操作系统

实　　验　　报　　告

课程名称	操作系统实验	课程编号	0906553
实验项目名称	磁盘调度算法		
学号		年级	
姓名		专业	计算机科学与技术
学生所在学院	计算机科学与技术学院	指导教师	
实验室名称地点			

哈尔滨工程大学

计算机科学与技术学院

磁盘调度算法

一．实验概述：

1.实验名称：磁盘调度算法

2.实验目的：

1）通过学习 EOS 实现磁盘调度算法的机制，掌握磁盘调度算法执行的条件和时机；

2）观察 EOS 实现的 FCFS、SSTF 和 SCAN 磁盘调度算法，了解常用的磁盘调度算法；

3）编写 CSCAN 和 N-Step-SCAN 磁盘调度算法，加深对各种扫描算法的理解。

3.实验类型：验证、设计

4.实验内容：

1）准备实验，创建一个 EOS Kernel 项目；

2）验证先来先服务（FCFS）磁盘调度算法；

3）验证最短寻道时间优先（SSTF）磁盘调度算法；

4）验证 SSTF 算法造成的线程“饥饿现象”；

5）验证扫描（SCAN）磁盘调度算法；

6）改写 SCAN 算法；

7）编写循环扫描（CSCAN）磁盘调度算法；

8）验证 SSTF、SCAN 及 CSCAN 算法中的“磁臂粘着”现象；

9）编写 N-Step-SCAN 磁盘调度算法。

二．实验环境

操作系统：windows XP

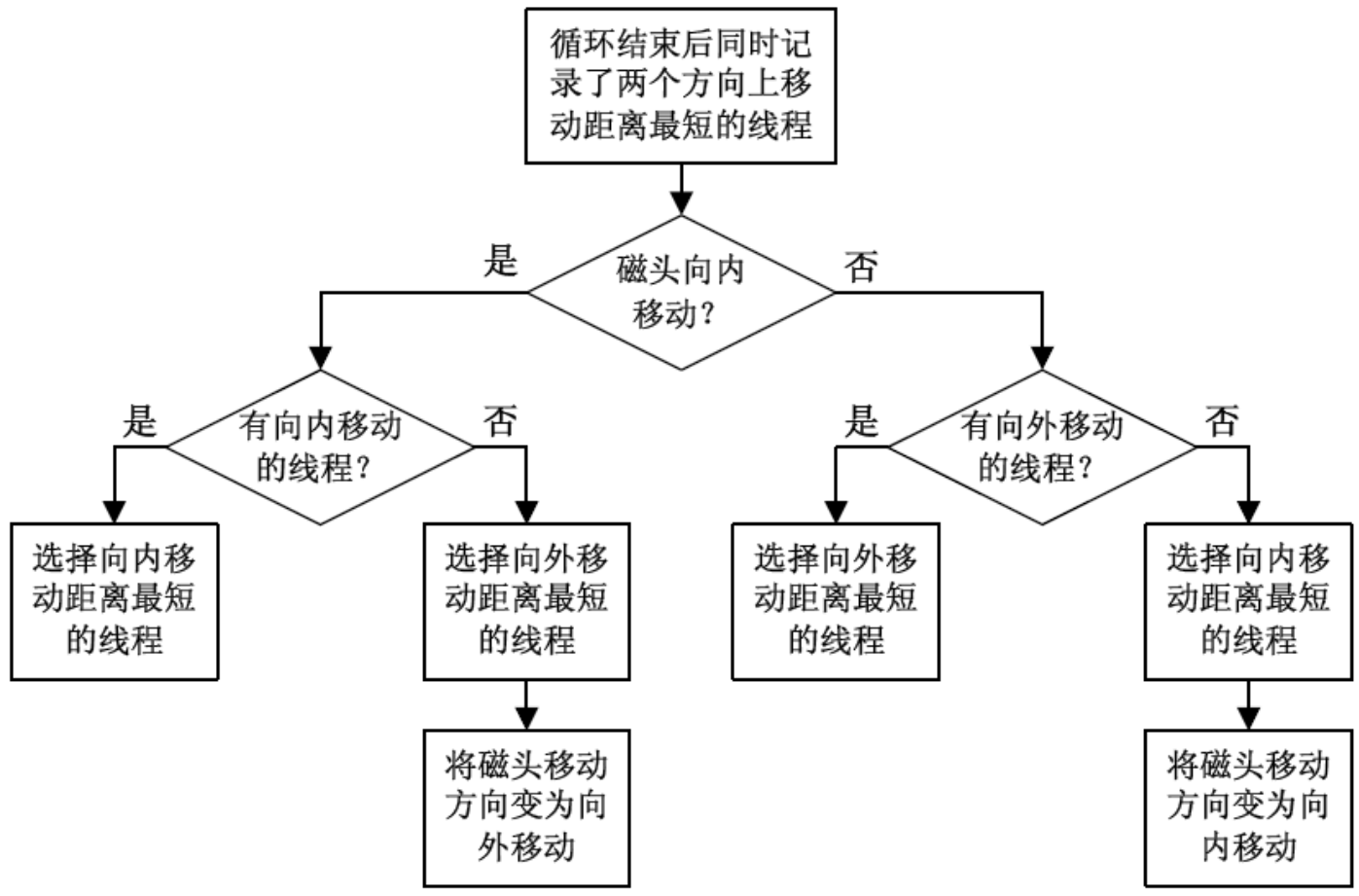
