

Lab4 Nachos 的文件系统Nachos 的硬盘是如何创建的:

1. 磁盘设备模拟了一个物理磁盘。Nachos 用宿主机中的一个文件来模拟一个单面物理磁盘，该磁盘由道组成，每个道由扇区组成，而每个扇区的大小是固定的。和实际的物理磁盘一样，Nachos 以扇区为读取/写入的最小单位，每个扇区有唯一的扇区地址，具体的计算方法是：

```
SectorSize * sectorNumber + MagicSize
```

该物理磁盘是一个异步的物理磁盘，同终端设备和网络设备一样，当系统发出读磁盘的请求，立即返回，只有具体的磁盘终端到来的时候，整个过程才算结束。

- 磁盘开始的4个字节是硬盘标识，其值为0x456789ab，用于指明该硬盘是一个 Nachos 硬盘：

```
#define MagicNumber    0x456789ab
#define MagicSize      sizeof(int)
#define DiskSize       (MagicSize + (NumSectors * SectorSize))
```

Nachos 在文件 `disk.cc` 中定义了磁盘类 `Disk` 的具体方法，具体包括初始化、销毁、读/写特定扇区、计算读写磁盘时间。初始化磁盘时，除初始化一些必要的参数外，还通过系统调用新建/打开一个名为 `*name` 的文件，该文件就是 Nachos 模拟的磁盘。由于在 FCB 中可以通过文件名来定位 `iNode`，因此通过文件名可唯一访问磁盘。新建文件后，首先把 `MagicNumber` 写入到开始位置，然后将文件指针便宜磁盘大小 `DiskSize` 的位移定位到文件尾，在文件尾写入 `0`，实际上代表 `EOF` 标志。

Nachos 中磁盘读写时间 = 寻道时间 + 扇区定位时间 + 数据传输时间。由于 Nachos 只是模拟了磁盘的物理结构，读写数据时以扇区为单位，因此对于给定的地址应计算磁道便宜量+扇区偏移量，并计算各自的时间。同时，定位磁道时，磁针有可能处于扇区中间，此时 Nachos 也模拟了扇区对准的时间。对准扇区后，再循环磁盘定位目标扇区。

- 磁盘 `Disk` 大小为 `DiskSize = MagicSize + (NumSectors * SectorSize)`，Nachos 并没有直接访问物理地址，而是通过偏移量来模拟磁盘地址的。

2. 查看 Nachos 硬盘上内容的方法:

- `./nachos -D` : 显示硬盘 DISK 中的文件系统;
- `od -c DISK` : 每一行输出 16 个字符，字符在文件中的偏移量（左边数字）以 8 进制表示;

```
0000000 T h i s   i s   t h e   s p r i
0000020 n g   o f   o u r   d i s c o n
0000040 t e n t . \n
0000046
```

- `hexdump -c DISK` : 每一行输出 16 个字符，字符在文件中的偏移量（左边数字）以 16 进制表示;

```
0000000 T h i s   i s   t h e   s p r i
0000010 n g   o f   o u r   d i s c o n
0000020 t e n t . \n
0000026
```

- `hexdump -C DISK` : 每一行输出 16 个字符，以 ASCII 形式输出文件内容。

00000000	54 68 69 73 20 69 73 20	74 68 65 20 73 70 72 69	This is the spr
00000010	6e 67 20 6f 66 20 6f 75	72 20 64 69 73 63 6f 6e	ng of our discon
00000020	74 65 6e 74 2e 0a		tent..
00000026			

3. 硬盘初始化的过程（如何在硬盘上创建一个文件系统）：

- Nachos 在 `disk.cc` 中调用构造函数初始化磁盘，通过打开文件并赋予该文件特定的大小来模拟磁盘大小，具体的初始化流程在 1 中介绍。对于文件系统，传入参数为 `-f` 时，Nachos 在函数 `Initialize` 中执行 `bool format = FALSE;` 接着调用 `synchDisk = new SynchDisk("DISK");` 初始化磁盘，该类的私有成员包括磁盘类 `Disk`，用于读写的锁和信号量，方法包括读扇区、写扇区、读写扇区完毕后的后处理操作。在其后调用 `fileSystem = new FileSystem(format);` 初始化了文件系统。

