

27 | 文件系统：项目成果要归档，我们就需要档案库

2019-05-29 刘超

趣谈Linux操作系统

[进入课程 >](#)



讲述：刘超

时长 15:54 大小 14.58M



咱们花了这么长的时间，规划了会议室管理系统，这样多个项目执行的时候，隔离性可以得到保证。但是，会议室里面保存的资料还是暂时的，一旦项目结束，会议室会被回收，会议室里面的资料就丢失了。有一些资料我们希望项目结束也能继续保存，这就需要有一个和项目运行生命周期无关的地方，可以永久保存，并且空间也要比会议室大的多。

文件系统的功能规划

要知道，这些资料才是咱们公司的财富，是执行多个项目积累下来的，是公司竞争力的保证，需要有一个地方归档。这就需要我们有一个存放资料的档案库，在操作系统中就是**文件系统**。我们应该如何组织规划文件系统这个档案库呢？

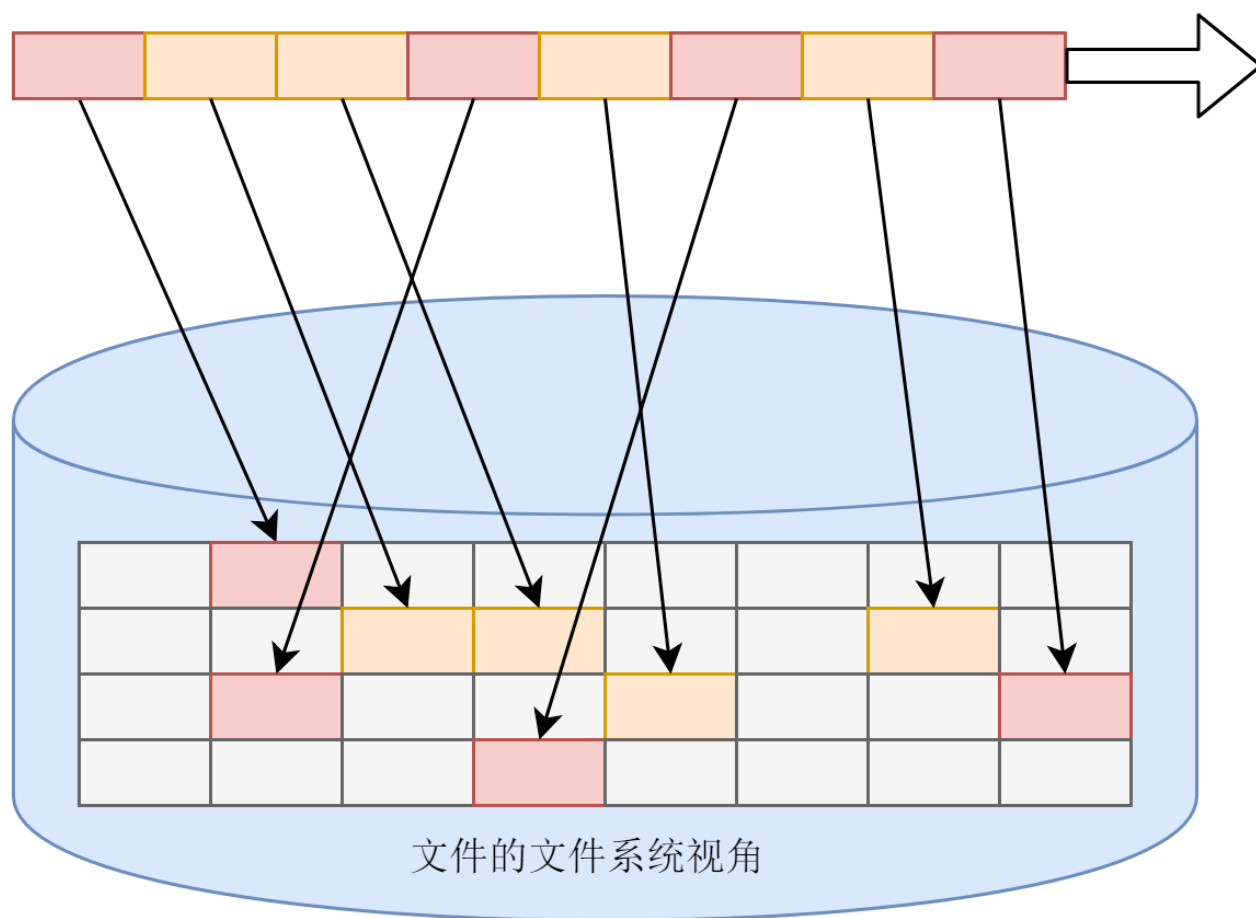
对于运行的进程来说，内存就像一个纸箱子，仅仅是一个暂存数据的地方，而且空间有限。如果我们想要进程结束之后，数据依然能够保存下来，就不能只保存在内存里，而是应该保存在外部存储中。就像图书馆这种地方，不仅空间大，而且能够永久保存。

