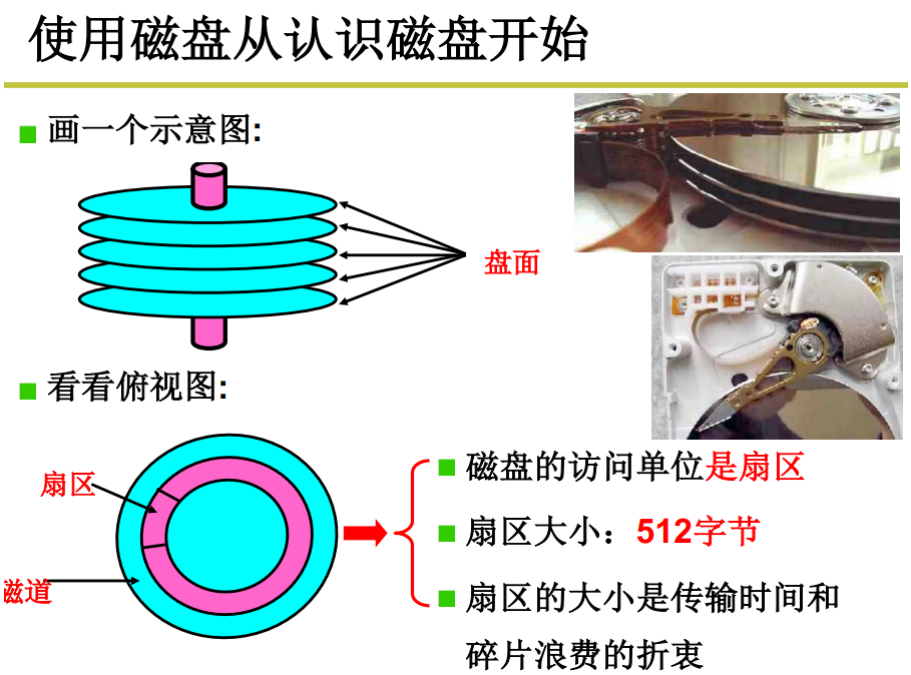
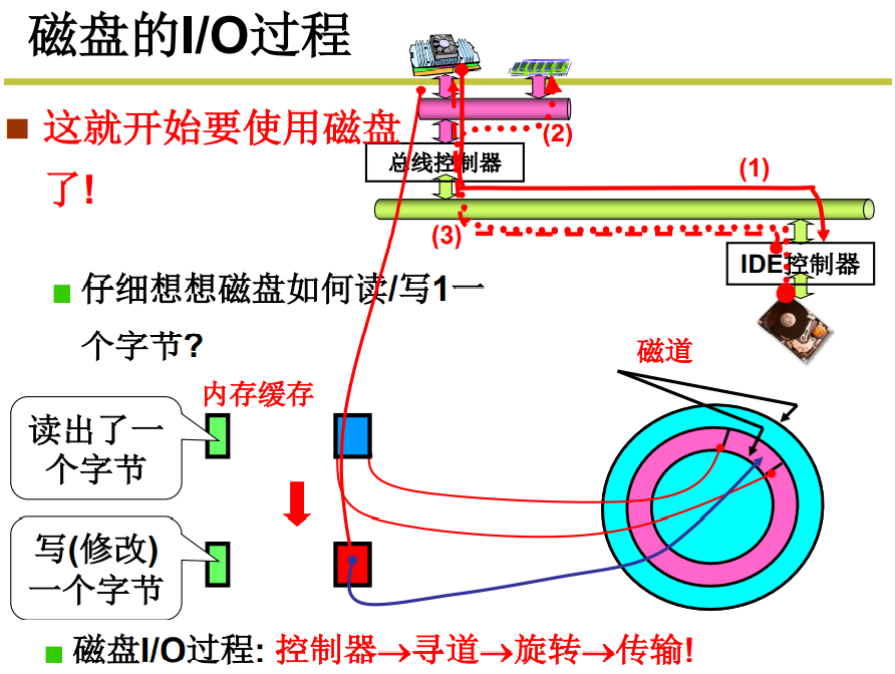
哈工大操作系统-L28生磁盘的使用

