Operating Systems

Introduction to Lab 8

File System

Department of Computer Science & Technology
Tsinghua University
IIIS

Outline

- The Architecture of ucore File System
- The Simple File System
- Virtual File System
- I/O Device Interfaces
- Work Flow

The Architecture of ucore File System

FS测试用例::usr/*.c

write::usr/libs/file.c
sys_write::usr/libs/syscall.c
syscall::usr/libs/syscall.c
sys_write::/kern/syscall/syscall.c

sysfile_write::/kern/fs/sysfile.c
file_write::/kern/fs/file.c
vop_write::/kern/fs/vfs/inode.h

How to select concrete FS?

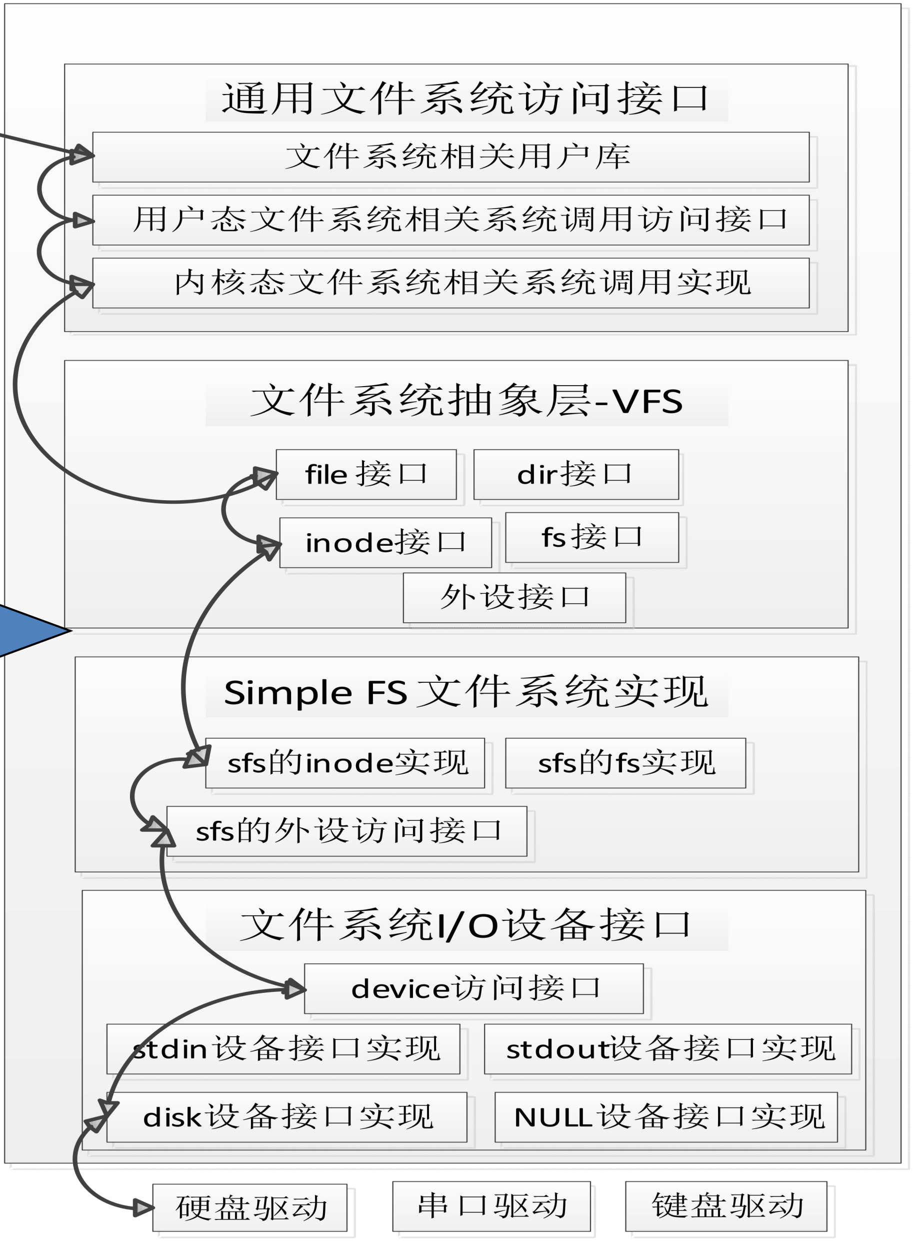
sfs_write::/kern/fs/sfs/sfs_inode.c
-> sfs_io() -> sfs_io_nolock()

sfs_wbuf::/kern/fs/sfs/sfs_io.c
-> sfs_rwblock_nolock()

