

图解 Linux 文件系统

原创 songsong001 Linux内核那些事 4月11日

之前我写过有关 Linux 文件系统源码分析的文章，但从源码角度分析文件系统略显枯燥（对新手不友好），所以这次主要通过图文的方式来讲解 Linux 文件系统的原理，而不用陷入源代码的深渊之中。

一、硬盘简介

在介绍文件系统前，我们先来了解一下 硬盘 。

众所周知，内存在断电后数据就会丢失，所以现代计算机都通过 硬盘 来进行数据存储。也就是说，硬盘中的数据在断电后依然能够保存下来。

现在比较流行的硬盘分为： 机械硬盘（HDD） 和 固态硬盘（SSD） 。由于本文重点介绍的对象是 文件系统 ，所以对于硬盘的原理就不进行过多的介绍。下面是 机械硬盘 和 固态硬盘 的对照图：

机械硬盘



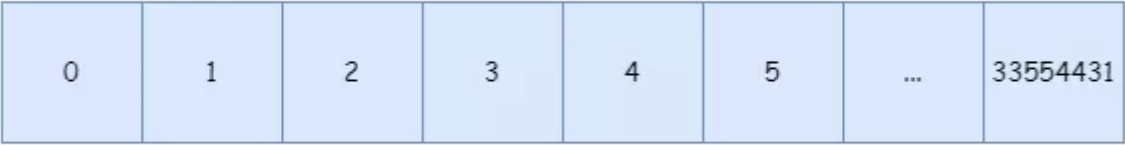
固态硬盘



Linux内核那些事

我们可以把硬盘想象成一个巨大的数组，而数组的每个元素代表一个数据块，如下图：

128GB 的硬盘



Linux内核那些事

在 Linux 内核中，每个数据块定义为 4KB 的大小，所以一个 128GB 的硬盘可以分为 33554432 个数据块，内核就是以数据块的编号来对硬盘进行读写操作的。