我们最常用的外部存储就是硬盘，数据是以文件的形式保存在硬盘上的。为了管理这些文件，我们在规划文件系统的时候，需要考虑到以下几点。

第一点，文件系统要有严格的组织形式，使得文件能够以块为单位进行存储。这就像图书馆里，我们会给设置一排排书架，然后再把书架分成一个个小格子，有的项目存放的资料非常多，一个格子放不下，就需要多个格子来进行存放。我们把这个区域称为存放原始资料的仓库区。

第二点，文件系统中也要有索引区，用来方便查找一个文件分成的多个块都存放在了什么位置。这就好比，图书馆的书太多了，为了方便查找，我们需要专门设置一排书架，这里面会写清楚整个档案库有哪些资料，资料在哪个架子的哪个格子上。这样找资料的时候就不用跑遍整个档案库，在这个书架上找到后，直奔目标书架就可以了。

文件的用户视角



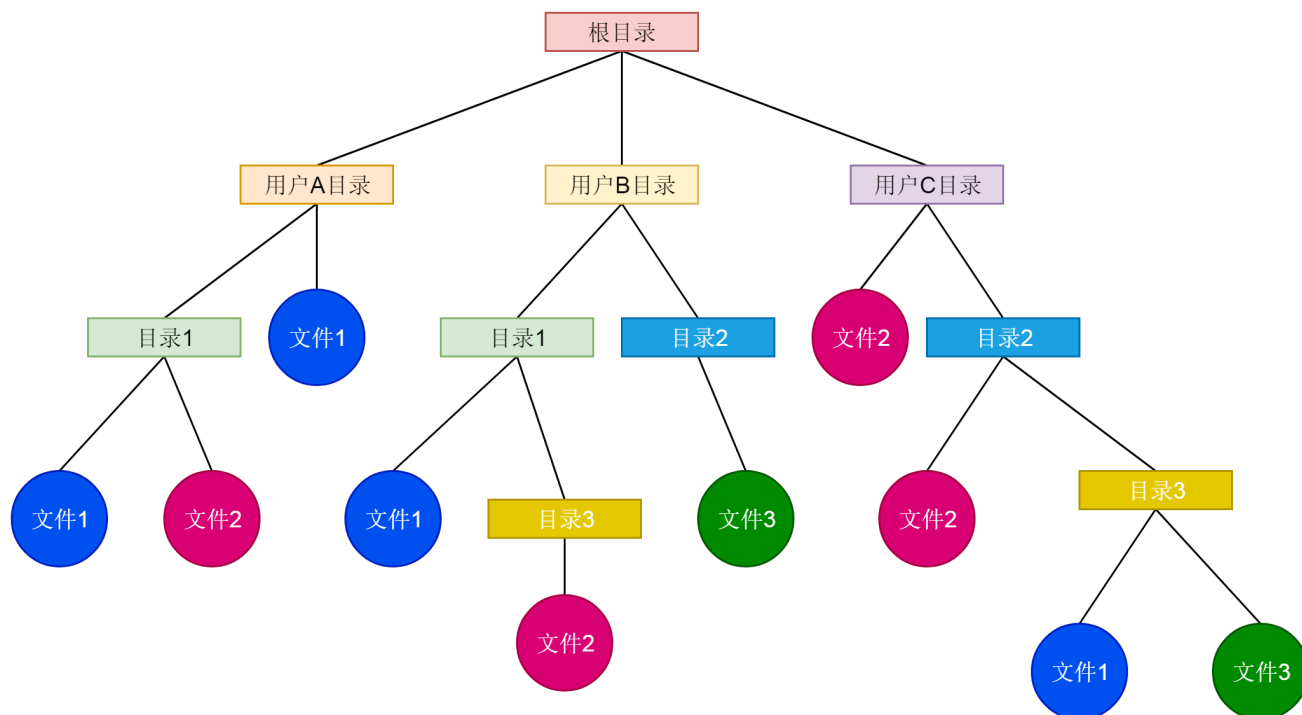
第三点，如果文件系统中有的文件是热点文件，近期经常被读取和写入，文件系统应该有缓存层。这就相当于图书馆里面的热门图书区，这里面的书都是畅销书或者是常常被借还的图书。因为借还的次数比较多，那就没有必要每次有人还了之后，还放回遥远的货架，我们可以专门开辟一个区域，放置这些借还频次高的图书。这样借还的效率就会提高。