读写完毕后，要把信号量增加或把锁释放。

- Nachos 模拟的磁盘是异步设备。当发出访问磁盘的请求后立即返回，当从磁盘读出或写入数据结束后，发出磁盘中断，说明一次磁盘访问真正结束。Nachos 是一个多线程的系统，如果多个线程同时对磁盘进行访问，会引起系统的混乱。所以必须作出这样的限制：同时只能有一个线程访问磁盘当发出磁盘访问请求后，必须等待访问的真正结束。这两个限制就是实现同步磁盘的目的。
- 当线程向磁盘设备发出读访问请求后，等待磁盘中断的到来。一旦磁盘中断来到，中断处理程序执行 `semaphore->V()` 操作，`ReadSector` 得以继续运行。对磁盘同步写也基于同样的原理。在 `Disk` 构造函数中，Nachos 将引导块和大小写入磁盘中；在 `FileSystem` 构造函数中，Nachos 将位图、根目录文件写入其中：

语法：`FileSystem (bool format)`

参数：format: 是否进行格式化的标志

功能：在同步磁盘的基础上建立一个文件系统。当 format 标志设置时，建立一个新的文件系统；否则使用原来文件系统的内容。具体有：

- 如果 format 标志没有设置，则使用原有的文件系统
- 打开位图文件和目录文件，返回
- 2 如果 format 标志设置，则生成一个新的文件系统
- 生成新的位图和空白的根目录
- 生成位图 `FileHeader` 和目录 `FileHeader`
- 在位图中标识 0 和 1 号扇区被占用（虽然此时还没有占用）
- 为位图文件和目录文件申请必要的空间，如果申请不到，系统出错返回
- 将位图 `FileHeader` 和目录 `FileHeader` 写回 0 和 1 号扇区（这时候位图文件和目录文件所在的位置已经确定）
- 打开位图文件和目录文件
- 将当前的位图和目录写入相应的文件中（位置确定，内容终于可以写入）而
- 且这两个文件保持打开状态

4. 了解 Nachos 文件系统提供了哪些命令：

目前 Nachos 已实现初始化磁盘、挂载文件、删除文件、打印磁盘内容、查看位示图文件、按不同格式打印文件等操作。还需要实现扩展文件系统等操作。

5. 文件系统命令的实现原理：

```

class FileSystem {
public:
    FileSystem(bool format);           // 生成文件系统
    bool Create(char *name, int initialSize); // 生成一个文件
    OpenFile* Open(char *name);       // 打开一个文件
    bool Remove(char *name);          // 删除一个文件
    void List();                      // 列出文件系统所有文件
    void Print();                    // 列出文件系统所有文件及其内容
private:
    OpenFile* freeMapFile;           // 位图文件打开结构
    OpenFile* directoryFile;        // 根目录文件打开结构
};

```

- **Create** 方法：在当前的文件系统中创建一个固定大小的文件。首先，在根目录下搜寻该文件名，如果搜索到，出错返回；如果没有搜索到，申请文件 FileHeader 所需的空間，为新文件申请 FileHeader，根据新文件的大小申请相应块数的扇区，将所有有变化的数据结构写入磁盘。最终，如果生成成功，返回 TRUE，否则返回 FALSE。

在 Nachos 的文件系统中，对目录对象和位图对象的操作应该是临界区操作。因为如果两个线程同时需要向同一个目录中写入一个文件，可能会出现两个线程同时申请到同一个目录项；在空闲块分配时，也会出现相类似的情况。但是目前 Nachos 没有对此进行处理。

- **Open** 方法：在当前的文件系统中打开一个已有的文件，首先在根目录下搜寻该文件名，如果搜索到，寻找其对应的扇区位置，然后打开该文件并返回打开文件结构。
- **Remove** 方法：在当前的文件系统中删除一个已有的文件，首先在根目录下搜寻该文件名，如果没有搜索到，返回 FALSE；反之，如果搜索到，打开该文件并返回打开文件控制块，将该文件从目录中删除，释放 FileHeader 所占用的空间，释放文件数据块占用的空间，将对位图和目录的修改写回磁盘。如果删除成功，返回 TRUE；否则返回 FALSE。