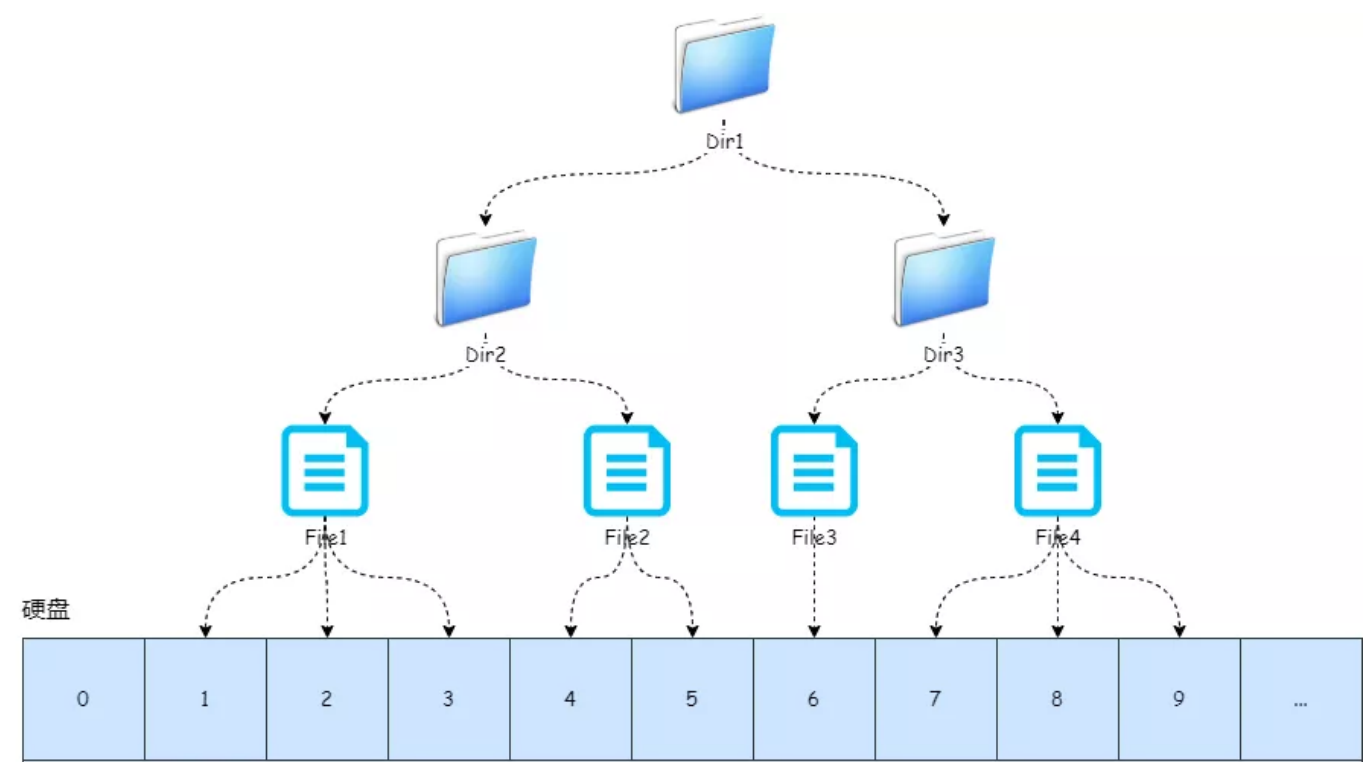
二、什么是文件系统

前面说过，内核是以数据块的形式来对硬盘进行读写的，但是这对人类来说是非常不直观的，因为我们不可能记住每一个数据块保存了什么数据。

为了让用户在使用上更方便和直观，Linux 内核抽象出两个概念来管理硬盘中的数据： 文件（File） 和 目录（Directory） 。

- 文件：用于保存数据。
- 目录：用于保存文件列表，当然目录也可以保存目录。

由于数据是保存在硬盘数据块中，所以文件只需要记录哪些数据块属于当前文件即可。如下图所示：



Linux内核那些事

从上图可以看出，目录中既可以保存文件，也可以保存目录。而文件中保存的是属于当前文件的数据块编号，所以当读写文件时，只需要找到文件对应的数据块进行读写即可。

三、MINIX 文件系统实现

现在，我们以 MINIX 文件系统来详细介绍文件系统的设计原理。由于 MINIX 文件系统非常简单，所以适合用于教学使用。

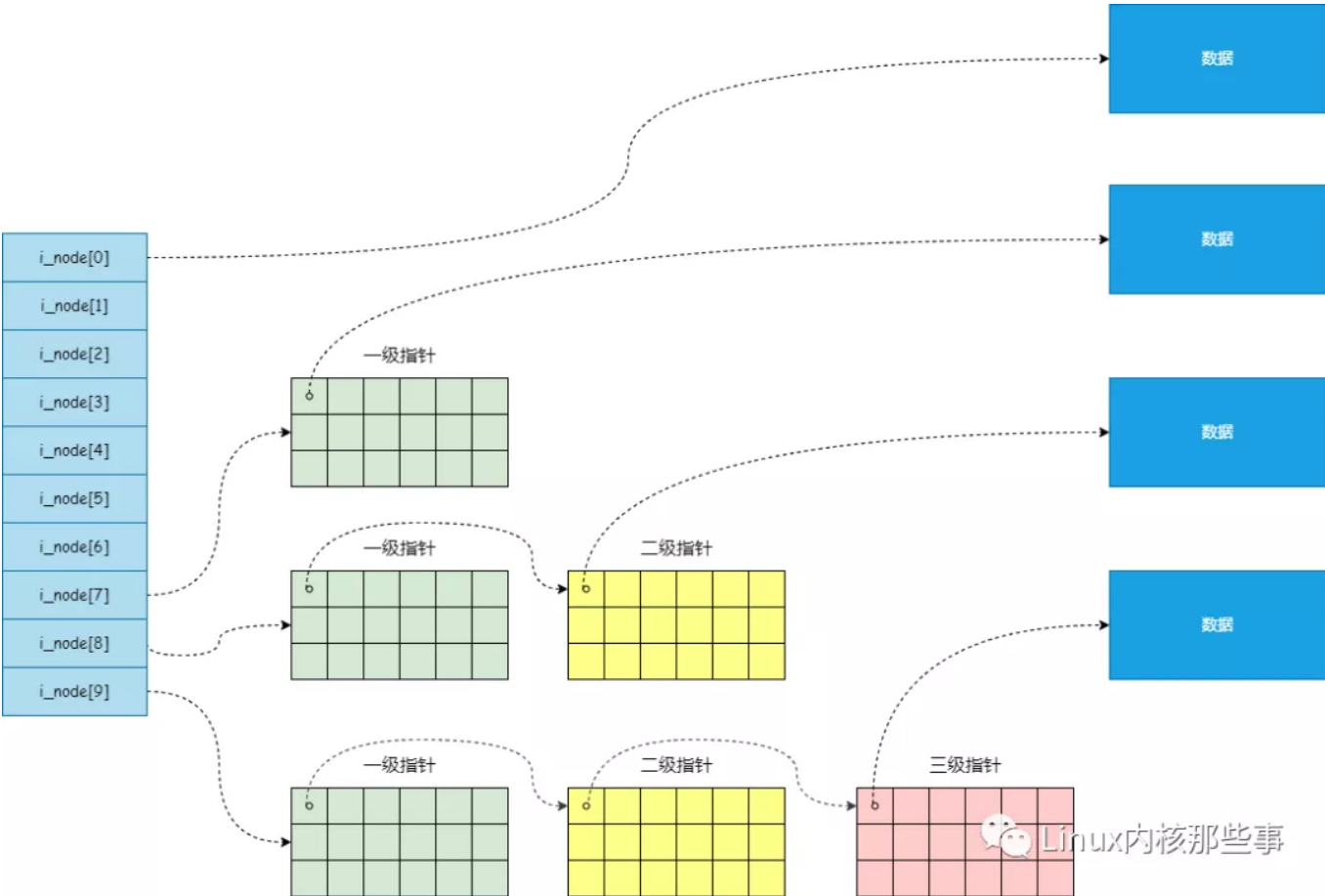
1. MINIX 文件与目录

在 MINIX 文件系统中，以 `minix2_inode` 对象来描述一个文件。我们来看看 `minix2_inode` 的定义：

```
1  struct minix2_inode {
2      __u16 i_mode;        // 模式
3      __u16 i_nlinks;      // 链接数
4      __u16 i_uid;         // 所属用户UID
5      __u16 i_gid;         // 所属组ID
6      __u32 i_size;        // 文件大小
7      __u32 i_atime;       // 访问时间
8      __u32 i_mtime;       // 修改时间
9      __u32 i_ctime;       // 创建时间
10     __u32 i_zone[10];    // 文件数对应的数据块编号
11 };
```

我们需要特别关注 `minix2_inode` 对象的 `i_zone` 字段，它就是用来记录属于当前文件的数据块编号。从定义来看，`i_zone` 是一个用于 10 个元素的整型数组，那么是否就说明 MINIX 的文件只能保存 40 KB 的数据呢？

答案是否定的，因为 MINIX 文件系统将 `i_zone` 数组分为 4 个部分：前 7 个元素直接指向保存数据的数据块编号，也就是数据会直接存储在这些数据块上，而第 8 个元素是一级间接指向，第 9 个元素是二级间接指向，第 10 个元素是三级间接指向。我们通过下图来说明这个关系：



