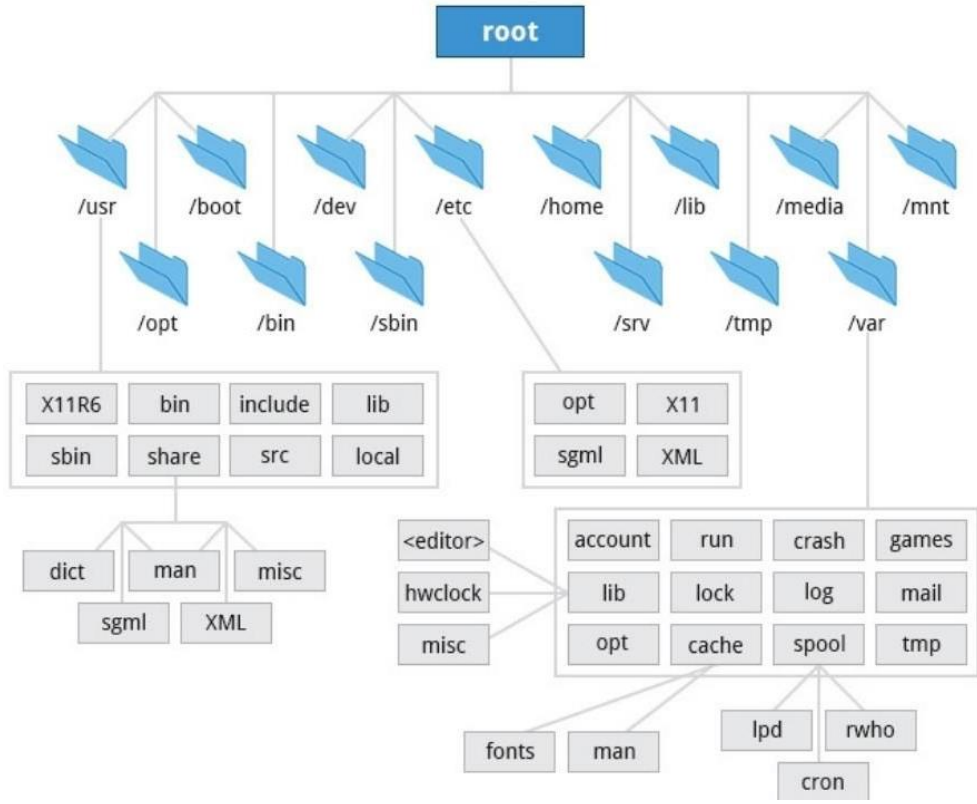
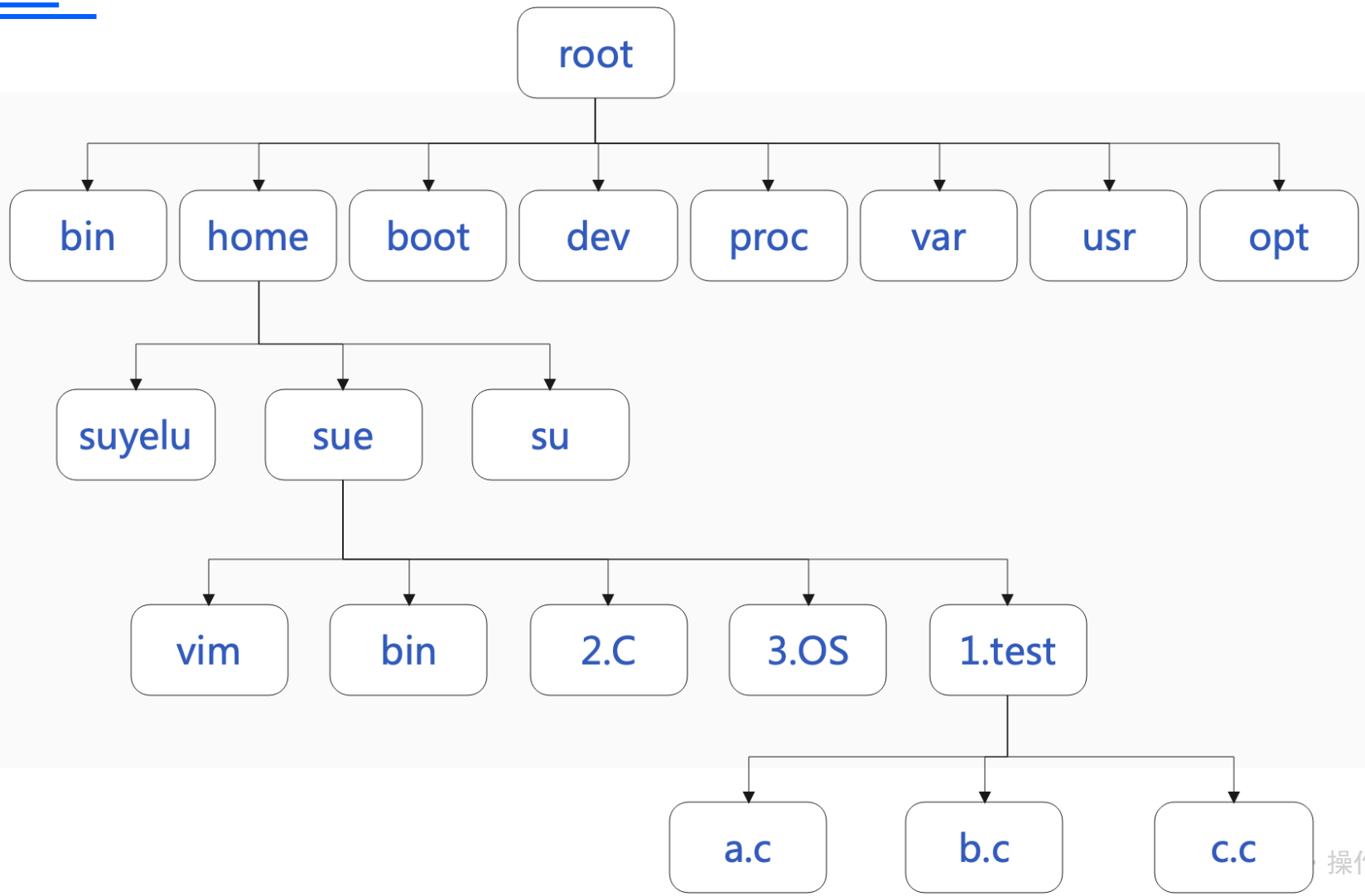
操作系统编程·实战