编译器：Tevalaton OS Lab

语言：C

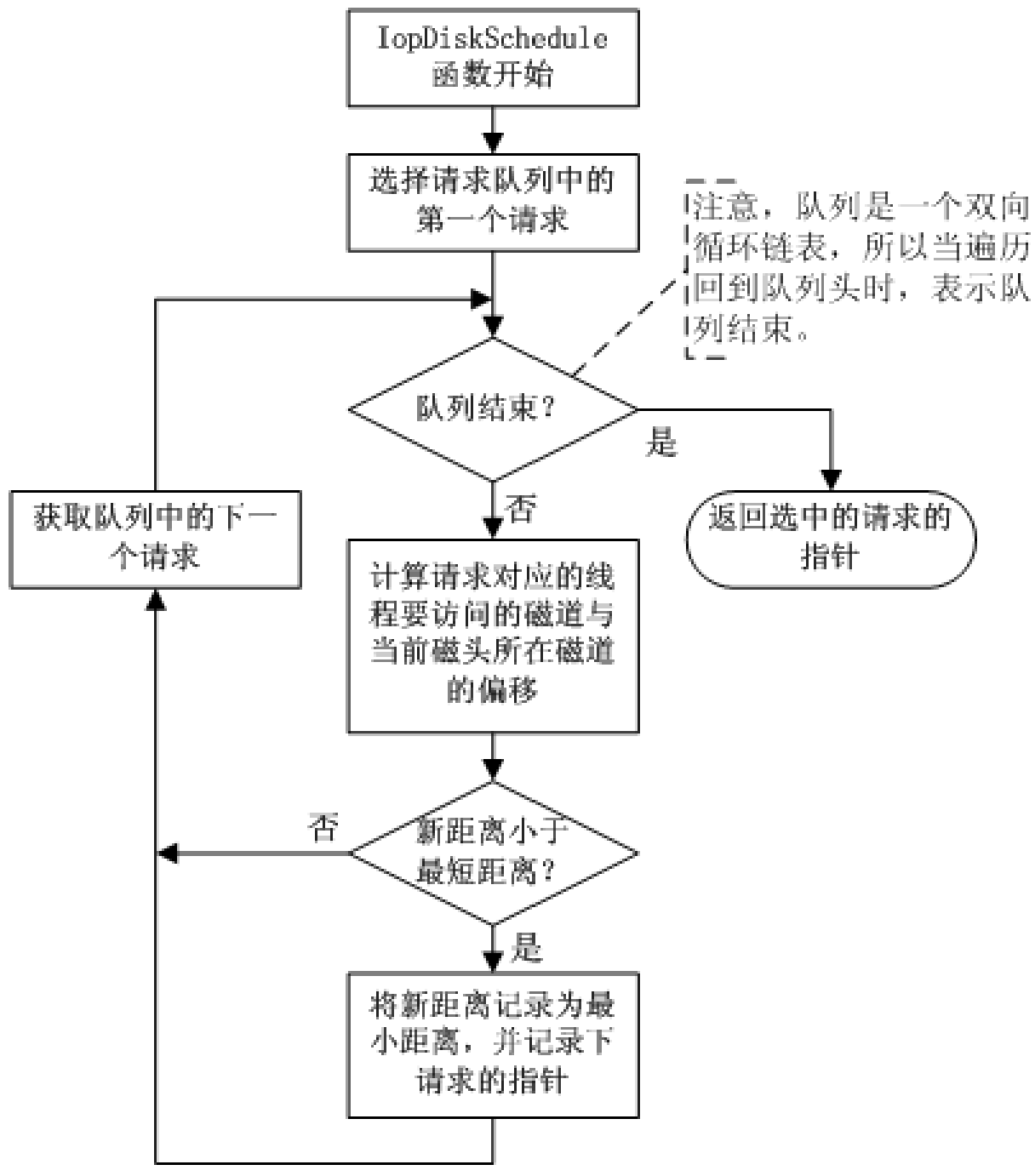
三．实验过程

1.设计思路和流程图：

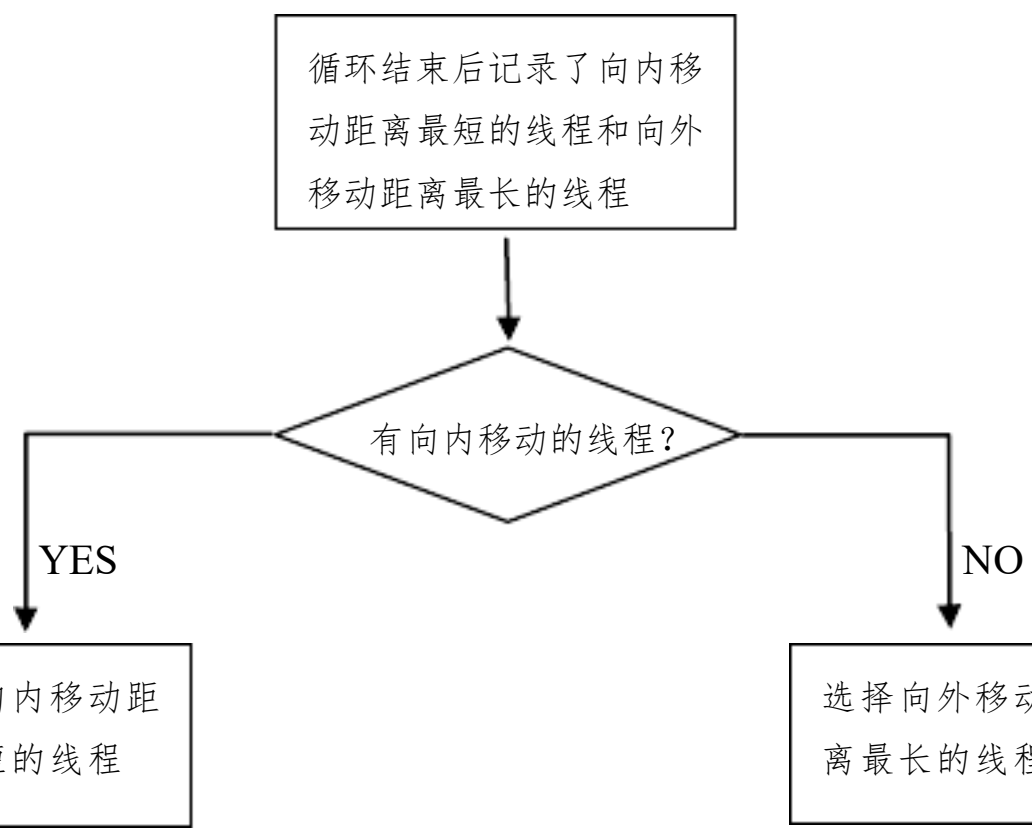
SCAN 算法流程图：



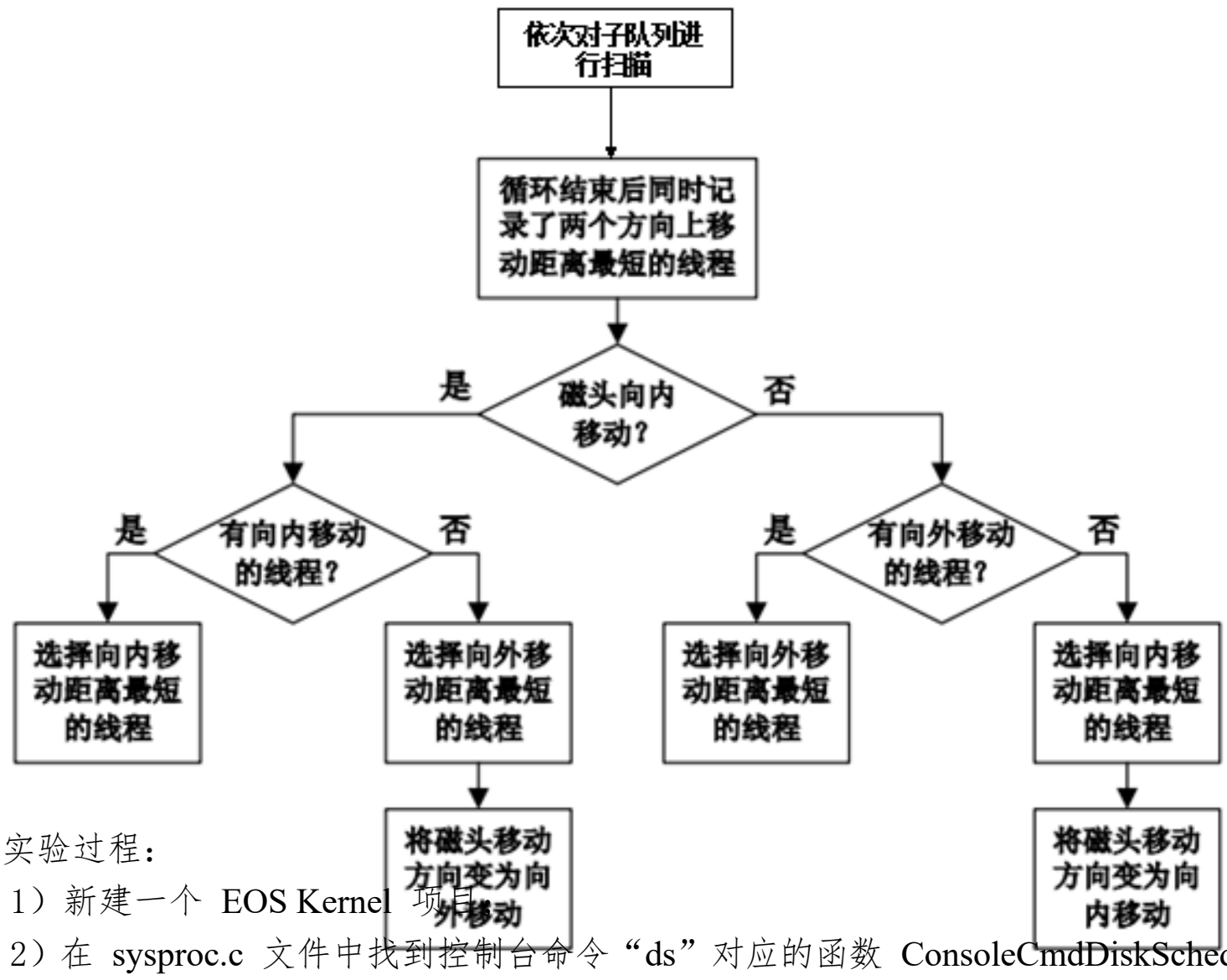
SSTF 算法的流程图：



CSACN 流程图：



N-STEP-SCAN 算法调度：

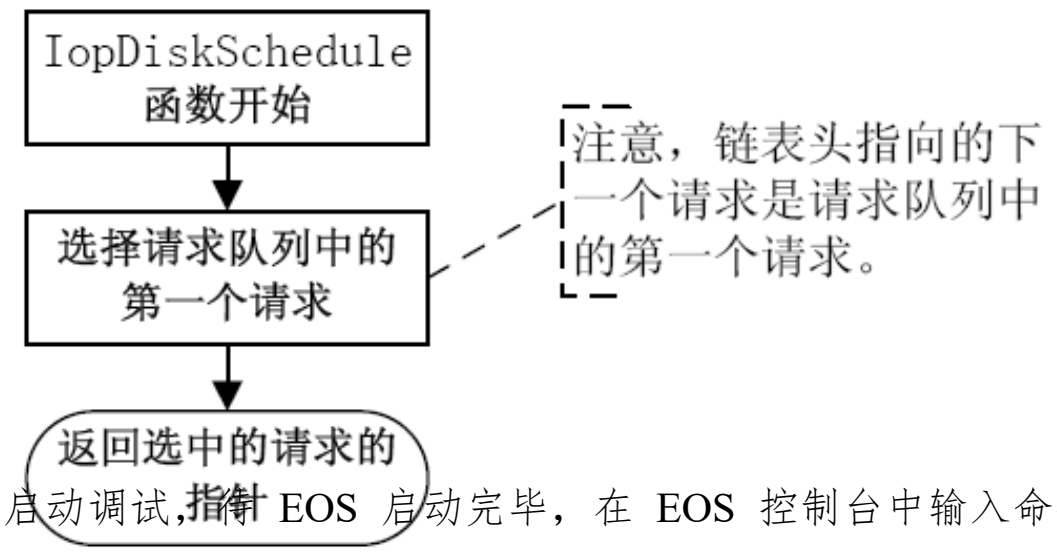


2.实验过程：

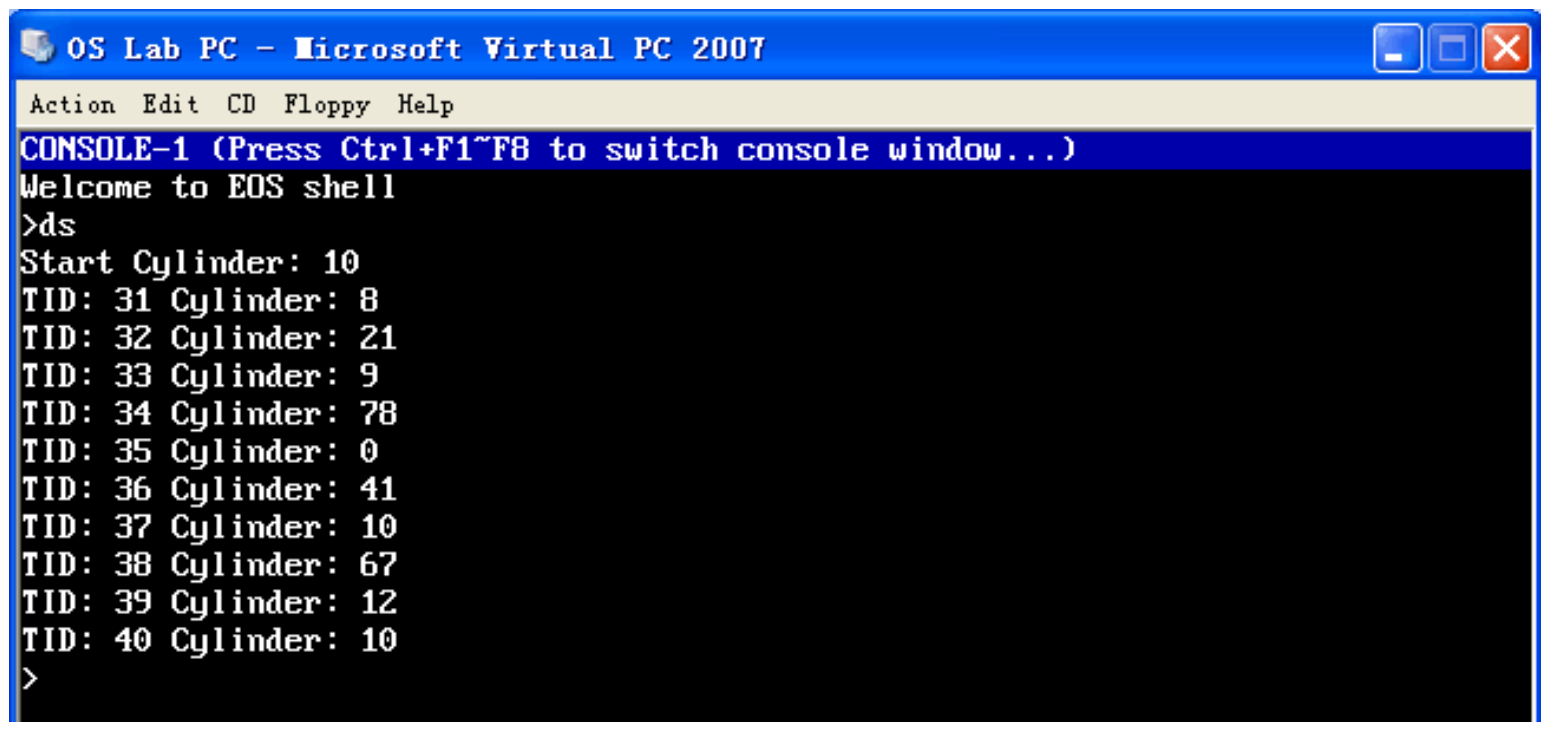