通过这种多级指向的方式，一个 MINIX 文件就可以保存超过 40KB 的数据。

有描述文件的对象，那么也应该有描述目录的对象吧？在 MINIX 文件系统中，目录也是使用 minix2_inode 对象来描述的。那么怎么区分文件和目录呢？

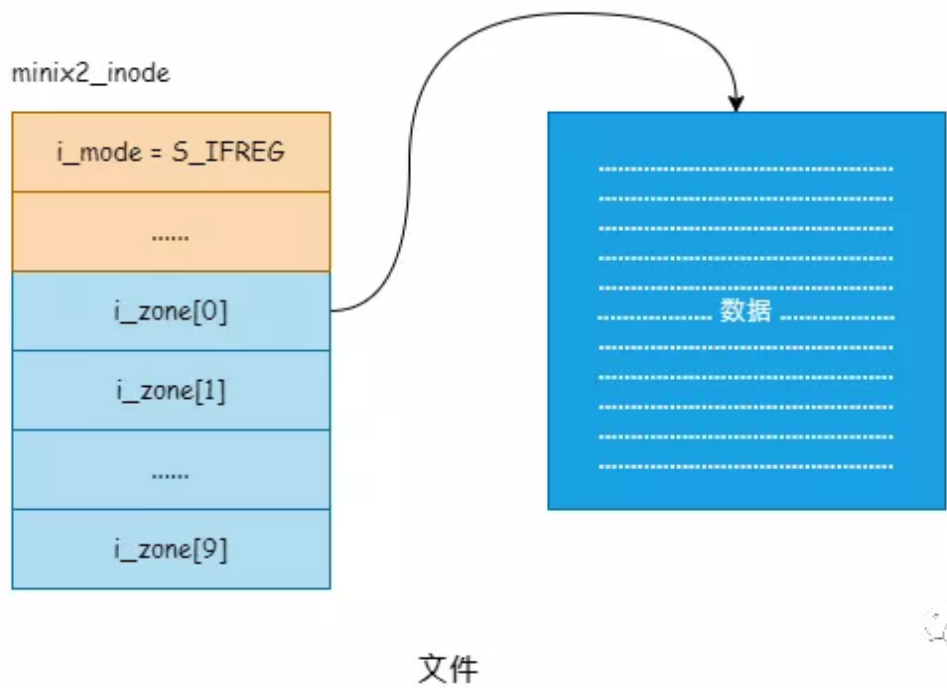
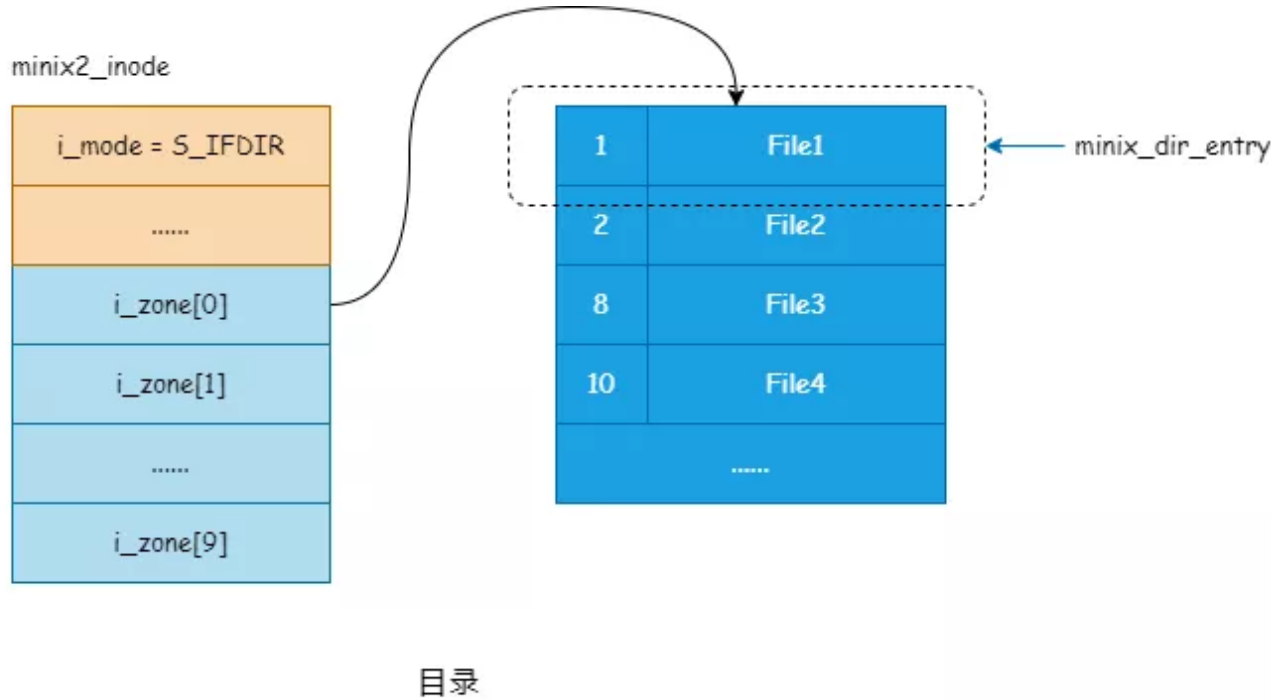
在 minix2_inode 对象中有个名为 i_mode 的字段，它保存着 minix2_inode 对应的类型，普通文件使用 S_IFREG 标志来表示，而目录使用 S_IFDIR 来表示。所以从本质来看，目录也是一种特殊的文件。

普通文件的数据块保存的是文件的数据，那么目录的数据块保存的是什么？答案就是文件列表，而文件列表的每个表项使用 minix_dir_entry 对象表示，定义如下：

```
1 struct minix_dir_entry {
2     __u16 inode;
3     char name[0];
4 };
```

- inode：当前文件对应的 minix2_inode 对象所处于 inode 数组的索引，我们暂时可以忽略此字段的作用，下面将会介绍。
- name：用于记录当前文件的文件名，由于文件名的长度是不固定的，所以这里使用了柔性数组（大小可变的数据）来表示。