1.磁盘常识

1.1磁盘结构与读写流程

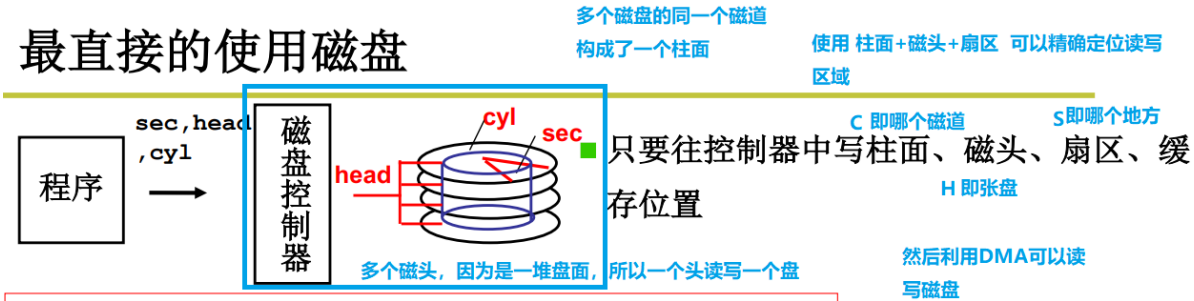




- 磁盘访问单位是扇区
- 一个扇区是512B
- 磁盘读写的三部曲：移动、旋转、读写
  - 移动磁头到磁道
  - 旋转磁盘使得磁头在相应扇区
  - 读写

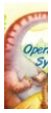
1.2最直接的使用磁盘的方式

只要往控制器中写柱面、磁头、扇区、缓存位置，就可以读写磁盘了。




```
void do_hd_request(void)
{   ...hd_out(dev,nsect,sec,head,cyl,WIN_WRITE,...);
   port_write(HD_DATA,CURRENT->buffer,256);}
```

```
void hd_out(drive, nsect, sec, head, cyl, cmd...)
{  port = HD_DATA; //数据寄存器端口(0x1f0)
   outb_p(nsect,++port); outb_p(sect,++port);
   outb_p(cyl,++port); outb_port(cyl>>8,++port);
   outb_p(0xA0|(drive<<4)|head, ++port);
   outb_p(cmd, ++port); }
```

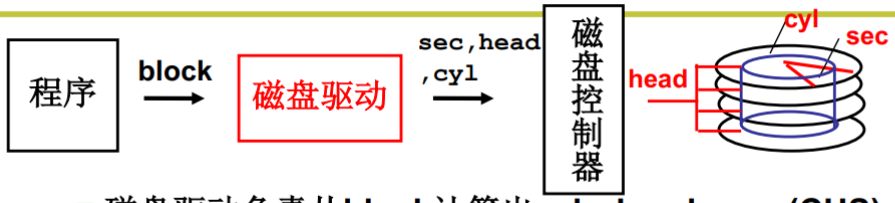


2.通过盘块号读写磁盘(一层抽象)

- 系统负责将盘块号计算出CHS的三维编号。
- 通过最小化磁盘访问时间来进行编址。
- 经过分析，多个扇区组成一个盘块，**以盘块为单位进行读写会使得读写速度提升**

- 因此，一层抽象后，程序以盘块为单位进行磁盘的读写，操作系统负责将盘块号翻译成CHS再发给控制器
- 而盘块为单位进行读写是能提升读写速度的
- 这是**以空间效率来换取时间效率**

通过盘块号读写磁盘(一层抽象)



- 磁盘驱动负责从block计算出cyl，head，sec(CHS)

