File Clone

This project requires that you write a C or C++ application to clone a file. You will need to be able to copy the entire contents of a file to a different location and update the meta information for the newly created file to match the original file.

1. Your program **shall** accept the name of a file on the command line. This should be the first argument for the program. This is the file that will be copied. It is herein referred to as the *source*.
   (a) If the source file does not exist or cannot be read, your program **shall** report an error and exit.
2. Your program **shall** accept a destination on the command line. This can be a path or a file.
   (a) If the destination is a path, your program **shall** use the same name for the destination file as the source file. This is what the `cp` command does.
   (b) If the destination is a file, your program **shall** use that file name for the destination file. This is what the `cp` command does.
      i. If the destination file exists, your program **shall** overwrite the file if, and only if, the *-f* option was specified. If the destination exists and *-f* is not provided, report an error and exit.
3. Your program **shall** copy all data from the source file to the destination file. Do not add anything extra. These files should be identical according to the `diff` command.
4. The destination file will have the same meta data as the source file:
   (a) The owner of the destination file **shall** be the same as the owner of the source file.
   (b) The group of the destination file **shall** be the same as the group of the source file.
   (c) The permission bits for the destination file **shall** be the same as the permission bits for the source file.
   (d) The last modified time for the destination file **shall** be the same as the last modified time for the source file. You can truncate to the nearest second.

   BONUS The type of the destination file will be the same as the type of the source file. Completing this requirement will give you three (3) bonus points. You will have to support directories, regular files, and symbolic links. (You may need to do extra work for the directory to work. Test with `diff`.)
5. Your program **shall** meet the basic requirements listed below:
   (a) Your program **shall** print error information for each error in so much as your program was directly responsible for the method call that produced the error.

(b) Your program **shall** terminate after encountering an unrecoverable error (after printing error information).

(c) You **shall** submit all source code and a makefile in a tar file.

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

   ii. The built program **shall** be able to run in a 32 bit x86 Lubuntu VM.

BONUS You will get two (2) 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++).

Example

The behavior of the application may be different depending on the input file type. A regular file should work perfectly. There are no technical limitations for copying a regular file (aside from what we expect). Directories, however, cannot be read from like normal files. Copying a directory is much more complicated and so you are not required to correctly handle this type. Instead, you should be able to create an output file and duplicate the meta data like you would for a normal file, but the contents won't be correct. It is reasonable to always operate on the target of symbolic links.

If you are trying to get the three bonus points for supporting file type correctly, you must be able to handle both symbolic links and directories. Symbolic links should be duplicated correctly as one would expect (linking to the original file when possible). Directories should have the same content for any type of regular file or symbolic link. Do not copy other files. Think about how to make two directories "identical" in terms of their "contents".