Linux 下文件和目录的本质区别和组成

目录的递归原理

遇到目录自己调自己,

遇到文件,把文件放到数组中,同时自己调用自己

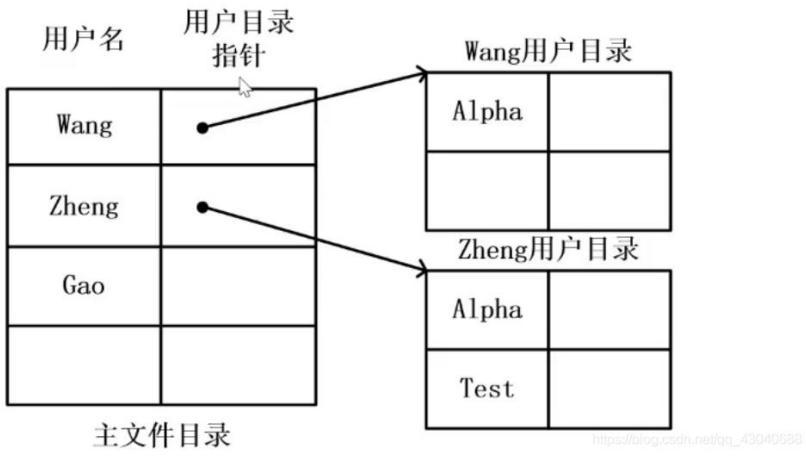
核心功能
检索文件–（如根据文件名、属性、内容）
目录结构

1. 单级目录

文件名	物理地址	文件说明	状态位
文件名1			
文件名2			
……			

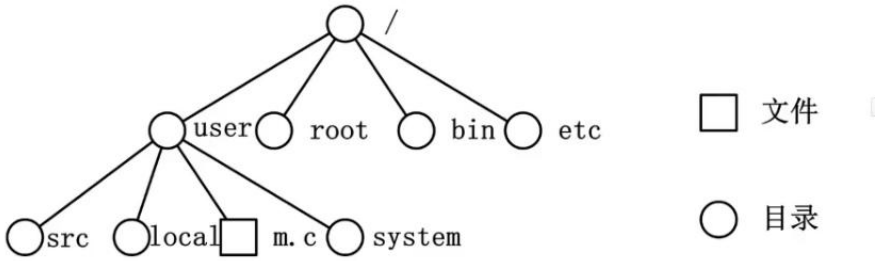
最重要的两个字段是文件名和物理地址
实现的方式是线性表
优点
十分简单
缺点
不允许文件名重复

