- CPSC 275: Introduction to Computer Systems

CPSC 275: Introduction to Computer Systems

Fall 2025

- Syllabus
- Schedule
- Resources
- Upload
- Solution

Homework 30

In these exercises, you will empirically compare the performance of low-level Unix system calls with standard I/O library functions when copying large files. The goal is to understand how buffering and the overhead associated with switching between user-space and kernel-space affect I/O performance and overall file copying speed.

1. To ensure a meaningful comparison that demonstrates the effect of loop iterations and buffer size, the input file must be significantly larger than the maximum buffer size. We will use a 100 MB file. Use the `dd` command to create a file of 100 MB with random content.

   ```
   # Creates a 100 MB file (100 blocks of 1 MB)
   $ dd if=/dev/urandom of=input.bin bs=1M count=100
   ```

2. Write a C program, `copy1.c`, that copies a file using only the system calls `open()`, `read()`, `write()`, and `close()`.

   ```
   $ ./copy1 srcfile destfile [buffer_size]
   ```

   Use `clock_gettime()` with `CLOCK_MONOTONIC` to measure the elapsed time precisely. Record the start time immediately before the copy loop and the end time immediately after the loop finishes. Finally, print the total execution time in milliseconds.

3. Write a C program, `copy2.c`, that copies a file using standard I/O: `fopen()`, `fread()`, `fwrite()`, and `fclose()`. Run the program with the same input file and the same buffer size used in `copy1.c`. Measure and print the elapsed time.

4. Repeat the experiments (running both `copy1` and `copy2` with the 100 MB file) using the following buffer sizes for the internal read/write operations: 1B, 64B, 4KB, 64KB, 1MB.

5. Answer the following questions based on your empirical results:
   A. For the smallest buffer size (e.g., 1 byte), which version is faster, and why? (Hint: Consider the cost of user-space/kernel-space switching).
   B. How does performance change as the buffer size increases for both programs? At what buffer size do the performance gains start to diminish?

C. Does standard I/O always outperform system calls? Why or why not? Consider situations where the OS's native buffering might be more or less efficient than the standard I/O's user-space buffering.
D. What role does the user-space buffering provided by `fread`/`fwrite` (in addition to the buffer size you defined) play in the performance compared to repeated `read`/`write` system calls?
E. Suppose you were writing a high-performance file copying tool. In what situations might you choose pure system calls versus standard I/O?

- **Welcome: Sean**
  - LogOut