Writing to Files | Coursera

coursera.org/learn/interacting-system-managing-memory/supplement/OFKPW/writing-to-files

Writing to Files

Writing to Files

The other operation we may wish to perform is to write to a file. As with reading from a file, there are a variety of options to write to a file. One option—useful if we are printing formatted text—is the **fprintf** function. This function behaves the same as the familiar **printf** function, except that it takes an additional argument (before its other arguments), which is a **FILE *** specifying where to write the output.

You can also use **fputc** to write a single character at a time, or **fputs** to write a string without any format conversions. That is, if you do **fputs("%d")** it will just print "%d"to the file directly rather than attempting to convert an integer and print the result.

Finally, if you want to print non-textual data, your best choice is **fwrite**, which has the prototype:

```
1
size_t fwrite(const void * ptr, size_t size, size_t nitems, FILE * stream);
```

The arguments are much the same as those given to **fread**, except that the data is read from the buffer pointed to by **ptr** and written to the stream (whereas **fread** reads from the stream and writes into the buffer pointed to by **ptr**).

All of these methods write at the current position in the file and advance it accordingly. Furthermore, any of these methods of writing to a file may fail, some of which are detected immediately, and others of which are detected later. See the relevant man pages for the functions you are using to see how they might fail and what values the return to indicate failures.

The reason that the failures may be detected later is that the C library functions may buffer the data, and not immediately request that the OS write it. Even once the application makes the requisite system calls to write the data, the OS may buffer it internally for a while before actually writing it out to the underlying hardware device. The motivation for not writing immediately is performance: making a system call is a bit of a slow process, and writing to a hard disk is quite slow. Furthermore, writing a disk

becomes less efficient as you write smaller quantities of data to it—there are fixed overheads to find the location on the disk which can be amortized over large writes—so the OS tries to buffer up writes until there is significant data that can be written at once.

Completed