问题：如何编址?为什么这样编址？ block相邻的盘块可以快速读出

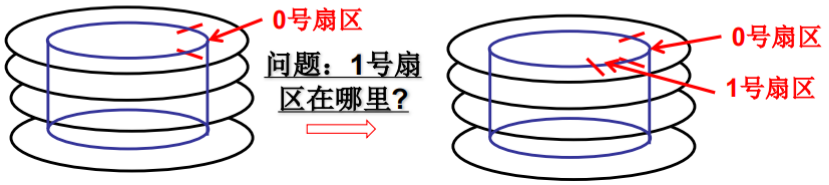
磁盘访问时间 = 写入控制器时间 + 寻道时间 + 旋转时间 + 传输时间

50M/秒,约0.3ms

12 ms to 8 ms

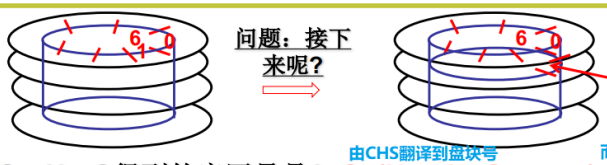
7200转/分钟:半周 4 ms

磁盘访问的时间主要花在了寻道，其次是旋转



因此相邻盘块号的块应该放在相同的磁道

从CHS到扇区号，从扇区到盘块



问题：C、H、S得到的扇区号是？ C×(Heads×Sectors) + H×Sectors + S

由CHS翻译到盘块号

而由盘块号%磁道的扇区数目=扇区号，然后可以推导C、H

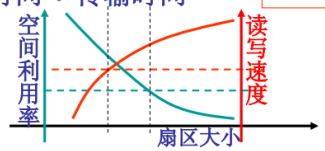
Heads是磁头数目

Sectors是一个磁道的扇区数目

磁盘访问时间 = 写入控制器时间 + 寻道时间 + 旋转时间 + 传输时间

共计约10ms

扇区大小固定，但操作系统可以每次读/写连续的几个扇区(盘块)



- 从扇区到盘块: 每次读写1 K：碎片0.5K；读写速度100K/秒；每次读写1 M：碎片0.5M；读写速度约40M/秒

再接着使用磁盘：程序输出block

```
static void make_request()
{   struct requset *req;
    req=request+NR_REQUEST;
    req->sector=bh->b_blocknr<<1;
    add_request(major+blk_dev,req); }
```

一个盘块2扇区

问题：Linux 0.11盘块多大?

block = C×(Heads×Sectors) + H×Sectors + S ➔ S = block%Sectors

```
void do_hd_request(void)
{ unsigned int block=CURRENT->sector;
  __asm__("divl %4":"=a"(block),"=d"(sec):"0"(block),
       "1"(0),"r"(hd_info[dev].sect));
  __asm__("divl %4":"=a"(cyl),"=d"(head):"0"(block),
           "1"(0),"r"(hd_info[dev].head));
  hd_out(dev,nsect,sec,head,cyl,WIN_WRITE,...);
  ... }
```

hd out 为使用磁盘

3.多进程通过队列使用磁盘(第二次抽象)

多个进程通过队列使用磁盘(第二层抽象)

- 多个磁盘访问请求出现在请求队列怎么办?
- 调度的目标是什么? 调度时主要考察什么?

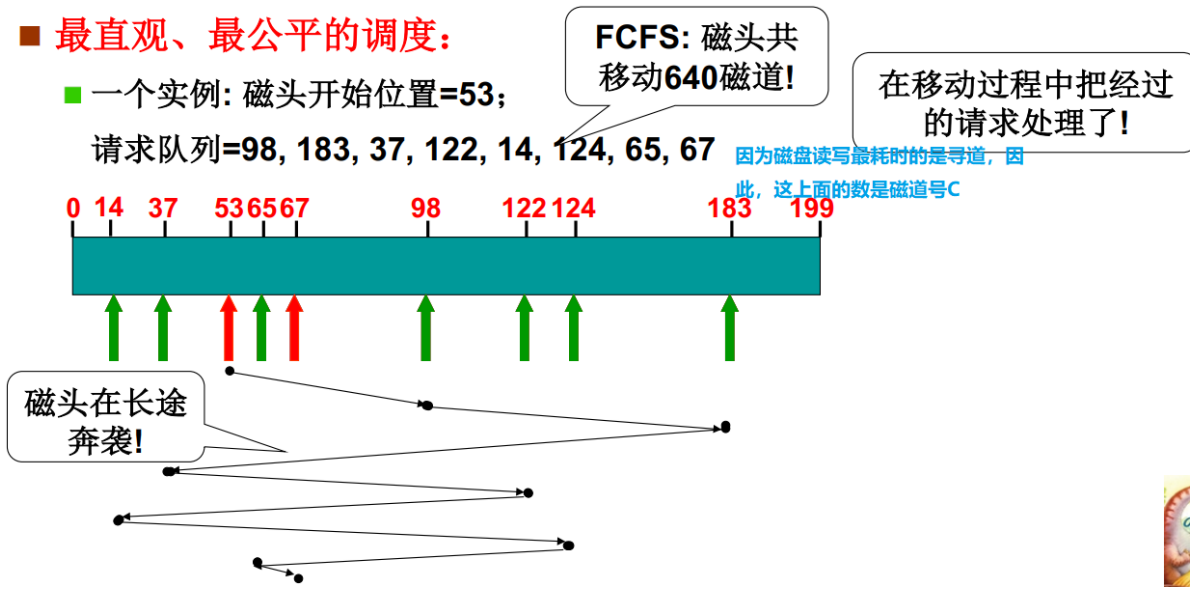
目标当然是平均访问延迟小!