第四点，文件应该用文件夹的形式组织起来，方便管理和查询。这就像在图书馆里面，你可以给这些资料分门别类，比如分成计算机类、文学类、历史类等等。这样你也容易管理，项目组借阅的时候只要在某个类别中去找就可以了。

在文件系统中，每个文件都有一个名字，这样我们访问一个文件，希望通过它的名字就可以找到。文件名就是一个普通的文本。当然文件名会经常冲突，不同用户取相同的名字的情况还是会经常出现的。

要想把很多的文件有序地组织起来，我们就需要把它们成为目录或者文件夹。这样，一个文件夹里可以包含文件夹，也可以包含文件，这样就形成了一种树形结构。而我们可以将不同

的用户放在不同的用户目录下，就可以一定程度上避免了命名的冲突问题。



如图所示，不同的用户的文件放在不同的目录下，虽然很多文件都叫“文件 1”，只要在不同的目录下，就不会有问题。

有了目录结构，定位一个文件的时候，我们还会分**绝对路径**（Absolute Path）和**相对路径**（Relative Path）。所谓绝对路径，就是从根目录开始一直到当前的文件，例如“/ 根目录 / 用户 A 目录 / 目录 1 / 文件 2”就是一个绝对路径。而通过 cd 命令可以改变当前路径，例如“cd / 根目录 / 用户 A 目录”，就是将用户 A 目录设置为当前目录，而刚才那个文件的相对路径就变成了“./ 目录 1 / 文件 2”。

第五点，Linux 内核要在自己的内存里面维护一套数据结构，来保存哪些文件被哪些进程打开和使用。这就好比，图书馆里会有个图书管理系统，记录哪些书被借阅了，被谁借阅了，借阅了多久，什么时候归还。

好了，这样下来，这文件系统的几个部分，是不是就很好理解、记忆了？你不用死记硬背，只要按照一个正常的逻辑去理解，自然而然就能记住了。接下来的整个章节，我们都要围绕这五点展开解析。


文件系统相关命令行

在 Linux 命令的那一节，我们学了一些简单的文件操作的命令，这里我们再来学几个常用的。

首先是**格式化**，也即将一块盘使用命令组织成一定格式的文件系统的过程。咱们买个硬盘或者 U 盘，经常说要先格式化，才能放文件，说的就是这个。

使用 Windows 的时候，咱们常格式化的格式为**NTFS** (New Technology File System)。在 Linux 下面，常用的是 ext3 或者 ext4。