dop_io::/kern/fs/devs/dev.h

disk0_io::/kern/fs/devs/dev_disk0.c

ide_write_secs::/kern/driver/ide.c



通用文件系统访问接口

文件系统相关用户库

用户态文件系统相关系统调用访问接口

内核态文件系统相关系统调用实现

文件系统抽象层-VFS

file接口 | dir接口

inode接口 | fs接口

外设接口

Simple FS 文件系统实现

sfs的inode实现 | sfs的fs实现

sfs的外设访问接口

文件系统I/O设备接口

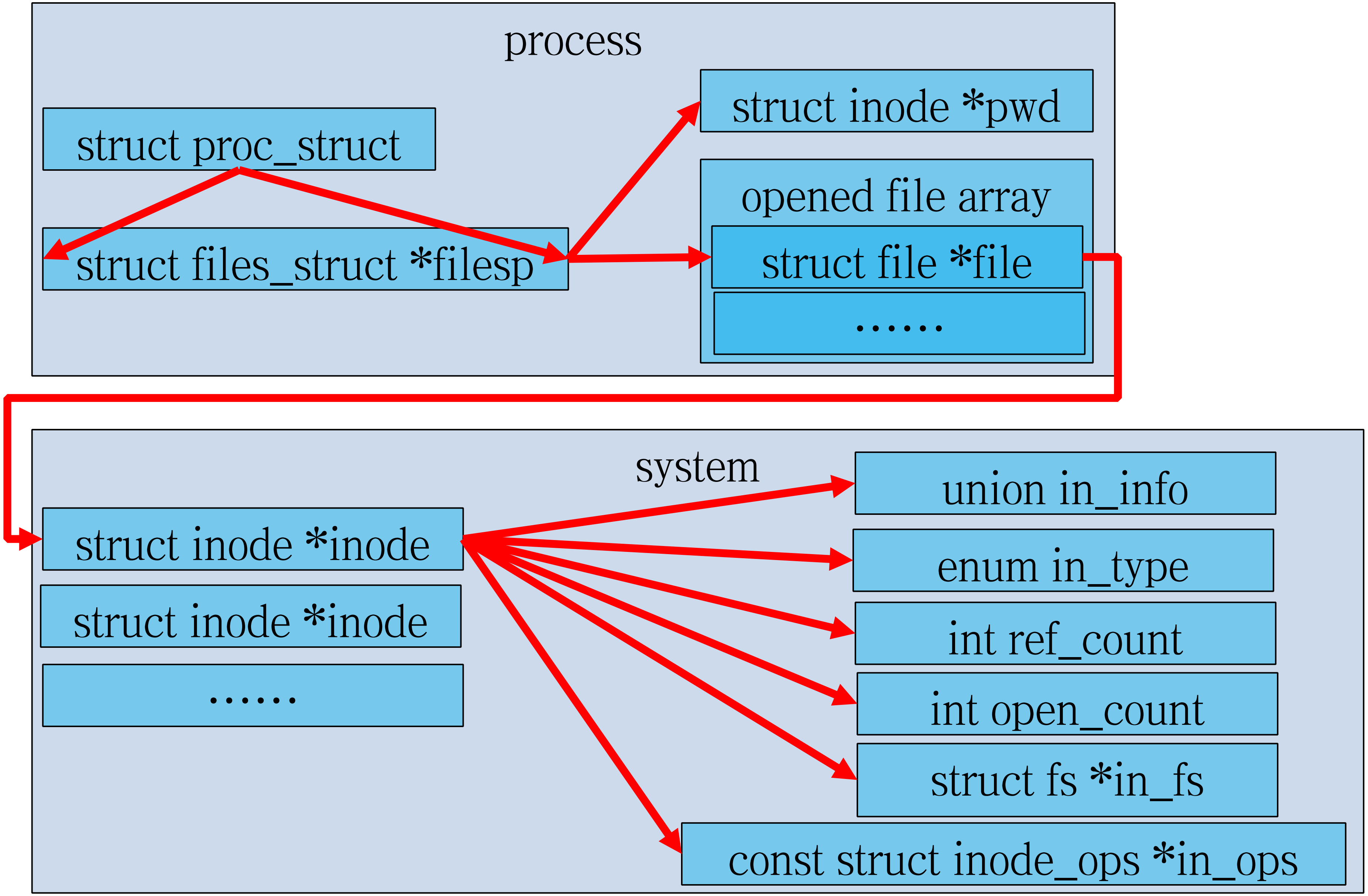
device访问接口

stdin设备接口实现 | stdout设备接口实现

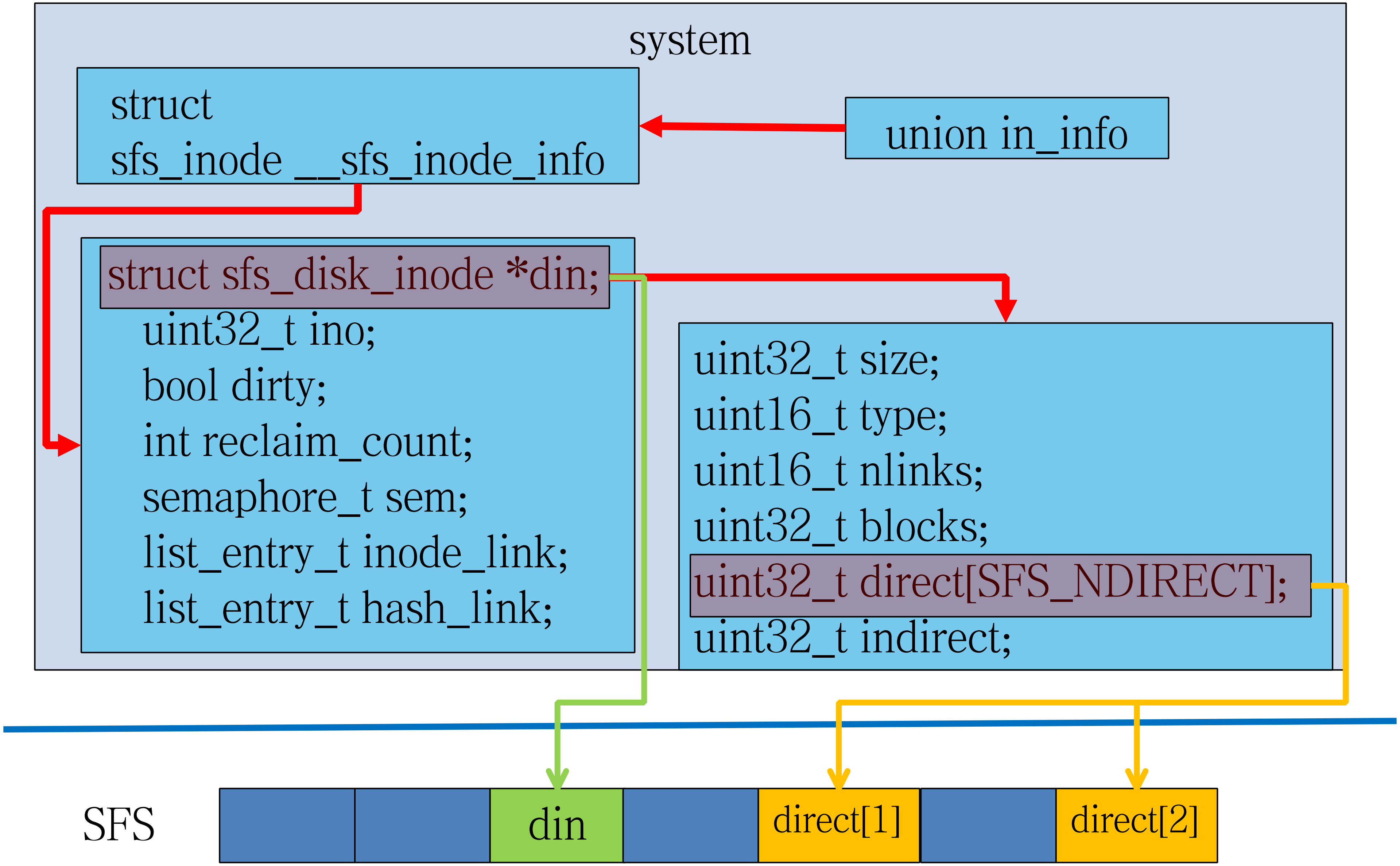
disk设备接口实现 | NULL设备接口实现

硬盘驱动 | 串口驱动 | 键盘驱动

The Key Data Structures



The Key Data Structures



system

struct
sfs_inode __sfs_inode_info

union in_info

struct sfs_disk_inode *din;
uint32_t ino;
bool dirty;
int reclaim_count;
semaphore_t sem;