1）新建一个 EOS Kernel 项目。

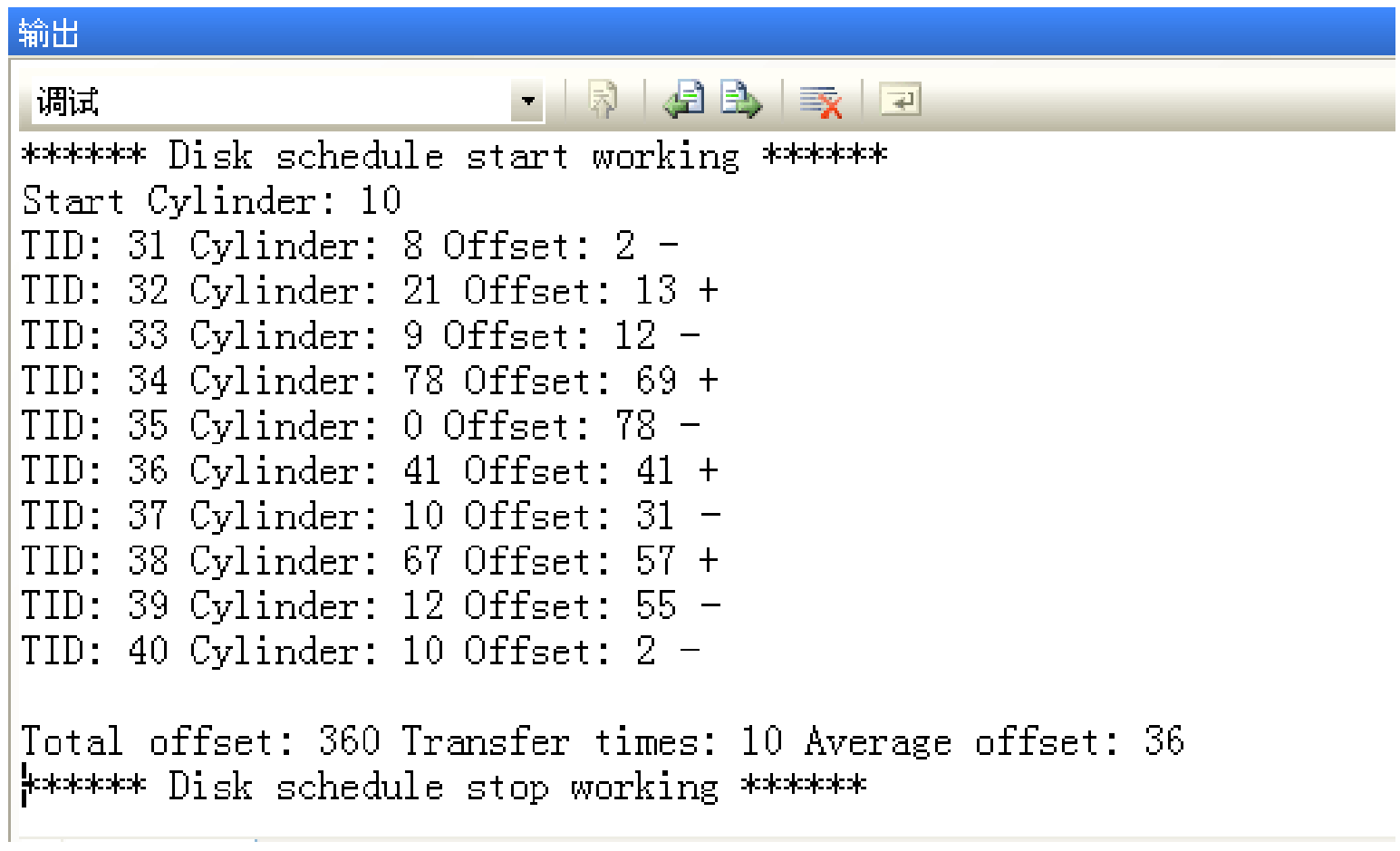
2）在 sysproc.c 文件中找到控制台命令“ds”对应的函数 ConsoleCmdDiskSchedule。“ ds” 命令专门用来测试磁盘调度算法。阅读该函数中的源代码，目前该函数使磁头初始停留在磁道 10， 其它被阻塞的线程依次访问磁道 8、21、9、78、0、41、10、67、12、10；

3）打开 io/block.c 文件，在 第 378 行找到磁盘调度算法函数 IopDiskSchedule。阅读该函数中的源代码，目前此函数实现了 FCFS 磁盘调度算法，流程图如下：