当一个 Linux 系统插入了一块没有格式化的硬盘的时候，我们可以通过命令**fdisk -l**，查看格式化和没有格式化的分区。

 复制代码

```
1 # fdisk -l
2
3
4 Disk /dev/vda: 21.5 GB, 21474836480 bytes, 41943040 sectors
5 Units = sectors of 1 * 512 = 512 bytes
6 Sector size (logical/physical): 512 bytes / 512 bytes
7 I/O size (minimum/optimal): 512 bytes / 512 bytes
8 Disk label type: dos
9 Disk identifier: 0x000a4c75
10
11
12   Device Boot      Start         End      Blocks   Id  System
13  /dev/vda1  *        2048     41943006     20970479+   83   Linux
14
15
16 Disk /dev/vdc: 107.4 GB, 107374182400 bytes, 209715200 sectors
17 Units = sectors of 1 * 512 = 512 bytes
18 Sector size (logical/physical): 512 bytes / 512 bytes
19 I/O size (minimum/optimal): 512 bytes / 512 bytes
```

例如，从上面的命令的输出结果可以看出，vda 这块盘大小 21.5G，是格式化了的，有一个分区 /dev/vda1。vdc 这块盘大小 107.4G，是没有格式化的。


我们可以通过命令**mkfs.ext3**或者**mkfs.ext4**进行格式化。

 复制代码

```
1 mkfs.ext4 /dev/vdc
```

执行完这个命令后，vdc 会建立一个分区，格式化为 ext4 文件系统的格式。至于这个格式是如果组织的，我们下一节仔细讲。

当然，你也可以选择不将整块盘格式化为一个分区，而是格式化为多个分区。下面的这个命令行可以启动一个交互式程序。

 复制代码

```
1 fdisk /dev/vdc
```

在这个交互式程序中，你可以输入 **p** 来打印当前分了几个区。如果没有分过，那这个列表应该是空的。

接下来，你可以输入 **n** 新建一个分区。它会让你选择创建主分区 primary，还是扩展分区 extended。我们一般都会选择主分区 p。

接下来，它会让你输入分区号。如果原来没有分过区，应该从 1 开始。或者你直接回车，使用默认值也行。

接下来，你可以一路选择默认值，直到让你指定这个分区的大小，通过 +sizeM 或者 +sizeK 的方式，默认值是整块盘都用上。你可以输入 +5620M 分配一个 5G 的分区。这个时候再输入 p，就能看到新创建的分区了，最后输入 w，将对分区的修改写入硬盘。

分区结束之后，可能会出现 vdc1, vdc2 等多个分区，这个时候你可以 mkfs.ext3 /dev/vdc1 将第一个分区格式化为 ext3，通过 mkfs.ext4 /dev/vdc2 将第二个分区格式化为 ext4。


格式化后的硬盘，需要挂在到某个目录下面，才能作为普通的文件系统进行访问。

 复制代码

```
1 mount /dev/vdc1 / 根目录 / 用户 A 目录 / 目录 1
```


例如，上面这个命令就是将这个文件系统挂在到 “/ 根目录 / 用户 A 目录 / 目录 1” 这个目录下面。一旦挂在过去，“/ 根目录 / 用户 A 目录 / 目录 1” 这个目录下面原来的文件 1 和文件 2 就都看不到了，换成了 vdc1 这个硬盘里面的文件系统的根目录。

有挂载就有卸载，卸载使用 **umount** 命令。

 复制代码

```
1 umount / 根目录 / 用户 A 目录 / 目录 1
```

前面我们讲过，Linux 里面一切都是文件，那从哪里看出是什么文件呢？要从 `ls -l` 的结果的第一位标识位看出来。

- 表示普通文件；


d 表示文件夹；

c 表示字符设备文件，这在设备那一节讲解；

b 表示块设备文件，这也在设备那一节讲解；

s 表示套接字 socket 文件，这在网络那一节讲解；

l 表示符号链接，也即软链接，就是通过名字指向另外一个文件，例如下面的代码，instance 这个文件就是指向了 /var/lib/cloud/instances 这个文件。软链接的机制我们这一章会讲解。

 复制代码

```
1 # ls -l
2 lrwxrwxrwx 1 root root 61 Dec 14 19:53 instance -> /var/lib/cloud/instances
```

