哈爾濱工業大學（深圳）

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

操作系统 (Operating System)

实验五：基于FUSE的青春版EXT2文件系统

2024年秋季

目录

实验目的

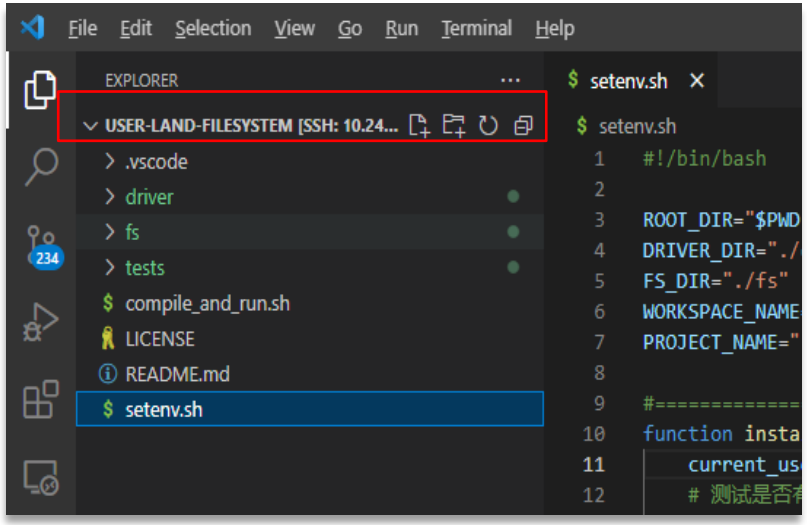
- 以Linux系统中的EXT2文件系统为例，熟悉该文件系统内部数据结构的组织方式和基本处理流程；

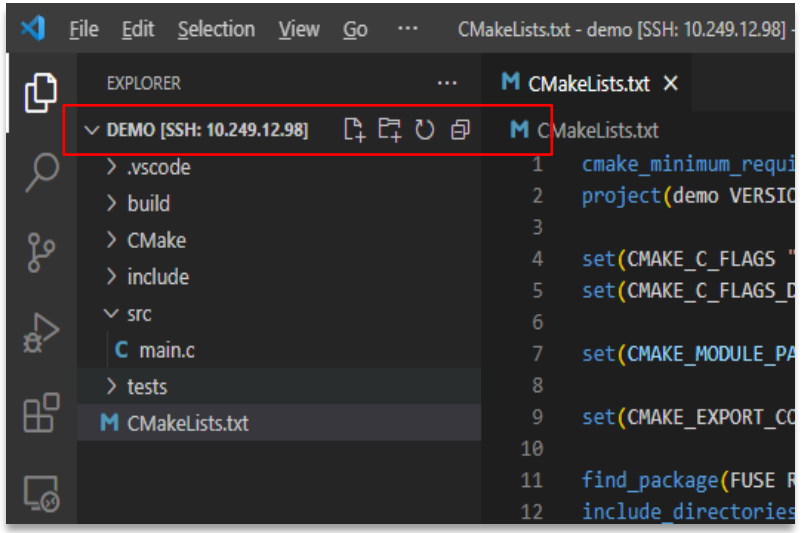
- 基于 FUSE 设计并实现一个可以真正在Linux上跑的文件系统。

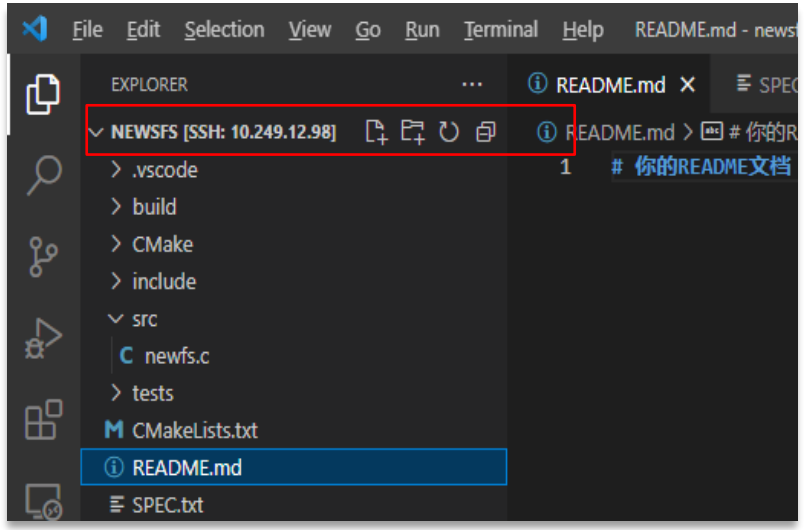
实验内容

- **实验包**： git clone https://gitee.com/ftutorials/user-land-filesystem.git

- **实验环境**：运行**./setenv.sh**后，即可按照指导一步一步来建立环境
  - ✓如需编译任务一，请在 **./fs/demo** 文件夹下打开VSCode软件。
  - ✓如需编译任务二，请在 **./fs/newfs** 文件夹下打开VSCode软件。