我们通过下图来展示文件与目录所指向的数据内容的区别：



Linux内核那些事

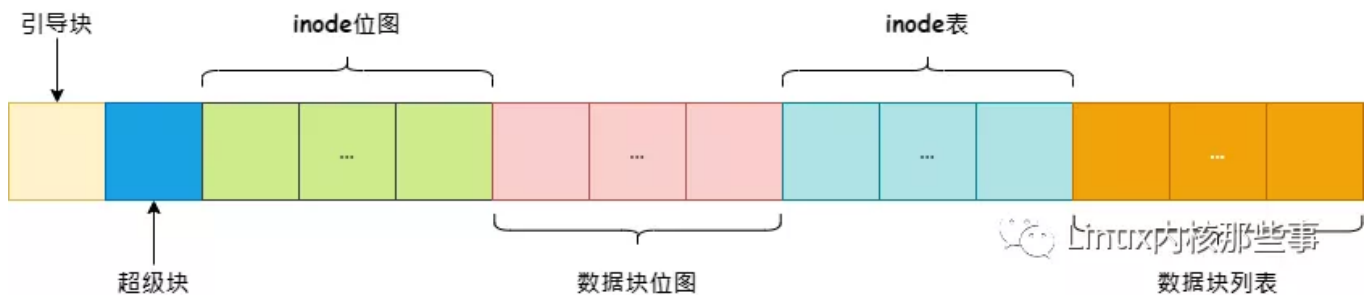
上图展示了文件与目录两个明显的区别：

- 文件的 `i_mode` 字段设置为 `S_IFREG`，而目录的 `i_mode` 字段设置为 `S_IFDIR`。
- 文件的 `i_zone` 字段指向的数据块保存的是文件的数据，而目录的 `i_zone` 字段指向的数据块保存的是文件列表。

2. MINIX 文件系统格式化

现在，我们基本了解 MINIX 文件系统对文件与目录的存储方式了，接下来我们将会介绍 MINIX 文件系统怎么管理硬盘中的文件和目录，也就是我们常说的 格式化 。

前面说过，我们可以把硬盘当成一个由数据块组成的巨大数组，那么 MINIX 文件系统会把硬盘划分为以下几个部分，如下图所示：



下面我们对这几个部分进行解说：

- 引导块：占用一个数据块，用于操作系统启动时使用，我们可以忽略。
- 超级块：占用一个数据块，用于保存文件系统的信息，MINIX 文件系统使用 `minix_super_block` 对象来保存文件系统的信息，如 `inode`位图 占用几个数据块、`数据块位图` 占用几个数据块等。
- `inode`位图：占用若干个数据块，用于描述 `inode`表中哪些成员已经被使用，每个位表示一个 `inode` 的使用情况。
- `数据块位图`：占用若干个数据块，用于描述 `数据块列表` 中哪些成员已经被使用，每个位表示一个数据块的使用情况。
- `inode`表：占用若干个数据块，由多个 `minix2_inode` 对象组成，每个 `minix2_inode` 对象表示一个文件或目录。
- `数据块列表`：占用若干个数据块，用于保存文件的数据。

上图就是 MINIX 文件系统在硬盘中的格式化结构，我们先来看看 超级块 记录的信息有哪些，超级块是由 `minix_super_block` 对象表示，其定义如下：

```

1  struct minix_super_block {
2      __u16 s_ninodes;           // inode表的元素个数
3      __u16 s_nzones;           // 数据块列表的元素个数(v1版本)
4      __u16 s_imap_blocks;       // inode位图占用的数据块数量
5      __u16 s_zmap_blocks;       // 数据块位图占用的数据块数量
6      __u16 s_firstdatazone;     // 第一个数据块起始号
7      __u16 s_log_zone_size;
8      __u32 s_max_size;          // 文件最大尺寸
9      __u16 s_magic;             // 魔数(用于识别MINIX文件系统)
10     __u16 s_state;             // 文件系统状态

```

```
11      __u32 s_zones;           // 数据块列表的元素个数(v2版本)
12  };
```

minix_super_block 每个字段的作用都在注释中进行了说明，通过 minix_super_block 对象我们可以了解到 MINIX 文件系统的信息。

