设备管理习题 参考答案

姓名　　学号

第一部分　写操作

一、概念题

1、同步 IO

进程入睡等待 IO 完成，被内核唤醒。同步 IO 操作期间，进程阻塞，不能执行其它任务。

2、异步 IO

进程在发起 IO 请求后继续执行，而非睡眠等待。进程使用异步方式，可以启动多个并行的 IO 操作，可以在等待 IO 传输过程中执行计算任务。异步 IO，是高性能计算利器，但需要进程（应用程序）主动查询 IO 操作状态。

3、为什么读是同步的，写是异步的

为什么读操作是同步的？因为进程要处理从磁盘读入的文件数据。

为什么写操作是异步的？因为写磁盘最终是 DMA 和磁盘控制器硬件做的，它们将进程产生的新数据持久化存储在磁盘上。进程后续无需，也不能再处理这块数据了。

4、磁盘数据块为什么要先读后写

磁盘写操作以数据块为单位，会用内存中的整块数据覆盖磁盘数据块。所以，正确的操作步骤是（1）先读，确保缓存块中包含磁盘数据块中的旧数据（2）后写，将新数据存入缓存块（3）硬件（DMA 控制器和磁盘控制器）将既包含新数据，又包含旧数据的缓存块写回磁盘。如果没有先读操作，内存中的错误信息会覆盖磁盘数据块中的部分数据，导致文件内容出错。

5、延迟写操作的优点和不足

优点：合并多个写操作，减少写 IO 数量。缩短写操作的平均响应时间，减少磁盘磨损。

确定：引入文件系统不一致性，增加了丢失数据的风险。

6、Unix 系统何时将脏缓存写回磁盘

（1）数据块写至尾部

（2）LRU 缓存是脏缓存，分配用来装其它数据块之前

（3）关机时所有脏缓存写回磁盘

（4）磁盘卸载时，属于这张磁盘的所有脏缓存写回磁盘

（5）系统定期执行一次 sync 操作，将所有 SuperBlock，脏的 DiskInode 和脏缓存块写回磁盘。Unix V6++，30s。

脏缓存指数据块已修改但未写回磁盘的缓存块。

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

- IO：0 次。
- 入睡：数据块空闲，不睡。数据块，其它进程在用，会睡。

2、读磁盘数据块，缓存不命中

- IO：1 次，读入目标数据块。此外，缓存分配会产生脏缓存刷回磁盘的异步写 IO。
- 入睡：1 次，等待磁盘读操作结束。此外，自由缓存队列空或没有干净的缓存块时，进程会因为需要为数据块分配缓存块而入睡。

3、写磁盘数据块，缓存命中

- 写 IO：写至缓存底部，1 次（立即）。未写至缓存底部，0 次（延迟）。
- 入睡：数据块空闲，不睡。数据块，其它进程在用，会睡。但，不会因为写操作入睡。

4、写磁盘数据块，缓存不命中

- 读 IO：需要先读，1 次。此外，缓存分配会产生脏缓存刷回磁盘的异步写 IO。
- 写 IO：写至缓存底部，1 次（立即）。未写至缓存底部，0 次（延迟）。
- 入睡：1 次，等待磁盘读操作结束。此外，自由缓存队列空或没有干净的缓存块时，进程会因为需要为数据块分配缓存块而入睡。

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，PC 的调度状态 和 Buffer[i]的使用状态。

- PA 执行 read 系统调用
  - 调度状态。PA 为 55#磁盘数据块申请得到一块缓存，发起读 IO 请求后，sleep(&m_buf[7], -50)，入睡等待 IO 完成、B_DONE 变为 1。
  - Buffer[7]的使用状态：分配给 55#磁盘数据块，在 IO 请求队列。B_BUSY 为 1，B_DONE 为 0。
- PB 执行 write 系统调用
  - 调度状态。PB 等待复用 55#数据块，sleep(&m_buf[7],-50)，入睡等待缓存块解锁、B_BUSY 变为 0。
  - Buffer[7]的使用状态：分配给 55#磁盘数据块，在 IO 请求队列。B_BUSY 为 1，B_DONE 为 0，B_WANTED 为 1。
- PC 执行 read 系统调用
  - 同 PB。
- 55#扇区 IO 完成

调度：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。

Buffer[7] 的使用状态：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）。

调度：PA 先用缓存块中的数据，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，唤醒 PB 和 PC。PB 和 PC 依次上台，分别执行 write 系统调用和 read 系统调用的下半段，写、读缓存块中偏移量依次为 200 和 500 的字节。谁先执行，不得而知，对最终结果没有影响。