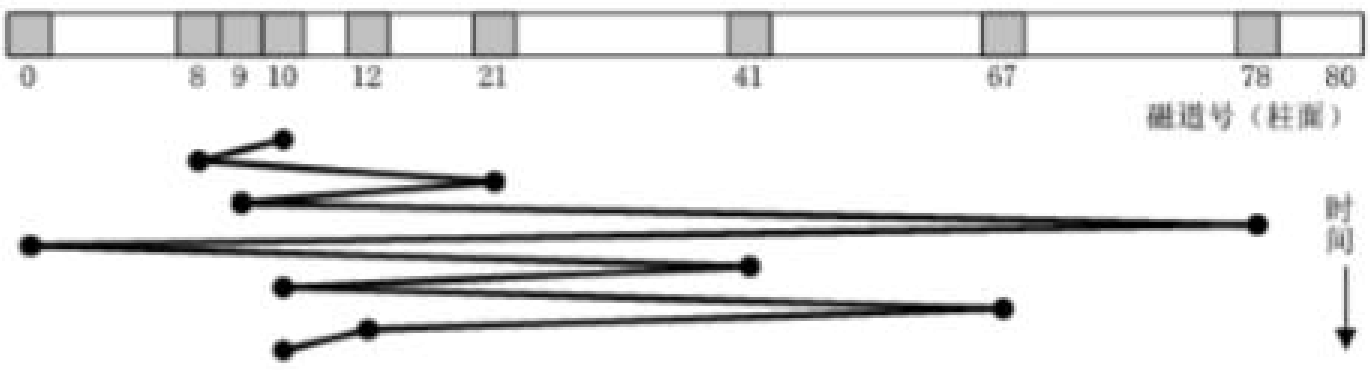
4）生成项目，启动调试，待 EOS 启动完毕，在 EOS 控制台中输入命令“ds”后按回车；





在 EOS 控制台中会首先显示磁头的起始位置是 10 磁道，然后按照线程被阻塞的顺

序依次显示线程的 信息(包括线程 ID 和访问的磁道号)。磁盘调度算法执行的过程中，在 OS Lab 的“输出”窗口中也会首 先显示磁头的起始位置，然后按照线程被唤醒的顺序依次显示线程信息（包括线程 ID、访问的磁道号、磁 头移动的距离和方向），并在磁盘调度结束后显示此次调度的统计信息(包括总寻道数、寻道次数和平均 寻道数)。对比 EOS 控制台和“输出”窗口中的内容，可以发现 FCFS 算法是根据线程访问磁盘的先后顺序 进行调度的。下图显示了本次调度执行时磁头移动的轨迹：



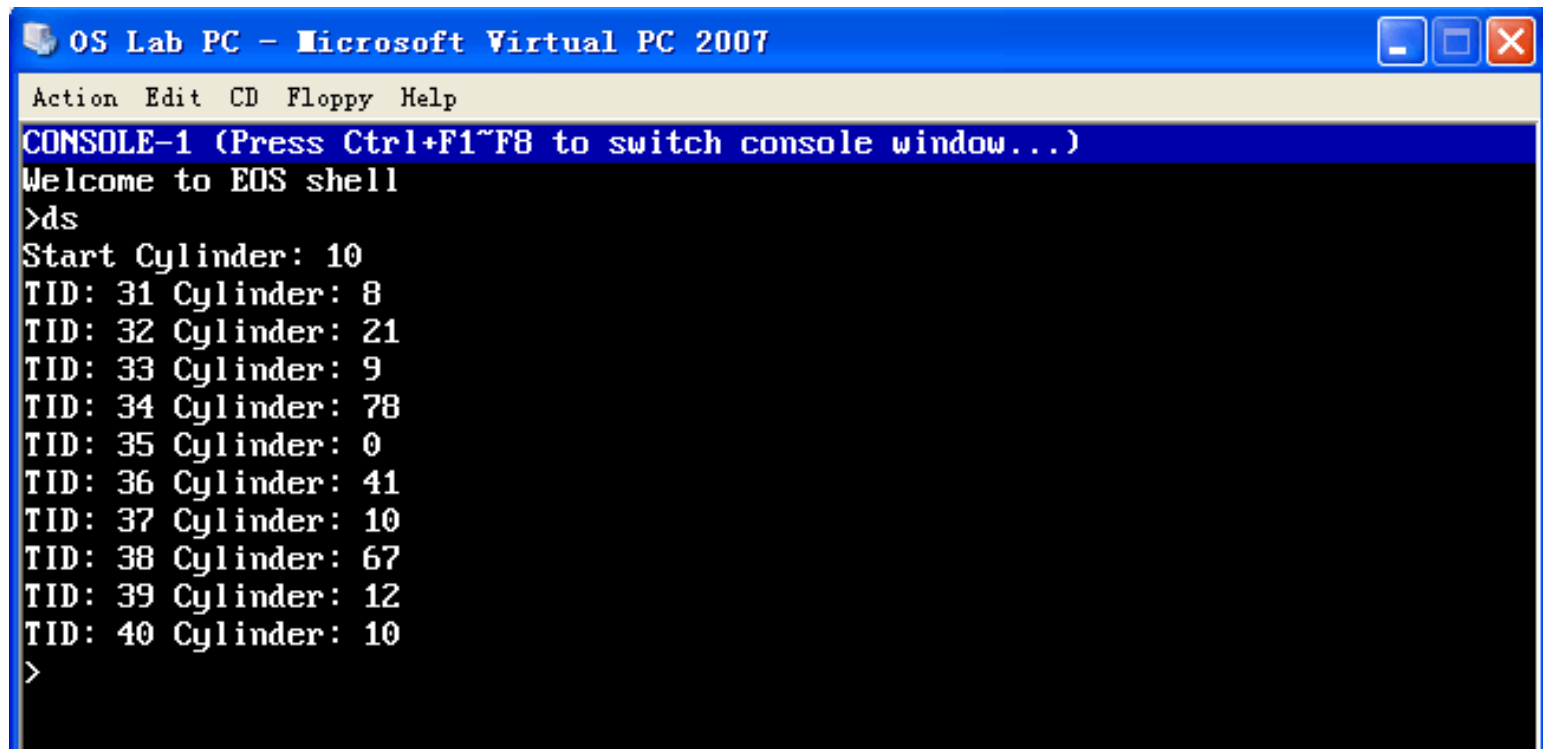
5）打开 sstf.c 文件，该文件提供的 IopDiskSchedule 函数实现了 SSTF 磁盘调度算法，其中应注意：

