设备管理习题

姓名 曾崇然　　学号 2152809

第一部分　写操作

一、概念题

1、同步 IO

提出块设备读写请求的进程需要等待 IO 操作结束，然后再继续执行后续操作。

2、异步 IO

提出块设备读写请求的进程无需等待 IO 操作结束就可以继续执行后续操作。

3、为什么读是同步的，写是异步的

因为数据处理往往是先读入信息然后加工，所以读是同步的；

而输出数据与加工处理后一批数据一般是并不冲突的，可以并行进行，因此写是异步的。

4、磁盘数据块为什么要先读后写

保护数据块不被改写的部分。

5、延迟写操作的优点和不足

延迟写操作是指如果缓存没有写满，考虑到短时间内可能还会继续写下去，不急于进行写块设备操作，保持缓存中的内容，将写块设备操作推迟到某个恰当时刻进行。

优点：（1）写操作不用等 IO 完成，耗时降低很多（2）合并写操作，减少写 IO 操作的数量。

不足：牺牲文件系统一致性，会丢失文件系统更新。

6、Unix 系统何时将脏缓存写回磁盘

①LRU 的脏缓存块被分配给其它数据块时；

②写至缓存底部时；

③磁盘卸载 unmount 时；

④关机时。

二、以下操作引发几次 IO？进程会不会睡？不考虑预读。

1、 读磁盘数据块，缓存命中

缓存命中，数据直接从内存中获取，0 次 IO，不会睡眠。

2、 读磁盘数据块，缓存不命中

缓存不命中，需要从磁盘中读取数据，1 次 IO，进程等待数据读取时陷入睡眠。

3、 写磁盘数据块，缓存命中

缓存命中，两种情况：

①若缓存处于空闲状态，可以直接使用，0 次 IO，不会睡眠

②若缓存正被某个进程使用，进入睡眠，待该缓存使用完毕被释放时再被唤醒，唤醒后该进程写缓存 0 次 IO。但是另外那个进程使用缓存完毕释放之前，进行 1 次 IO 将缓存内容写到相应设备上。

4、 写磁盘数据块，缓存不命中

缓存不命中，需要在自由缓存队列中进行分配，两种情况：

（1）自由缓存队列空，进入睡眠，待某个缓存被释放进入自由缓存队列时再被唤醒，唤醒后该进程写缓存 0 次 IO。但是释放缓存的那个进程使用缓存完毕释放之前，进行 1 次 IO 将缓存内容写到相应设备上。

（2）自由缓存队列非空，从队首取出缓存，若该缓存延迟写标志被设置，则将缓存内容异步写到相应设备上，1 次 IO，然后要求分配缓存的进程立即重复分配缓存操作，直到找到第一个未设置延迟写标志的缓存。有几个脏缓存就会异步进行几次 IO 操作，如果全都是脏缓存就会陷入睡眠，类似于上述情况（1）。

三、T1 时刻，PA、PB、PC 进程先后访问文件 A，PA read 4#字节，PB write 200#字节，PC read 500#字节。已知文件 A 的 0#逻辑块 存放在 55#扇区。T1 时刻缓存不命中。自由缓存队列不空，所有自由缓存不脏（不带延迟写标识），队首缓存块 Buffer[7]。

1、请分析如下时刻进程 PA，PB 的调度状态 和 Buffer[i]的使用状态。

- PA 执行 read 系统调用

  PA 执行 read 系统调用，需要读取 4#字节，缓存不命中，Buffer[7]是 LRU 自由缓存，且不脏，分配用来装新的数据块（55#扇区）。刷新 m_buf[7]：dev=0, blkno=55,B_READ=1，B_DONE=0，送 IO 请求队列之后，PA 执行 IOWait 函数入睡：sleep(&m_buf[7], -50)，高优先权睡眠。等待 IO 完成，也就是 m_buf[7]的 B_DONE 变 1。

  Buffer[7]状态：上锁(B_BUSY==1)，在 IO 请求队列；数据不可用（B_DONE==0）。进程 PA 持锁。

- PB 执行 write 系统调用

  PB 执行 write 系统调用，需要写入 200#字节，缓存命中，但已经设置 B_BUSY 标志，PB 执行 GetBlk 函数时入睡：sleep(＆m_buf[7], -50)，高优先权睡眠。等待持锁进程 PA 执行 Brelse( )解锁，也就是 Buf[7] 的 B_BUSY 变 1，B_WANTED 置 1。

  Buffer[7] 状态：上锁(B_BUSY==1)，在 IO 请求队列；数据不可用（B_DONE==0）。进程 PA 持锁。有进程等待使用其中的数据（B_WANTED==1）。