文件及目录

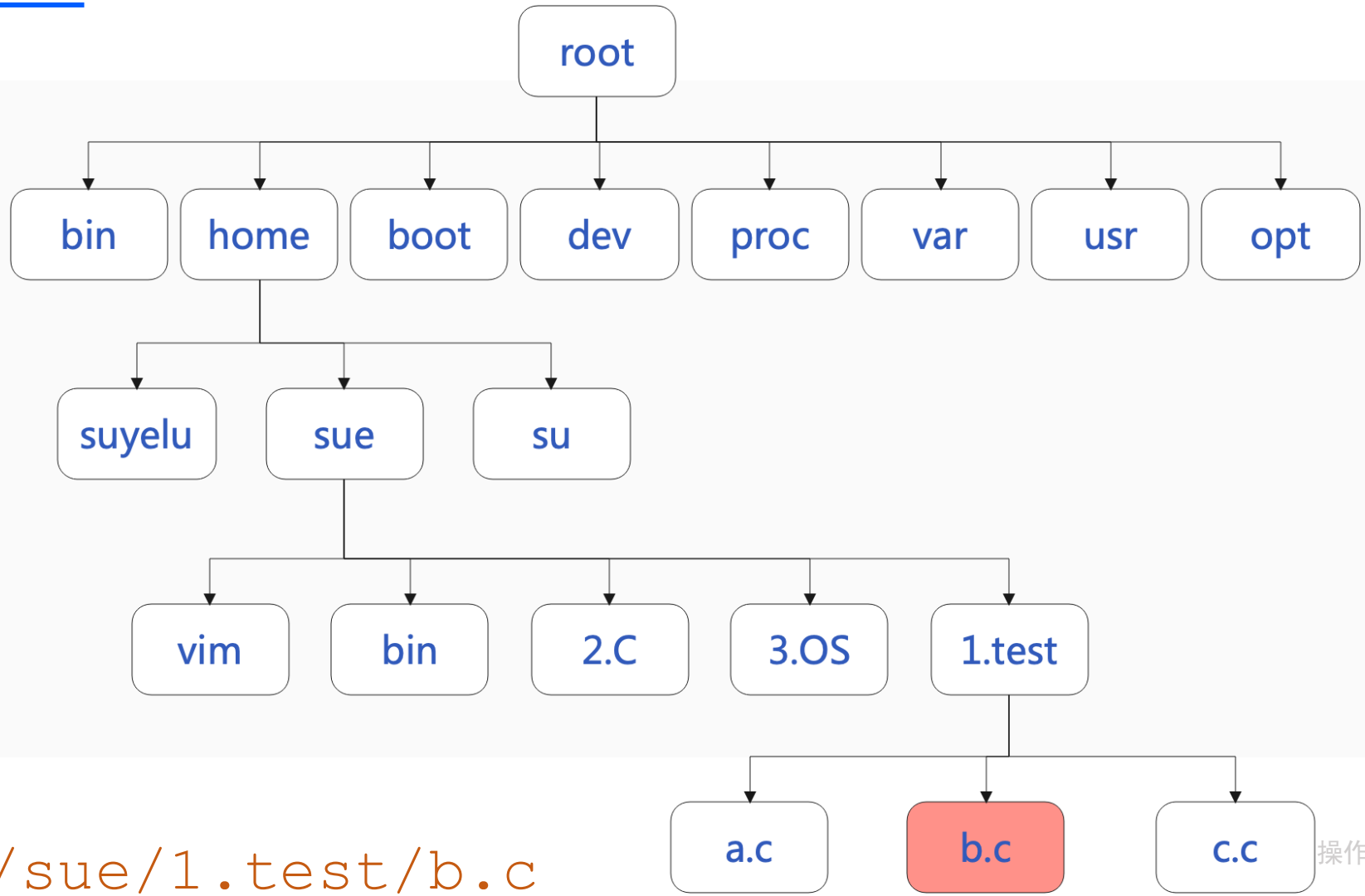
目录树


root
/usr
/boot
/dev
/etc
/home
/lib
/media
/mnt
/opt
/bin
/sbin
/srv
/tmp
/var
X11R6
bin
include
lib
sbin
share
src
local
opt
X11
sgml
XML
dict
man
misc
sgml
XML
<editor>
hwclock
misc
account
run
crash
games
lib
lock
log
mail
opt
cache
spool
tmp
fonts
man
lpd
rwho
cron