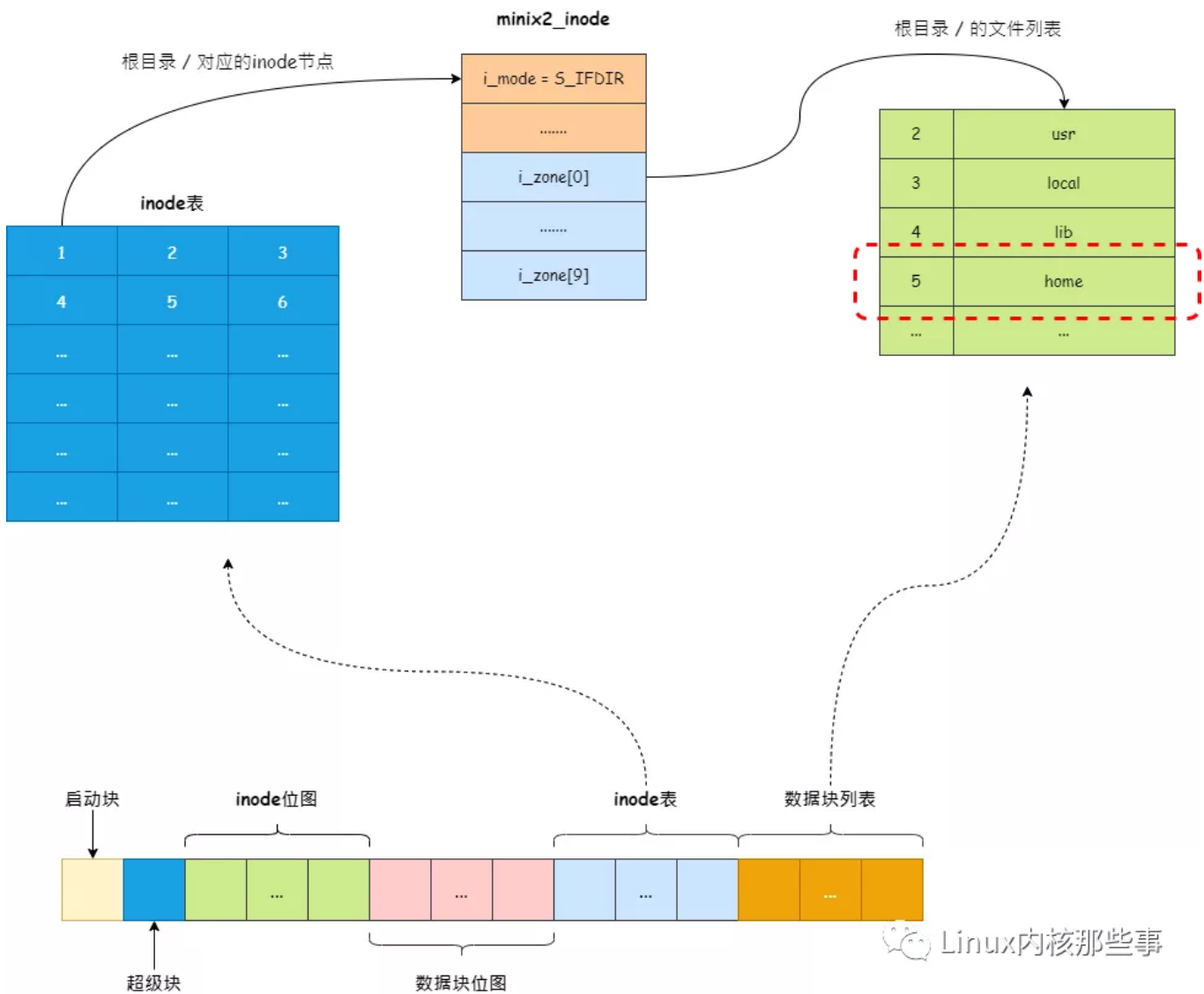
3. 读取文件过程

了解了 MINIX 文件系统的结构组织，现在我们介绍一下 MINIX 文件系统读取文件的过程。

例如，我们要读取 /home/file.txt 文件的内容，MINIX 文件系统是怎么准确地查找到文件并且读取其中的内容呢？下面我们进行分步来描述这个过程。

第一步：读取根目录

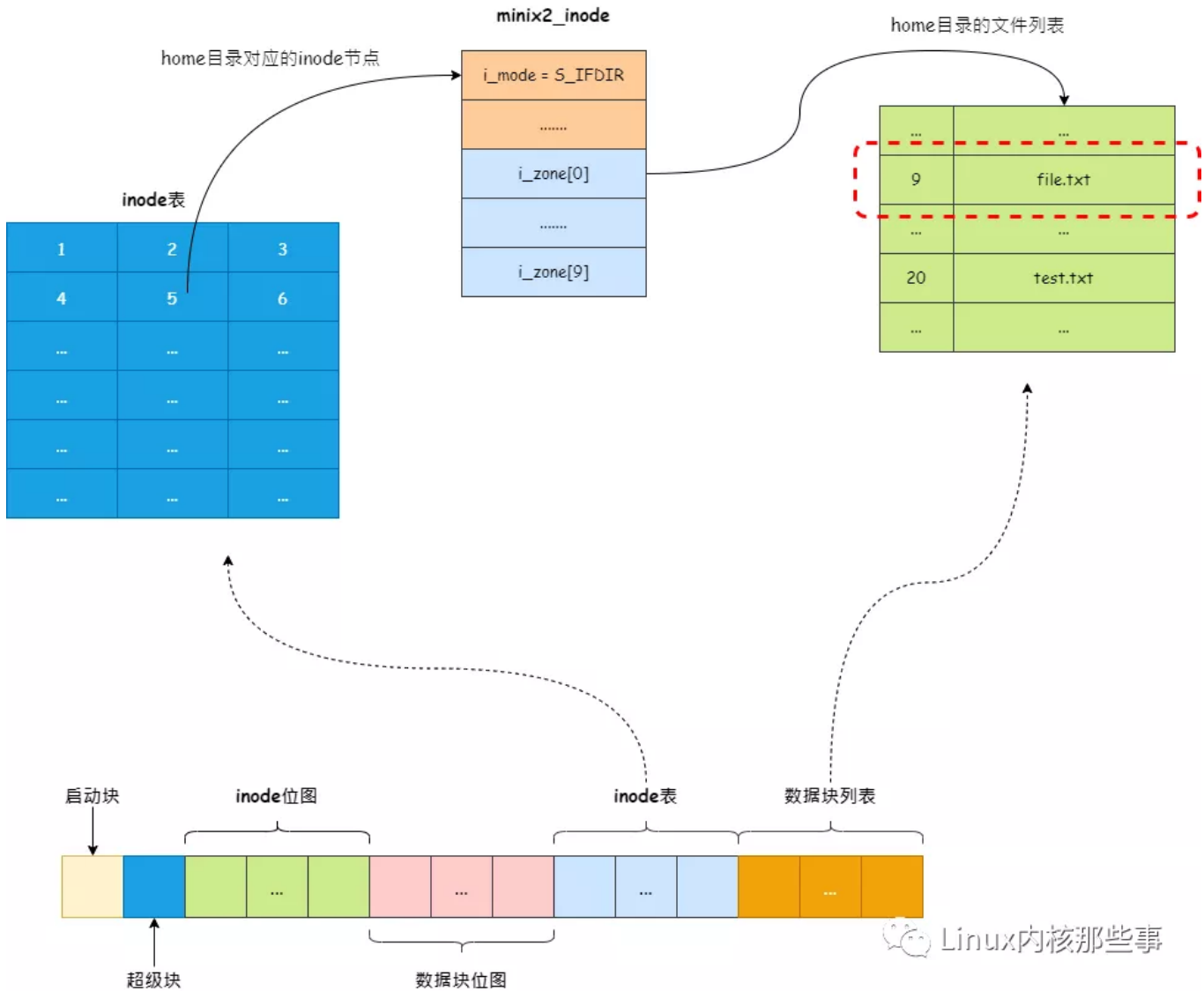
要读取 /home/file.txt 文件，首先要从根目录 / 开始，MINIX 文件系统约定根目录使用 inode表 的第一个元素进行存储。如下图：



如上图所示，根目录使用 `inode`表 的第一个元素进行存储，然后从根目录的文件列表中查找目录 `home` 。从上图可以看出， `home` 目录的 `inode`索引 为 5，表示 `home` 目录存储在 `inode`表 的第 5 个元素中。