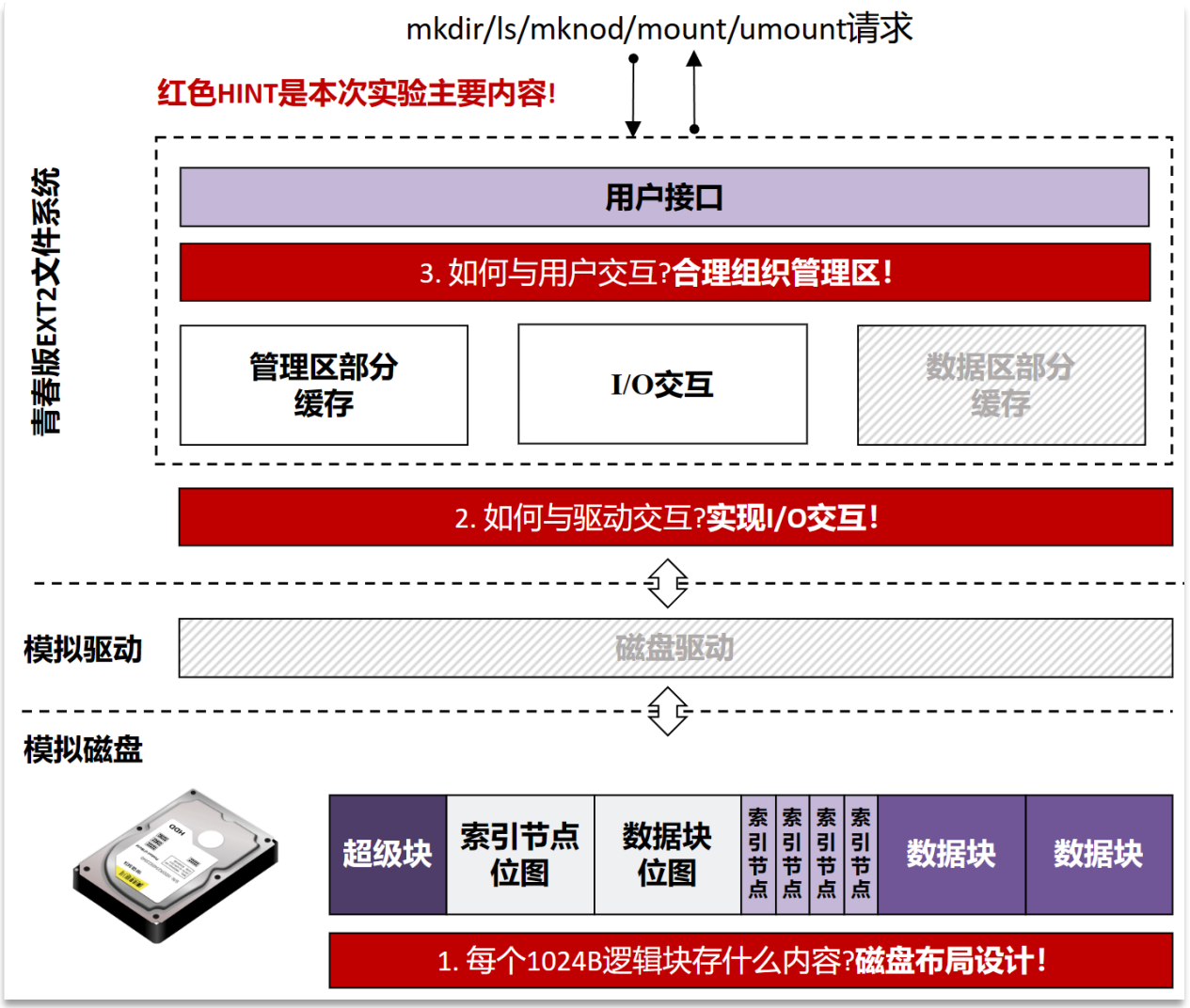
任务一：Vscode打开demo目录　　任务二： Vscode打开newfs目录

实验资源

- **实验指导书：** https://os-labs.pages.dev/lab5/part1/
- **实验环境搭建及调试演示视频**
  - https://www.bilibili.com/video/BV16ESFYeEUg/
- **查看虚拟磁盘的演示视频**
  - 使用Hex Editor查看磁盘布局： https://www.bilibili.com/video/BV1CN411g7KG

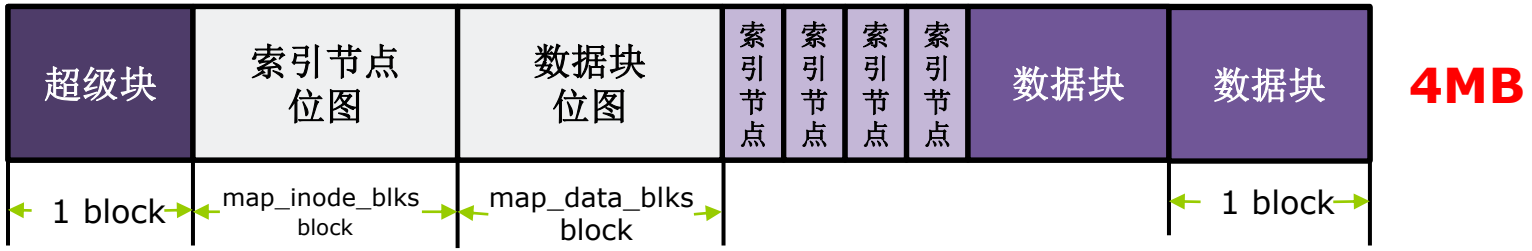
实验原理：实验总体结构

- **模拟磁盘**：**4MB**
- **模拟驱动**（ddriver.c）：
  - **I/O大小为512B**
  - ddriver_open/ddriver_close
  - ddriver_read/ddriver_write
  - ddriver_seek/ddriver_ioctl
- **青春版EXT2文件系统**：
  - ✓**磁盘布局设计**
  - ✓**实现与I/O交互**
  - ✓**合理组织管理区**