在系统中如何查找一个文件

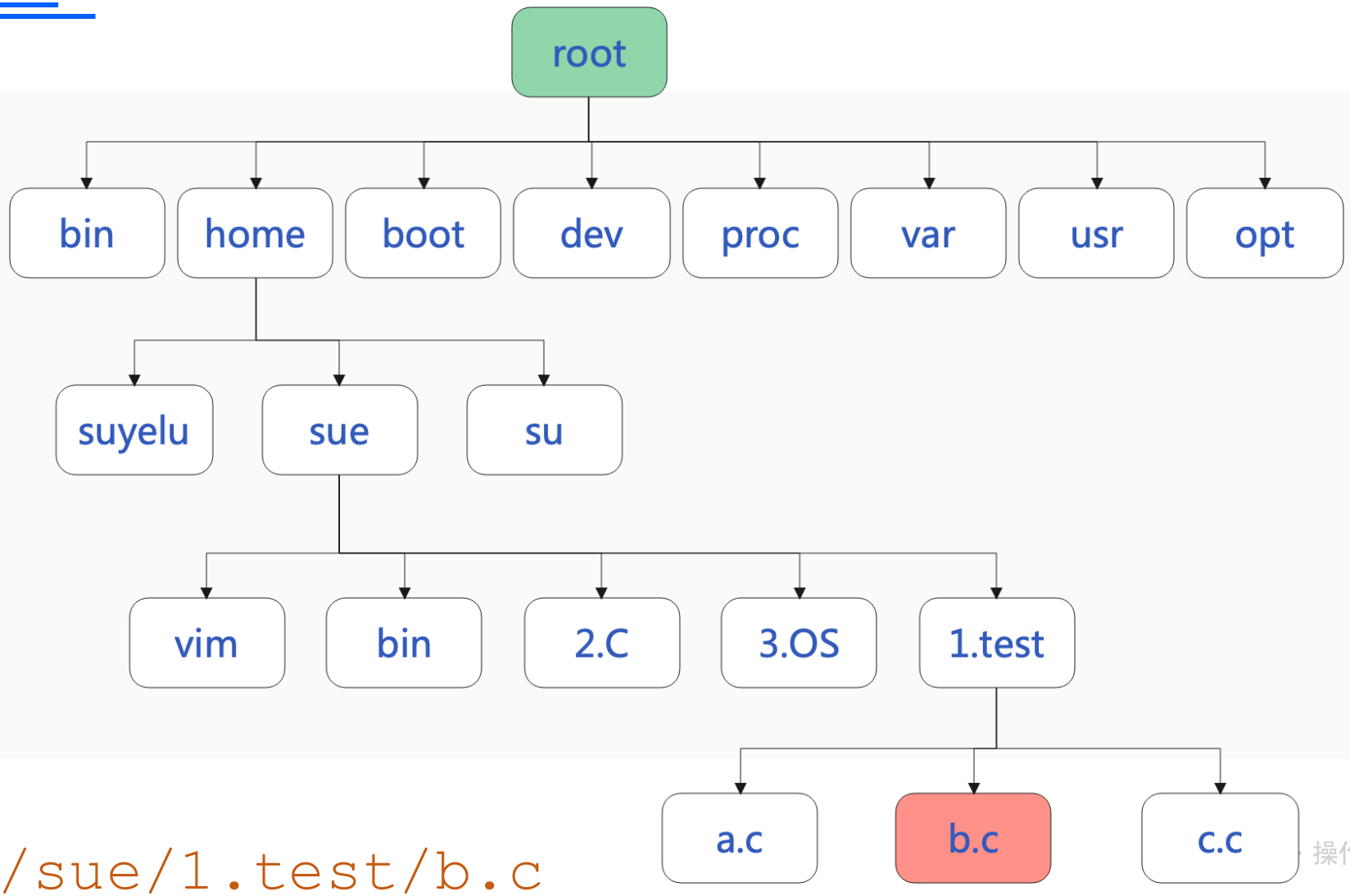


在系统中如何查找一个文件



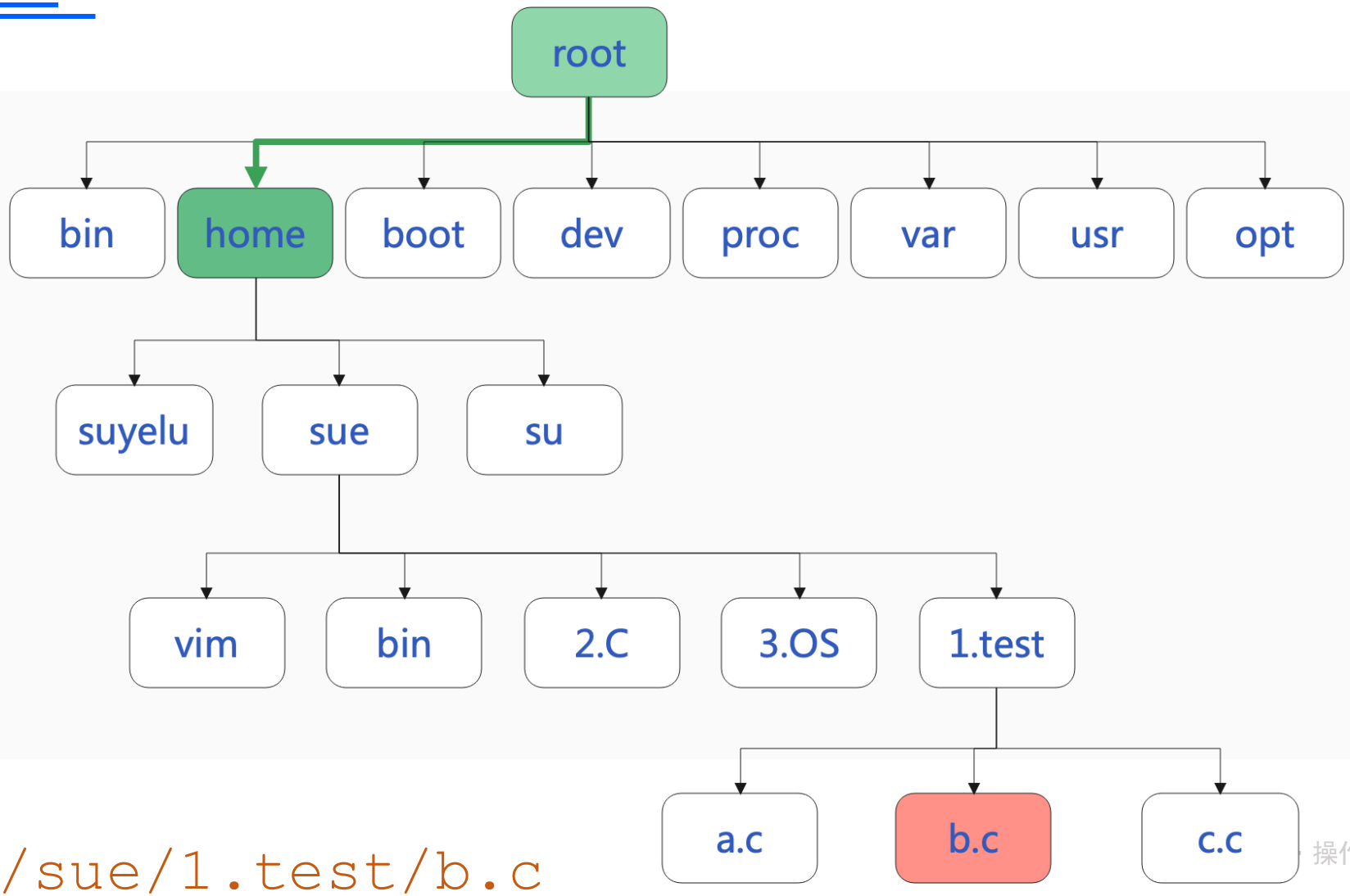
/home/sue/1.test/b.c

在系统中如何查找一个文件