- PC 执行 read 系统调用

  PC 执行 read 系统调用，需要读取 500#字节，缓存命中，但已经设置 B_BUSY 标志，PC 执行 GetBlk 函数时入睡：sleep(＆m_buf[7], -50)，高优先权睡眠。等待持锁进程 PA 执行 Brelse( )解锁，也就是 Buf[7] 的 B_BUSY 变 1，B_WANTED 置 1。

  Buffer[7] 状态：上锁(B_BUSY==1)，在 IO 请求队列；数据不可用（B_DONE==0）。进程 PA 持锁。有进程等待使用其中的数据（B_WANTED==1）。

- 55#扇区 IO 完成

  中断处理程序执行 IODone( )函数，m_buf[7] B_DONE=1 置 1，唤醒 PA、PB 和 PC。PA、PB 和 PC 就绪，之后依次上台运行。

  Buffer[7] 状态：上锁(B_BUSY==1)，不在 IO 请求队列，在设备缓存队列，不在自由缓存队列；数据可用（B_DONE==1）。进程 PA 持锁。有进程等待使用其中的数据（B_WANTED==1）。

  若 PA 先上台运行，B_DONE==1，PA 执行 IOmove 将 4#字节复制进用户空间。解锁缓存（B_BUSY=0，B_WANTED=0），送自由缓存队列队尾；然后 PB 和 PC 串行互斥访问缓存 Buffer[7]中的数据。

若 PB/PC 先上台运行，B_BUSY=1。PB/PC，sleep(& m_buf[7], -50)再次入睡。Buf[7]的 B_WANTED 标识置 1。然后 PA 上台运行。B_DONE=1, PA 执行 IOmove 将 4#字节复制进用户空间。解锁缓存（B_BUSY=0, B_WANTED=0），送自由缓存队列队尾；B_WANTED==1, 唤醒 PB/PC。然后 PB 和 PC 串行互斥访问缓存 Buffer[7]中的数据。

2、题干所有部分不变，PB write 511#字节。问题 2 与问题 1 独立。

不同之处主要在于 PB 和 PC 串行互斥访问缓存 Buffer[7] 中的数据时。

四、一个磁盘组有 100 个柱面，每个柱面有 8 个磁道（磁道号=磁头号），每根磁道 8 个扇区，每个扇区 512 字节。现有含 6400 个记录的文件，每条记录 512 字节。文件从 0 柱面、0 磁道、0 扇区顺序存放。

1、若同根磁道，相邻磁盘数据块存放在相邻物理扇区（现代硬盘，磁盘数据缓存的尺寸：整根磁道）

（1）3680#记录存放的位置。 柱面号=？，磁道号=？，扇区号=？

柱面号 57#，磁道号 5#，扇区号 0#

（2）78#柱面，6#磁道，6#扇区存放该文件的第几个记录？

5046#

2、若同根磁道，相邻磁盘数据块错开一个物理扇区（老式硬盘，磁盘数据缓存的尺寸：一个扇区）

（1）3680#记录存放的位置。 柱面号=？，磁道号=？，扇区号=？

柱面号 57#，磁道号 5#，扇区号 5#

（2）78#柱面，6#磁道，6#扇区存放该文件的第几个记录？

5040#

五、假定磁盘的移动臂现在正处在第 8 柱面，有如下 6 个请求者等待访问磁盘。
假设寻道时间>>旋转延迟。请列出最省时间的响应次序：

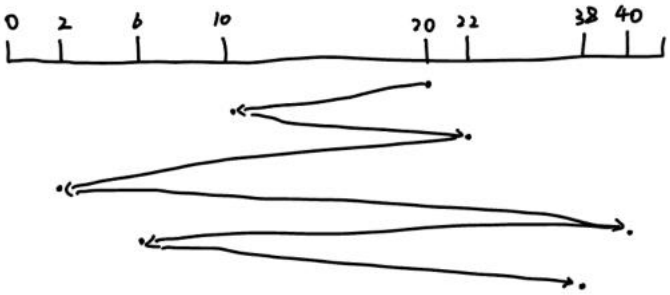
序号	柱面号	磁头号	扇区号
（1）	9	6	3
（2）	7	5	6
（3）	15	20	6
（4）	9	4	4
（5）	20	9	5
（6）	7	15	2

