要通过网络访问的远程设备。设备和其他特殊文件也可以通过VFS访问。接下来，我们将描述第一个被Linux广泛使用的文件系统ext2（second extended file system）。随后，我们将讨论ext3文件系统中所作的改进。所有的Linux都能处理有多个磁盘分区且每个分区上有一个不同文件系统的情况。

1. Linux 虚拟文件系统

为了使应用程序能够与在本地或远程设备上的不同文件系统进行交互，Linux采用了一个被其他UNIX系统使用的方法：虚拟文件系统。VFS定义了一个基本的文件系统抽象以及这些抽象上允许的操作集合。调用上节中提到的系统调用访问VFS的数据结构，确定要访问的文件所属的文件系统，然后通过存储在VFS数据结构中的函数指针调用该文件系统的相应操作。

图10-30总结了VFS支持的四个主要的文件系统结构。其中，超级块包含了文件系统布局的重要信息，破坏了超级块将会导致文件系统无法访问。每个i节点（i-node, index-node的简写，但是从来不这样称呼它，而一些人省略了“-”并称之为i节点）表示某个确切的文件。值得注意的是在Linux中，目录和设备也当作是文件，所以它们也有自己对应的i节点。超级块和i节点都有相应的结构，由文件系统所在的物理磁盘维护。

对象	描述	操作
Superblock	特定的文件系统	read_inode,sync_fs
Dentry	目录项，路径的一个组成部分	create,link
I-node	特定的文件	d_compare,d_delete
File	跟一个进程相关联的打开文件	read,write

图10-30 VFS支持的文件系统抽象

为了便于目录操作及路径（比如/usr/ast/bin）的遍历，VFS支持dentry数据结构，它表示一个目录项。这个数据结构由文件系统在运行过程中创建。目录项被缓存在dentry_cache中，比如，dentry_cache会包含/，/usr，/usr/ast的目录项。如果多个进程通过同一个硬连接（即相同路径）访问同一个文件，它们的文件对象都会指向这个cache中的同一个目录项。

file数据结构是一个打开文件在内存中的表示，并且在调用open系统调用时被创建。它支持read、write、sendfile、lock等上一节中提到的系统调用。

在VFS下层实现的实际文件系统并不需要在内部使用与VFS完全相同的抽象和操作，但是必须实现跟VFS对象所指定的操作在语义上等价的文件系统操作。这四个VFS对象中的operations数据结构的元素都是指向底层文件系统函数的指针。

2. Linux ext2 文件系统

接下来，我们介绍在Linux中最流行的磁盘文件系统：ext2。第一个Linux操作系统使用MINIX文件系统，但是它限制了文件名长度并且文件长度最大只能是64MB。后来MINIX被第一个扩展文件系统，ext文件系统取代。ext可以支持长文件名和大文件，但由于它的效率问题，ext被ext2代替，ext2在今天还在广泛使用。

ext2的磁盘分区包含了一个如图10-31所示的文件系统。块0不被Linux使用，而通常用来存放启动计算机的代码。在块0后面，磁盘分区被划分为若干个块组，划分时不考虑磁盘的物理结构。每个块组的结构如下：

第一个块是超级块，它包含了该文件系统的信息，包括i节点的个数、磁盘块数以及空闲块链表的起始位置（通常有几百个项）。下一个是组描述符，存放了位图（bitmap）的位置、空闲块数、组中的i节点数，以及组中目录数等信息，这个信息很重要，因为ext2试图把目录均匀地分散存储到磁盘上。

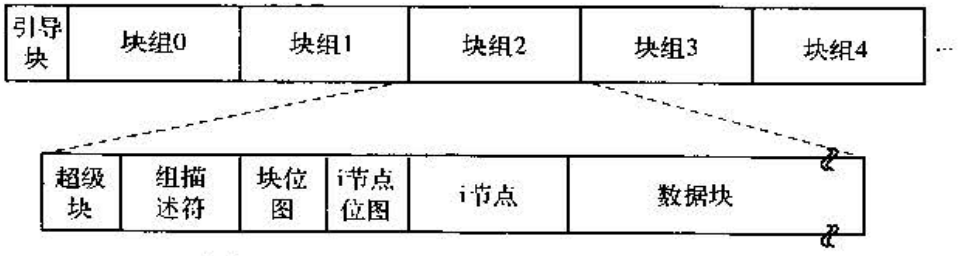

图10-31 Linux ext2文件系统的磁盘布局