6. 硬盘空闲块的管理方法：

- 硬盘采用位示图 (BitMap) 的思想管理硬盘的空闲块，即根据硬盘的扇区数建立一个位置图，当一个扇区空闲，位示图中对应的位为 0，否则为 1。位示图也是一个文件，由文件头+数据块组成，文件头保存在第 0 号扇区中，数据块存放位示图数据。
- 由于硬盘共有 1024 个扇区，需要 1024 位表示每个扇区的状态，因此位示图文件大小应为 $1024/8=128$ 字节，故位示图文件也可以恰好存储在一个扇区中。

```

class BitMap {
public:
    BitMap(int nitems); // 初始化位图，将所有位置0
    ~BitMap();
    void Mark(int which); // 标记第 which 位为占用
    void Clear(int which); // 清除第 which 位的占用状态
    bool Test(int which); // 测试第 which 位是否被占用
    int Find();           // 从头开始找第一个未被占用的位，标志其为占用
    int NumClear();       // 返回还有多少位空闲
    void Print();         // 打印位图

    void FetchFrom(OpenFile *file); // 读位图
    void WriteBack(OpenFile *file); // 写位图

private:

```

```

    int numBits;           // 8
    int numWords;          // 32
    unsigned int *map;
};

```

7. 目录文件的结构与管理:

- 一般的文件系统都采用树状目录结构，有的 UNIX 文件系统还有目录之间的勾连，形成图状文件系统结构。Nachos 则比较简单，只有一级目录，也就是只有根目录，所有的文件都在根目录下。而且根目录中可以存放的文件数是有限的。Nachos 文件系统的根目录同样也是通过文件方式存放的，它的 inode 占据了 1 号扇区。
- Nachos 采用一级目录表结构，目录表中的每个单元是一个名为 `DirectoryEntry` 的结构：

```

class DirectoryEntry {
public:
    bool inUse;           // 文件是否被占用(1B)
    int sector;           // 文件头所在扇区号(4B)
    char name[FileNameMaxLen + 1]; // 文件名称(9+1=10B)
};

```

文件目录表大小由下语句定义：

```

#define NumDirEntries      10
#define DirectoryFileSize  (sizeof(DirectoryEntry) * NumDirEntries)

```

由于C++有空间对其操作，因此 `sizeof(DirectoryEntry) == 20`，理论上来说，文件目录表所占空间最大可以是10条记录即 $10 \times 20 = 200\text{B}$ ，但由于 Nachos 限制了根目录文件头只能在第二个扇区，因此实际上只能容纳 $128/20=6$ 个文件。当访问一个文件时，根据文件名查找目录表，找到该文件文件头所在的扇区号，文件头中包含了文件大小、文件数据所占用的块数以及为文件数据所分配的盘块号。当删除一个文件时，只是将该文件对应的目录项中的 `inUser` 标记位清 0，其余两项并没有清除，便于文件的恢复。

8. 文件的结构与文件数据块的分配方法:

- 一个文件由“文件头+数据块”组成，文件头相当于 `iNode`，说明文件的具体属性以及文件的数据块在硬盘上的位置信息，Nachos 的文件头结构如下：

```

int numBytes;           // 文件大小，以字节为单位(4B)
int numSectors;         // 文件所分配的扇区数(4B)
int dataSectors[NumDirect]; // 文件每个数据块所对应的扇区号(120B)
// 总：128B

```

一个文件头限制最多占一个扇区，文件的扇区分配索引表最大空间为120B，因此一个文件最多占30个扇区，即 $30 \times 128\text{KB} = 3840\text{B}$ 。Nachos 分配文件数据块时采用了 UNIX 中的直接块管理方式，一个文件的各数据块可以存放不同的扇区中，但系统为文件分配数据块时，尽量将文件的数据分配到连续的扇区中，并依次将各数据块所在的扇区号记录在数组 `dataSectors[NumDirect]` 中。

9. 一个文件系统命令执行后硬盘的布局:

- `hexdump -C DISK` :

```
00000000  ab 89 67 45 80 00 00 00 01 00 00 00 02 00 00 00 |..gE.....|
00000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000080  00 00 00 00 c8 00 00 00 02 00 00 00 03 00 00 00 |.....|
00000090  04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000000a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
```

