## 实验十三 扫描 FAT12 文件系统管理的软盘

1. **实验目的**

 通过查看 FAT12 文件系统的扫描数据，并调试扫描的过程，理解 FAT12 文件系统管理软盘的方式。

 通过改进 FAT12 文件系统的扫描功能，加深对 FAT12 文件系统的理解。

1. **实验内容**

**2.1准备实验**

请读者按照下面方法之一在本地创建一个 EOS 内核项目，用于完成本次任务：

**方法一：从 CodeCode.net 平台领取任务**

读者需要首先登录 CodeCode.net 平台领取本次实验对应的任务，从而在 CodeCode.net 平台上创建个人项目，然后使用 OS Lab 提供的“从 Git 远程库新建项目”功能将个人项目克隆到本地磁盘中。

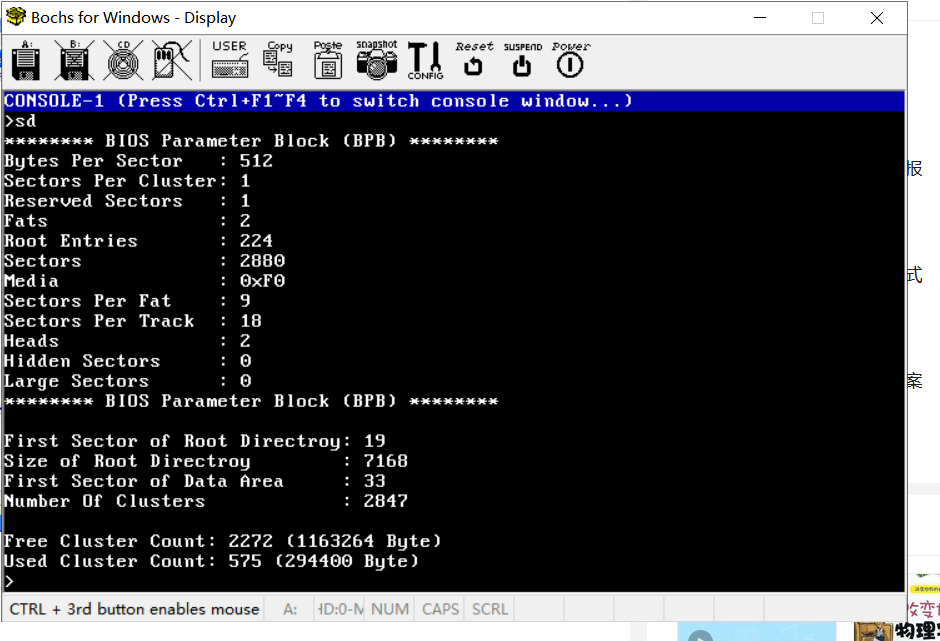
**方法二：不从 CodeCode.net 平台领取任务**

如果读者不使用 CodeCode.net 平台，就需要使用 OS Lab 提供的“从 Git 远程库新建项目”功能直接将实验模板克隆到本地磁盘中，实验模板的 URL 为

https://www.codecode.net/engintime/os-lab/Project-Template/eos-kernel.git。

2.2**阅读控制台命令“sd”相关的源代码，并查看其执行的结果**

通过执行指导书的步骤 1-3，学习到了操作系统中 FAT 的数据结构，每个表项都与数据区中的一个簇相对应，而且表项的序号也是与簇号一一对应的，在控制台执行 sd 命令 以后查看到 FAT 文件系统的具体内容，如：每个分区大小为 512 字节，每个簇有对应一个分区，根目录的第一个分区标号是 19，该目录大小为 7168，数据区的第一个分区是33 号，一共有 2847 个簇（2283 个空闲，564 个占用）等。



将卷控制块中缓存的 BIOS Parameter Block，以及卷控制块中的其他重要信息进行输出。

Bytes Per Sector:区字节数

Sectors Per Cluster:每簇的扇区数；

Reserved sectors:保留扇区

Fats：fat表

Root Entries:根目录项数;