2.两级目录



分为主文件目录和用户目录，增加了用户管理

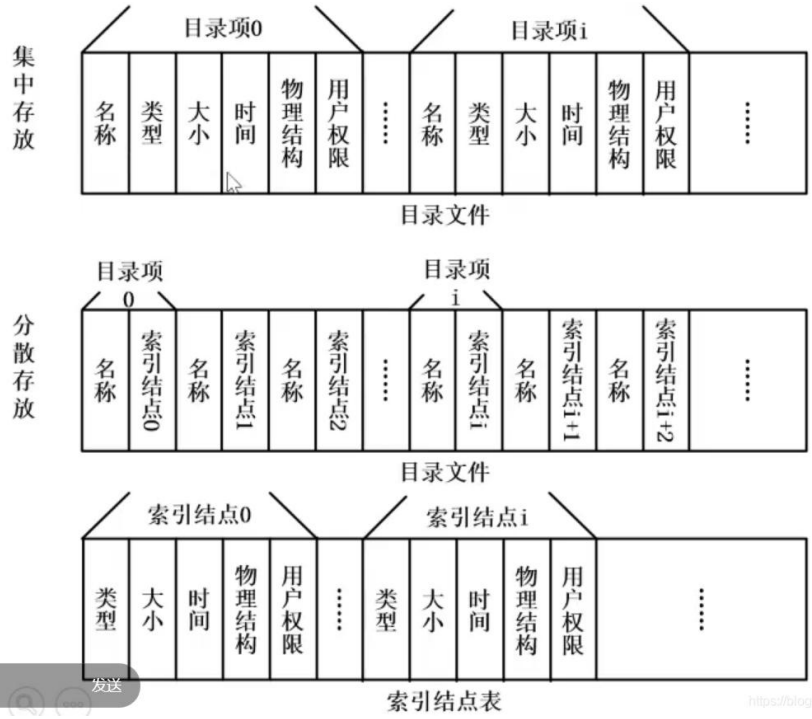
3.多级目录



两级目录和多级目录都是树结构
分为目录对象和文件对象
文件的位置使用路径描述
问题聚焦：实现文件的存储

- 目录文件是一种特殊的文件，由目录或文件的控制信息组成。
- 目录文件结构有两种。
  - 控制信息集中存储。
  - 控制信息分散存储。

目录文件
目录文件信息叫做目录项



皇天惊虞收集

集中存放
优点

目录简单
因为文件名称是最关键信息，其他信息会造成存储空间过大。所以将其他属性信息进行压缩，形成分散存放。（其他属性信息放在索引中）
所以采用分散存放的形式，需要根据需求选择目录文件存储的信息

文件目录
文件目录就是我们很熟悉的 Windows 操作系统的文件夹

文件控制块
实现文件目录的关键数据结构
目录本身就是一种结构文件,由一条条记录组成.每条记录对应一个放在目录下的文件。
目录文件中的一条记录就是一个文件控制块"FCB"
FCB 中包含了文件的一些基本信息

文件名
- 物理地址
- 逻辑结构
- 物理结构
- 存取控制信息
- 是否可读/可写
- 禁止访问的用户名单
- 使用信息(如文件的建立时间,修改时间)
- FCB 实现了文件名到文件之间的映射,旨在实现按名存取.

功能
搜索
当用户要使用一个文件时,系统要根据文件名搜索目录,找到该文件对应的目录项

创建文件
创建一个新文件时,需要在其所属的目录中增加一个目录项

删除文件
当删除一个文件时,需要在其目录中删除相应的目录项

显示目录
用户可以请求显示目录的内容,如显示该目录中所有的文件及相应属性

修改目录
某些文件属性保存在目录中,因此这些属性变化需要相应的目录项(譬如重命名)。

目录结构

皇天惊虞收集

单级目录结构[不允许重名]
早期操作系统并不支持多级目录,整个操作系统中只简历一张目录表,每个文件占一个目录项。
不允许文件重名,不适应多用户操作系统。

两级目录结构[不能对文件进行分类]
分为主文件目录和用户文件目录

主文件目录记录用户名及其相应用户文件目录的存放位置

用户文件目录由该用户的 FCB 组成

允许不同用户的文件重名。文件名虽然相同,但是对应的其实是不同的文件。

两级目录结构不允许用户的文件重名,也可以在目录上实现访问限制(检查此时登录的用户名是否匹配)。

多级目录结构(树型目录)[不方便文件共享]

但是两级目录结构依然缺乏灵活性,用户不能对自己的文件进行分类。

用户要访问某个文件时要用文件路径名标识文件,文件路径名是个字符串。各级目录之间用'/'隔开。从根目录出发的路径称为绝对路径。

我们可以设置当前目录,也就是相对路径,节省系统资源。

在引入了当前目录和相对路径后, 磁盘的 I/O 次数减少了。这就提升了访问文件的效率。

总结:
树形目录结构可以很方便地对文件进行分类,层次结构清晰,也能够更有效的进行文件的管理和保护.但是,树形结构不便于实现文件的共享。为此,提出了"无环图目录结构".

无环图目录结构
无环图目录结构在树形目录结构的基础上,增加了一些指向同一节点的有向边,使整个目录成为一个有向无环图。可以更方便地实现多个用户间的文件共享。

可以为不同的文件名指向同一个文件,甚至可以指向同一个目录。

需要为每个共享节点设置一个共享计数器,用于记录此时有多少个地方在共享该节点。用户提出删除节点的请求时,只是删除该用户的 FCB,并使共享计数器-1,并不会直接删除共享节点。只有共享计数器为 0 的时候,才会真正的删除共享节点。

注意:共享文件不同于复制文件,在共享文件中,由于各用户指向的是同一个文件,因此只要其中一个用户修改了文件数据,那么所有用户都可以看到文件数据的变化。

索引目录

索引节点是对 FCB 的改进,其实在查找各级目录的时候,只需要用到文件名,只有文件名匹配时,才需要读出文件的其他信息。因此可以考虑让目录表瘦身来提升效率。

在 Linux 和其他类 Unix 文件系统中，文件和目录本质上都是由 inode 和数据块组成的：

文件：文件的 inode 存储了元数据（如所有者、权限、文件大小、创建和修改时间等）以及指向数据块的指针。这些数据块存储了文件的实际内容。

目录：目录的 inode 同样存储了元数据和指向数据块的指针。不同的是，这些数据块存储的是一系列目录项，每个目录项包含一个文件名和一个 inode 号。这样，我们就可以通过文件名找到对应的 inode，并进一步找到文件或者子目录的内容。