实验原理：磁盘布局设计

- Ext2文件系统将盘块分成两大类：保存元数据(管理数据)的**元数据盘块**，以及存放文件内容数据的**数据盘块。**



①**超级块**： 包含整个系统的总体信息；

②**索引节点位图** ： 记录着**索引节点表**的使用情况，**用1个比特**记录某一个索引节点是否被使用；

③**数据块位图**： 记录着**数据块**的使用情况， **用1个比特**记录某一个数据块是否被占用；

④**索引节点（inode）**： 记录着文件的元数据，每个文件都与一个inode对应；

⑤**数据块**： 记录文件内容，若文件太大时，会占用多个 block 。

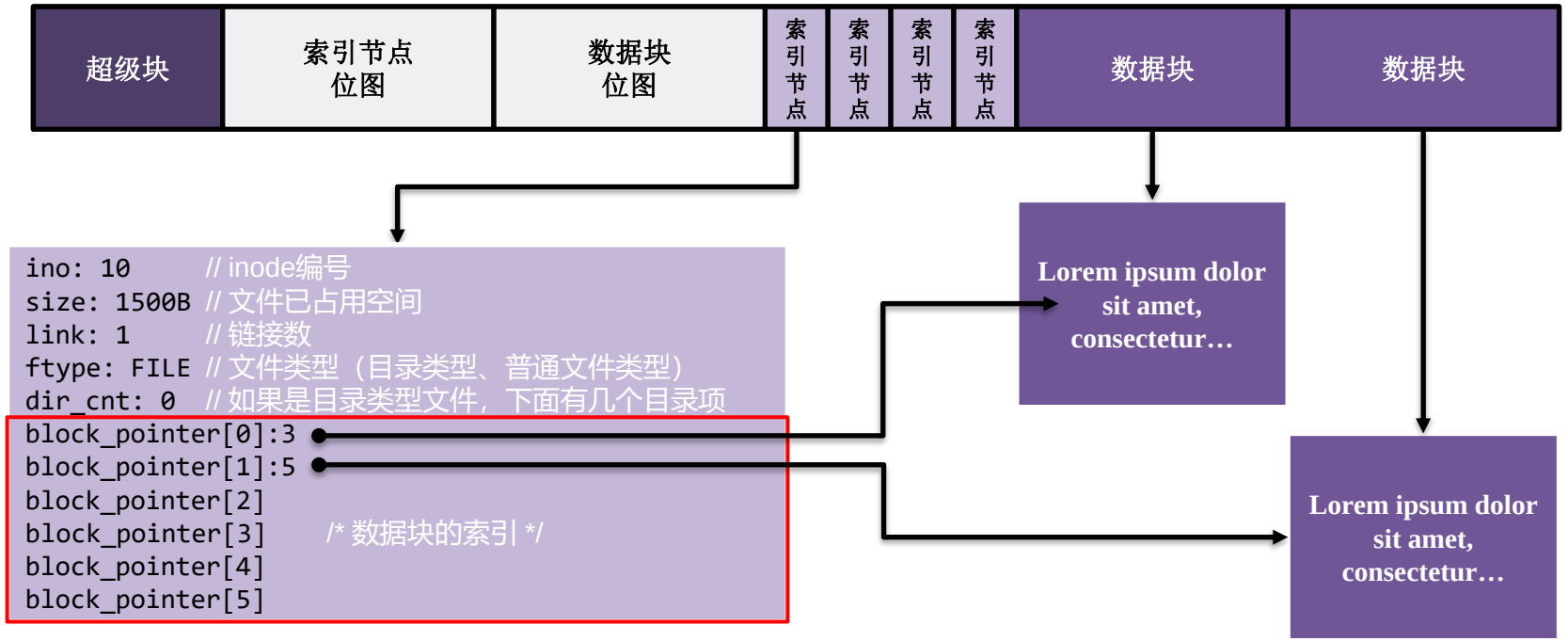
实验原理：超级块

- 超级块是存放**文件系统全局性**的数据结构。
  - **幻数**。用来识别磁盘上是否有文件系统、文件系统类型、文件是否损坏等。
  - **逻辑块信息**。用来管理的逻辑块大小及块数。
  - **磁盘布局分区信息** 。
  - **支持的限制**。如最大支持inode数、文件最大大小等字段。
  - **根目录索引**。根目录的索引节点编号可以自行设定。
  - **其他信息**。如添加时间戳、挂载次数等。

```
struct super_block_d
{
    /* 幻数 */
    uint32_t    magic_number;   // 幻数

    /* 逻辑块信息 */
    int blks_size;          // 逻辑块大小
    int blks_nums;          // 逻辑块数

    /* 磁盘布局分区信息 */
    int sb_offset;          // 超级块于磁盘中的偏移，通常默认为0
    int sb_blks;            // 超级块于磁盘中的块数，通常默认为1

    int ino_map_offset;     // 索引节点位图于磁盘中的偏移
    int ino_map_blks;       // 索引节点位图于磁盘中的块数

    ... // 数据块位图同理

    ... // 索引节点区同理

    ... // 数据块区同理

    /* 支持的限制 */
    int ino_max;            // 最大支持inode数
    int file_max;           // 支持文件最大大小

    /* 根目录索引 */
    int root_ino;           // 根目录对应的inode

    /* 其他信息 */
    ...
}
```