list_entry_t inode_link;
list_entry_t hash_link;

uint32_t size;
uint16_t type;
uint16_t nlinks;
uint32_t blocks;
uint32_t direct[SFS_NDIRECT];
uint32_t indirect;

SFS

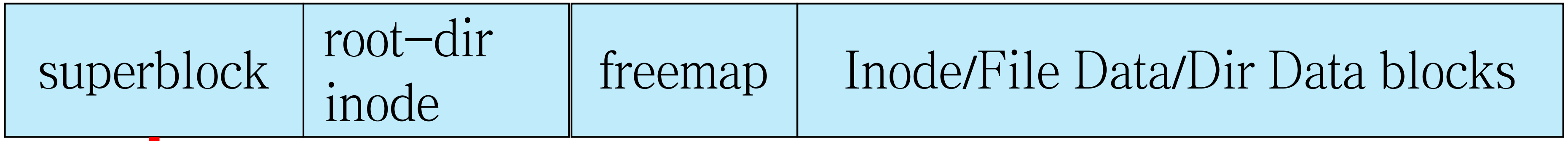
din

direct[1]

direct[2]

The Simple FS

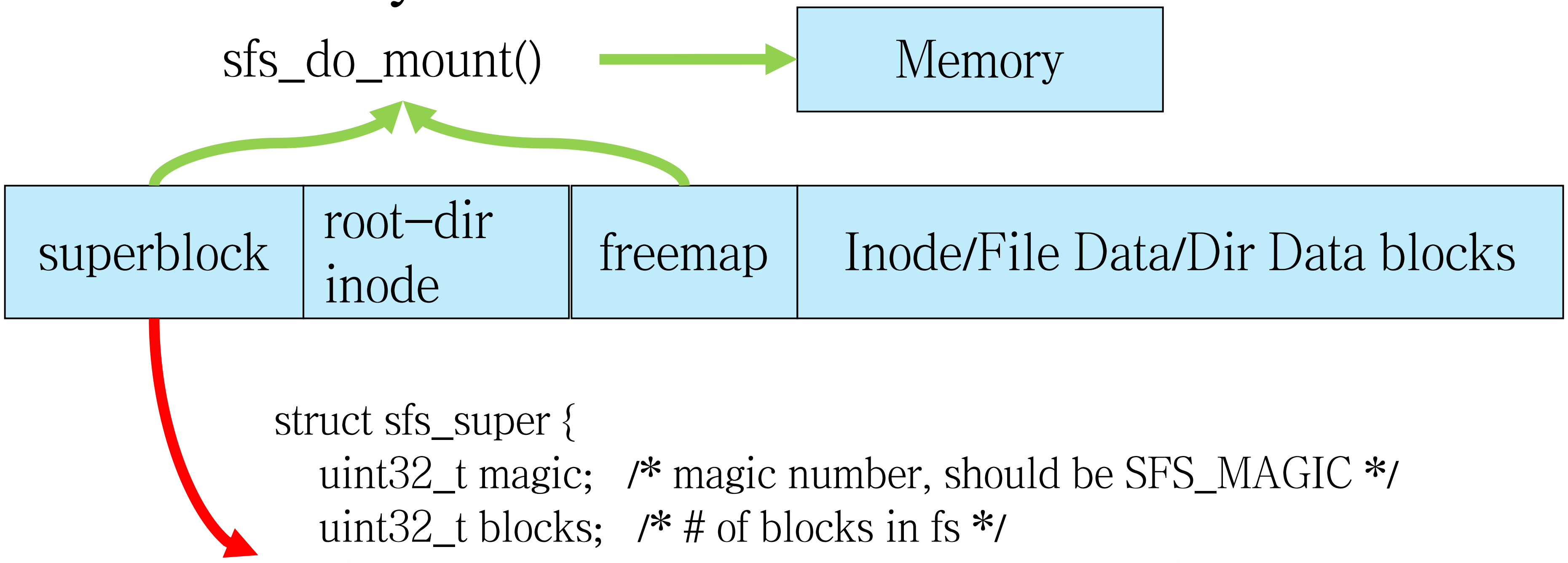
- File Types
  - Regular file; Directory; Link file; Device; Pipe.
- SFS Layout in Disk



```
struct sfs_super {
    uint32_t magic;   /* magic number, should be SFS_MAGIC */
    uint32_t blocks;   /* # of blocks in fs */
    uint32_t unused_blocks;   /* # of unused blocks in fs */
    char info[SFS_MAX_INFO_LEN + 1];    /* infomation for sfs  */
};
```