文件系统相关系统调用

看完了命令行，我们来看一下，如何使用系统调用在操作文件？我们先来看一个完整的例子。

 复制代码

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5
6
7 int main(int argc, char *argv[])
8 {
9
10
11     int fd = -1;
12     int ret = 1;
13     int buffer = 1024;
14     int num = 0;
15
16
17     if((fd=open("./test", O_RDWR|O_CREAT|O_TRUNC))== -1)
18     {
19         printf("Open Error\n");
20         exit(1);
21     }
22
23
24     ret = write(fd, &buffer, sizeof(int));
25     if( ret < 0)
26     {
27         printf("write Error\n");
28         exit(1);
29     }
30     printf("write %d byte(s)\n",ret);
31
32
33     lseek(fd, 0L, SEEK_SET);
34     ret= read(fd, &num, sizeof(int));
35     if(ret== -1)
36     {
37         printf("read Error\n");
38         exit(1);
39     }
40     printf("read %d byte(s), the number is %d\n", ret, num);
41
42
43     close(fd);
44
45
46     return 0;
47 }
```


当使用系统调用 `open` 打开一个文件时，操作系统会创建一些数据结构来表示这个被打开的文件。下一节，我们就会看到这些。为了能够找到这些数据结构，在进程中，我们会为这个打开的文件分配一个文件描述符 `fd` (File Descriptor)。

文件描述符，就是用来区分一个进程打开的多个文件的。它的作用域就是当前进程，出了当前进程这个文件描述符就没有意义了。`open` 返回的 `fd` 必须记录好，我们对这个文件的所有操作都要靠这个 `fd`，包括最后关闭文件。

在 `Open` 函数中，有一些参数：

`O_CREAT` 表示当文件不存在，创建一个新文件；

`O_RDWR` 表示以读写方式打开；

`O_TRUNC` 表示打开文件后，将文件的长度截断为 0。

接下来，`write` 要用于写入数据。第一个参数就是文件描述符，第二个参数表示要写入的数据存放位置，第三个参数表示希望写入的字节数，返回值表示成功写入到文件的字节数。


`lseek` 用于重新定位读写的位置，第一个参数是文件描述符，第二个参数是重新定位的位置，第三个参数是 `SEEK_SET`，表示起始位置为文件头，第二个参数和第三个参数合起来表示将读写位置设置为从文件头开始 0 的位置，也即从头开始读写。

`read` 用于读取数据，第一个参数是文件描述符，第二个参数是读取来的数据存到指向的空间，第三个参数是希望读取的字节数，返回值表示成功读取的字节数。

最终，`close` 将关闭一个文件。

对于命令行来讲，通过 `ls` 可以得到文件的属性，使用代码怎么办呢？

我们有下面三个函数，可以返回与打开的文件描述符相关的文件状态信息。这个信息将会写到类型为 `struct stat` 的 `buf` 结构中。

 复制代码

```
1 int stat(const char *pathname, struct stat *statbuf);
2 int fstat(int fd, struct stat *statbuf);
3 int lstat(const char *pathname, struct stat *statbuf);
4
```


```

5
6 struct stat {
7     dev_t      st_dev;          /* ID of device containing file */
8     ino_t      st_ino;          /* Inode number */
9     mode_t     st_mode;         /* File type and mode */
10    nlink_t     st_nlink;        /* Number of hard links */
11    uid_t       st_uid;          /* User ID of owner */
12    gid_t       st_gid;          /* Group ID of owner */
13    dev_t       st_rdev;         /* Device ID (if special file) */
14    off_t       st_size;         /* Total size, in bytes */
15    blksize_t   st_blksize;      /* Block size for filesystem I/O */
16    blkcnt_t    st_blocks;       /* Number of 512B blocks allocated */
17    struct timespec st_atim;     /* Time of last access */
18    struct timespec st_mtim;     /* Time of last modification */
19    struct timespec st_ctim;     /* Time of last status change */
20 };

```

函数 `stat` 和 `lstat` 返回的是通过文件名查到的状态信息。这两个方法区别在于，`stat` 没有处理符号链接（软链接）的能力。如果一个文件是符号链接，`stat` 会直接返回它所指向的文件的属性，而 `lstat` 返回的就是这个符号链接的内容，`fstat` 则是通过文件描述符获取文件对应的属性。

接下来我们来看，如何使用系统调用列出一个文件夹下面的文件以及文件的属性。

 复制代码

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6  #include <sys/stat.h>
7  #include <dirent.h>
8
9
10 int main(int argc, char *argv[])
11 {
12     struct stat sb;
13     DIR *dirp;
14     struct dirent *direntp;
15     char filename[128];
16     if ((dirp = opendir("/root")) == NULL) {
17         printf("Open Directory Error%s\n");
18         exit(1);
19     }
20     while ((direntp = readdir(dirp)) != NULL){

