设备管理例题一

同济大学计算机系操作系统　　邓蓉

例题 1、

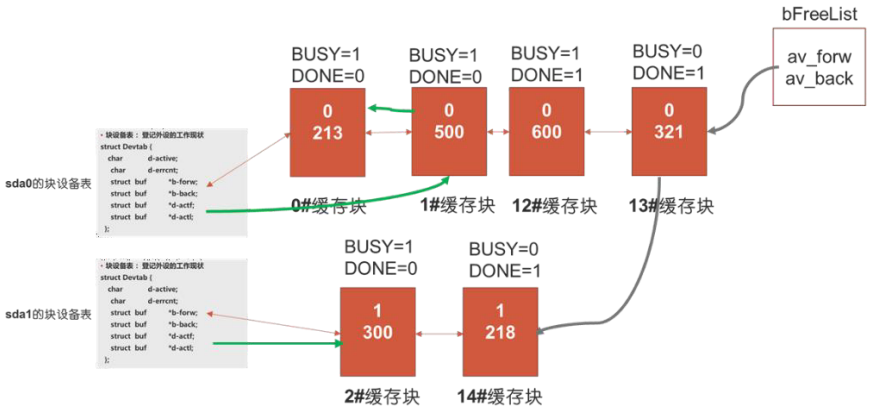

图 1、缓存队列。灰色，自由缓存队列；红色，设备缓存队列；绿色，IO 请求队列

sda0，0#硬盘，major为0，minor为0，设备号是0

sda1，1#硬盘，major为0，minor为1，设备号是1

图中的标记：0#缓存块分配给sda0，213#扇区

【参考答案】

1、写出系统的自由缓存队列，磁盘sda0，sda1 的设备缓存队列和 IO 请求队列

- 自由缓存队列：<0,321>➔<1,218>
- sda0 的设备缓存队列：<0,213> ➔ <0,500> ➔ <0,600> ➔ <0,321>
- sda0 的 IO 请求队列：<0,500> ➔ <0,213>
- sda1 的设备缓存队列：<1,300>➔<1,218>
- sda1 的 IO 请求队列：

<1,300> 2、sda0 正在进行同

步读操作:

- 内存地址：Buffer[1]
- 扇区：500
- IO 完成时，磁盘中断处理程序
  - 会启动新的 IO 请求吗？会，IO 213#扇区
  - 会唤醒进程吗？会，因为读操作是同步的。

3、sda1 正在进行异步写操作:

- 内存地址：Buffer[2]
- 扇区：300
- IO 完成时，磁盘中断处理程序
  - 会启动新的 IO 请求吗？一定不会，执行完此 IO 后 IO 请求队列为空
  - 会唤醒进程吗？一定不会。没有进程睡眠等待异步IO完成

缓存块可能同时出现在2个队列，所以缓存控制块中有4根链表指针。

例题 2：在 UNIX V6++中，试说明缓存控制块 Buf 有无可能，在什么样的条件下出现下列情况：

（1） 同时处在自由 Buf 和一个设备 Buf 队列中；可能, Brelse 释放的自由缓存块

（2）同时处在某一设备 Buf 队列和 I/O 请求队列中；可能, IO 操作尚未完成的所有缓存块都是这样的

（3）只处在某一设备Buf 队列中；可能。IO 完成（B_DONE 是 1）但 IOmove 还没有完成的缓存块。

（4）只处在I/O 请求队列中，不在设备缓存队列里。可能。负责传送进程图像的 SwBuf。

（5）同时出现在自由 Buf 队列和某个设备的 I/O 请求队列中 不可能, 自由缓存不可能在 IO 请求队列，反之亦然。

（6）同时出现在一类设备的 Buf 队列、另一类设备的 I/O 请求队列中；不可能, 一个缓存块不可能分配给 2 个设备。

例题 3、T0 时刻，PA、PB 进程先后访问文件A，PA read 偏移量为4的字节，PB write 偏移量为4的字节。已知文件 A 的 0#逻辑块（文件的 0#~511#字节） 存放在 666#扇区。T0 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。
  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（666#扇区）。刷新 m_buf[7]：dev=0, blkno=666, 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。

```
while( (bp->b_flags & Buf::B_DONE) == 0 )
    u.u_procp->Sleep((unsigned long)bp, -50);
```

  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
  缓存命中，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。

```
while( (bp->b_flags & Buf::B_BUSY) == 0 ) {
    bp->b_flags |= Buf::B_WANTED
    u.u_procp->Sleep((unsigned long)bp, -50); }
```

  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）。
- 666#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 和PB。PA、PB 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。系统先后发生以下行为。
  - 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，唤醒PB;
  - PB 上台运行。B_BUSY==0，PB 锁住缓存，先读后写，将新数据写入 Buffer[7] 4#字节后，置延迟写标记（脏标记）B_DELWR。解锁缓存，标记缓存块中数据可用（是最新版本）B_DONE=1，送自由缓存队列队尾。（没 IO 的）
- 若 PB 先上台运行。系统先后发生以下行为。
  - B_BUSY==1。PB，sleep(& m_buf[7], -50)再次入睡。置 1 Buf[7]的 B_WANTED 标识。
    - Buffer[7] 状态：上面绿的。
  - 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，唤醒PB。
  - 上面紫的。

- PA 进程能够读到PB 进程写入的新数据吗?
  不能。因为，IO 完成时，一定是 PA 先使用缓存块中的数据。

例题 4：进程读磁盘文件，存在不入睡的可能吗？何时？ 存在。需要读入的数据命中自由缓存。
另外，缓存命中，进程也会睡的。。。如果其它进程正在使用缓存池中的数据块。

例题 5：时刻 T1，数据块 556、600、782、891、900 缓存不命中。已知：将IO请求放入IO请求队列后至IO操作完成，进程平均需要等待 T。从缓存复制数据到用户空间，平均耗时 t。T>>t 。请比较使用缓存池的IO 和 未使用缓存池的IO，系统的性能。指标1，IO请求的平均耗时；指标2，完成整个IO请求序列系统的总耗时。

1、读操作序列：556，556，556，600，782，891，900，556，556，556
不使用缓存池 数据从磁盘直接复制到用户空间，IO请求平均耗时 T。完成所有读操作，IO 10次，系统的总耗时 10T 。
使用缓存池 数据从磁盘复制到缓存。之后进程从缓存中取用数据，复制到自己的用户空间。完成整个序列，IO 5次。总耗时，5T+10t。IO请求平均耗时 T/2+t 。
磁盘高速缓存的作用 重用IO数据。缩短读操作平均耗时。减少 IO 请求次数，有利于缩短 IO 操作的平均耗时，提高IO子系统的吞吐率。

例题6、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。

缺点：引入文件系统不一致性，增加了丢失数据的风险。

6、什么样的缓存是脏缓存，Unix V6++系统何时将脏缓存写回磁盘

脏缓存存放的数据块已被修改，但未写回磁盘。Unix V6++系统将脏缓存写回磁盘的时机包括：

（1） 数据块写至尾部

（2） 脏缓存成为LRU缓存，分配它用之前

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

例题7、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。