- `nachos -cp test/small small` :

```
root@ecs-a3e0:~/OS/nachos-3.4-SDU/code/filesys# hexdump -C DISK
00000000  ab 89 67 45 80 00 00 00 01 00 00 00 02 00 00 00 |..gE.....|
00000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000080  00 00 00 00 c8 00 00 00 02 00 00 00 03 00 00 00 |.....|
00000090  04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000000a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000100  00 00 00 00 7f 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000180  00 00 00 00 01 00 00 00 05 00 00 00 73 6d 61 6c |.....smal|
00000190  6c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |l.....|
000001a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000280  00 00 00 00 26 00 00 00 01 00 00 00 06 00 00 00 |....&.....|
00000290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000300  00 00 00 00 54 68 69 73 20 69 73 20 74 68 65 20 |....This is the |
00000310  73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
00000320  73 63 6f 6e 74 65 6e 74 2e 0a 00 00 00 00 00 00 |scontent.....|
00000330  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00020004
```

- `nachos -cp test/medium medium` :


```

root@ecs-a3e0:~/OS/nachos-3.4-SDU/code/filesys# hexdump -C DISK
00000000  ab 89 67 45 80 00 00 00 01 00 00 00 02 00 00 00 |..gE.....|
00000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000080  00 00 00 00 c8 00 00 00 02 00 00 00 03 00 00 00 |.....|
00000090  04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000000a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000100  00 00 00 00 ff 03 00 00 00 00 00 00 00 00 00 00 |.....|
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000180  00 00 00 00 01 00 00 00 05 00 00 00 73 6d 61 6c |.....small|
00000190  6c 00 00 00 00 00 00 00 01 00 00 00 07 00 00 00 |l.....|
000001a0  6d 65 64 69 75 6d 00 00 00 00 00 00 00 00 00 00 |medium.....|
000001b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000280  00 00 00 00 26 00 00 00 01 00 00 00 06 00 00 00 |....&.....|
00000290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000300  00 00 00 00 54 68 69 73 20 69 73 20 74 68 65 20 |....This is the |
00000310  73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
00000320  73 63 6f 6e 74 65 6e 74 2e 0a 00 00 00 00 00 00 |scontent.....|
00000330  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000380  00 00 00 00 98 00 00 00 02 00 00 00 08 00 00 00 |.....|
00000390  09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000003a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000400  00 00 00 00 54 68 69 73 20 69 73 20 74 68 65 20 |....This is the |
00000410  73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
00000420  73 63 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 20 69 |scontent..This i|
00000430  73 20 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 |s the spring of |
00000440  6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a |our discontent..|
00000450  54 68 69 73 20 69 73 20 74 68 65 20 73 70 72 69 |This is the spri|
00000460  6e 67 20 6f 66 20 6f 75 72 20 64 69 73 63 6f 6e |ng of our discon|
00000470  74 65 6e 74 2e 0a 54 68 69 73 20 69 73 20 74 68 |tent..This is th|
00000480  65 20 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 |e spring of our |
00000490  64 69 73 63 6f 6e 74 65 6e 74 2e 0a 00 00 00 00 |discontent.....|
000004a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00020004