The Simple FS

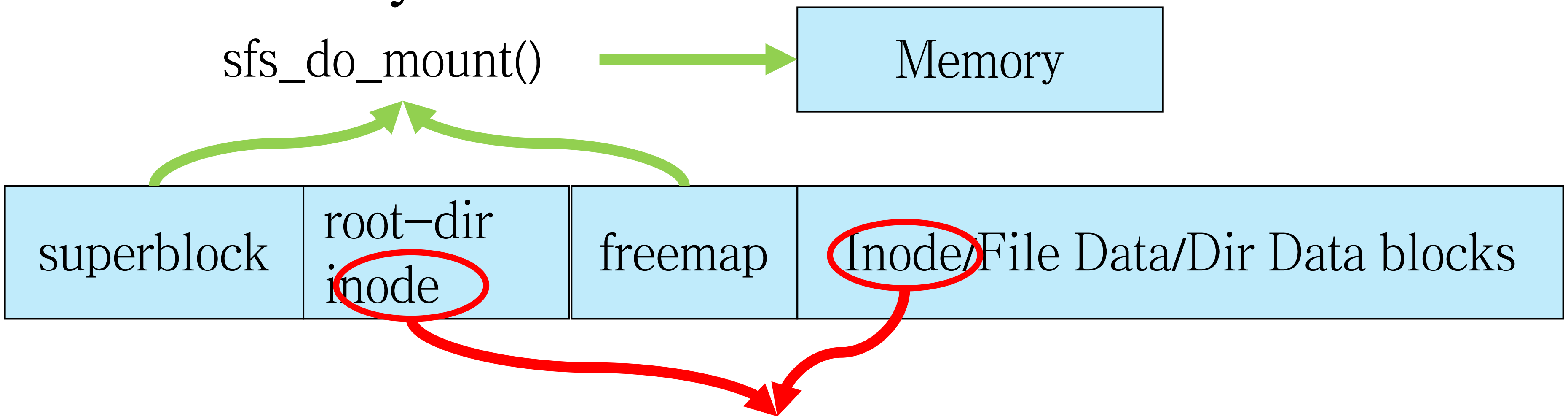
- File Types
  - Regular file; Directory; Link file; Device; Pipe.
- SFS Layout in Disk



```
struct sfs_super {
    uint32_t magic;   /* magic number, should be SFS_MAGIC */
    uint32_t blocks;   /* # of blocks in fs */
    uint32_t unused_blocks;   /* # of unused blocks in fs */
    char info[SFS_MAX_INFO_LEN + 1];    /* infomation for sfs  */
};
```

The Simple FS

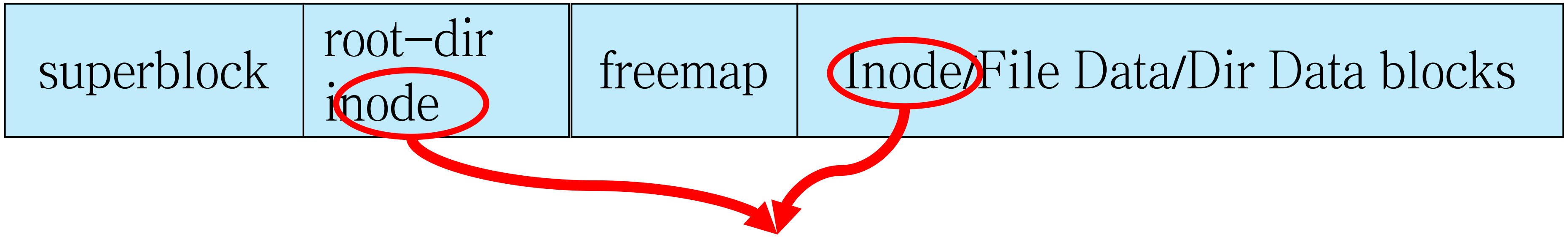
- File Types
  - Regular file; Directory; Link file; Device; Pipe.
- SFS Layout in Disk



```
struct sfs_disk_inode {
    uint32_t size;                      如果inode表示常规文件，则size是文件大小
    uint16_t type;                      inode的文件类型
    uint16_t nlinks;                    此inode的硬链接数
    uint32_t blocks;                    此inode的数据块数的个数
    uint32_t direct[SFS_NDIRECT];  此inode的直接数据块索引值（有SFS_NDIRECT个）
    uint32_t indirect;                  此inode的一级间接数据块索引值
};
```