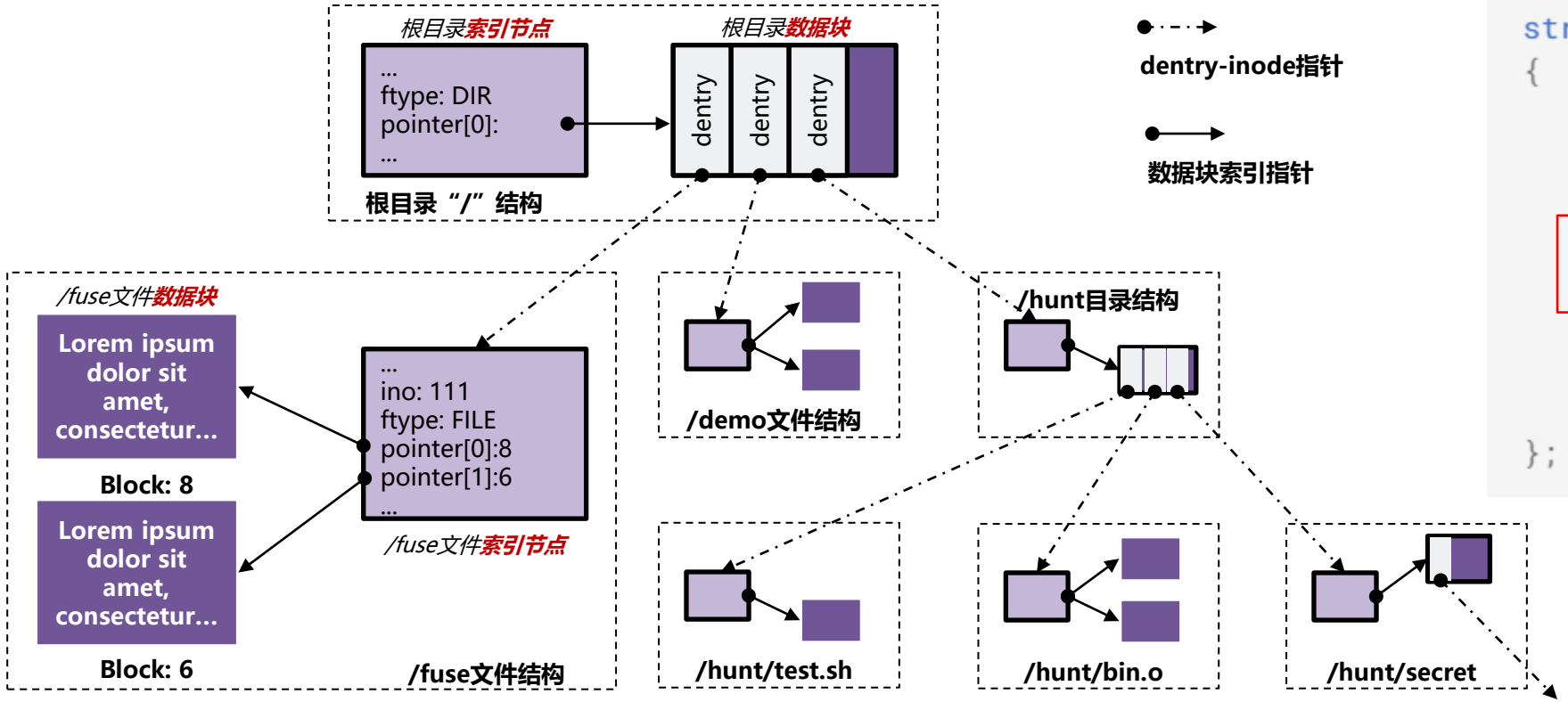
实验原理：文件的表示

- 文件是由**索引节点 inode** 和**数据 data** 构成，包括普通文件 FILE 和目录文件 DIR。
  - 普通文件（file）



实验原理：文件的表示

- 目录文件（dir）：采用UNIX/Linux的文件/文件名解耦方式实现。
  - ✓目录项（dentry）：指向了文件对应的索引节点，并保存了该文件的文件名。通过dentry的方式支持硬链接：一个文件对应多个文件名。




```
struct dentry_d
{
    /* inode编号 */
    int     ino;

    /* 文件名称 */
    char    file_name[MAX_FILE_NAME];

    /* 文件类型 */
    FILE_TYPE       ftype;
};
```

实验原理：in-Mem与to-Disk数据结构

- **位于内存（in-Mem）**：指动态维护着，更加灵活的结构，只要是**方便文件系统进行管理**的所需要的**字段**，都可以动态的在文件系统运行时添加到in-Mem的结构中进行维护。