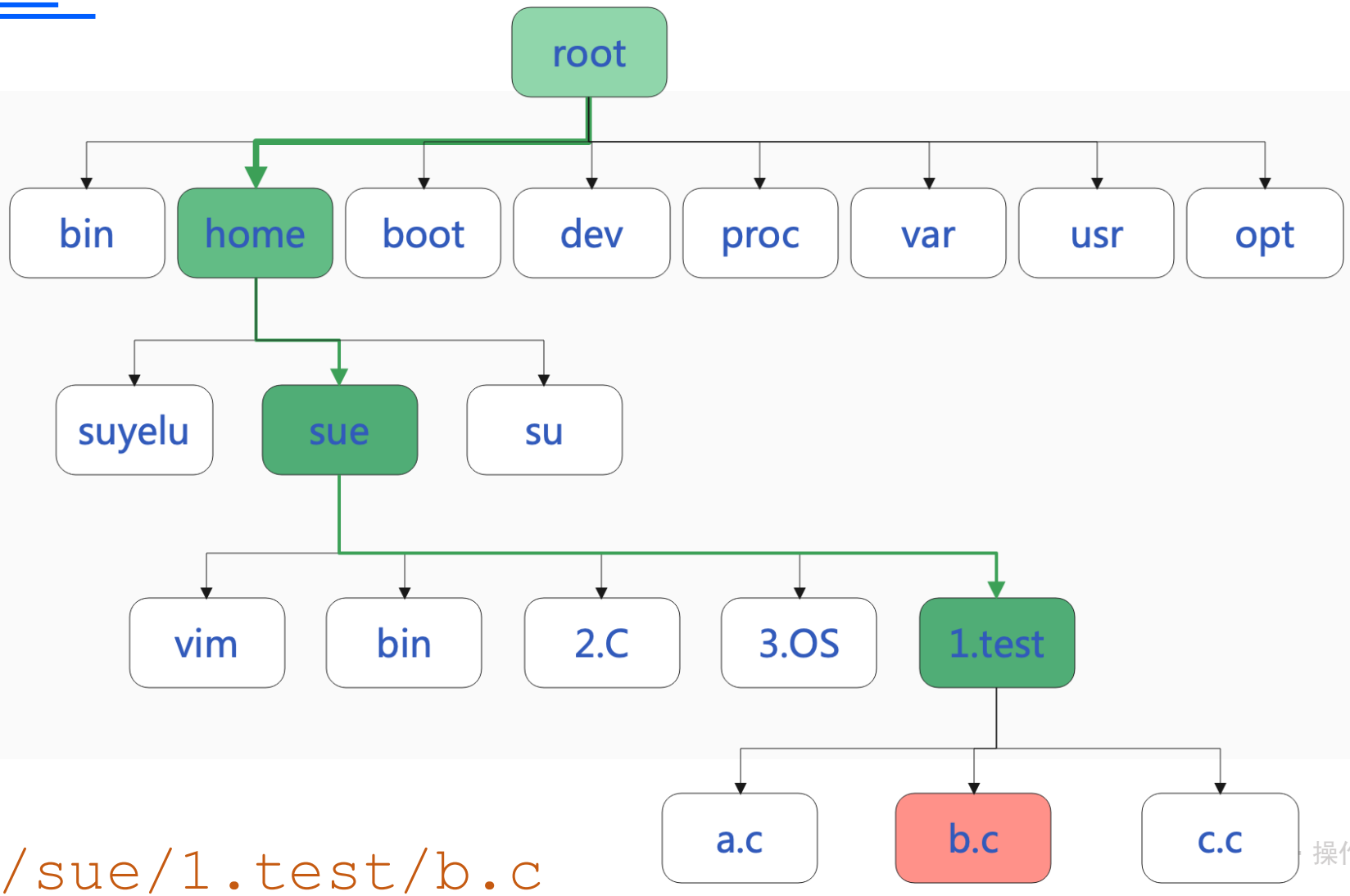
/home/sue/1.test/b.c

在系统中如何查找一个文件



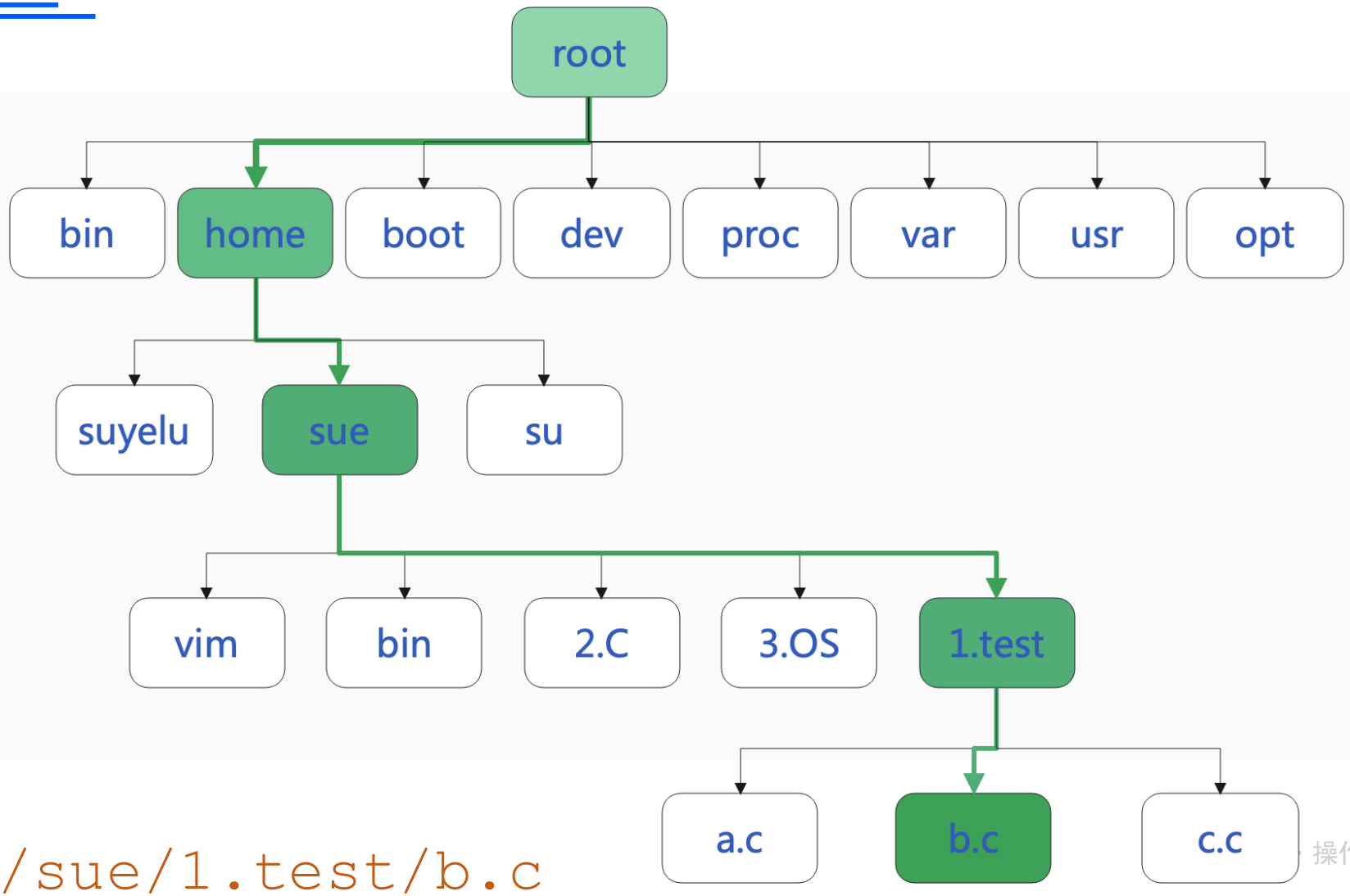
/home/sue/1.test/b.c

在系统中如何查找一个文件



/home/sue/1.test/b.c

在系统中如何查找一个文件