The Simple FS

- File Types
  - Regular file; Directory; Link file; Device; Pipe.
- SFS Layout in Disk



```
struct sfs_disk_inode {
    …
    uint32_t direct[SFS_NDIRECT];  此inode的直接数据块索引值（有SFS_NDIRECT个）
    uint32_t indirect;              此inode的一级间接数据块索引值
};
```

```
struct sfs_disk_entry {
    uint32_t ino;       索引节点所占数据块索引值
    char name[SFS_MAX_FNAME_LEN + 1];     文件名
}
```

The Simple FS

- Inode in Memory (\kern\fs\sfs\sfs.h)

```
struct sfs_inode {
    struct sfs_disk_inode *din;            /* on−disk inode */
    uint32_t ino;                          /* inode number */
    uint32_t flags;                        /* inode flags */
    bool dirty;                            /* true if inode modified */
    int reclaim_count;                     /* kill inode if it hits zero */
    semaphore_t sem;                       /* semaphore for din */
    list_entry_t inode_link;               /* entry for linked−list in sfs_fs */
    list_entry_t hash_link;           /* entry for hash linked−list in sfs_fs */
};
```

The Simple FS

- Inode in Memory (\kern\fs\sfs\sfs.h, sfs_inode.c)

```
struct sfs_inode {
    struct sfs_disk_inode *din;            /* on−disk inode */
    uint32_t ino;                           /* inode number */
    uint32_t flags;                        /* inode flags */
    bool dirty;                             /* true if inode modified */
    int reclaim_count;                     /* kill inode if it hits zero */
    semaphore_t sem;                       /* semaphore for din */
    list_entry_t inode_link;                /* entry for linked−list in sfs_fs */
    list_entry_t hash_link;          /* entry for hash linked−list in sfs_fs */
};

sfs_bmap_load_nolock(···);  sfs_bmap_truncate_nolock(···);
sfs_dirent_read_nolock(···); sfs_dirent_write_nolock(···);
sfs_dirent_search_nolock(···);
```

The Simple FS

- Inode Operation (\kern\fs\sfs\sfs_inode.c)

```
static const struct inode_ops sfs_node_fileops = {
    .vop_magic                    = VOP_MAGIC,
    .vop_open                     = sfs_openfile,
    .vop_close                    = sfs_close,
    .vop_read                     = sfs_read,
    .vop_write                    = sfs_write,
    ……
};

   static const struct inode_ops sfs_node_dirops = {
       .vop_magic                    = VOP_MAGIC,
       .vop_open                     = sfs_opendir,
       .vop_close                    = sfs_close,
       .vop_getdirentry                = sfs_getdirentry,
   .vop_lookup                   = sfs_lookup,
       ……
   };
```

Virtual File System

- file (\kern\fs\file.h)

```
struct file {
    enum {
        FD_NONE, FD_INIT, FD_OPENED, FD_CLOSED,
    } status;                            //访问文件的执行状态
    bool readable;                       //文件是否可读
    bool writable;                      //文件是否可写
    int fd;                              //文件在filemap中的索引值
    off_t pos;                          //访问文件的当前位置
    struct inode *node;                 //该文件对应的内存inode指针
    atomic_t open_count;                //打开此文件的次数
};
```

Virtual File System

- file interface (\kern\fs\file.h)

```
struct file {
    enum {
       FD_NONE, FD_INIT, FD_OPENED, FD_CLOSED,
    } status;                              //访问文件的执行状态
    bool readable;                         //文件是否可读
    bool writable;                         //文件是否可写
    int fd;                                //文件在filemap中的索引值
    off_t pos;                             //访问文件的当前位置
    struct inode *node;                    //该文件对应的内存inode指针
    atomic_t open_count;                   //打开此文件的次数
};

 struct files_struct {
     struct inode *pwd;      // inode of present working directory
     struct file *fd_array;  // opened files array
     int files_count;        // the number of opened files
     semaphore_t files_sem;  // lock protect sem
 };
```

Virtual File System

- inode interface(\kern\fs\vfs\inode.h)

```
struct inode {
    union {                    //包含不同文件系统特定inode信息的union成员变量
        struct device __device_info;        //设备文件系统内存inode信息
        struct sfs_inode __sfs_inode_info;    //SFS文件系统内存inode信息
    } in_info;
    enum {
        inode_type_device_info = 0x1234,
        inode_type_sfs_inode_info,
    } in_type;                          //此inode所属文件系统类型
    int ref_count;          //此inode的引用计数
    int open_count;        //打开此inode对应文件的个数
    struct fs *in_fs;           //抽象的文件系统，包含访问文件系统的函数指针
    const struct inode_ops *in_ops;    //抽象的inode操作，包含访问inode的函数指针
};
```

