# Lab1-Report

21302010042

侯斌洋

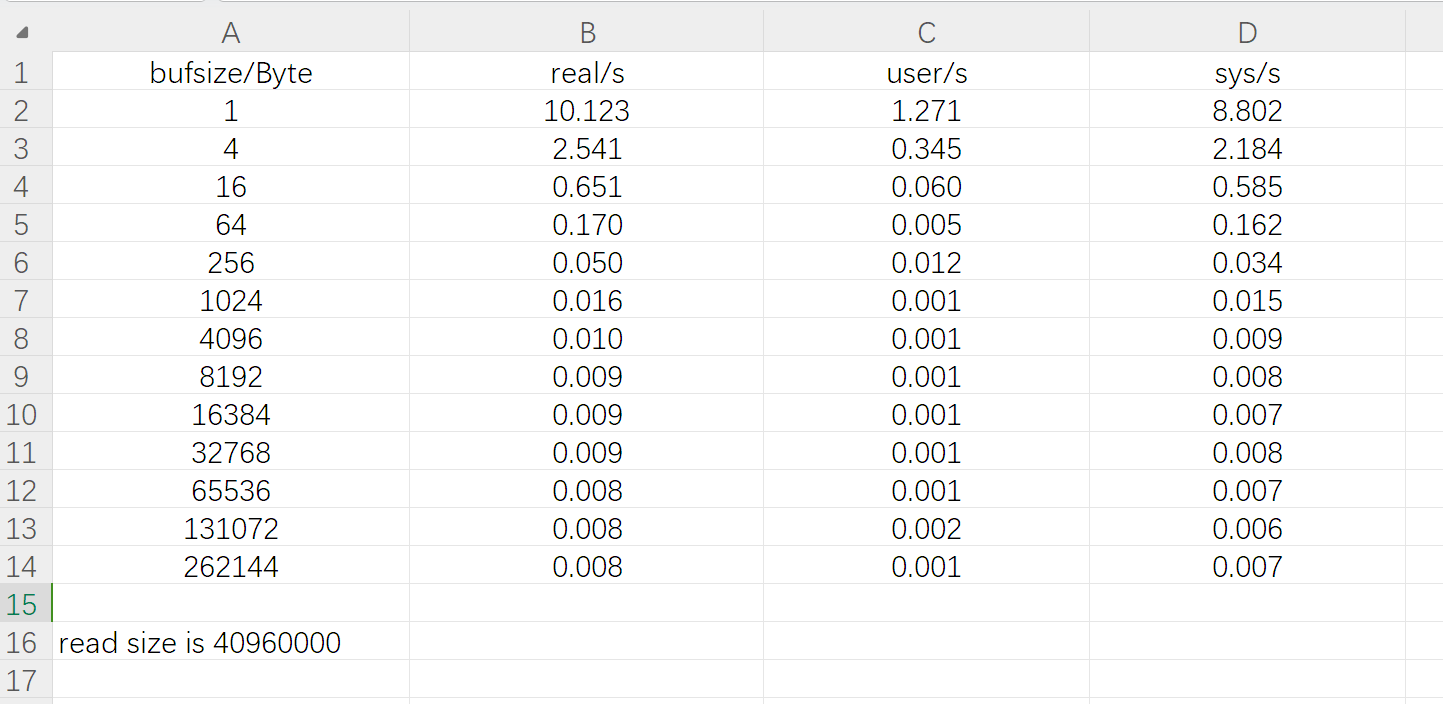
**实验环境：ubuntu22.04**

1：实验数据