/home/sue/1.test/b.c

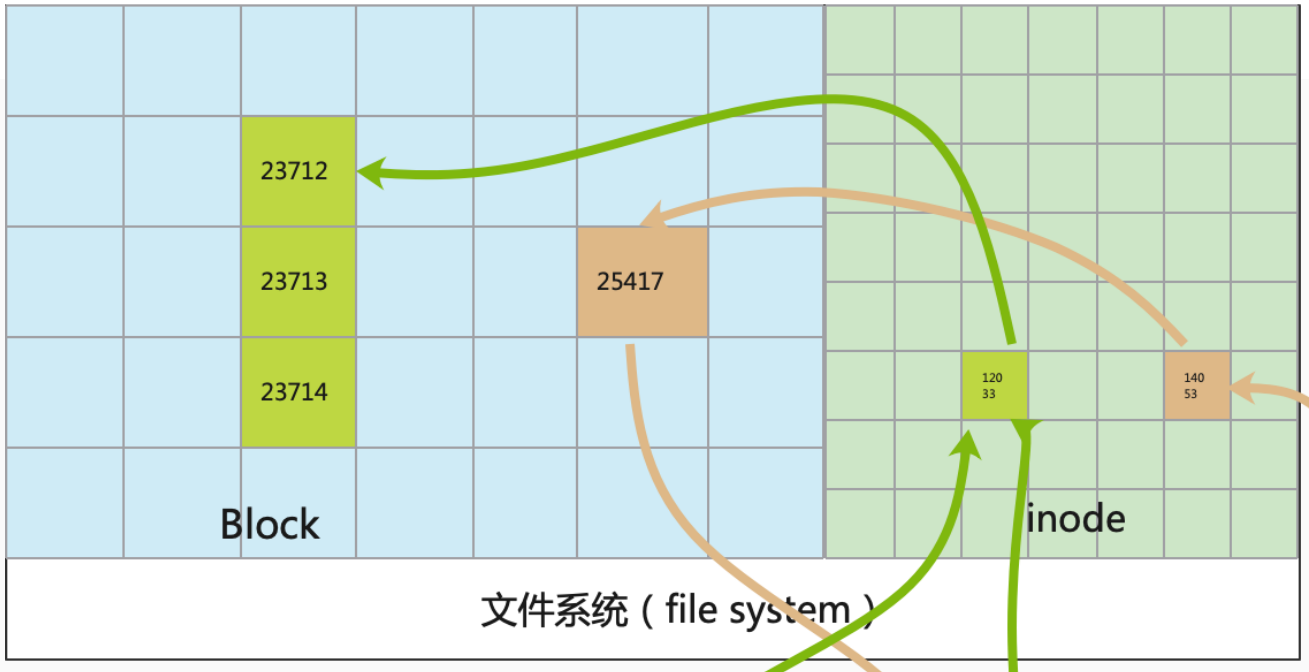
查找文件的细节

1. 每个进程都知道两个特殊目录的具体位置
    1. 当前目录（上下文工作目录）
    2. 根目录
2. 每个目录都有一张表，存储着当前文件夹下的文件列表
    1. 样式：