寻道时间是主要矛盾!

- 给调度算法，仍然从FCFS开始...

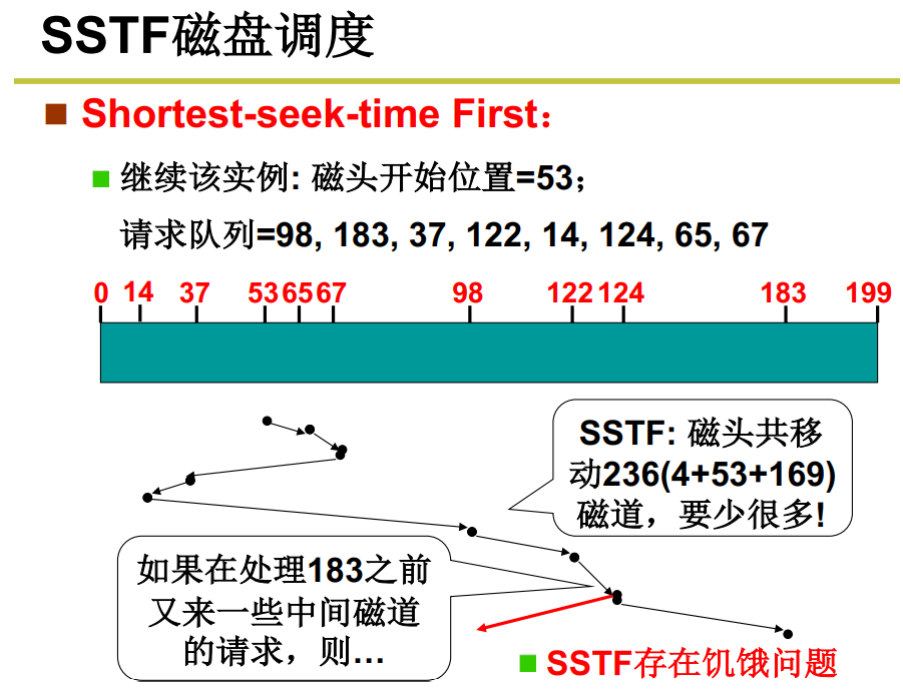
3.1磁盘调度--FCFS先来先服务

FCFS磁盘调度算法



- FCFS会使得**磁道移动的耗时增加**
- 我们分析发现，在长途移动的过程中已经访问了后面需要访问的磁道，因此为何不顺便处理了呢，由此生出SSTF

3.2磁盘调度--SSTF最短寻道时间优先



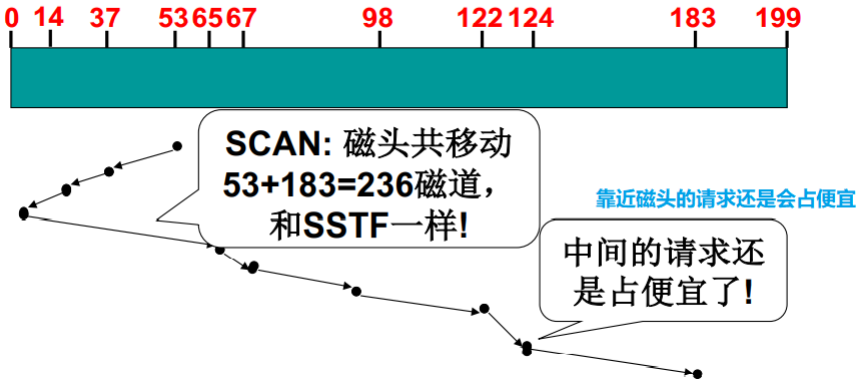
- 总寻道时间减少
- 但是会出现一个问题，即磁头附件的磁道访问的几率比远离磁头的大，会造成磁盘圆心和圆周的磁道访问次数少，存**在饥饿**。