```

```

21     sprintf(filename, "/root/%s", direntp->d_name);
22     if (lstat(filename, &sb) == -1)
23     {
24         printf("lstat Error%s\n");
25         exit(1);
26     }
27
28
29     printf("name : %s, mode : %d, size : %d, user id : %d\n", direntp->d_name, sb.st_mode,
30
31
32     }
33     closedir(dirp);
34
35
36     return 0
37 }

```



`opendir` 函数打开一个目录名所对应的 DIR 目录流。并返回指向 DIR 目录流的指针。流定位在 DIR 目录流的第一个条目。

`readdir` 函数从 DIR 目录流中读取一个项目，返回的是一个指针，指向 `dirent` 结构体，且流的自动指向下一个目录条目。如果已经到流的最后一个条目，则返回 `NULL`。

`closedir()` 关闭参数 `dir` 所指的目录流。

到这里，你应该既会使用系统调用操作文件，也会使用系统调用操作目录了。下一节，我们开始来看内核如何实现的。

总结时刻

这一节，我们对于文件系统的主要功能有了一个总体的印象，我们通过下面这张图梳理一下。

在文件系统上，需要维护文件的严格的格式，要通过 `mkfs.ext4` 命令来格式化为严格的格式。

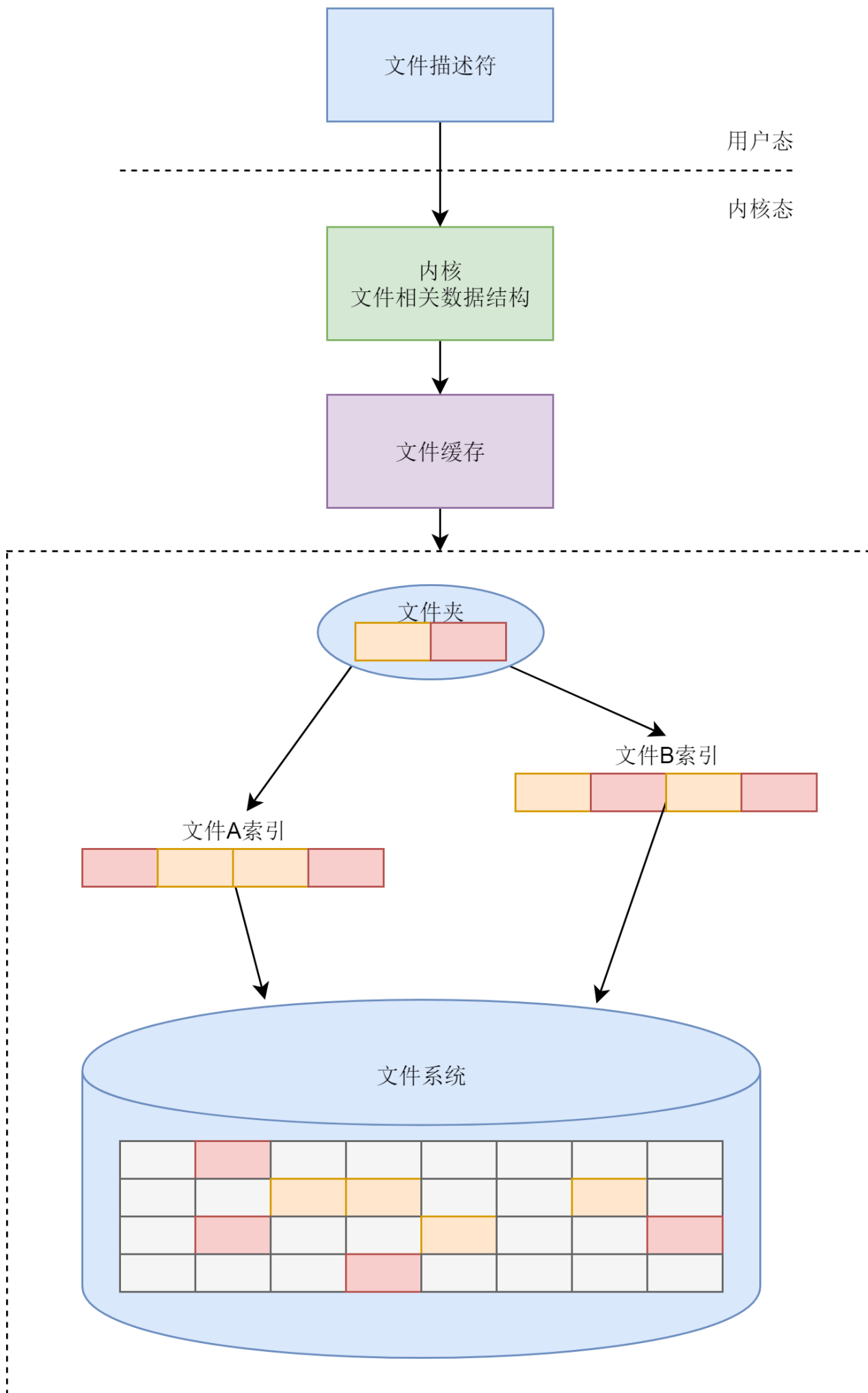
每一个硬盘上保存的文件都要有一个索引，来维护这个文件上的数据块都保存在哪里。

文件通过文件夹组织起来，可以方便用户使用。

为了能够更快读取文件，内存里会分配一块空间作为缓存，让一些数据块放在缓存里面。

在内核中，要有一整套的数据结构来表示打开的文件。

在用户态，每个打开的文件都有一个文件描述符，可以通过各种文件相关的系统调用，操作这个文件描述符。



课堂练习