文件名	inode

3. inode中存储着文件的元数据和文件在外存中地址
    1. 元数据包括：文件类型、权限、所有者、大小、时间等
    2. 根据外存地址可以读取文件真实内容

文件系统中的FCB



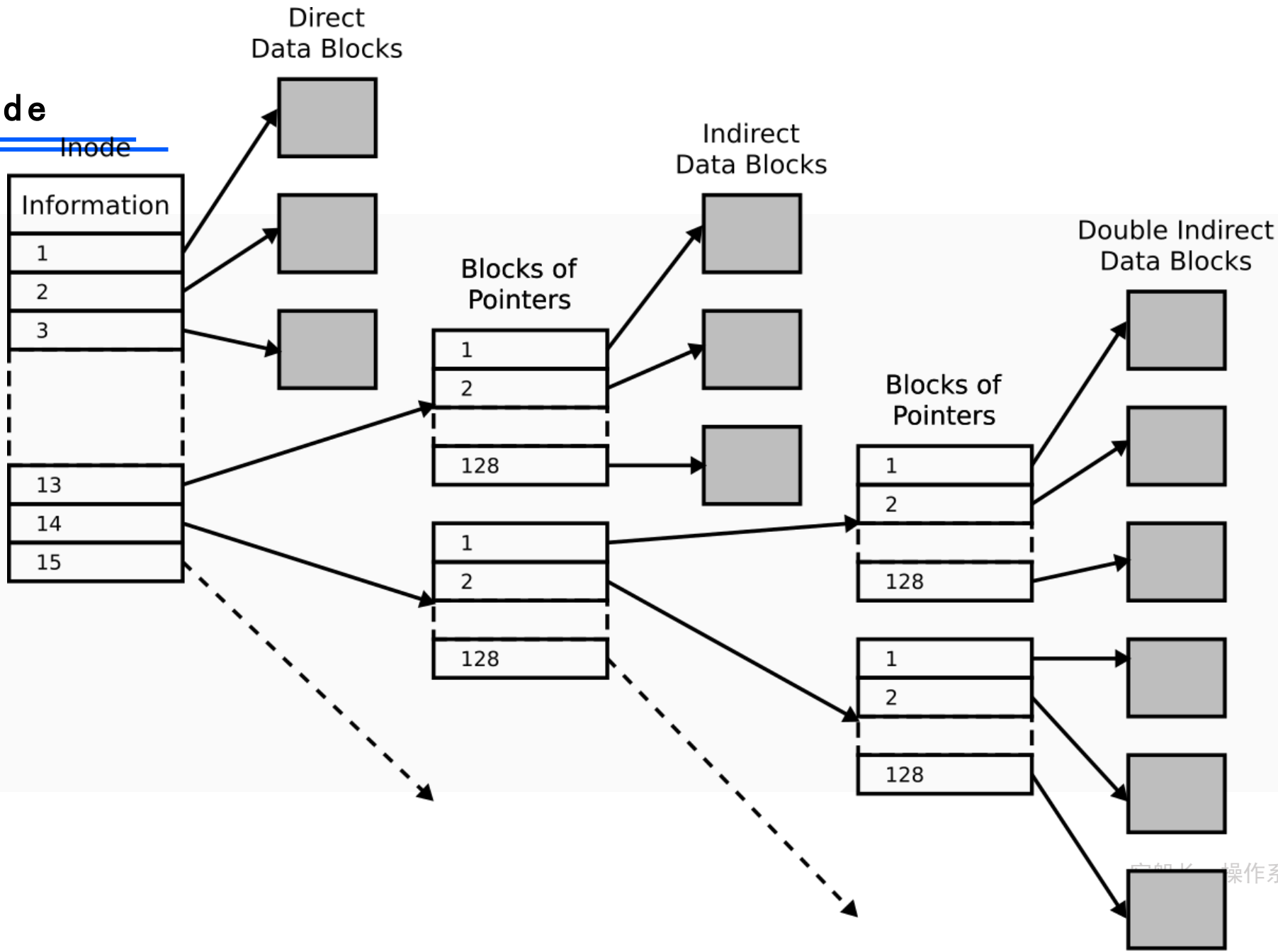
文件及路径	Inode
/home/haizeix/File A	12033
/home/zhangsan/File B	12033
/tmp/File C	14053