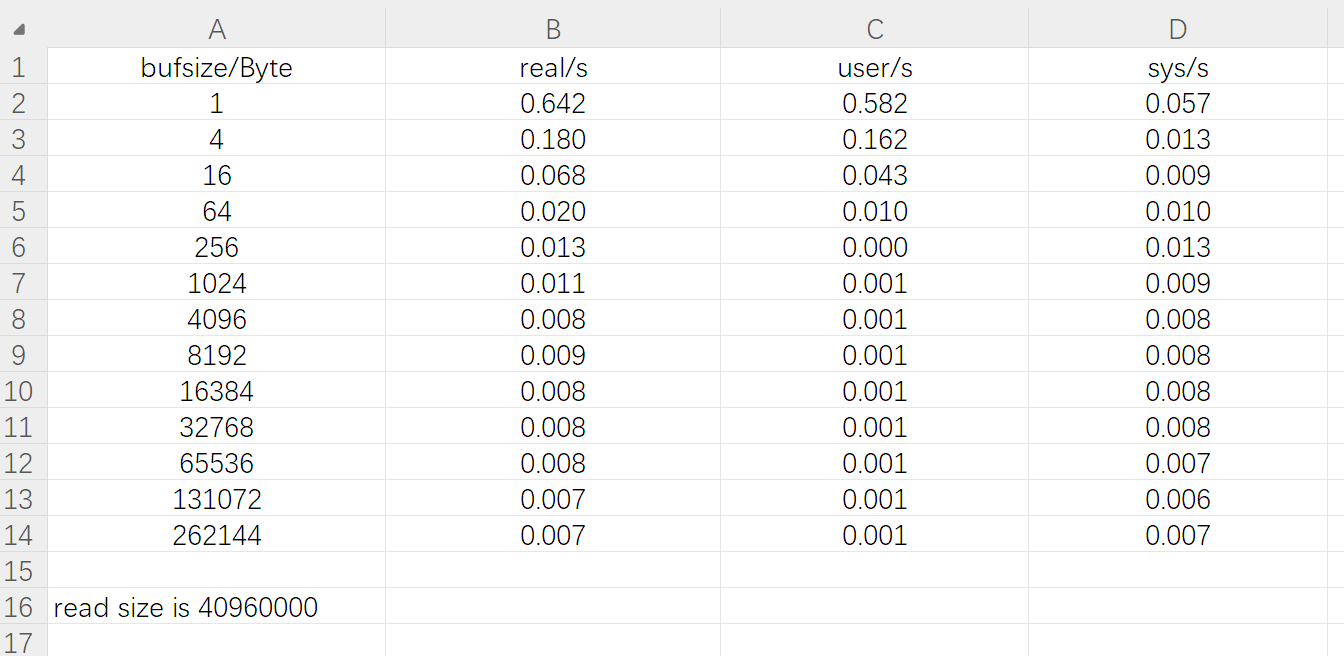
实验数据源文件为lab1\_test\_data.xlsx。

以下为截图：

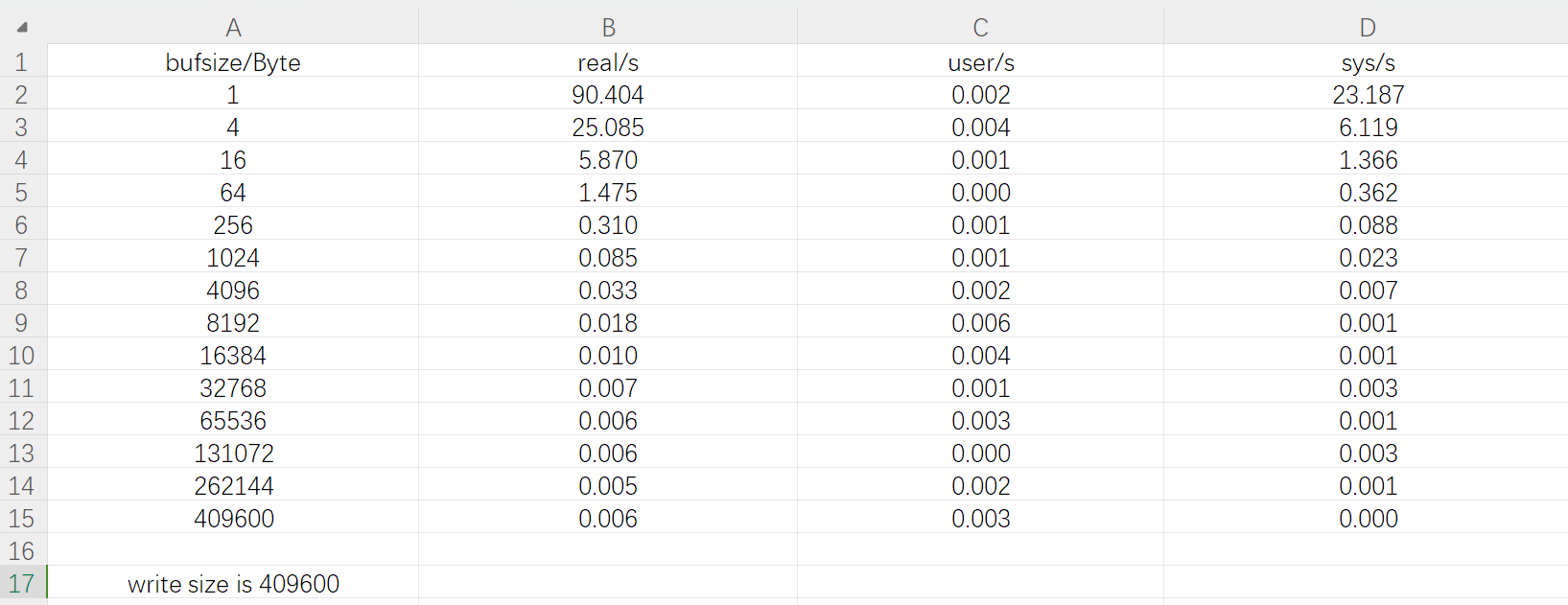
（1）测试read()



