What is this?

A few examples of system calls to use in class before diving into the wonderful world of forking processes. The examples are as follows:

- `write_to_file.c`: opens a file for writing and dumps a small text message. the file is created if needed with permissions 0640 (unless running in WSL, then all bets are off because who knows how Microsoft implemented the OS service call)
- `append_to_file.c`: same as `write_to_file.c`, except the file is opened in append mode. NOTE: writing includes the terminator ('\0') character, so expect funny business if opening it with a simple application that assumes that's the EOF delimiter.
- `read_file.c`: reads the contents of `output_dump.txt`, and shows them using `printf`. Will fail gracefully if the file doesn't exist yet. Showcases basic memory management, though subtler strategies (e.g., memory mapping) are left untouched.
- `flush_file.c`: a simple copy of `write_to_file.c` that manually trims the size of the file to 0 bytes. useful when one wants to flush the contents.

Utility files:

- `output_dump.txt`: generated and modified by `write_to_file.c`, `append_to_file.c` and `flush_file.c`. Read by `read_file.c`. Used to show basic manipulation strategies.
- `output_dump2.txt`: a custom, manually made `output_dump.txt`. copy to `output_dump.txt` for use or modify the hardcoded path to target in `read_file.c`. Designed to show a slightly longer example, and how the code does not break with it.
- `Makefile`: standard Makefile. Contains rules to generate and clean all the code. Will not remove text files; those are up to the user to clean up.

Topics touched upon

1. Basic file manipulation (opening, reading, writing) in C.
2. Basic memory management (allocation, deallocation) in C.

Topics *not* touched upon

1. Console input handling with `argc` and `argv`.
2. More sophisticated strategies (e.g., memory mapping), suitable for large files.
3. Void pointers.
4. Structs (briefly mentioned, but not elaborated on)

Closing remarks

The contents are provided as-is, with the bare minimum testing and as much documentation as was humanly reasonable. The goal is not to showcase perfectly optimized C code nor best coding practices, but to quickly and briefly exemplify system calls related to the very, barebone basics of file and memory management. For that purpose, these examples are sufficient, though they could probably be improved upon greatly.

For anything more involved, including forking processes, mutexes, synchronization, race conditions (and their solution), struct handling... These examples fall woefully short. Caution is, thus, strongly advised.