然而，这只是构成文件或目录的基本部分。在实际使用中，还需要考虑到如何组织这些文件和目录（比如创建文件系统的目录结构），以及如何处理权限、所有权和其他安全性问题。这些都是构建和管理文件系统的重要部分。

inode 详解

inode 这个词来自于 index node 的缩写，即索引节点。它在 Unix 类型的文件系统中被用来表示文件系统对象（如文件和目录）的元数据。

在文件系统被初始化（例如，通过格式化操作）时，inode 会被创建并存储在磁盘上的特定区域。每个 inode 都有一个唯一的编号，系统通过这个编号来识别文件。

每个 inode 包含以下信息：

文件类型：普通文件、目录、字符设备、块设备、管道、链接、套接字等。

文件权限和所有者：读、写、执行权限，用户 ID（所有者）和组 ID。

时间戳：文件创建时间、最后访问时间、最后修改时间等。

文件大小。

指向文件数据块的指针。

注意，inode 不存储文件名。在 Unix 类型的文件系统中，文件名是存储在目录的数据块中的。目录是一种特殊类型的文件，其数据块中存储了一组目录项，每个目录项都是一个文件名和一个 inode 号的映射。

通过这种方式，用户可以通过文件名来访问文件，而系统通过文件名找到对应的 inode，进而找到文件的数据。这就是 Unix 类型文件系统中的 inode 工作原理。

inode 的数量是在文件系统初始化时决定的，通常根据文件系统的大小和预期的文件数量来计算。这意味着，即使磁盘还有剩余空间，如果所有的 inode 都已经被使用，也无法再创建新的文件。

两者的区别

在 Linux 系统中，目录和文件的底层数据结构都是通过 inode (索引节点) 来进行管理的。inodes 存储有关文件系统对象（如文件和目录）的元数据，比如对象的所有者、权限、创建和修改日期以及物理数据位置等信息。

然而，目录和文件在这种管理方式中有着本质的区别：

文件：文件的 inode 存储了指向文件内容所在的数据块的指针，文件的内容就保存在这些数据块中。文件的 inode 还保存了文件的各种属性，如文件大小、创建时间、所有者等信息。

目录：目录的 inode 不是直接指向数据内容，而是指向一种特殊的数据结构，我们通常称之为目录项（Directory Entries）。每一个目录项包括两部分，一是文件名，二是指向该文件（或子目录）的 inode 的指针。所以，目录实际上是一个特殊的文件，它的内容是一种映射关系，即文件名到 inode 的映射。因此，目录可以包含其他文件或目录，这就构成了我们常见的文件系统的树状结构。

以上就是 Linux 下目录和文件的底层区别。目录的这种结构使得它可以包含文件和其他目录，从而形成复杂的文件系统。

文件的组成

在 Linux 中，文件本质上是由 inode 和数据块构成的。

inode 是文件的元数据，包括了文件的所有者、权限、大小、创建和修改的时间戳等信息，以及指向实际存储文件数据的数据块的指针。每个 inode 有一个唯一的编号，系统通过这个编号来识别文件。

而数据块则是存储了文件的实际内容，比如文本、图片、视频等。

所以，一个文件在文件系统中的存在，实际上就是一个 inode 和一系列数据块的组合。inode 提供了关于文件的元数据和找到文件数据的路径，数据块则存储了文件的实际内容。

文件名和目录项则是在目录的数据结构中定义的。目录包含了一组目录项，每个目录项都是一个文件名和一个 inode 编号的对应关系。通过这种方式，用户可以通过文件名来访问文件，而系统通过文件名找到对应的 inode，进而找到文件的数据。

目录的组成

在 Linux 中，目录本质上也是由 inode 和数据块构成的。

不过，目录和文件的数据块中存储的内容是不同的：

文件的数据块中存储的是文件的实际内容，比如文本、图片、音频、视频等。

目录的数据块中存储的是一系列目录项。每一个目录项包含一个文件名和一个 inode 号。这样，用户可以通过文件名找到文件，系统则通过 inode 号找到实际的文件内容。

这就是目录和文件在底层数据结构上的区别。总的来说，目录和文件都是由 inode 和数据块组成的，但是它们的数据块中存储的内容有所不同。