第二步：读取 `home` 目录

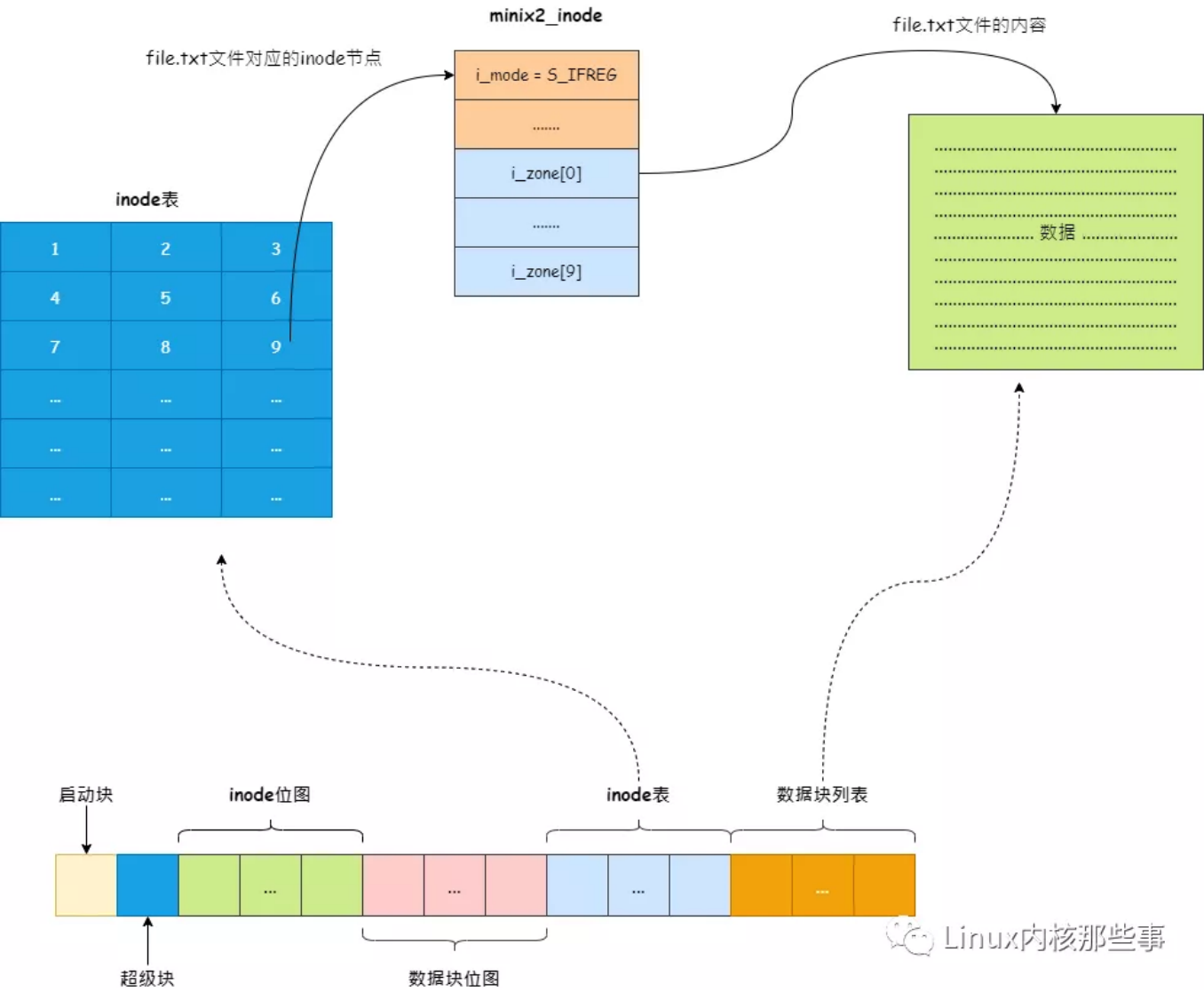
知道 `home` 目录的 `inode`索引 为 5 后，再读取 `inode`表 的第 5 个元素，然后再从 `home` 目录的文件列表中查找文件 `file.txt` ，过程如下图：



如上图所示，从 `home` 目录的文件列表中找到 `file.txt` 文件的 `inode`索引 为 9，所以现在可以通过读取 `inode`表的第 9 个元素来获得 `file.txt` 文件对应的 `inode` 节点。

第三步：读取 `file.txt` 文件的内容

现在我们已经知道了 `file.txt` 文件对应的 `inode`索引 ，所以从 `inode`表中读取第 9 个元素即可获得 `file.txt` 文件的 `inode`节点 ，然后就可以通过 `inode`节点的 `i_zone` 字段所指向的数据块来读取文件的内容，如下图所示：



Linux内核那些事

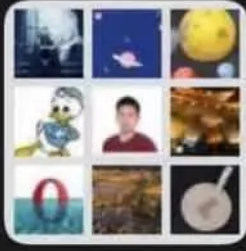
如上图所示，通过读取 `inode表` 的第 9 个元素获得 `file.txt` 文件的 `inode节点` 后，可以通过 `inode节点` 的 `i_zone` 字段所指向的数据块读取文件的内容。

另外说明一下，`inode位图` 和 `数据块位图` 用于创建文件时，快速查找哪些 `inode节点` 和 `数据块` 没有被使用的。

四、总结

本文通过 MINIX 这种简单的文件系统来介绍怎么设计一个文件系统，虽然 Linux 系统有多种文件系统，但其基本思想都是怎么有效地管理硬盘的数据。所以，掌握 MINIX 文件系统的设计对理解其他不同的文件系统有非常大的帮助。


技术交流群



Linux 内核那些事读者群



该二维码 7 天内 (4 月 14 日前) 有效, 重新进入将更新

 Linux内核那些事

喜欢此内容的人还喜欢

CPU 是如何理解 01 二进制的？

码农的荒岛求生

主板上这家伙，要当CPU和内存的中间商！

编程技术宇宙

内核级pyhon:编译python编译器和语法修改

Coding迪斯尼