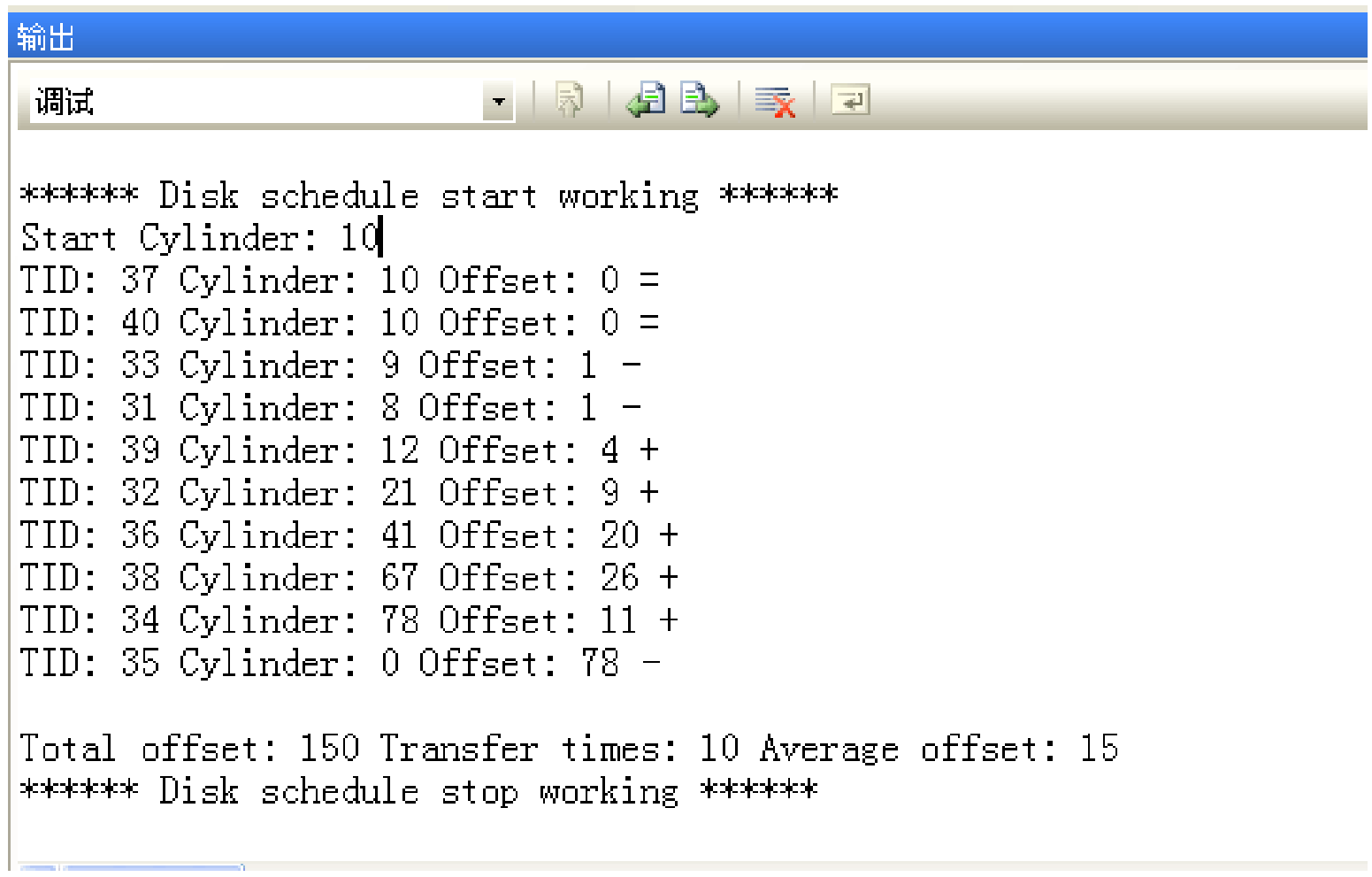
①变量 Offset 是有符号的长整型,用来表示磁头的偏移(包括距离和方向)。Offset 大于 0 时表示 磁头向内移动(磁道号增加);小于 0 时表示磁头向外移动(磁道号减少);等于 0 时表示磁头没 有移动。而名称以“Distance”结尾的变量都是无符号长整型，只表示磁头移动的距离（无方向）。 所以在比较磁头的偏移和距离时，或者在将偏移赋值给距离时，都要取偏移的绝对值（调用 C 库 函数 abs)。本实验在实现其它磁盘调度算法时也同样遵守此约定;

②在开始遍历之前，将最小距离（ShortestDistance）初始化为最大的无符号长整型数，这样，第 一次计算的距离一定会小于最小距离，从而可以使用第一次计算的距离来再次初始化最小距离。 本实验在实现其它磁盘调度算法时也同样使用了此技巧。

6）生成项目，启动调试，待 EOS 启动完毕，在 EOS 控制台中输入命令“ds”后按回车;

对比 EOS 控制台和“输出”窗口中的内容（特别是线程 ID 的顺序），可以发现，SSTF 算法唤醒线程的 顺序与线程被阻塞的顺序是不同的。图 18-4 显示了本次调度执行时磁头移动的轨迹。对比 SSTF 算法与 FCFS 算法在“输出”窗口中的内容，可以看出，SSTF 算法的平均寻道数明显低于 FCFS 算法。





7）验证 SSTF 算法造成的线程“饥饿现象”，使用 SSTF 算法时，如果不断有新线程要求访问磁盘，而且其所要访问的磁道与当前磁头所在磁道的 距离较近，这些新线程的请求必然会被优先满足，而等待队列中一些老线程的请求就会被严重推迟，从而 使老线程出现“饥饿”现象。

8）修改 sysproc.c 文件 ConsoleCmdDiskSchedule 函数中的源代码，仍然使磁头初始停留在磁道 10，而让其它线程依次访问磁道 78、21、9、8、11、41、10、67、12、10，生成项目，启动调试，待 EOS 启动完毕，在 EOS 控制台中输入命令“ds”后按回车；