- **刷回磁盘（to-Disk）**：指要在磁盘中静态存储的结构，也就是要写回磁盘的结构，包含一些**静态的、最基本的、必须要写回磁盘**的字段。

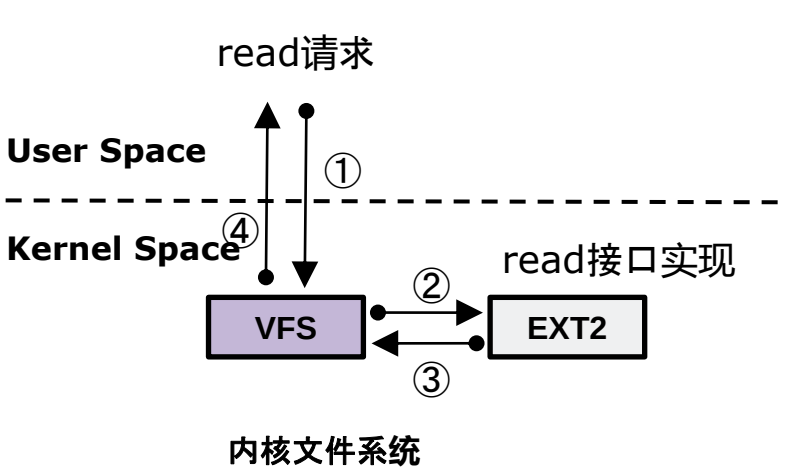
```c
/* to-Disk */
struct dentry_d
{
    /* inode编号 */
    int         ino;
    /* 文件名 */
    char        fname[MAX_FILE_NAME];
    /* 文件类型 */
    FILE_TYPE   ftype;
};
```

```c
/* in-Mem */
struct dentry
{
    /* inode编号 */
    int         ino;
    /* 文件名 */
    char        fname[MAX_FILE_NAME];
    /* 文件类型 */
    FILE_TYPE   ftype;
    /* 多的字段示例 */
    struct  dentry* parent; // 父目录的dentry
    struct  dentry* brother; // 兄弟的dentry
    struct  inode*  inode; // 文件对应的inode
};
```

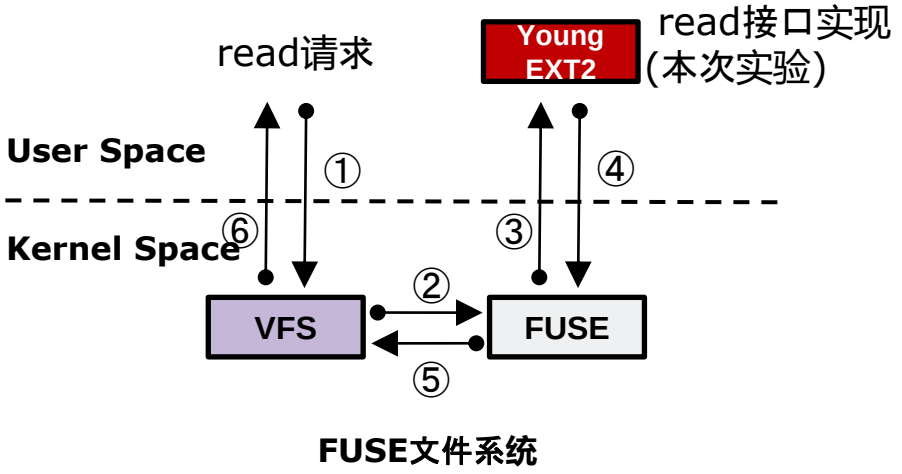
- 多出来的这些字段，不是刷回磁盘所必须的，但是可能是文件系统运行时所需要的，或者方便文件系统进行实现所额外添加的字段。

实验原理：FUSE的使用

- FUSE（Filesystem in User Space）将文件系统的实现从内核态搬到了用户态。
  - **Step ①~②**：如果在该目录中有相关操作（如调用read命令）时，请求会经过Linux虚拟文件系统VFS层到FUSE的内核模块；
  - **Step ③~④**：FUSE内核模块根据请求类型，调用用户态应用注册的函数；
  - **Step ⑤~⑥**：将处理结果通过VFS返回给系统调用。



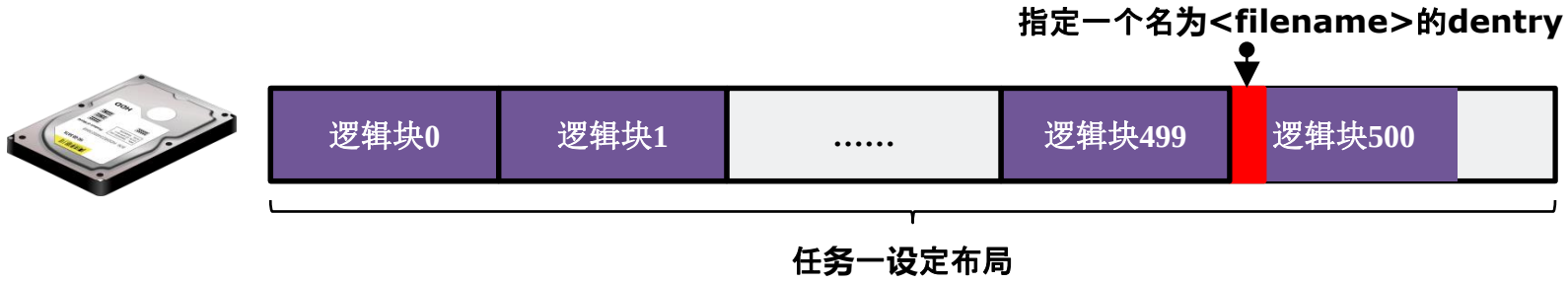
内核文件系统



FUSE文件系统

实验任务一：简单的文件系统demo

- 仅需实现 **ls** 命令，并且 **ls** 时只会显示某个预设的文件名<filename>。
- 假设逻辑块500（**逻辑块大小为1024B，两个I/O块**）为根目录的数据块，这个数据块只有一个dentry，也就是名为<filename>的dentry。
- 出于简单示例，除了上述的块，磁盘其他块都是空的，无需做其他复杂考虑。



