Closing Files | Coursera

coursera.org/learn/interacting-system-managing-memory/supplement/DveF8/closing-files

Closing Files

Closing Files

After you are finished with a file, you should close it with the **fclose** function, which has the following prototype:

```
1

int fclose(FILE * stream);
```

This function takes one argument, specifying which stream to close. Closing the stream sends any buffered write data to the OS, and then asks the OS to close the associated file descriptor. After calling **fclose** on a stream, you may no longer access it (a call to **fgetc**, **fprintf**, etc. is erroneous, though exactly what will happen is undefined).

Observe that **fclose** returns anint. This return value indicates the success (0 is returned) or failure (EOF is returned, and **errno** is set accordingly) of the **fclose**operation. The **fclose**operation can fail for a variety of reasons, the most serious of which arise in circumstances where the data cannot actually be written to the underlying hardware device (*e.g.*, disk drive)—for example, the disk is full, or the file resides on a remote file system and network connectivity has been lost. Failure of **fclose** for a file you have been writing to is a serious situation—it means that the data your program has tried to write may be lost. You should therefore, always check the return value of **fclose**to see if it failed.

However, what you do in response to the failure of **fclose**is a bit of a difficult question. You cannot try again, as performing any operation on the stream you attempted to close—including another call to **fclose**—is erroneous (and results in undefined behavior). What you do, however, is highly situation-dependent.

In an interactive program, you may wish to inform the user of the problem, and she may be able to take corrective action before proceeding. For example, suppose you are writing an editor (like emacs). If the user attempts to save their work but the disk is full, she would much rather be told that the save failed (and why). The user could then proceed to free up disk space, and save again (which would involve **fopen**ing the file again,

fwriteingall the data, the **fclose**ing that stream—not retrying to**fclose** the original stream). By contrast, if the editor ignored the return value of **fclose**and *failed silently* — not informing the user of the problem, she may quit, losing all of her work.

In other situations, you may not be directly interacting with the user (or a user capable of remedying the situation: imagine if you are writing some web service—the user you are interacting with typically has no ability to administer the system). You still will want to detect the problem and take some sort of corrective action.

We will not concern ourselves with complex corrective actions when **fclose** fails—printing an error message suffices. However, you should get in the habit of checking its return value. This way, when you are working on real programs, you will check the return value by habit, and at least think about what you should do if it fails.