Inode	Blocks
12033	23712, 23713, 23714
14053	25417

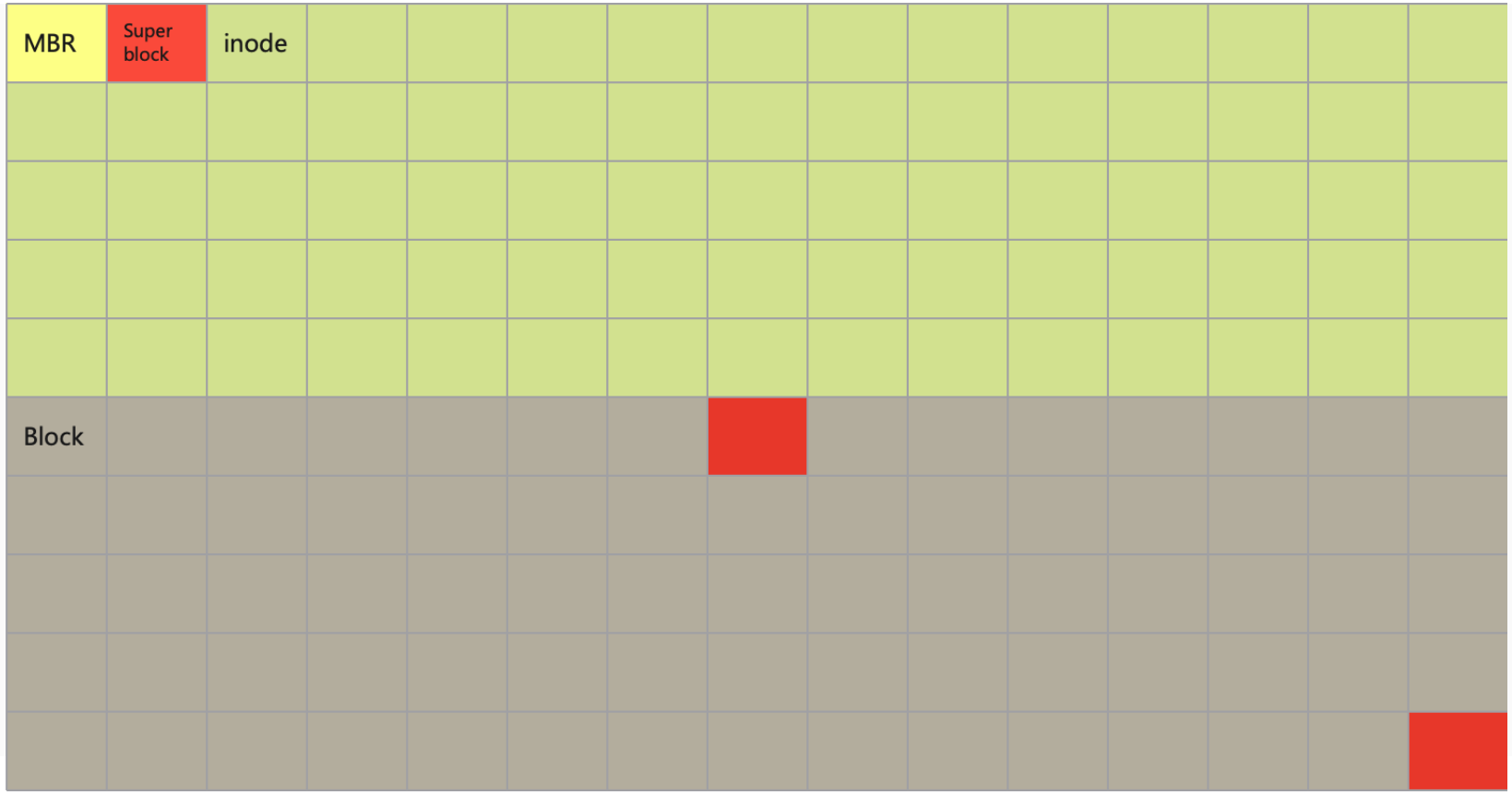
Hard Link of each other

Soft Link of File A

inode


Inode
Information
1
2
3
13
14
15
Direct
Data Blocks
Indirect
Data Blocks
Blocks of
Pointers
1
2
128
1
2
128
Double Indirect
Data Blocks
Blocks of
Pointers
1
2
128
1
2
128

inode、block和super block



ls命令

文件类型及权限字符串

链接数

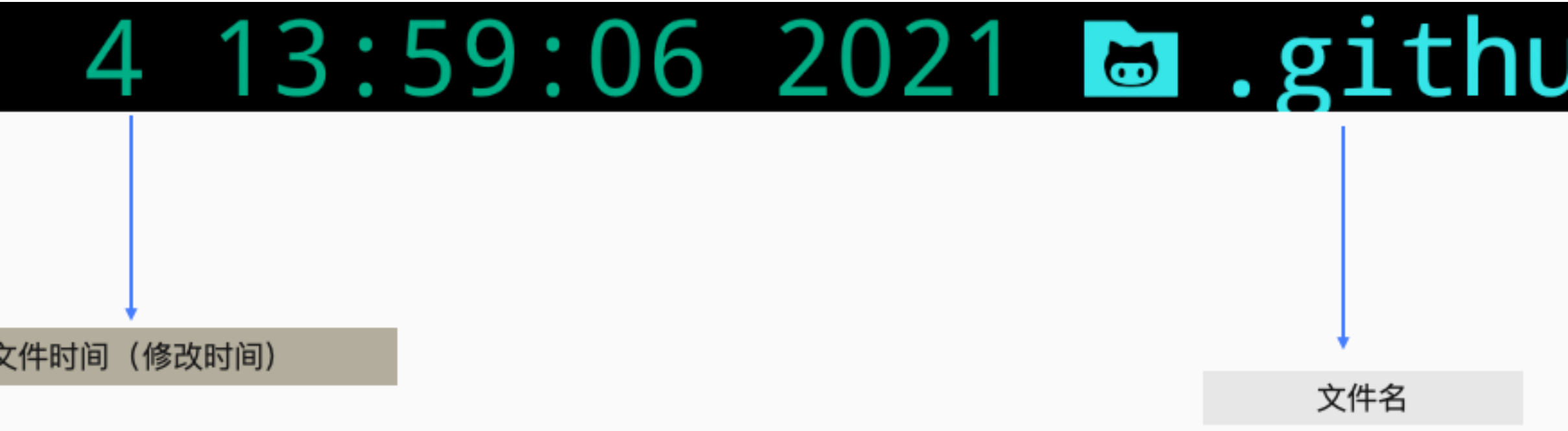
文件所属用户

文件所属组

Type	User			Group			Other		
d	r	w	x	r	-	x	r	-	x
	1	1	1	1	0	1	1	0	1
	2^2	2^1	2^0	2^2	0	2^0	2^2	0	2^0
	4	2	1	4	0	1	4	0	1
	7			5			5		