Sectors：扇区

Media 媒体

Sectors per Fat ：每fat中扇区数

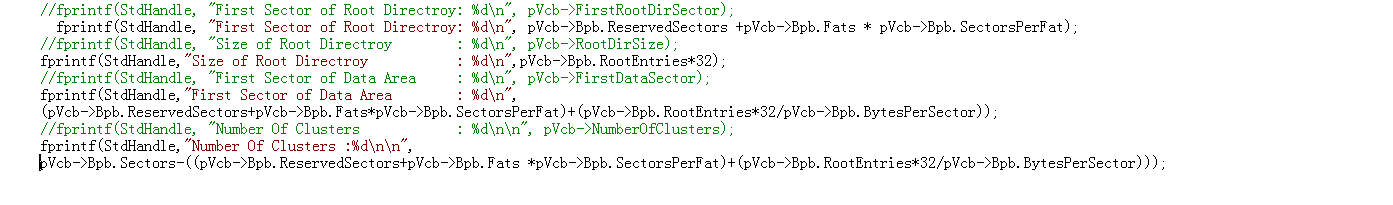
Sectors per track：每磁道扇区数

Heads 磁头

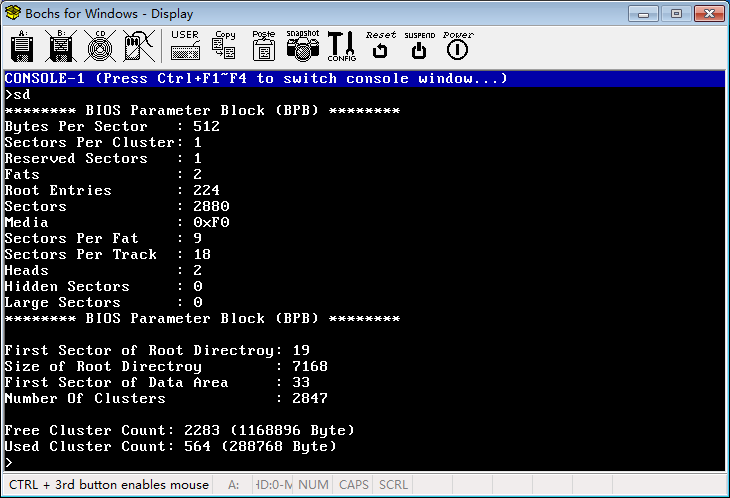
Hidden Sectors 隐藏扇区

Large Sectors：大扇区

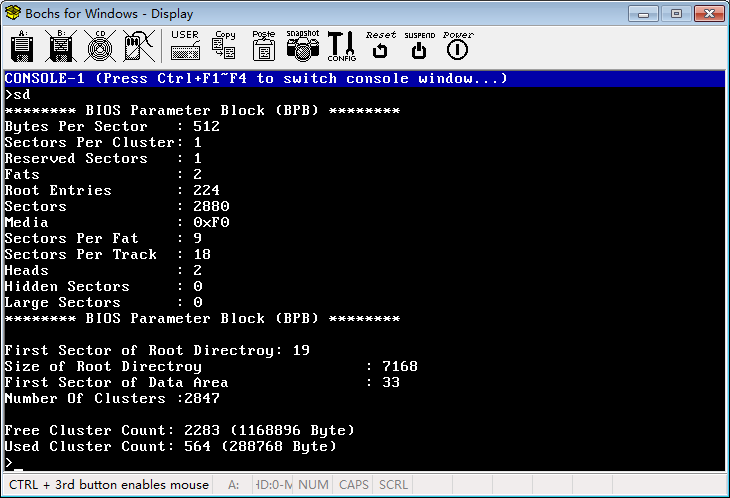
**2.3 根据 BPB 中的信息计算出其他信息**



与原来直接获取信息的输出对比如下：



（直接输出的信息）



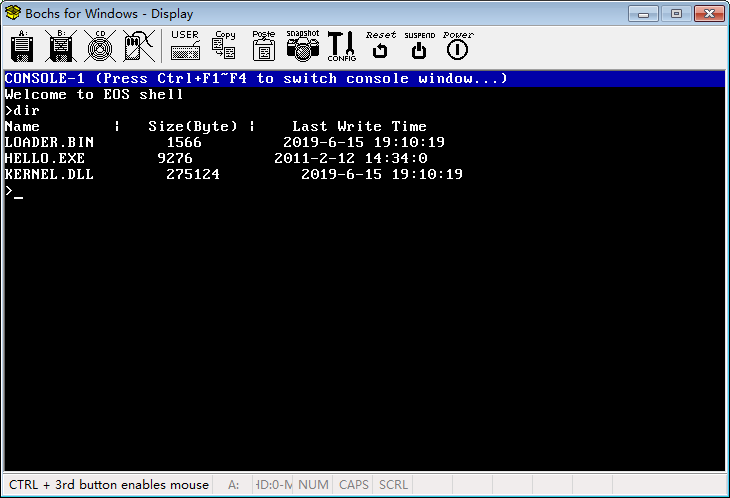