```

- `nachos -cp test/big big`:

```

root@ecs-a3e0:~/OS/nachos-3.4-SDU/code/filesys# hexdump -C DISK
00000000 ab 89 67 45 80 00 00 00 01 00 00 00 02 00 00 00 |...gE.....|
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000080 00 00 00 00 c8 00 00 00 02 00 00 00 03 00 00 00 |.....|
00000090 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000100 00 00 00 00 ff ff 00 00 00 00 00 00 00 00 00 00 |.....|
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000180 00 00 00 00 01 00 00 00 05 00 00 00 73 6d 61 6c |.....smal|
00000190 6c 00 00 00 00 00 00 00 01 00 00 00 07 00 00 00 |l.....|
000001a0 6d 65 64 69 75 6d 00 00 00 00 00 00 01 00 00 00 |medium.....|
000001b0 0a 00 00 00 62 69 67 00 00 00 00 00 00 00 00 00 |....big.....|
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000280 00 00 00 00 26 00 00 00 01 00 00 00 06 00 00 00 |....&.....|
00000290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000300 00 00 00 00 54 68 69 73 20 69 73 20 74 68 65 20 |....This is the |
00000310 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
00000320 73 63 6f 6e 74 65 6e 74 2e 0a 00 00 00 00 00 00 |scontent.....|
00000330 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000380 00 00 00 00 98 00 00 00 02 00 00 00 08 00 00 00 |.....|
00000390 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000003a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000400 00 00 00 00 54 68 69 73 20 69 73 20 74 68 65 20 |....This is the |
00000410 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
00000420 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 20 69 |scontent..This i|
00000430 73 20 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 |s the spring of |
00000440 6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a |our discontent..|
00000450 54 68 69 73 20 69 73 20 74 68 65 20 73 70 72 69 |This is the spri|
00000460 6e 67 20 6f 66 20 6f 75 72 20 64 69 73 63 6f 6e |ng of our discon|
00000470 74 65 6e 74 2e 0a 54 68 69 73 20 69 73 20 74 68 |tent..This is th|
00000480 65 20 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 |e spring of our |
00000490 64 69 73 63 6f 6e 74 65 6e 74 2e 0a 00 00 00 00 |discontent.....|
000004a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000500 00 00 00 00 60 02 00 00 05 00 00 00 0b 00 00 00 |....`.....|
00000510 0c 00 00 00 0d 00 00 00 0e 00 00 00 0f 00 00 00 |.....|
00000520 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000580 00 00 00 00 54 68 69 73 20 69 73 20 74 68 65 20 |....This is the |
00000590 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
000005a0 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 20 69 |scontent..This i|
000005b0 73 20 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 |s the spring of |
000005c0 6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a |our discontent..|
000005d0 54 68 69 73 20 69 73 20 74 68 65 20 73 70 72 69 |This is the spri|

```

```

00000430 73 20 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 |s the spring of |
00000440 6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a |our discontent..|
00000450 54 68 69 73 20 69 73 20 74 68 65 20 73 70 72 69 |This is the spri|
00000460 6e 67 20 6f 66 20 6f 75 72 20 64 69 73 63 6f 6e |ng of our discon|
00000470 74 65 6e 74 2e 0a 54 68 69 73 20 69 73 20 74 68 |tent..This is th|
00000480 65 20 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 |e spring of our |
00000490 64 69 73 63 6f 6e 74 65 6e 74 2e 0a 00 00 00 00 |discontent.....|
000004a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000500 00 00 00 00 60 02 00 00 05 00 00 00 0b 00 00 00 |....`.....|
00000510 0c 00 00 00 0d 00 00 00 0e 00 00 00 0f 00 00 00 |.....|
00000520 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000580 00 00 00 00 54 68 69 73 20 69 73 20 74 68 65 20 |....This is the |
00000590 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
000005a0 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 20 69 |scontent..This i|
000005b0 73 20 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 |s the spring of |
000005c0 6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a |our discontent..|
000005d0 54 68 69 73 20 69 73 20 74 68 65 20 73 70 72 69 |This is the spri|
000005e0 6e 67 20 6f 66 20 6f 75 72 20 64 69 73 63 6f 6e |ng of our discon|
000005f0 74 65 6e 74 2e 0a 54 68 69 73 20 69 73 20 74 68 |tent..This is th|
00000600 65 20 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 |e spring of our |
00000610 64 69 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 |discontent..This|
00000620 20 69 73 20 74 68 65 20 73 70 72 69 6e 67 20 6f | is the spring o|
00000630 66 20 6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 |f our discontent|
00000640 2e 0a 54 68 69 73 20 69 73 20 74 68 65 20 73 70 |..This is the sp|
00000650 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 73 63 |ring of our disc|
00000660 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 20 69 73 20 |ontent..This is |
00000670 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 6f 75 |the spring of ou|
00000680 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 |r discontent..Th|
00000690 69 73 20 69 73 20 74 68 65 20 73 70 72 69 6e 67 |is is the spring|
000006a0 20 6f 66 20 6f 75 72 20 64 69 73 63 6f 6e 74 65 | of our disconte|
000006b0 6e 74 2e 0a 54 68 69 73 20 69 73 20 74 68 65 20 |nt..This is the |
000006c0 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 |spring of our di|
000006d0 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 20 69 |scontent..This i|
000006e0 73 20 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 |s the spring of |
000006f0 6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a |our discontent..|
00000700 54 68 69 73 20 69 73 20 74 68 65 20 73 70 72 69 |This is the spri|
00000710 6e 67 20 6f 66 20 6f 75 72 20 64 69 73 63 6f 6e |ng of our discon|
00000720 74 65 6e 74 2e 0a 54 68 69 73 20 69 73 20 74 68 |tent..This is th|
00000730 65 20 73 70 72 69 6e 67 20 6f 66 20 6f 75 72 20 |e spring of our |
00000740 64 69 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 |discontent..This|
00000750 20 69 73 20 74 68 65 20 73 70 72 69 6e 67 20 6f | is the spring o|
00000760 66 20 6f 75 72 20 64 69 73 63 6f 6e 74 65 6e 74 |f our discontent|
00000770 2e 0a 54 68 69 73 20 69 73 20 74 68 65 20 73 70 |..This is the sp|
00000780 72 69 6e 67 20 6f 66 20 6f 75 72 20 64 69 73 63 |ring of our disc|
00000790 6f 6e 74 65 6e 74 2e 0a 54 68 69 73 20 69 73 20 |ontent..This is |
000007a0 74 68 65 20 73 70 72 69 6e 67 20 6f 66 20 6f 75 |the spring of ou|
000007b0 72 20 64 69 73 63 6f 6e 74 65 6e 74 2e 0a 54 68 |r discontent..Th|
000007c0 69 73 20 69 73 20 74 68 65 20 73 70 72 69 6e 67 |is is the spring|
000007d0 20 6f 66 20 6f 75 72 20 64 69 73 63 6f 6e 74 65 | of our disconte|
000007e0 6e 74 2e 0a 00 00 00 00 00 00 00 00 00 00 00 00 |nt.....|
000007f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00020004