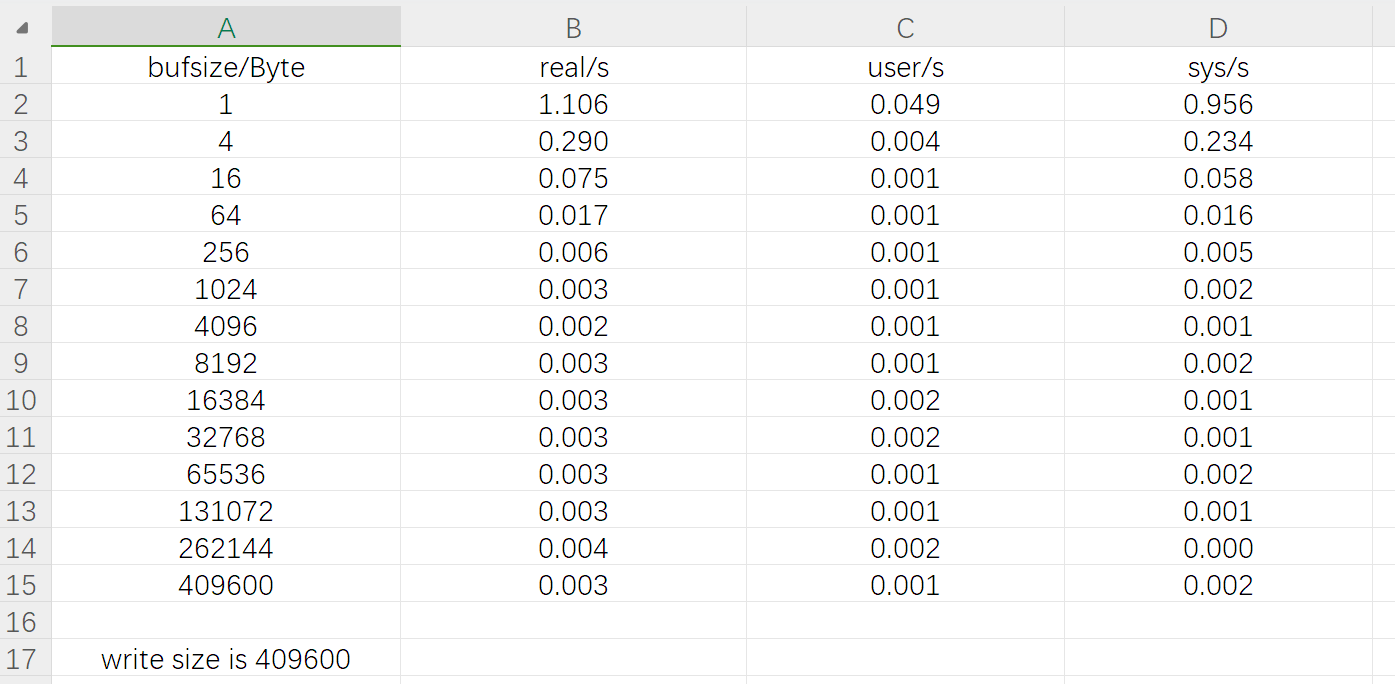
（2）测试fread()