你可以试着将一块空闲的硬盘，分区成为两块，并安装不同的文件系统，进行挂载。这是 Linux 运维人员经常做的一件事情。

欢迎留言和我分享你的疑惑和见解，也欢迎你收藏本节内容，反复研读。你也可以把今天的内容分享给你的朋友，和他一起学习、进步。

 极客时间

趣谈 Linux 操作系统

像故事一样的操作系统入门课

刘超

网易杭州研究院
云计算技术部首席架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 26 | 内核态内存映射：如何找到正确的会议室？

下一篇 28 | 硬盘文件系统：如何最合理地组织档案库的文档？

精选留言 (8)

 写留言



权奥

2019-05-30

👍 1

fdisk有个坑是最大只能分2T，对于超过2T的分区需求可以使用parted

展开 ▾



活的潇洒

2019-05-29

👍 1

决心从头把计算机所有的基础课程全部补上，夯实基础，一定要坚持到最后
day27笔记: <https://www.cnblogs.com/luoahong/p/10943864.html>



Mr.差不多

2019-05-29

👍 1

您好，老师想问一下，通过程序获得文件夹下面所有文件，以什么作为排序的准则返回呢？还是随机的？



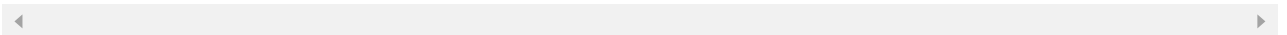
Sharry

2019-05-29

👍 1

在 Android NDK 源码中多处能够看虚拟文件系统这样的名词, 期待后面的课程

作者回复: 是的，也是基于Linux



烈日融雪

2019-05-30

👍

本期内容理解度很高，是我看这栏目以来最轻松的一期， 😊

展开 ▾



jkhcw

2019-05-30

👍

Linux文件索引采用的是哪种数据结构？红黑树还是B+树

展开 ▾



Zj_scute

2019-05-29

👍

终于有一讲看得轻松一点了，哈哈

展开 ∨



why

2019-05-29



- 文件系统的功能

- 以块为单位的存储组织形式
- 要有索引, 方便查找
- 热点文件应该有缓存
- 可以以文件夹形式组织, 方便管理...

展开 ∨