实验任务一：简单的文件系统demo

- **Step 1**：配置开发环境，使用**VSCode**打开 **fs/demo文件夹**。
- **Step 2**：理解demo代码。
- **Step 3**：通过**ddriver**磁盘驱动接口完成全局超级块填充：**demo_mount**
- **Step 4**：完成遍历目录逻辑，从**第500个逻辑块**读出demo_dentry：**demo_readdir**
- **Step 5**：完成显示文件属性函数：**demo_getattr**
- **Step 6**：通过测试。

DDRIVER磁盘驱动接口：

```
int ddriver_open(char *path)
int ddriver_close(int fd)
int ddriver_read(int fd, char *buf, size_t size)
int ddriver_write(int fd, char *buf, size_t size)
int ddriver_ioctl(int fd, unsigned long cmd, void *ret)
int ddriver_seek(int fd, off_t offset, int whence)
```

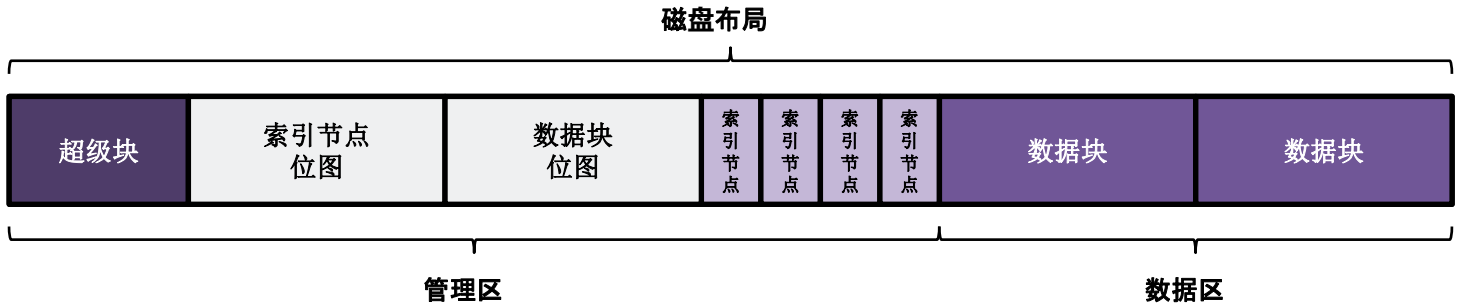
实验任务二：基于FUSE的青春版EXT2文件系统

- 系统结构参考EXT2，要求有SUPER_BLOCK、DATA_MAP、INODE_MAP等主要结构
  - ✓ 挂载/卸载文件系统， **fusermount**
  - ✓ 创建文件，**touch**
  - ✓ 创建文件夹，**mkdir**
  - ✓ 查看文件夹下的文件，**ls**
- **选做**：
  - ✓ 间接索引和二级间接索引
  - ✓ 删除操作（rm命令，需要支持rm -r）
  - ✓ 文件移动（mv）
  - ✓ 读写文件（支持vim，vscode直接修改文件）
  - ✓ 软硬链接（ln命令）