（3）测试带O\_SYNC的write()



（4）测试不带O\_SYNC的write()



*数据特点：*

1. 系统CPU要比用户CPU调用的时间更长，时钟时间也主要由系统CPU组成。

在buffsize从1增大到4096的过程中，时钟时间明显减少，从4096增大到262144的过程中时钟时间变化很小，考虑到time测试程序时间的误差，可以认为buffsize超过4096后时钟时间基本保持稳定。

1. buffsize小于64时时钟时间主要由用户CPU组成，大于64时主要由系统CPU组成。

在buffsize从1增大到4096的过程中，时钟时间明显减少，从4096增大到262144的过程中时钟时间变化很小，考虑到time测试程序时间的误差，可以认为buffsize超过4096后时钟时间基本保持稳定。

###fread与read数据进行对比的话，可以看出在buffsize小于4096时fread的时###钟时间明显更短，而buffsize大于4096后差距不大。

1. buffsize小于1024时系统CPU要比用户CPU时间明显更长，大于1024时差距不明显。

系统CPU和用户CPU之和明显小于时钟时间，此特点为（3）所独有。

在buffsize从1增大到32768的过程中，时钟时间明显减少，从32768增大到409600的过程中时钟时间变化很小，

1. buffsize小于64时系统CPU要比用户CPU时间明显更长，大于64时差距不明显。

在buffsize从1增大到1024的过程中，时钟时间明显减少，从1024增大到262144的过程中时钟时间变化很小

###带O\_SYNC和不带O\_SYNC的write相比，不带O\_SYNC的write时钟时间明显###更短。

2：分析与思考

在buffsize较小时，循环调用read，write的次数会很多，而read等系统调用内部存在大量的代码，故系统调用的时间会明显大于用户CPU。而当buffsize较大时，循环次数减少，系统调用的次数也减少，故系统CPU的时间会明显减少，到达一定范围时，调用次数减少不那么明显了，由于单次系统调用本身开销也较大，故会保持稳定。

对于（3）中系统CPU和用户CPU之和明显小于时钟时间的特点，这是由于read，fread，不带O\_SYNC的write在进行IO时CPU不会一直等待，而是切换到其他进程，而带O\_SYNC的write则会一直等待IO慢慢完成。此时time算在了时钟时间里但由于并未执行任何代码而只是等待，故会存在real与sys+user的差距。

read和fread的buffsize均在4096左右出现临界值，这是由于磁盘上的一页大概就是4096的大小，由于磁盘上的文件是以页来区分的，每一页都需要单独寻找，故如果磁盘每次找到的一页能被完全读取，则就可以达到最大IO效率。所以会在4096处出现临界值。而之所以fread比read在buffsize较小时效率明显高，是因为fread是包装函数，函数内部自动设立了较大的缓冲区，故效率相对较高。

对于（3）（4）中buffsize效率临界出现在32768和256，则可能与write的内部实现，文件系统空间分配以及磁盘的IO方式等多种因素有关，不过可以看到的是（3）（4）都存在一个临界值可以使IO效率更高，说明write对处理某个大小的整块的效率也更高。

3：提高IO效率

1. 一般情况下可以尽量避免使用read等系统调用，转而使用fread等包装函数。因为使用系统调用时需要仔细考虑buffsize的设置，否则会严重影响IO效率，而使用fread等包装函数则不需要考虑buffsize的问题，包装函数内部会自动设置较为合理的缓冲区。但由于fread进行了包装，内部存在一些冗余代码，可能也会影响程序效率，要根据具体场景进行取舍。
2. 如果对所使用的设备有一定了解的话，比如知道磁盘一页的大小，文件系统的运作方式等，则可以在编写程序时使用合适的buffsize来提高IO效率。
3. 系统调用会占用很多CPU时间，在使用read,write等系统调用循环读取时要注意减少循环的次数以减少系统调用的开销。