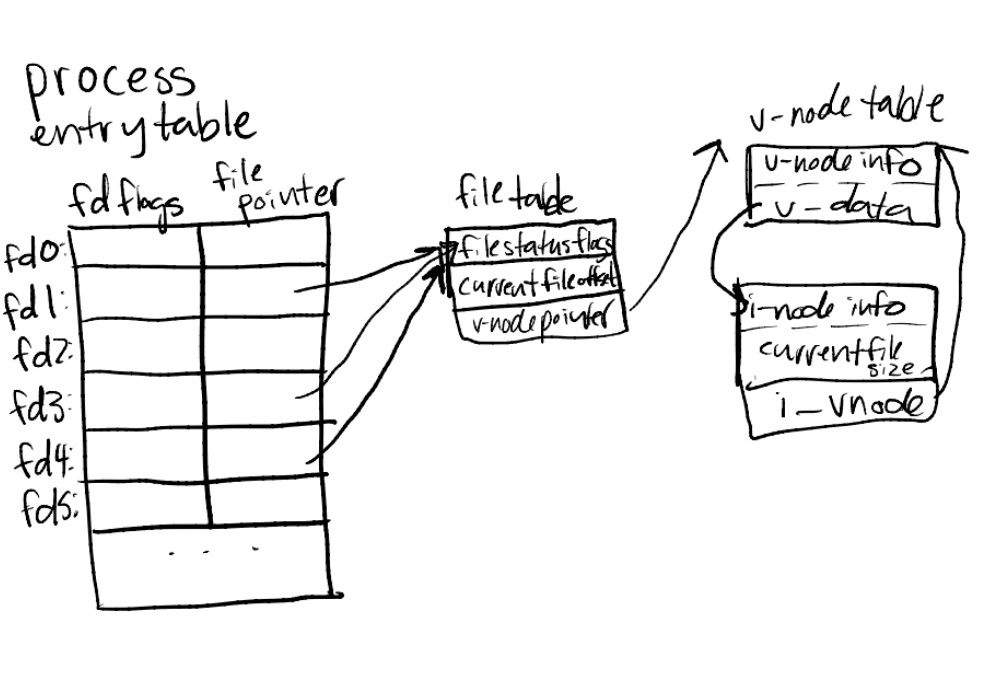
1. Program attached separately. This includes the files:

   - my_dup2_test
   - ReadMeQ4.md
   - Makefile (used for q1 and q4 compiling)

2.



For F_SETFD, the file descriptor flag would be affected.

For F_SETFL, the file status flags would be affected, such as O_APPEND, O_NONBLOCK, O_SYNC, etc.
. Only fd 1 would be affected by SETFL since it is only setting the descriptor flag of the one fd.

3. A hard link that gives the same inode number to files with different names. A soft link is a text string used as a shortcut by the operating to create a path to another file. With a soft link, there is a different inode between the file and the reference and if the original file is deleted, the soft link will no longer work.

For example, consider if you have a file called **King.txt**.

1) If you want to create a hard link, you can use the command `ln King.txt King_copy.txt`

```
[hertza@csci-gnode-02 ~]$ ln King.txt King_copy.txt
[hertza@csci-gnode-02 ~]$ ls
apue.3e       hw1-hertz.txt           hw2-hertz.txt  Lab1         src.3e.tar.gz
bashrc        hw2_1_AH.txt            hw3_AH         Lincoln.txt  tmp
csci3751      hw2_2_AH.txt            Kennedy.txt    Makefile     typescript
DSPD          hw2_3_AH.txt            King_copy.txt  Networks
find-result   hw2-hertz-script-4.txt  King.txt       OOP
```

King_copy.txt is now the same file as King.txt and points to the same inode.

2) Now, create a soft link to a new file named King_symbolic_copy.txt. Use the command 'ln -s King.txt Kint_symbolic_copy.txt`.

```
[hertza@csci-gnode-02 ~]$ ln -s King.txt King_symbolic_copy.txt
[hertza@csci-gnode-02 ~]$ ls
apue.3e         hw2_1_AH.txt            Kennedy.txt             Makefile
bashrc          hw2_2_AH.txt            King_copy.txt           Networks
csci3751        hw2_3_AH.txt            King_symbolic_copy.txt  OOP
DSPD            hw2-hertz-script-4.txt  King.txt                src.3e.tar.gz
find-result     hw2-hertz.txt           Lab1                    tmp
hw1-hertz.txt   hw3_AH                  Lincoln.txt             typescript
```

3) Remove the original file using the command `rm King.txt`

```
[hertza@csci-gnode-02 ~]$ rm King.txt
[hertza@csci-gnode-02 ~]$ ls
apue.3e         hw2_1_AH.txt            Kennedy.txt             Networks
bashrc          hw2_2_AH.txt            King_copy.txt           OOP
csci3751        hw2_3_AH.txt            King_symbolic_copy.txt  src.3e.tar.gz
DSPD            hw2-hertz-script-4.txt  Lab1                    tmp
find-result     hw2-hertz.txt           Lincoln.txt             typescript
hw1-hertz.txt   hw3_AH                  Makefile
```

4)Run `cat King_copy.txt` and you will be able to still view the contents of the file even though the original file was deleted.

5) Run `cat King_symbolic_copy.txt`

You will receive a "No such file or directory" error.

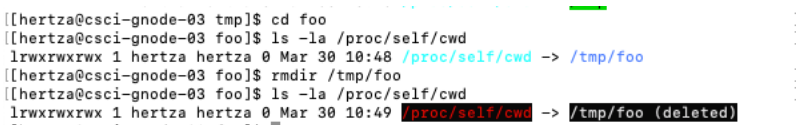
```
[hertza@csci-gnode-02 ~]$ cat King_symbolic_copy.txt
cat: King_symbolic_copy.txt: No such file or directory
```

This exercise demonstrates the difference between a hard and soft link.

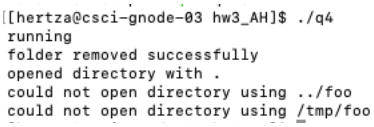
4.

1) The rmdir() command does succeed.

2) When the program tries to read the current directory, it is able to open that directory, because you set the current working directory (cwd) there, which is remembered by the kernel and the directory won't entirely go away until it is no longer the cwd. For example, if I go to the directory foo so that it is the cwd, and then I delete it, if I check what the cwd is, the kernel still remembers, it just shows it as deleted.



3) Program is found in the file q4.c, along with a makefile and ReadMeQ4. Here is the expected output (also found in the ReadMe).



5. The purpose of both fflush() and fsync() is to write data. However, as a C library function, fflush(aFilePointer) is part of the stdio library, which operates on the file structure. The fsync(2) system call operates on the kernel and can tell the OS that it needs a buffer on the hardware storage.

If you ran these two in succession, the output stream would be written to a buffer that is associated to the operating system first when you ran fflush().

Then, fsync(fileno(aFilePointer)) would cause the OS to write a buffer to your physical media.