ENCE360 Lab 9: Buffered I/O

Objectives

This lab explores the performance of using buffered and non-buffered I/O. We will compare it to the `mmap()` copying we did in Lab 4.

The aims of this lab are for you to understand the performance benefits and tradeoffs of buffered I/O. This lab is relatively short, to give you a little extra time to work on your assignment. Get onto it!

Preparation

Download and extract your Lab 9 files from `lab9BufferedIO.zip` on Learn. This contains the files `fread_fwrite.c`, `read_write.c`, `perf.sh`, and `Makefile`.

Also, copy your solution programs `mmap.c` and `realloc.c` from lab 4 into this lab's directory. We will compare performance to them.

Simply executing `make` in the lab directory should build all programs. In addition, running `./perf.sh` will measure the performance and assess the correctness of each program.

Once you've completed the tasks below, make sure you also complete the quiz on the quiz server.

Program `read_write.c`

`read()` and `write()` are the system calls the OS supplies for file I/O. The OS performs no process-level buffering on these operations, which is why they are thread-safe.

Implement copying where the comments indicate; the files are already opened. We supply an intermediary user-space array `buffer` to use.

Check correctness by running `./perf.sh` before continuing.

Program `fread_fwrite.c`

`fread()` and `fwrite()` are C library functions that use system calls under the hood. They internally buffer operations to minimise the number of system calls that need to be made. Implement copying using the standard library functions instead.

Compare Performance

Run `./perf.sh` with all programs working. How do your `read_write.c` and `fread_fwrite.c` compare at different user-space buffer sizes? How do they differ? Why? Explain the differences between these approaches and `mmap.c`.