（间接计算输出的信息）

通过对比两次结果一致，从中学习到 FAT12 的主磁盘结构大致如下：



**2.4 阅读控制台命令“dir”相关的源代码，并查看其执行的结果**

通过执行指导书的步骤 1-3，看到了如下现象：

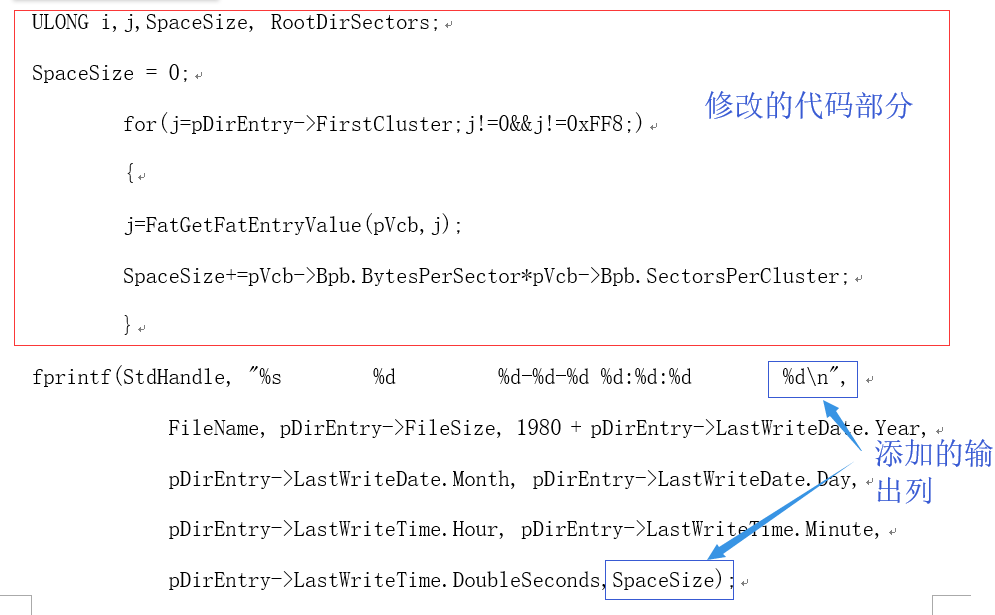


展示了根目录的相关信息，说明文件系统中的卷控制块 VCB 保存了目录的文件名、文件大小以及修改时间等相关信息。

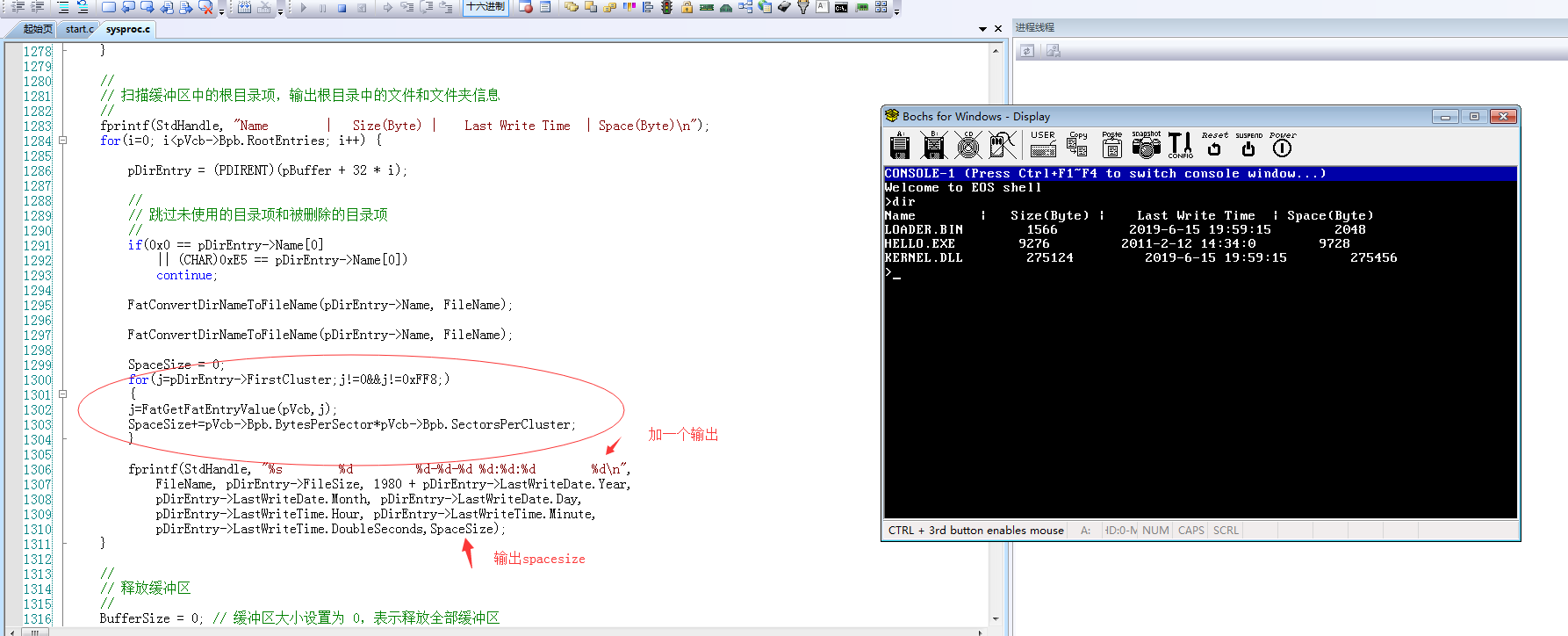
**2.5 输出每个文件所占用的磁盘空间的大小**

通过执行指导书的步骤 3.5.1 和 3.5.2，学习到了文件系统中文件大小和所占磁盘空间大 小是不同的，导致这种现象的原因是文件不是连续存在簇上的，可能会分散的占用不同的簇，所占磁盘的大小是以簇为基本单位的，是簇大小的整倍数，所以会>=文件大小。具体操作如下：

修改的代码部分：



运行结果如下：



1. **思考练习**

**（思考一）在 ConsoleCmdScanDisk 函数中扫描 FAT 表时，为什么不使用 FAT 表项的数量进行计数，而是使用簇的数量进行计数呢？而且为什么簇的数量要从 2 开始计数呢？**

答：FAT 是用于将数据区的磁盘空间分配给文件，被胡分成紧密排列的若干个表项，每个表项对应数据区的一个簇，表项的序号也是与簇号意义对应的；本来序号为 0和 1 的 FAT 表项应该对应于簇 0 和 1，但由于这两个表项被设置成了固定值，簇 0和簇 1 就没有存在的意义了， 这样数据区就起始于簇 2。

时间：2020.6.26

签名：