3.3磁盘调度--SCAN扫描调度算法

SCAN磁盘调度

- **SSTF+中途不回折：每个请求都有处理机会**
  - 继续该实例: 磁头开始位置=53；

    请求队列=98, 183, 37, 122, 14, 124, 65, 67

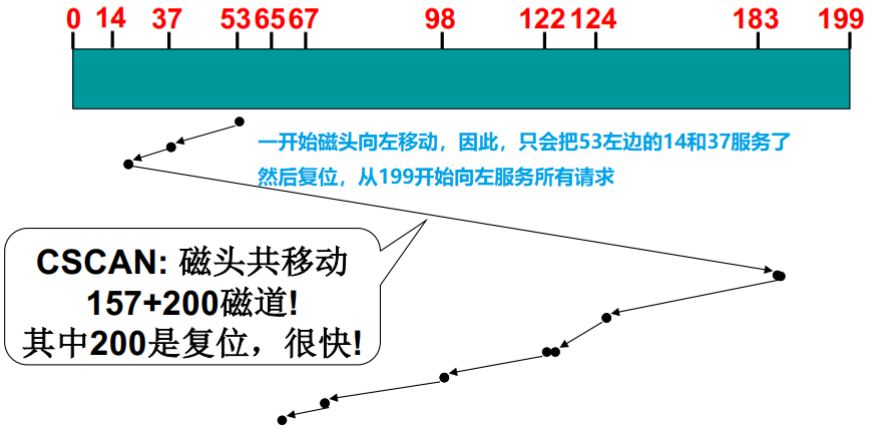


3.4磁盘调度--C-SCAN(电梯算法，真实运用)

C-SCAN磁盘调度(电梯算法)

- **SCAN+直接移到另一端：两端请求都能很快处理**
  - 继续该实例: 磁头开始位置=53；

    请求队列=98, 183, 37, 122, 14, 124, 65, 67



- 像电梯一样，磁盘从起点只往一个方向，把一个方向的请求都处理掉。
- **如果移动过程中出现移动方向相反的请求(无论这个请求离磁头近还是远)，先不处理**
- **等到磁头移动到另一端时，复位**，再从起点移动向另一个方向，处理路上遇到的请求。周而复始。

多个进程共同使用磁盘

```
static void make_request()
{   ...req->sector=bh->b_blocknr<<1;   算出扇区号
    add_request(major+blk_dev,req); }  加入请求队列中
```

```
static void add_request(struct blk_dev_struct *dev,
struct request *req)
{ struct requset *tmp=dev->current_request;
  req->next=NULL; cli();  //关中断(互斥)  放在队列中时要进行临
  for(;tmp->next;tmp=tmp->next)            界区保护
   if((IN_ORDER(tmp,req)||!IN_ORDER(tmp,tmp->next))
       &&IN_ORDER(req,tmp->next)) break;     应该是一个循环队列
  req->next=tmp->next; tmp->next=req; sti();}
```

req->next就是我们要插入的位置

for循环扫描队列；理论上队头较小；tmp是当前的操作向量；

sector = C×(Heads×Sectors) + H×Sectors + S

因此插入的位置应该是大于tmp小于tmp->next的，插入到tmp后面

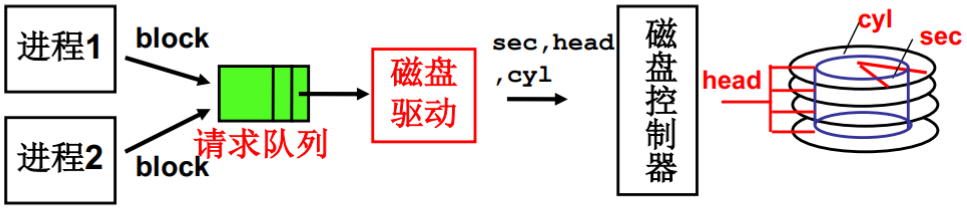
又或者遇到了首尾相接处，即tmp大于tmp->next，

那么我们也还是直接插入tmp后面成为队尾

```
#define IN_ORDER(s1, s2) \
  ((s1)->dev<(s2)->dev)||((s1)->dev == (s2)->dev\
     && (s1)->sector<(s2)->sector))
```

4.总结

生磁盘(raw disk)的使用整理



- (1) 进程“得到盘块号”，算出扇区号(sector)
- (2) 用扇区号make req，用电梯算法add_request
- (3) 进程sleep_on
- (4) 磁盘中断处理

```
static void read_intr(void)
{   end_request(1);
    do_hd_request(); }
```

唤醒进程!

- (5) do_hd_request算出cyl,head,sector
- (6) hd_out调用outp(…)完成端口写

- 下节课讲进程如何得到盘块号。实际上是通过文件的。
- 课外可以自学系统是如何使用缓冲区访问磁盘的。