Virtual File System

- inode interface(\kern\fs\vfs\inode.h)

```
struct inode {
   ......
  struct fs *in_fs;          //抽象的文件系统，包含访问文件系统的函数指针
  const struct inode_ops *in_ops;   //抽象的inode操作，包含访问inode的函数指针
};

struct inode_ops {
    unsigned long vop_magic;
    int (*vop_open)(struct inode *node, uint32_t open_flags);
    int (*vop_close)(struct inode *node);
    int (*vop_read)(struct inode *node, struct iobuf *iob);
    int (*vop_write)(struct inode *node, struct iobuf *iob);
    int (*vop_getdirentry)(struct inode *node, struct iobuf *iob);
    int (*vop_create)(struct inode *node, const char *name, bool excl, struct inode
**node_store);
int (*vop_lookup)(struct inode *node, char *path, struct inode **node_store);
......
 };
```

Virtual File System

- inode interface(\kern\fs\vfs\inode.h)

```
  struct inode {
     ……
     struct fs *in_fs;            //抽象的文件系统，包含访问文件系统的函数指针
     const struct inode_ops *in_ops;   //抽象的inode操作，包含访问inode的函数指针
  };


struct fs {
    union {
        struct sfs_fs __sfs_info;
    } fs_info;                                      // filesystem–specific data
    enum {
        fs_type_sfs_info,
    } fs_type;                                      // filesystem type
    int (*fs_sync)(struct fs *fs);                  // Flush all dirty buffers to disk
    struct inode *(*fs_get_root)(struct fs *fs);    // Return root inode of filesystem.
    int (*fs_unmount)(struct fs *fs);               // Attempt unmount of filesystem.
    void (*fs_cleanup)(struct fs *fs);              // Cleanup of filesystem.???
};
```

I/O Device Interface

- device interface(\kern\fs\devs\dev.h)

```
struct device {
    size_t d_blocks;    //设备占用的数据块个数
    size_t d_blocksize;  //数据块的大小
    int (*d_open)(struct device *dev, uint32_t open_flags);  //打开设备的函数指针
    int (*d_close)(struct device *dev); //关闭设备的函数指针
    int (*d_io)(struct device *dev, struct iobuf *iob, bool write); //读写设备的函数指针
    int (*d_ioctl)(struct device *dev, int op, void *data); //用ioctl方式控制设备的函数指针
}
```

I/O Device Interface

- device interface(\kern\fs\devs\dev.h \vfs\vfsdev.c)

```
struct device {
    size_t d_blocks;    //设备占用的数据块个数
    size_t d_blocksize;  //数据块的大小
    int (*d_open)(struct device *dev, uint32_t open_flags);  //打开设备的函数指针
    int (*d_close)(struct device *dev); //关闭设备的函数指针
    int (*d_io)(struct device *dev, struct iobuf *iob, bool write); //读写设备的函数指针
    int (*d_ioctl)(struct device *dev, int op, void *data); //用ioctl方式控制设备的函数指针
}


typedef struct {
    const char *devname;
    struct inode *devnode;
    struct fs *fs;
    bool mountable;
    list_entry_t vdev_link;
} vfs_dev_t;
```

I/O Device Interface

- stdout device
  - Initialization

kern_init-->fs_init-->dev_init-->dev_init_stdout --> dev_create_ino③

① ② ③ ④ --> stdout_device_init ④

--> vfs_add_dev ⑤

① kern\init\init.c

② kern\fs\fs.c

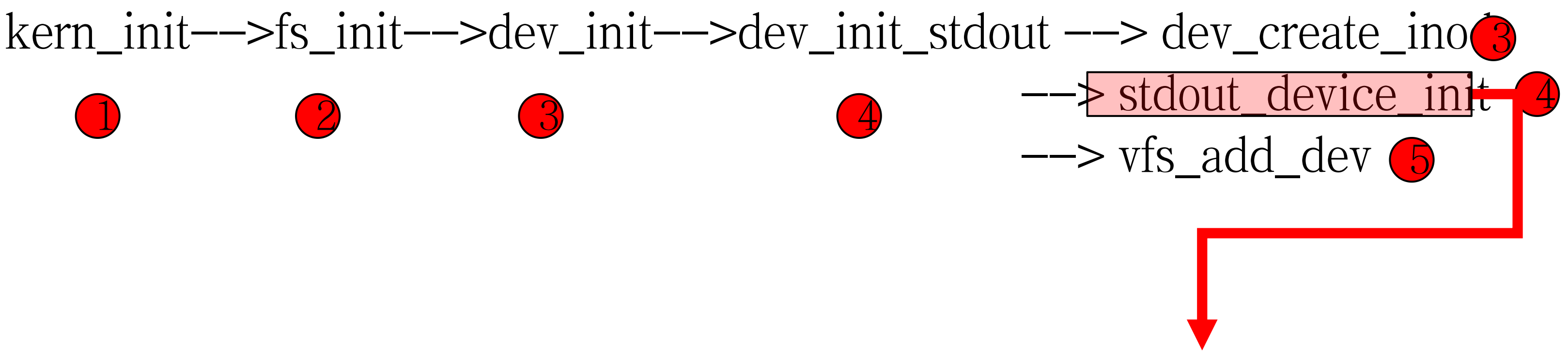
③ kern\fs\devs\dev.c

④ kern\fs\devs\dev_stdout.c

⑤ kern\fs\vfs\vfsdev.c

I/O Device Interface

- stdout device
  - Initialization



kern_init-->fs_init-->dev_init-->dev_init_stdout --> dev_create_inode ③

① ② ③ ④

--> stdout_device_init ④

--> vfs_add_dev ⑤

① kern\init\init.c

② kern\fs\fs.c

③ kern\fs\devs\dev.c

④ kern\fs\devs\dev_stdout.c

⑤ kern\fs\vfs\vfsdev.c

```
static void
stdout_device_init(struct device *dev) {
    dev->d_blocks = 0;
    dev->d_blocksize = 1;
    dev->d_open = stdout_open;
    dev->d_close = stdout_close;
    dev->d_io = stdout_io;
    dev->d_ioctl = stdout_ioctl;
}
```