最短寻道优先，（2）（6）（1）（4）（3）（5）

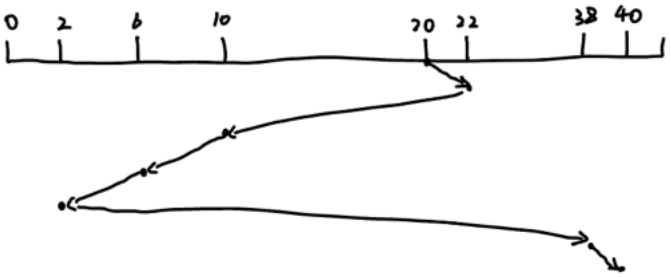
六、当前磁盘读写位于柱面号 20，此时有多个磁盘请求以下列柱面号顺序送至磁盘驱动器：10，22，2，40，6，38。寻道时，移动一个柱面需要 6ms。

1、按下列 3 种算法计算所需寻道时间，画磁头移动轨迹。

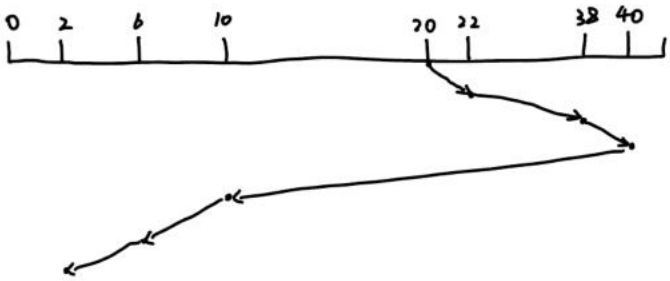
（1）先来先服务



（2）SSTF（下一个最临近柱面）



（3）电梯调度算法（当前状态为向上）



2、为什么电梯调度算法（磁头 向最远请求磁道距离目前磁头位置最近的方向 移动）性能优于另两种算法?

最小化平均响应时间： 电梯算法能够在一个方向上连续地服务磁盘请求，从而减少了寻道的平均时间。相比之下，FCFS 可能导致磁头在磁道上的随机移动，增加了平均响应时间。

减少寻道时间的不确定性：SCAN 算法会沿着一个方向移动，这减少了寻道的不确定性。而 FCFS 和 SSTF 算法可能导致磁头在不同磁道之间频繁切换，增加了磁头移动的不确定性，从而增加了平均寻道时间。

公平性：SCAN 算法具有较好的公平性，因为每个磁道都有机会被访问到，而不会出现像 FCFS 那样可能出现的饥饿情况。

七、外设的独占和共享

1、 从硬件驱动的角度，所有外设必须独占使用。为什么?

外设通常需要访问共享资源，如内存、寄存器等。如果多个应用或进程同时尝试访问同一外设，可能导致资源冲突。

2、多道系统，硬盘是共享设备。并发执行的多个任务可以同时访问硬盘。

内核引入了（ 自由缓存队列 ）填内核数据结构 将必须独占使用的物理硬盘

改造成逻辑上的共享设备。

3、打印机是必需独占使用的外设。并发 2 个打印任务，两个输出内容交织在一起，打印结果是不可用的。Spooling 技术借助硬盘这个共享设备，将原先必须独占使用的打印机改造成能够同时为多个用户提供打印服务的共享设备。思路是：

- 系统维护一个 FIFS 的打印队列。
- 进程需要打印时，
  - 新建一个临时磁盘文件，命名之。把要打印的内容写入这个文件。
  - 生成一个打印作业控制块，包含进程 ID，用户 ID，临时文件名……，送打印队列尾。
  - 进程返回。无需等待打印 IO 完成。
- 打印机完成当前打印任务后，取打印队列队首打印作业控制块，构造针对打印机的新的 IO 命令。

PS1：相对打印机，磁盘是很快的设备。所以，Spooling 技术同时也改善了打印机这个 IO 子系统的响应速度。Spooling 是个很不错的 IO 优化技术，对照期末自己在打印店看到的现象，体会下 Spooling 技术。

PS2：教科书上 Spooling 技术相关的，有输入井和输出井这两个概念。打印机的输出井就是一个文件夹，用来存放临时文件（上面高亮的那个）。

八、证明题 选做（1）io 请求集合固定时，第一步磁头向 最远端请求磁道距离磁头当前所在磁道距离最远的方向移动。这个版本的电梯算法理论最优（2）SSTF 不是最优。

（1）假设磁头当前所在磁道为 i，最左端请求磁道 l，最右端请求磁道为 r，且 r-i>i-l，即磁头先向右端移动最优。首先磁头要么向左一直移动到 l，然后向右一直移动到 r；要么向右一直移动到 r，然后向左一直移动到 l。不可能一下向左一下向右，因为这样 i 附近的磁道会 被重复扫描很多次。所以只要讨论为什么先向右比先向左更优。

（2）饥饿问题：如果有一个持续产生请求的源点，在磁头附近有大量请求时，可能会导致远离磁头的请求一直得不到服务，因为每次都有更近的请求需要被处理。