```

- 当系统中同时有 3 个文件时，磁盘的前三个扇区内容如下所示：

```

00000000 ab 89 67 45 80 00 00 00 01 00 00 00 02 00 00 00 |..gE.....|
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000080 00 00 00 00 c8 00 00 00 02 00 00 00 03 00 00 00 |.....|
00000090 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000000a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000100 00 00 00 00 ff ff 00 00 00 00 00 00 00 00 00 00 |.....|
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*

```


- 0x0~0x3 字节：硬盘开始的 4 个字节 (0x0~0x3) 是该磁盘的标识 (MagicNumber) ， 为 0x456789ab，显然数据采用小端存储。
- 0x4~0x83：扇区 0，128 字节；存放位示图文件头 (FCB, i-node) ；文件头由三部分组成： (int numBytes, int numSectors, int dataSectors[NumDirect];) ，即文件头所描述的三元组<文件的大小，占用的扇区数，各数据块所在扇区列表>，对应如下：
 - 0x4~0x7：4 个字节，位示图文件所占的字节数，值为 0x80，表示位示图文件大小为 0x80=128 字节；
 - 0x8~0xB：4 个字节，系统为位示图文件数据所分配的扇区数，其值为 0x1，表示位示图文件数据只需要一个扇区；
 - 0xC~0xF：位示图文件数据块所在的扇区号，其值为 0x2，说明系统将位示图文件的数据保存在第 2 号扇区中。
- 0x84~0x103：扇区 1，128 字节；存放目录表文件头；对应文件头由三部分 (int numBytes, int numSectors, int dataSectors[NumDirect]) 如下：
 - 0x84~0x87：4 个字节，目录表文件大小，值为 0xC8，表示目录表文件大小为 0xC8=200 字节；
 - 0x88~0x8B：4 个字节，系统为存放目录表所分配的扇区数，其值为 0x02，表示目录文件数据需要 2 个扇区 (目录文件大小为 200 字节，需要占用两个扇区) ；
 - 0x8C~0x8F：系统为目录表的前半部分分配的扇区号，其值为 0x3；
 - 0x90~0x93：系统为目录表的后半部分分配的扇区号，其值为 0x4；
- 0x104~0x183：扇区 2，128 字节；位示图的数据块，存储位示图文件内容，值为 0xff ff。
- 0x184~0x203 扇区 3 & 0x204~0x283 扇区 4：这两个扇区存放目录表，其值如下：

```

00000180  00 00 00 00 01 00 00 00 05 00 00 00 73 6d 61 6c |.....small|
00000190  6c 00 00 00 00 00 00 00 01 00 00 00 07 00 00 00 |l.....|
000001a0  6d 65 64 69 75 6d 00 00 00 00 00 00 01 00 00 00 |medium.....|
000001b0  0a 00 00 00 62 69 67 00 00 00 00 00 00 00 00 00 |...big.....|
000001c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000280  00 00 00 00 26 00 00 00 01 00 00 00 06 00 00 00 |...&.....|
00000290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*

