Homework #8

Question 1 (10 pt.)

The goal of this assignment is implementing a small file manipulation command-line tool with the flexibility to perform different actions on existing files. Its first command-line argument specifies the action to be performed by the tool, and the rest of the command-line arguments are interpreted based on that particular action. This is the list of supported actions, and their arguments:

- `./files info <file>`. Provide information about an existing file, passed in argument `<file>`. The information should include the inode number for the file, the size of the file in bytes, and its access permissions. Permissions should be provided in the same format as command "`ls -l`", excluding the left-most character that identifies the inode type (for example: `rw-r--r--`). You can check the documentation for system call `stat` (`man 2 stat`) for details on how to extract this information.
- `./files link <src> <dest>`. Create a new hard link for file `<src>`, named `<dest>`.
- `./files symlink <src> <dest>`. Create a new soft link for file `<src>`, named `<dest>`.
- `./files rm <file>`. Remove `<file>`.

In all cases, the tool should provide proper error messages if an action failed to execute, or if the tool is invoked with the wrong syntax. Function `perror()`, defined in the standard C library, can be useful here.

Upload your program in a file named `hw8.c`. The program should compile correctly on the CoE machines with the following command:

```
$ gcc hw8.c -o files
```

Here are some execution examples:

```
$ ls
files   files.c   Makefile

$ ./files info hello
Error: No such file or directory

$ ./files info Makefile
Inode: 10753369
Size: 43
Permissions: rw-rw-r--

$ ./files symlink Makefile Makefile2
$ ls -l
[ ... ] Makefile
[ ... ] Makefile2 -> Makefile
```