Buffer[7] 的最终状态：B_DELWR==1，B_DONE==1，B_BUSY==0，是一块自由的脏缓存。

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

和第一小题一致。唯一的不同在于，本小题 PB 写至数据块底部，55#数据块需要异步写回磁盘。若其先于 PC 执行，PC 会再次入睡，等待中断处理程序解锁 55#数据块。

Buffer[7] 的最终状态：B_DELWR==0，B_DONE==1，B_BUSY==0，是一块自由的不脏的缓存。

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=？

（2）78#柱面，6#磁道，6#扇区存放该文件的第几个记录？

【参考答案】

柱面，从最外圈向内递增编号。从 0 开始，先编号 n 号柱面的所有磁道，后编号 n+1 号柱面的所有磁道。属于同一柱面的多个磁道，按磁头号递增对磁道依次编号。一根磁道，所有扇区递增编号，之后编号下一个扇区。

（1）每个柱面 8*8 = 64 个扇区。

3680/64 = 57　　柱面号是 57。

3680 % 64 = 32　　这个记录存放在 57#柱面的 32#扇区。

每根磁道 8 个扇区。

32 / 8 = 4　　32#扇区在 4#磁道（4#读写头管理的那个磁道）

32 % 8 = 0　　是 4#磁道的 0#扇区。

3680#记录存放的位置是：　57#柱面，4#磁道，0#扇区。

（2）78*64+6*8+6 = 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=？

（2）78#柱面，6#磁道，6#扇区存放该文件的第几个记录？

【参考答案】

（1）3680#记录存放的位置是：　57#柱面，4#磁道，0#扇区。

（2）78*64+6*8+3 = 5043

二、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

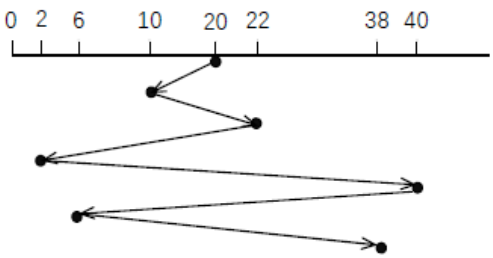
【参考答案】优先考虑缩短寻道时间。优化后的柱面访问次序：7，9，15，20

所以最省时间的相应次序应该是：（2，6），（1，4），3，5。括号内请求，次序可以互换。

三、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。

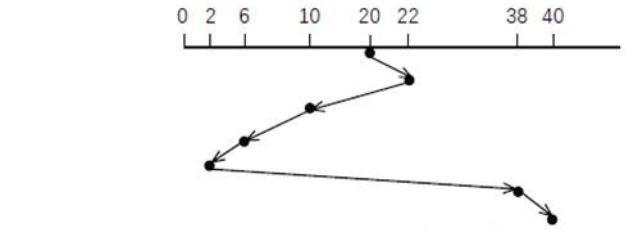
（1）先来先服务



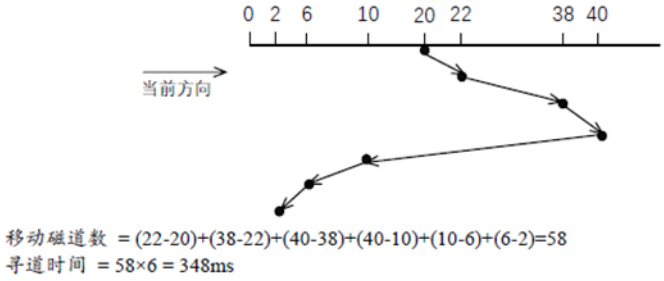
移动磁道数 =(20-10)+(22-10)+(22-2)+(40-2)+(40-6)+(38-6)=146

寻道时间 =146×6 = 876ms

（2）SSTF（下一个最邻近柱面）



（3）电梯调度算法（当前状态为向上）



2、为什么电梯调度算法性能优于另两种算法？**答：磁臂移动轨迹，折返距离短。**

四、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？

【参考答案】外设芯片中的数据缓存，命令、状态、数据寄存器只能存放一个请求的 IO 参数和 IO 数据。另外，对外设而言（以读操作为例），一次典型的 IO 操作包括：CPU 发 IO 命令，外设执行 IO 命令，数据寄存器中的新数据读入内存。全部完成后，CPU 才可以向外设发下一个 IO 命令。综上，从硬件驱动的角度，所有外设必须独占使用，没有例外。