```

每一个目录项占用空间为 20 字节，注意到目录项的内容为：


```

class DirectoryEntry {
public:
    bool inUse;                // 文件是否被占用(1B)
    int sector;                // 文件头所在扇区号(4B)
    char name[FileNameMaxLen + 1]; // 文件名称(9+1=10B)
};

```

由于对齐操作，0x05 表示第一个文件头所在扇区为第 5 号，0x73 6d 61 6c 6c 对应的 ASCII 码为 small，存放该文件的名称；最后四个字节值为 0x01，表示该文件已被占用；同理，第二个文件头所在扇区为第 7 号，第三个文件头所在扇区为第 10 号。

再执行 nachos -D：

 image-20220406105353717

可以发现，此时系统中存在 3 个文件，前 16 个扇区被占用，各文件所占扇区分别为 6、8~9、11~15。

10. 目前 Nachos 不能对文件进行扩展的原因及解决方案：

- 一个文件的大小是在创建时定的，在header，allocate时候，传入参数fileSize，按fileSize分配恰好合适大小的扇区数量；在openfile的write时，当写的位置超过fileSize时，会直接切掉超出部分，只是write不超出部分；所以一个文件的大小从创建开始是不会改变的，这导致 Nachos 文件扩展困难。

扇区号	起止字节	内容描述	内容	大小
	0x0~0x3	磁盘标识（魔数）	0x456789ab	4 字节
0	0x4~0x83	位示图文件头	Class FileHeader	128 字节
1	0x84~0x103	目录表文件头	Class FileHeader	128 字节
2	0x104~0x183	位示图文件数据块	Class BitMap	128 字节
3	0x184~0x203	目录表	Class Directory	128 字节
4	0x204~0x283	目录表	Class Directory	128 字节
5	0x284~0x303	第一个文件的文件头	Class FileHeader	128 字节
6	0x304~0x383	第一个文件的数据块		128 字节
7		(文件需要的块数不定)	
		第二个文件的文件头	Class FileHeader	128 字节
		第二个文件的数据块		128 字节
		
		第三个文件的文件头	Class FileHeader	128 字节
		第三个文件的数据块		128 字节
		
		第一个文件的数据块（扩展第一个文件时）		128 字节
		以此类推		

思考1：为什么将空闲块管理的位示图文件头，与目录表文件头存放在0号与1号这两个特殊的扇区中？

由于系统启动时需要根据目录文件的文件头访问根目录，因此为这两个特殊的结构分配到 0 号扇区与 1 号扇区这两个特殊的扇区中，是为了便于系统启动时从已知的、固定的位置访问它们。系统启动时可以很快地将它们装载入程序中，根据位示图文件头查找位示图的位置，去位示图中查看哪些扇区是空闲的，哪些扇区是满的；根据目录表文件头查找根目录表，去目录表中寻找文件信息。

思考2：Nachos 文件系统设计上的不足。

- 必须在文件生成时创建索引表。所以 Nachos 在创建一个文件时，必须给出文件的大小；而且当文件生成后，就不能改变文件的大小。
- 目前该文件系统没有 Cache 机制目前文件系统的健壮性不够强。
- 当正在使用文件系统时，如果突然系统中断，文件系统的内容可能不保证正确。

