line argument.
   (b) When no argument is provided to the program, the program **shall** use the current working directory of the process for the path.
   (c) If the path provided to the program is not a directory, the program **shall** print an error and terminate.
   (d) The program **shall** only list the file name. Do not display a directory path.
2. The program **shall** allow the user to specify a watch file after printing the directory list. This file should be one that is located in the directory that was printed.
3. The program **shall** print that the file was read any time the watch file is read after making the selection. Your program should not exit normally. Continue printing forever.
4. The program **shall** print that the file was written to any time the watch file is written to after making the selection. Your program should not exit normally. Continue printing forever.
5. The program **shall** close all file descriptors or directory streams that it directly opened before it terminates. If you used a function to open the descriptor (open, dup) you must close it. If the shell automatically opened it for you (standard in, standard out) you do not have to close it.
6. The program **shall** print error information for each error in so much as the program was directly responsible for the method call that produced the error.
7. The program **shall** terminate after encountering an unrecoverable error (after printing error information).
8. The program **shall** not print excessive information.
9. You **shall** submit all source code and a makefile in a tar file.

(a) The makefile **shall** be able to build your source in a 32 bit Lubuntu VM.

BONUS You will get three (3) bonus points for completing this homework using the C programming language. Your makefile must compile the solution correctly (regardless of whether you use C or C++).