ls命令

ls命令



两个系统调用

```
DIR *opendir(const char *name);
```

功能说明：

打开一个目录，并返回目录流。

```
struct dirent *readdir(DIR *dirp);
```

功能说明：

从目录流中读取下一个目录项。

实战：输出一个目录中的所有文件名

文件名：`list_file.c`

传入参数：目录，缺省为当前目录

输出：输出参数对应目录中的所有文件及目录

查找文件的细节

1. 每个进程都知道两个特殊目录的具体位置
    1. 当前目录（上下文工作目录）
    2. 根目录
2. 每个目录都有一张表，存储着当前文件夹下的文件列表
    1. 样式：

文件名	inode

3. inode中存储着文件的元数据和文件在外存中地址
    1. 元数据包括：文件类型、权限、所有者、大小、时间等
    2. 根据外存地址可以读取文件真实内容

```
struct stat {
    dev_t     st_dev;         /* ID of device containing file */
    ino_t     st_ino;         /* Inode number */
    mode_t    st_mode;        /* File type and mode */
    nlink_t   st_nlink;       /* Number of hard links */
    uid_t     st_uid;         /* User ID of owner */
    gid_t     st_gid;         /* Group ID of owner */
    dev_t     st_rdev;        /* Device ID (if special file) */
    off_t     st_size;        /* Total size, in bytes */
    blksize_t st_blksize;     /* Block size for filesystem I/O */
    blkcnt_t  st_blocks;      /* Number of 512B blocks allocated */

    /* Since Linux 2.6, the kernel supports nanosecond
       precision for the following timestamp fields.
       For the details before Linux 2.6, see NOTES. */

    struct timespec st_atim;  /* Time of last access */
    struct timespec st_mtim;  /* Time of last modification */
    struct timespec st_ctim;  /* Time of last status change */

#define st_atime st_atim.tv_sec      /* Backward compatibility */
#define st_mtime st_mtim.tv_sec
#define st_ctime st_ctim.tv_sec
};
```

stat 结构体

用户和组

```c
#include <sys/types.h>

struct passwd {
    char *pw_name;    // 用户名
    char *pw_passwd;  // 加密后的密码
    uid_t pw_uid;     // 用户 ID
    gid_t pw_gid;     // 主组 ID
    char *pw_gecos;   // 用户全名或相关信息（GECOS 字段）
    char *pw_dir;     // 主目录
    char *pw_shell;   // 登录时使用的 shell
};
```

```c
#include <sys/types.h>

struct group {
    char *gr_name;    // 组名
    char *gr_passwd;  // 加密后的密码
    gid_t gr_gid;     // 组 ID
    char **gr_mem;    // 组成员（以 NULL 结束的字符串数组）
};
```

如何读取文件内容

打开文件open()

open()函数介绍

- 头文件：<sys/types.h> <sys/stat.h> <fcntl.h>
- 原型：`int open(const char *pathname, int flags);`

  `int open(const char *pathname, int flags, mode_t mode);`
  - pathname：打开文件的路径名字
  - flags：一个或多个标志位的按位或组合
  - mode：表示文件的权限，如果flags位为O_CREAT则一定要有文件权限
- 返回值：成功返回fd，失败<0

打开文件open()

- flags 有一系列的常数值可以选择，可以同时选择多个使用**按位或运算符**连接起来，所以这些宏定义都是以O_开头的，表示or。
  - **必选项**：以下三个常数必定指定一个，表示打开文件的方式，且仅能指定一个
    O_RDONLY 、O_WRONLY、O_RDWR
  - **可选项**：可以同时指定零个或多个，和必选项按位或起来作为flags的参数

    O_CREAT　如果此文件不存在则表示新建文件，此时**必须提供第三个参数mode**

    O_EXCL　和标志位O_CREAT一起使用，如果文件已存在，open()调用失败，如果没有和O_CREAT一起使用则标志位失去意义

    O_TRUNC　如果文件存在且是个有可写权限的普通文件，则将文件长度截断为0字节

    O_APPEND 追加，的方式写入文件
- mode 指定文件权限，可以用八进制数表示，也可以用表示常数的宏按位或

  S_IRWXU、S_IRUSR、S_IRGRP…

关闭文件close()

close()函数介绍

- 头文件：<unistd.h>
- 原型：int close(int fd);
  - fd：文件描述符
- 返回值：成功返回0，失败返回-1

- close会取消当前进程的文件描述符fd与其关联的文件之间的映射。
- 当一个进程终止时，内核会对该进程所有尚未关闭的文件描述符调用close
- open的返回值，一定是当该进程所未使用的最小描述符

创建文件creat()

creat()函数介绍

- 头文件：<sys/types.h> <sys/stat.h> <fcntl.h>
- 原型：int creat(const char *pathname, mode_t mode);
  - pathname：打开文件的路径名
  - mode：表示文件的权限，同open()
- 返回值：成功返回fd，失败<0

读文件read()

read()函数介绍

- 头文件：<unistd.h>
- 原型：`ssize_t read(int fd, void *buf, size_t count);`
  - fd：文件描述符
  - buf：一块内存的首地址，用于存放读取的数据
  - count：读取的字节数
- 返回值：读取到的字节数，0表示文件结束

写文件write()

write()函数介绍

- 头文件：<unistd.h>
- 原型：`ssize_t write(int fd, const void *buf, size_t count);`
  - fd：文件描述符
  - buf：一块内存的首地址，存放写入fd的数据
  - count：写入的字节数
- 返回值：写入的字节数，0表示什么也没写入

拜拜