实验任务二：基于FUSE的青春版EXT2文件系统

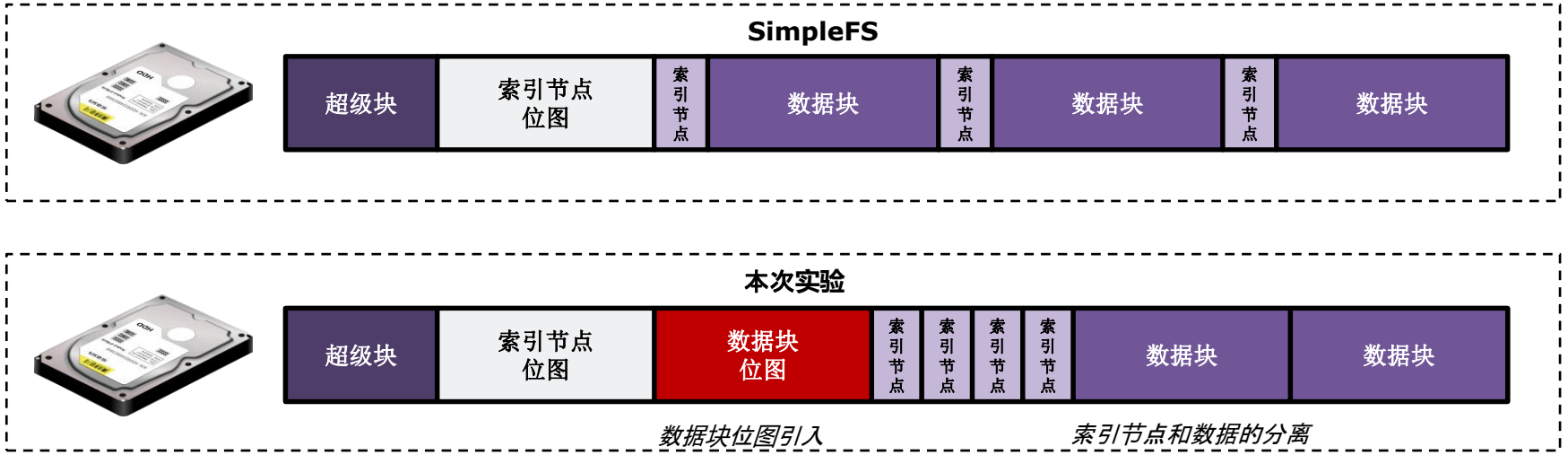
➢ 实验要求：

- ✓ 逻辑块大小应该为**1024B**，也就是两个磁盘的I/O单位（**512B**）。
- ✓ 该文件系统的磁盘布局应该按顺序依次包括：**超级块**、**索引节点位图**、**数据块位图**、**索引节点区**、**数据块区**。**五个区域缺一不可**。但具体的每个区域占多少个逻辑块，同学们可以自行设计。
- ✓ 实现**按需分配**文件数据块，利用好数据块位图，当文件需要新的数据块来写内容的时候才分配，**而不是采用预先分配**。
- ✓ 本次实验统一规定**不实现 “.“ 和 ".. “ 两个特殊目录**。



实验步骤：基于FUSE的青春版EXT2文件系统

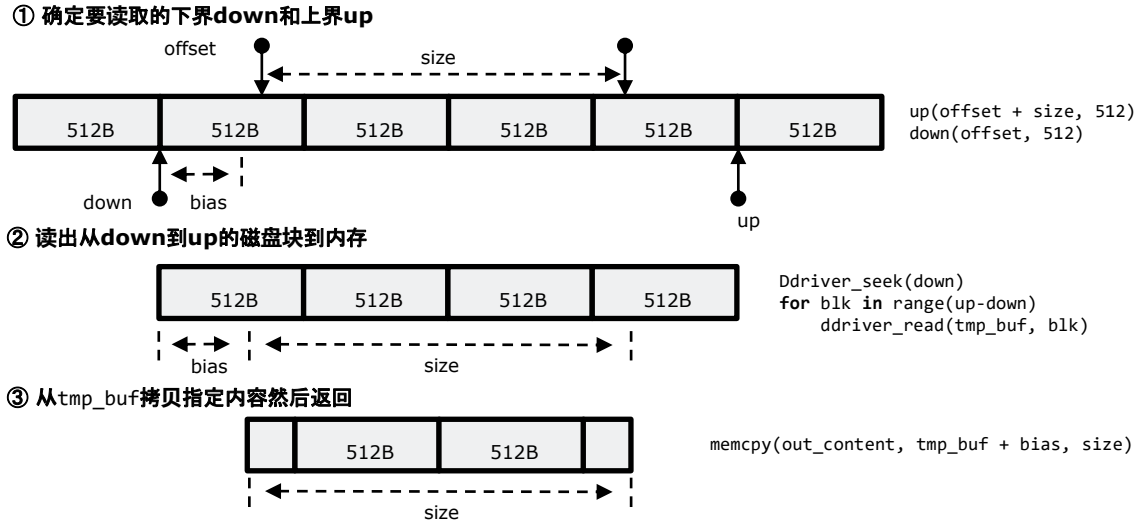
- **Step 1**：学习参考一个实现了完整功能的文件系统样例，simplefs
  - **逻辑块大小**不一样，Simplefs： **512B**， Young EXT2： **1024B**
  - **磁盘布局**不一样：
    - ✓ Young EXT2 磁盘布局应该包括：超级块、索引节点位图、数据块位图、索引节点区、数据块区五个区域。



实验步骤：基于FUSE的青春版EXT2文件系统

Step 2：封装对ddriver的访问代码

- 利用提供原始驱动接口ddriver_read和ddriver_write，完成一层封装。先把数据所在的磁盘块都读出来，然后再从这一部分读出的数据中读写相应的数据，若是写，则要把读入修改的部分再写回磁盘。
  - int your_read(int offset, void *out_content, int size);
  - int your_write(int offset, void *out_content, int size);



DDRIVER磁盘驱动接口：

int ddriver_open(char *path)
int ddriver_close(int fd)
int ddriver_read(int fd, char *buf, size_t size)
int ddriver_write(int fd, char *buf, size_t size)
int ddriver_ioctl(int fd, unsigned long cmd, void *ret)
int ddriver_seek(int fd, off_t offset, int whence)

实验步骤：基于FUSE的青春版EXT2文件系统

Step 3：实现文件系统接口

- 挂载/卸载文件系统，**fusermount**
  - **挂载函数** void* init(struct fuse_conn_info * conn_info);
  - **卸载函数** void* destroy(void* p);
- 创建文件，**touch**
  - **创建文件函数** int mknod(const char* path, mode_t mode, dev_t dev);
  - **修改文件访问时间** `int utimens(const char* path, const struct timespec tv[2]);`
- 创建文件夹，**mkdir**
  - **创建目录函数** int mkdir(const char* path, mode_t mode);
- 查看文件夹下的文件，**ls**
  - **获取文件属性函数** int getattr(const char* path, struct stat * stat);
  - **读取目录函数** int readdir(const char * path, void * buf, fuse_fill_dir_t filler, off_t offset, struct fuse_file_info * fi);

实验步骤：基于FUSE的青春版EXT2文件系统

- **Step 4**：通过实验测评
  - **填写好include目录下的 fs.layout 文件。描述你的磁盘布局情况，其中()内的数值代表占多少个块，BSIZE代表逻辑块的大小。**

```
| BSIZE = 1024 B |
| Super(1) | Inode Map(1) | DATA Map(1) | INODE(1) | DATA(*) |
```