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（ 磁盘高速缓存，IO 请求队列 ）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。是个很不错的 IO 优化技术呐。对照期末自己在学子打印资料看到的现象，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）。

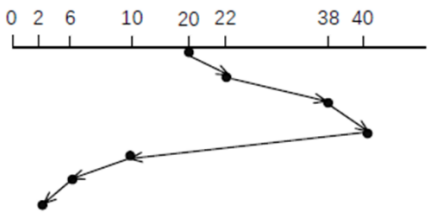
八、证明题 选做（1）io 请求集合固定时，第一步磁头向 最远端请求磁道距离磁头当前所在磁道距离**最近**的方向移动。这个版本的电梯算法理论最优（2）SSTF 不是最优。

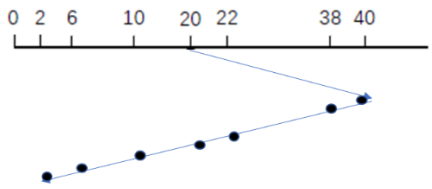
（1）
【参考答案】断言成立。
给定 IO 请求集合，所有 IO 请求∈[min，max]。min 和 max 是最小磁道号和最大磁道号。磁头向距离当前位置最近的端点移动，电梯调度算法给出的解理论最优。

证明一：错误的证明
第一种情况：若当前磁头 curr 不在集合[min，max]，电梯调度算法服务全部 IO 请求，只需磁头从当前位置移动至最远磁道：
磁臂移动距离 = （max – min）+ curr 到距离最近的磁道的距离。
具体而言，这个值是：

- max – curr，若 curr<=min
- curr - min，若 curr>=max

第二种情况：若当前磁头 curr∈[min，max]，向距离近的一端移动，之后折返向另一端移动。考虑磁臂移动距离，下面两张图是等价的。





磁臂移动距离 = （max – min）+ curr 到距离最近的磁道的距离。

总结：电梯调度算法磁臂移动总距离由两部分组成：

- 服务全部 IO 请求，磁臂移动距离：max-min。磁臂没有折返，这部分值不可优化。理由是只要重排请求次序，磁臂就会有折返，移动总距离会增加。
- curr 到距离最近的磁道的距离。这部分值也不可优化。理由是，磁头从当前位置移动到任何请求所在磁道，距离不会小于这个值。错在：这只是磁头移动的第一步。第一步移动距离最短，并不能保证完成整个请求序列磁头总移动距离最短。

所以，若 IO 请求集合恒定，电梯调度算法给出的是最优解。

证明二：磁道 min 和 max 处有 IO 请求。无论是怎样的响应序列，磁头一定要经过这两根磁道。

第一步，仅考虑 curr、min、max 这 3 根磁道。从 curr 出发，经过 min 和 max，磁头最短移动距离 =

distance[min , max] + min(distance[min , curr], distance[curr , max])

磁头移动轨迹：curr，离 curr 较近的一端 a，离 curr 较远的一端 b。

磁头在从 a 移动至 b 的过程中，服务全部 IO 请求。

这是电梯调度算法给出的解，命名为 bestSequence。

第二步：任选 IO 请求序列中的一个 IO 请求 x，插入 bestSequence。得到的响应序列长度不小于第一步产生的响应序列。证明如下：

x 的插入位置有 3 处：

- a 之前　　curr　　x　　a　　b

  curr，a 两点间直线距离最短：

  distance[curr，a]<= distance[curr，x]+ distance[x, y]。

  途径 x 到 a，轨迹长度增加 distance[x , a]。
- a ，b 之间　　curr　　a　　x　　b

  轨迹长度不变。
- b 之后　　curr　　a　　b　　x

  轨迹长度增加 distance[x , b]。

第三步：任选 IO 请求序列中的一个 IO 请求 y，插入从第二步衍生出的任意长度响应序列。得到的响应序列运动轨迹长度不小于已有响应序列。证明方法与第二步相同。

依次实施第一步、第二步，之后递归实施第三步可以构造出任意 IO 请求响应序列。由上述证明可知，生成的 IO 请求响应序列磁头运动轨迹长度不会小于第一步给出的 bestSequence。电梯调度算法给出的解理论最优。

证明有点长，应该还会有简洁的、更好的证明。

（2）SSTF 不是最优。

这个还好，找一个反例。第 6 题就是。