I/O Device Interface

- stdin device
  - Initialization

kern_init-->fs_init-->dev_init-->dev_init_stdin--> dev_create_inode ③

① ② ③ ④

--> stdin_device_init ④

--> vfs_add_dev ⑤

① kern\init\init.c

② kern\fs\fs.c

③ kern\fs\devs\dev.c

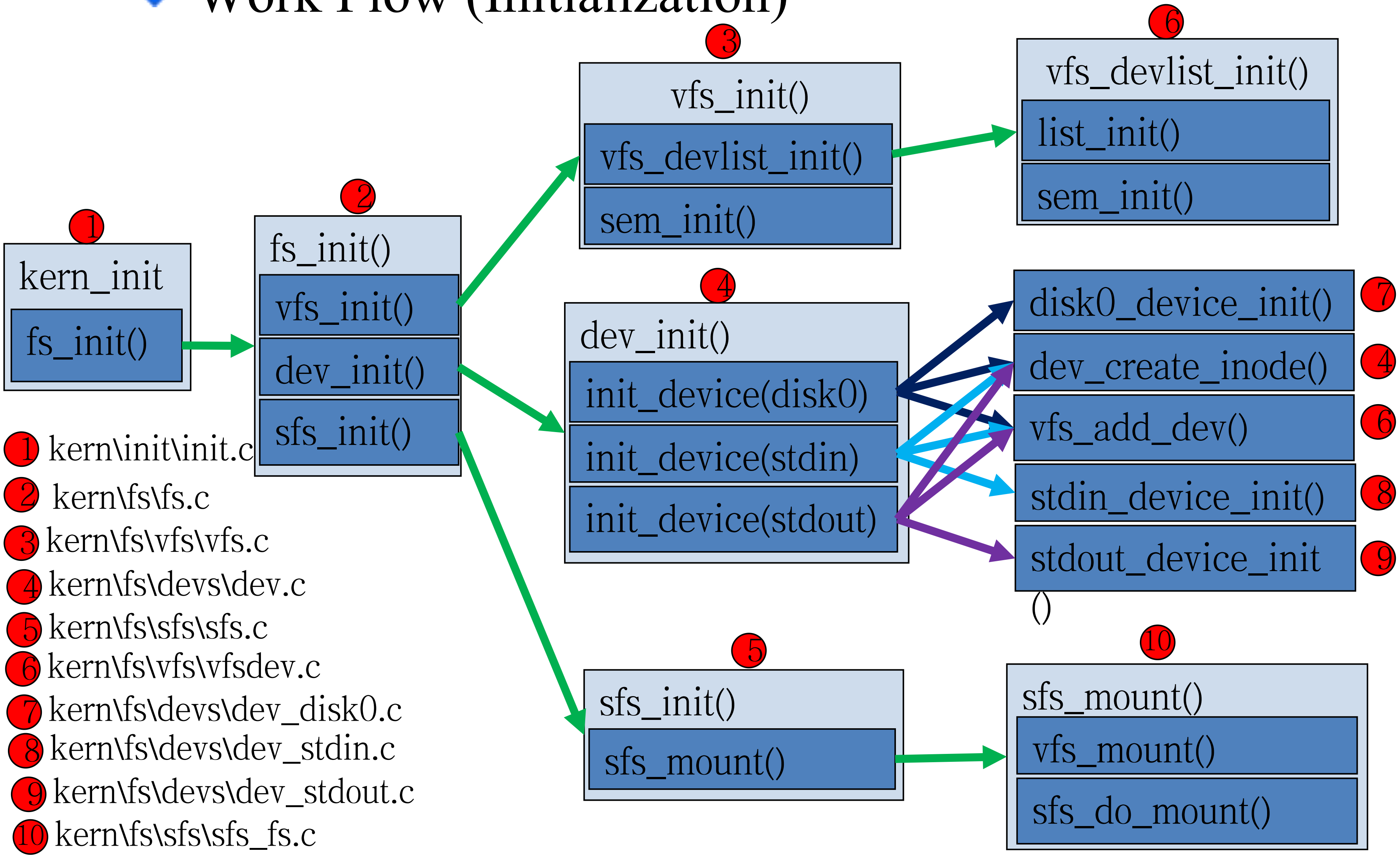
④ kern\fs\devs\dev_stdin.c

⑤ kern\fs\vfs\vfsdev.c

```
static void
stdin_device_init(struct device *dev) {
    dev->d_blocks = 0;
    dev->d_blocksize = 1;
    dev->d_open = stdin_open;
    dev->d_close = stdin_close;
    dev->d_io = stdin_io;
    dev->d_ioctl = stdin_ioctl;
     p_rpos = p_wpos = 0;
    wait_queue_init(wait_queue);
}
```

Work Flow & Key Functions and Data Structures

- Work Flow (Initialization)