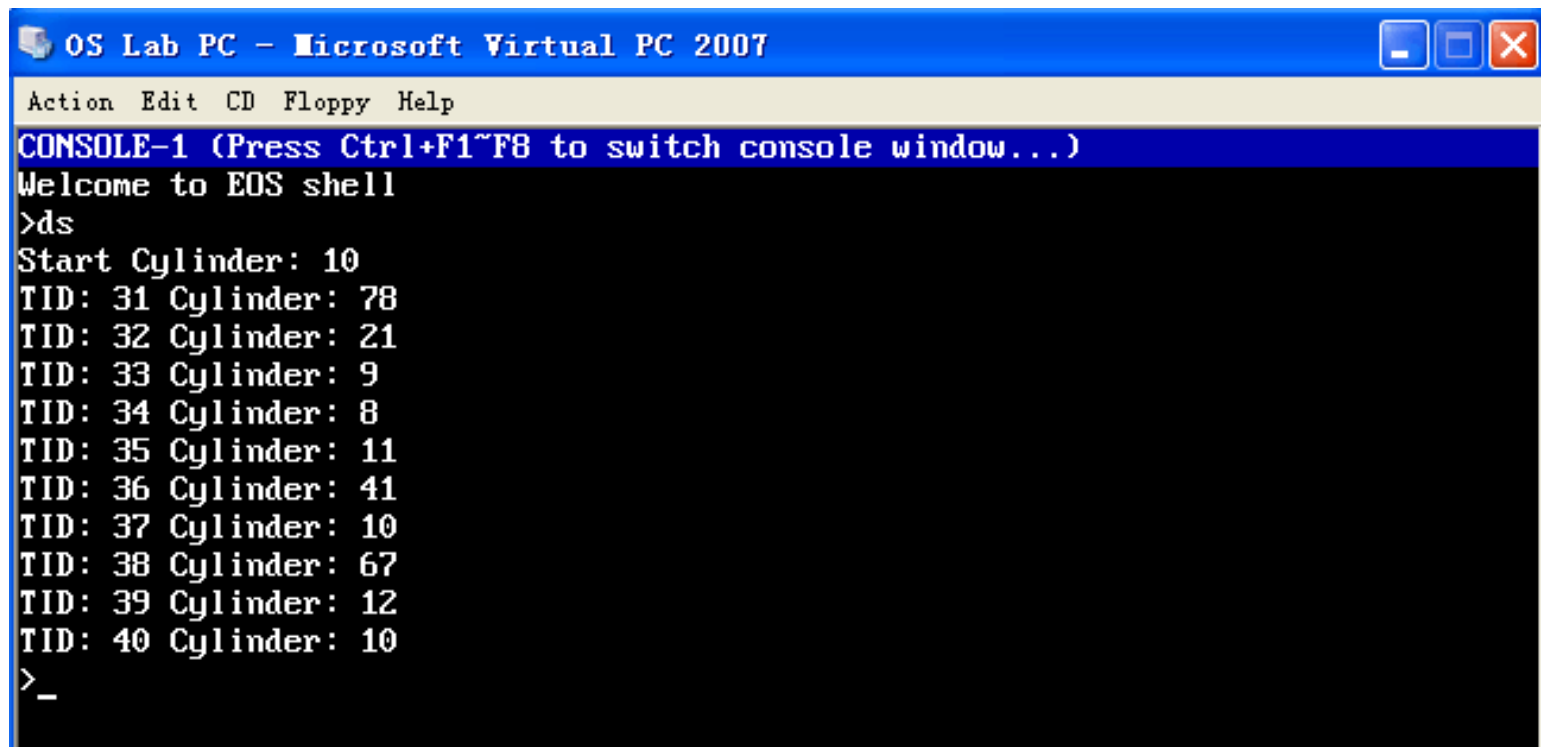
查看“输出”窗口中显示的内容，可以发现，虽然访问 78 号磁道的线程的请求第一个被放入请求队 列，但却被推迟到最后才被处理，出现了“饥饿”现象。如果不断有新线程的请求到达并被优先满足，则 访问 78 号磁道的线程的“饥饿”情况就会更加

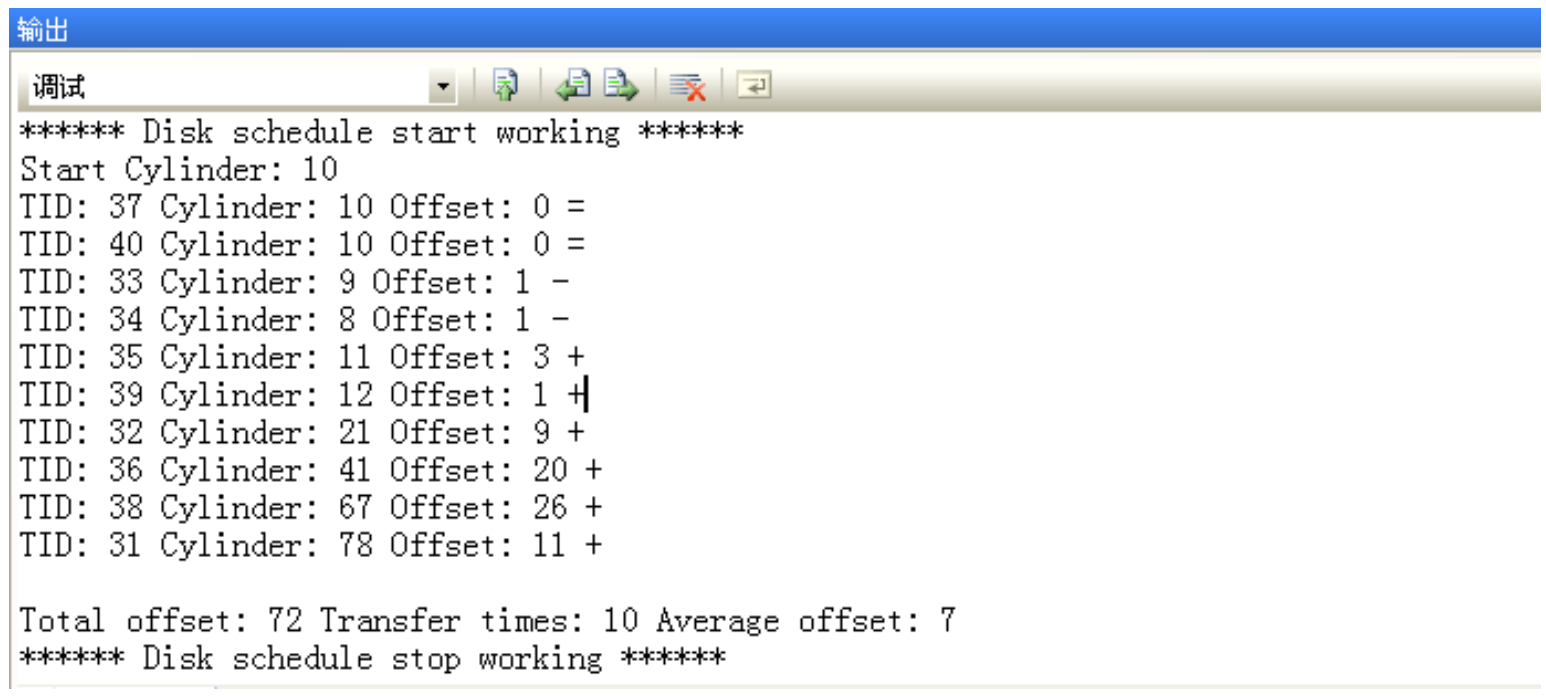
严重；

修改访问磁道顺序：

```
614         //
615         NewThreadAccessCylinder(StdHandle, 78);
616         NewThreadAccessCylinder(StdHandle, 21);
617         NewThreadAccessCylinder(StdHandle,  9);
618         NewThreadAccessCylinder(StdHandle,  8);
619         NewThreadAccessCylinder(StdHandle, 11);
620         NewThreadAccessCylinder(StdHandle, 41);
621         NewThreadAccessCylinder(StdHandle, 10);
622         NewThreadAccessCylinder(StdHandle, 67);
623         NewThreadAccessCylinder(StdHandle, 12);
624         NewThreadAccessCylinder(StdHandle, 10);
625
626         //
```