  - **运行测评程序**

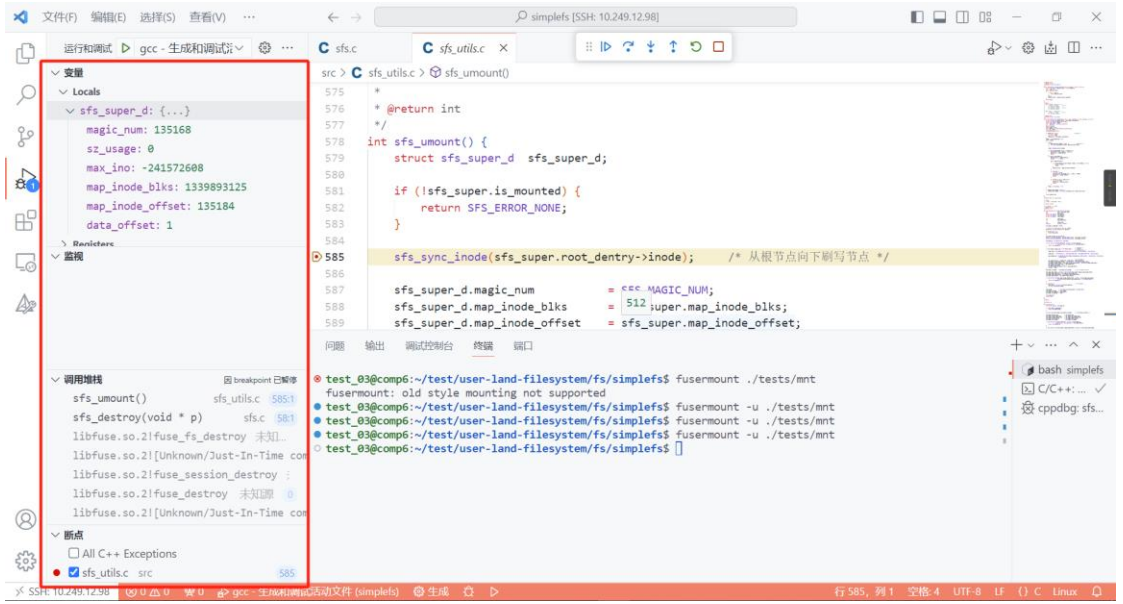
```
chmod +x test.sh && ./test.sh
```

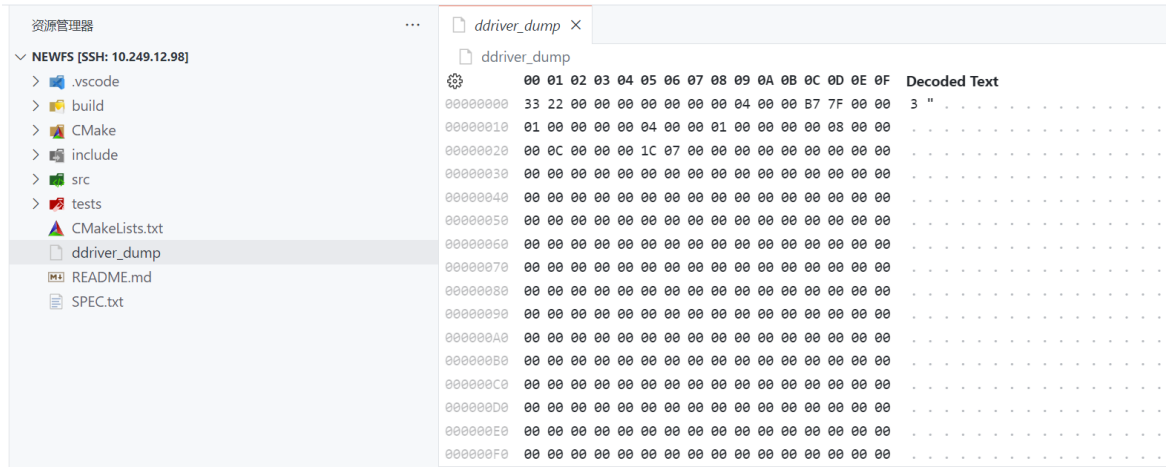
  - **基础分 30分，进阶分 34分**

```
Score: 30/30
pass: 恭喜你，通过所有测试 (30/30)
/home/students/test_03/sample/code/newfs/tests/mnt
test_03@comp6:~/sample/code/newfs/tests$
```

实验要点

- 调试技巧：
  - ✓ **Printf大法**。
  - ✓ 利用 **VSCode设置断点**，单步调试，在左侧窗口看看各个变量的值是否合理，或者找到crash的地方等等。
  - ✓ **注释掉部分代码**，来重新运行文件系统，看文件系统是否会发生crash，来定位到最终发生问题的地方。可以采用二分的思想来不断缩小注释的区间来定位。
  - ✓ 使用Hex Editor**查看磁盘镜像**。检查一下例如数据位图、索引节点位图的具体数值，数据写回位置是否和预期的相符等。
  - ✓ 请认真阅读实验指导书以及**常见问题**。





实验提交

- **请务必注意查看作业提交平台，按时提交代码及报告**
  - ✓ 提交截止时间：两个星期内
    - ✓ 实验报告
    - ✓ 实验代码

Hope you enjoyed the OS course!