meowLab实验报告

首先测试cat的效率，用时为184.8ms

任务3：缓冲区对齐的cat

1.1）将缓冲区对齐到系统内存可以优化CPU缓存行加载和存储，减少缓存未命中，减少内存访问次数；

由于硬件通常以页面为单位操作，对于内存访问操作更高效；

可以减少页面跨越、缓存行分裂，提高缓存命中率。

2）多次运行后任务二平均用时326.1ms，任务三平均308ms，支持但有时差异不明显，可能是现代硬件和操作系统优化了非对齐内存的处理。

2.malloc通常只保证基本对齐（如8或16字节对齐）以适应通用数据结构，不保证页面对齐，因为这会增加内存碎片。malloc的实现通常会在分配的内存前后添加额外的信息用于管理内存。

3.在对齐指针前存储原始指针，释放时通过存储的原始指针来释放内存。

任务4：设置缓冲区大小为文件系统块大小的整数倍的cat

1.文件系统以块为单位进行磁盘I/O操作。如果缓冲区大小是文件系统块大小的倍数，可以确保每次读取或写入都与底层磁盘操作对齐，减少部分块读取或额外的数据传输，从而提高效率。此外，操作系统可能对块大小对齐的I/O请求进行更有效的缓存和调度。

2.1）通过stat()系统调用获取目标文件的st_blksize，针对每个文件动态调整缓冲区大小，而不是假设全局一致。

2）检查st_blksize是否有效（非零且为2的幂，检查(blocksize & (blocksize - 1)) == 0），且不小于页大小。

任务5：考虑系统调用开销情况下的cat

这个实验脚本旨在通过系统化测试不同缓冲区大小对文件读写性能的影响，设计目标是优化I/O吞吐量。其核心逻辑包括使用`lcm`函数计算页面大小和文件系统块大小的最小公倍数以确定初始缓冲区大小，`get_buf_size`函数获取系统参数并对齐缓冲区，`measure_rate`函数通过模拟从`/dev/zero`读取1GB数据并写入`/dev/null`来测量速率，最后`run_experiment`函数通过多种倍数（如1、2、4等）重复测试三次取平均值，并用`matplotlib`绘制倍数与速率的折线图。这种设计通过对齐系统参数、重复测量和可视化分析，有效探索最佳缓冲区大小，适用于Linux/Unix环境，旨在提供性能优化参考。

倍数: 256, 缓冲区大小: 1048576 字节, 速率: 26377.6 MB/s

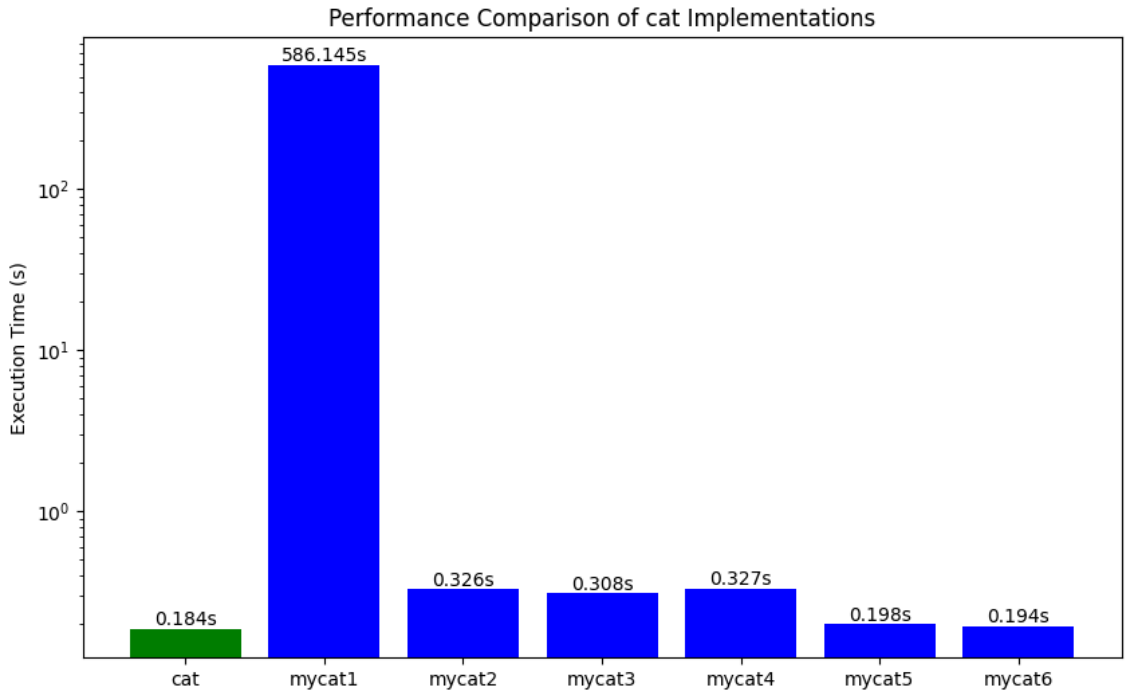
任务6：使用系统调用fdadvice的cat

1.使用POSIX_FADV_SEQUENTIAL告知系统我们将顺序读取文件，这提示内核可以预读更多数据，优化I/O性能。

2.顺序读取：增加预读窗口大小，提前读取更多数据到缓存

随机读取：减少预读或禁用预读，避免缓存污染

任务7：总结



上述改进分析：mycat1(无缓冲)性能最差(586s)，因为每个字节都需系统调用，添加缓冲区后mycat2性能提升约2000被，进一步内存对齐以及考虑文件系统块大小有一定改进，但提升不明显，考虑系统调用开销和增大缓冲区后mycat5和6提升较大，与cat差距较小。

mycat6 未完全达到 cat 的性能，可能因为 GNU cat 还使用了其他优化。

因此，适当的缓冲区大小是 I/O 性能的关键，需平衡系统调用开销和内存使用；内存和文件系统块对齐可显著提升性能；同时系统调用如 fadvise，通过与操作系统协作，可以进一步优化 I/O。