修改后执行“ds”命令的结果：





多次输入“ds”命令：

```
****** Disk schedule start working ******
Start Cylinder: 10
TID: 47 Cylinder: 10 Offset: 0 =
TID: 50 Cylinder: 10 Offset: 0 =
TID: 43 Cylinder: 9 Offset: 1 -
TID: 41 Cylinder: 8 Offset: 1 -
TID: 49 Cylinder: 12 Offset: 4 +
TID: 42 Cylinder: 21 Offset: 9 +
TID: 46 Cylinder: 41 Offset: 20 +
TID: 48 Cylinder: 67 Offset: 26 +
TID: 44 Cylinder: 78 Offset: 11 +
TID: 45 Cylinder: 0 Offset: 78 -

Total offset: 150 Transfer times: 10 Average offset: 15
```

9）对 SSTF 算法稍加改进后可以形成 SCAN 算法，可防止老线程出现“饥饿”现象。打开 scan.c 文件，该文件提供的 IopDiskSchedule 函数实现了 SCAN 磁盘调度算法。其中应注意下面几点：

①在 block.c 文件中的第 374 行定义了一个布尔类型的全局变量 ScanInside，用于表示扫描算法中 磁头移动的方向。该变量值为 TRUE 时表示磁头向内移动（磁道号增加）；值为 FALSE 时表示磁头 向外移动（磁道号减少）。该变量初始化为 TRUE，表示 SCAN 算法第一次执行时，磁头向内移动；

②在 scan.c 文件的 IopDiskSchedule 函数中使用了双重循环。第一次遍历队列时，查找指定方向 上移动距离最短的线程，如果在指定方向上已经没有线程，就变换方向，进行第二次遍历，同样 是查找移动距离最短的线程。在这两次遍历中一定能找到合适的线程。

10）使用 scan.c文件中 IopDiskSchedule 函数的函数体，替换 block.c 文件中 IopDiskSchedule 函 数的函数体，生成项目，启动调试，待 EOS 启动完毕，在 EOS 控制台中输入命令“ds”后按回车；

输出

调试

```
****** Disk schedule start working ******
Start Cylinder: 10
TID: 37 Cylinder: 10 Offset: 0 =
TID: 40 Cylinder: 10 Offset: 0 =
TID: 39 Cylinder: 12 Offset: 2 +
TID: 32 Cylinder: 21 Offset: 9 +
TID: 36 Cylinder: 41 Offset: 20 +
TID: 38 Cylinder: 67 Offset: 26 +
TID: 34 Cylinder: 78 Offset: 11 +
TID: 33 Cylinder: 9 Offset: 69 -
TID: 31 Cylinder: 8 Offset: 1 -
TID: 35 Cylinder: 0 Offset: 8 -

Total offset: 146 Transfer times: 10 Average offset: 14
```

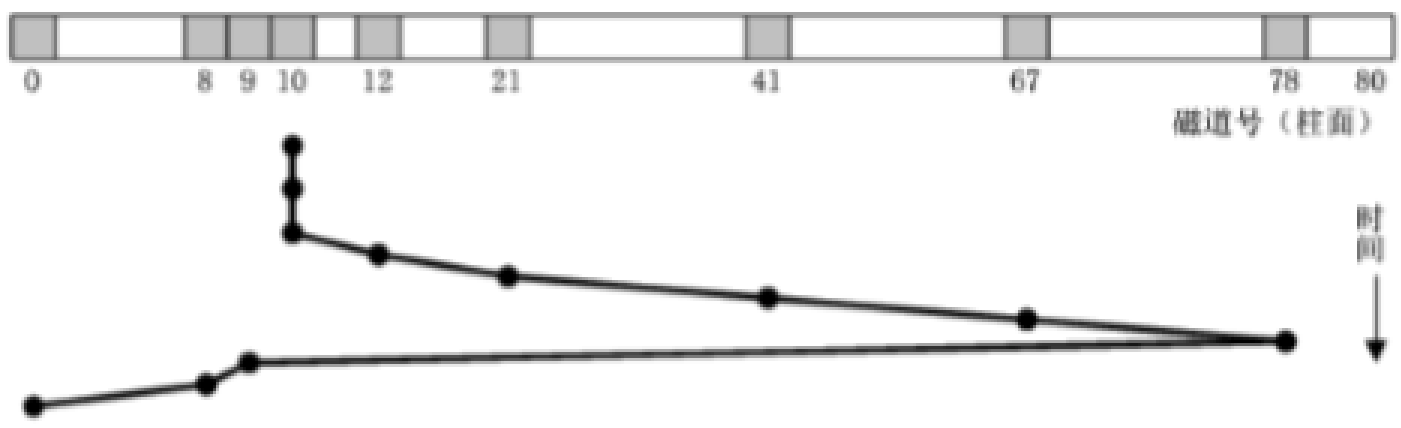
输出

调试

```
****** Disk schedule start working ******
Start Cylinder: 10
TID: 47 Cylinder: 10 Offset: 0 =
TID: 50 Cylinder: 10 Offset: 0 =
TID: 43 Cylinder: 9 Offset: 1 -
TID: 41 Cylinder: 8 Offset: 1 -
TID: 45 Cylinder: 0 Offset: 8 -
TID: 49 Cylinder: 12 Offset: 12 +
TID: 42 Cylinder: 21 Offset: 9 +
TID: 46 Cylinder: 41 Offset: 20 +
TID: 48 Cylinder: 67 Offset: 26 +
TID: 44 Cylinder: 78 Offset: 11 +

Total offset: 88 Transfer times: 10 Average offset: 8
```

SCAN 算法与 SSTF 算法在“输出”窗口中的内容，可以看出，SCAN 算法的平均寻道数有可能小于 SSTF 算法，所以说 SSTF 算法不能保证平均寻道数最少。下图显示了本次调度执行时磁头移动的轨迹：



11）改写 SCAN 算法，算法提示：

①在一次遍历中，不再关心当前磁头移动的方向，而是同时找到两个方向上移动距离最短的线程所 对应的请求，这样就不再需要遍历两次；

②在计算出线程要访问的磁道与当前磁头所在磁道的偏移后，可以将偏移分为三种类型：偏移为 0，表示线程要访问的磁道与当前磁头所在磁道相同，此情况应该优先被调度，可立即返回该线程对应的请求的指针；偏移大于 0，记录向内移动距离最短的线程对应的请求；偏移小于 0，记录向 外移动距离最短的线程对应的请求；

③循环结束后，根据当前磁头移动的方向选择同方向移动距离最短的线程，如果在同方向上没有线 程，就变换方向，选择反方向移动距离最短的线程；

流程如下所示：

SCAN 改写代码：

```
PREQUEST
IopDiskSchedule(
    VOID
    ){
    PLIST_ENTRY pListEntry;
```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/088020131002006034