Lab 1 Report

实验环境：macOS Sonoma 14.1

实验数据

`read()`效率

BUFFSIZE(B)	User(s)	System(s)	Total(s)	#Loops
1	4.83	15.32	20.60	33554432
4	1.21	3.85	5.11	8388608
16	0.31	0.97	1.32	2097152
64	0.08	0.24	0.74	524288
256	0.02	0.06	0.53	131072
1024	0.01	0.02	0.47	32768
4096	0.00	0.02	0.44	8192
16384	0.00	0.01	0.45	2048
65536	0.00	0.01	0.43	512
262144	0.00	0.01	0.42	128

`fread()`效率

BUFFSIZE(B)	User(s)	System(s)	Total(s)	#Loops
1	0.90	0.01	1.40	33554432
4	0.24	0.01	0.65	8388608
16	0.07	0.01	0.52	2097152
64	0.03	0.01	0.60	524288
256	0.01	0.01	0.50	131072
1024	0.01	0.01	0.50	32768
4096	0.00	0.01	0.44	8192
16384	0.00	0.01	0.46	2048
65536	0.00	0.01	0.44	512
262144	0.00	0.01	0.41	128

`write()` with `O_SYNC` flag效率

BUFFSIZE(B)	User(s)	System(s)	Total(s)	#Loops
1	5.76	370.12	943.66	33554432
4	1.49	92.74	242.21	8388608
16	0.37	22.79	90.40	2097152
64	0.09	5.72	15.93	524288
256	0.02	1.47	5.34	131072
1024	0.01	0.37	2.95	32768
4096	0.00	0.11	0.85	8192
16384	0.00	0.04	1.51	2048
65536	0.00	0.02	0.62	512
262144	0.00	0.01	0.40	128

`write()` without `O_SYNC` flag效率

BUFFSIZE(B)	User(s)	System(s)	Total(s)	#Loops
1	4.83	15.32	20.60	33554432
4	0.94	17.41	18.59	8388608
16	0.31	5.70	6.48	2097152
64	0.08	1.45	1.94	524288
256	0.02	0.37	0.79	131072
1024	0.01	0.11	0.50	32768
4096	0.00	0.03	0.44	8192
16384	0.00	0.02	0.47	2048
65536	0.00	0.02	0.52	512
262144	0.00	0.01	0.38	128

数据特点

`read()`

- 系统CPU时间大于用户CPU时间，一般为用户CPU时间的3倍左右

- 随着缓冲区大小`BUFFSIZE`的增大，用户CPU时间、系统CPU时间和时钟时间均减小，`BUFFSIZE`达到256之后三种时间变化较小，考虑到`time`的误差，可以认为基本保持不变

`fread()`

- 用户时间在随`BUFFSIZE`的增大而减小，在`BUFFSIZE`大于1024后小于系统时间
- 系统时间在不同`BUFFSIZE`下基本保持不变
- 与`read()`相比，`fread()`在`BUFFSIZE`从1到256区间内用时明显较小，对于更大的`BUFFSIZE`则区别不明显

`write()` with `O_SYNC`

- 系统时间占主导
- 缓冲区大小`BUFFSIZE`从1增大到256过程中，用户CPU时间、系统CPU时间和时钟时间均显著减小（大约与`BUFFSIZE`呈反比）；`BUFFSIZE`从1024继续增加，用户时间基本不变，其他两种时间继续减小
- 用户时间与系统时间之和与时钟时间之间有较大差距（小于时钟时间）

`write()` without `O_SYNC`

- 随着缓冲区大小`BUFFSIZE`的增大，用户CPU时间、系统CPU时间和时钟时间均减小，`BUFFSIZE`达到256之后时钟时间基本稳定
- 不带`O_SYNC`的`write()`系统时间和时钟时间明显小于带`O_SYNC`的

分析

- `read()`和`write()`中，当`BUFFSIZE`较小时，循环次数较大，会有很多次系统调用，这些系统调用耗时较长（可能自身周期数多，且需要os在user mode和kernel mode之间切换，即trap和return from trap），使得系统CPU时间往往比较长；`BUFFSIZE`较大后，系统调用的次数不多，故系统CPU时间明显减少。系统时间到达约0.01s后趋于稳定，可能是因为运行程序本身也涉及一次user mode和kernel mode的切换，加上不可避免的一次I/O，当程序中的系统调用次数很少时，造成的影响相对就不是那么明显了。
- `fread()`效率优于`read()`是因为它是标准库中封装好的函数，是buffered I/O，内部设置好了较大的buffer size从而可以减小系统调用次数，把系统时间控制在较小范围。
- 带`O_SYNC`的`write()`时钟时间明显大于用户时间与系统时间之和的原因是，其他几个函数在进行I/O时CPU会转而执行其他进程，而带`O_SYNC`的`write()`进程会一直占用CPU直到I/O完成，这部分时间也算在了时钟时间里

提高I/O效率

- 减少`read()`和`write()`等系统调用次数
- 设置合适的`BUFFSIZE`
- 使用`fread()`等标准库函数