1 kern\init\init.c

2 kern\fs\fs.c

3 kern\fs\vfs\vfs.c

4 kern\fs\devs\dev.c

5 kern\fs\sfs\sfs.c

6 kern\fs\vfs\vfsdev.c

7 kern\fs\devs\dev_disk0.c

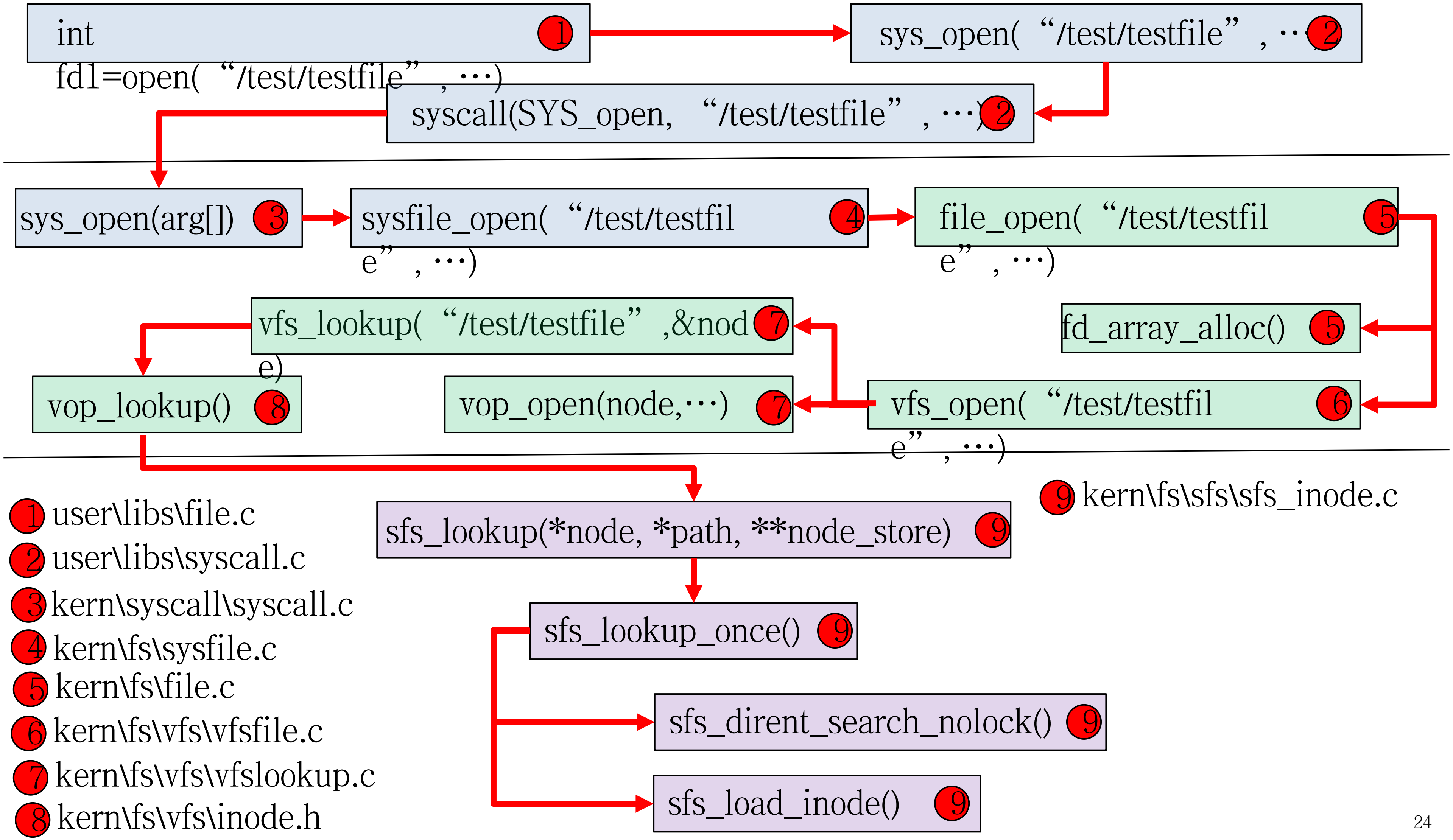
8 kern\fs\devs\dev_stdin.c

9 kern\fs\devs\dev_stdout.c

10 kern\fs\sfs\sfs_fs.c

Work Flow & Key Functions and Data Structures

◆ Work Flow (Open File)



int fd1=open(“/test/testfile” , …) ①

sys_open(“/test/testfile” , …) ②

syscall(SYS_open, “/test/testfile” , …) ②

sys_open(arg[]) ③

sysfile_open(“/test/testfile” , …) ④

file_open(“/test/testfile” , …) ⑤

fd_array_alloc() ⑤

vfs_open(“/test/testfile” , …) ⑥

vop_open(node,…) ⑦

vfs_lookup(“/test/testfile” ,&node) ⑦

vop_lookup() ⑧

sfs_lookup(*node, *path, **node_store) ⑨

sfs_lookup_once() ⑨

sfs_dirent_search_nolock() ⑨

sfs_load_inode() ⑨

① user\libs\file.c

② user\libs\syscall.c

③ kern\syscall\syscall.c

④ kern\fs\sysfile.c

⑤ kern\fs\file.c

⑥ kern\fs\vfs\vfsfile.c

⑦ kern\fs\vfs\vfslookup.c

⑧ kern\fs\vfs\inode.h

⑨ kern\fs\sfs\sfs_inode.c

That’s all. Thanks!