文件IO

从内存的角度看
1：从文件中读取数据到内存中
0:从内存将数据写入文件中

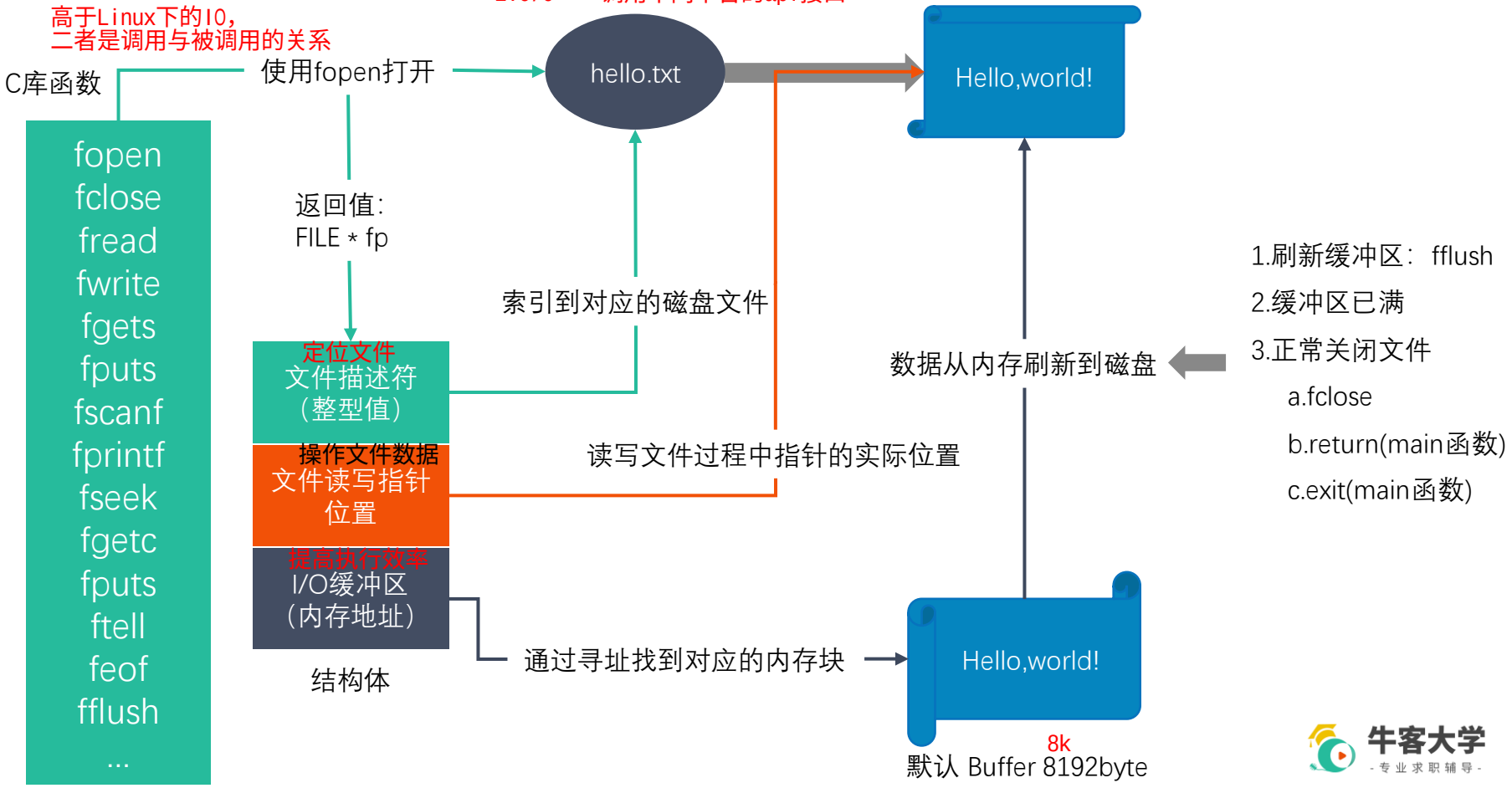
第三方库，不属于操作系统

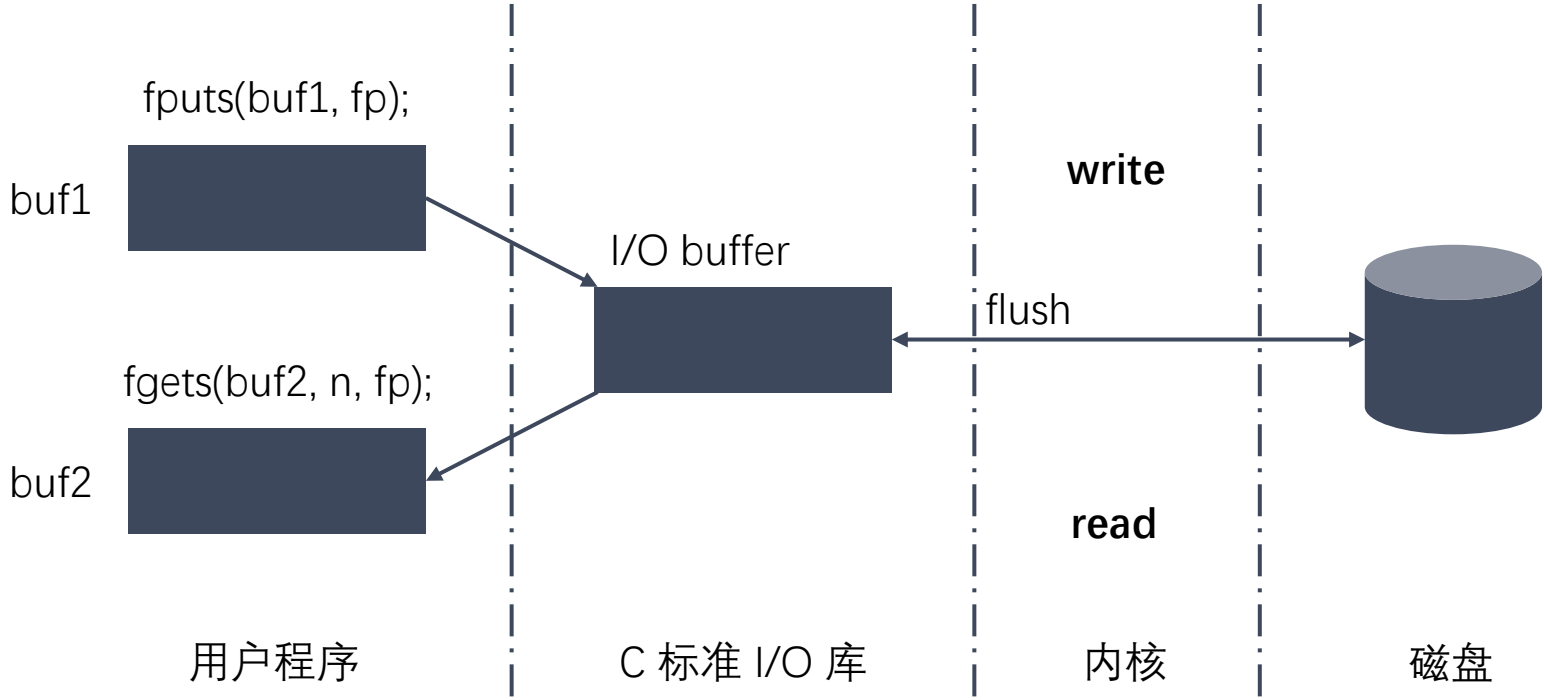
01 / 标准 C 库 IO 函数

高于Linux下的IO，
二者是调用与被调用的关系

跨平台的两种实现方式：
1.JAVA:在不同的平台有虚拟机
2.C/C++：调用不同平台的api接口

在网络通信时使用Linux系统的IO函数
对磁盘读写时为了提高效率，使用带缓冲区的标准C库函数



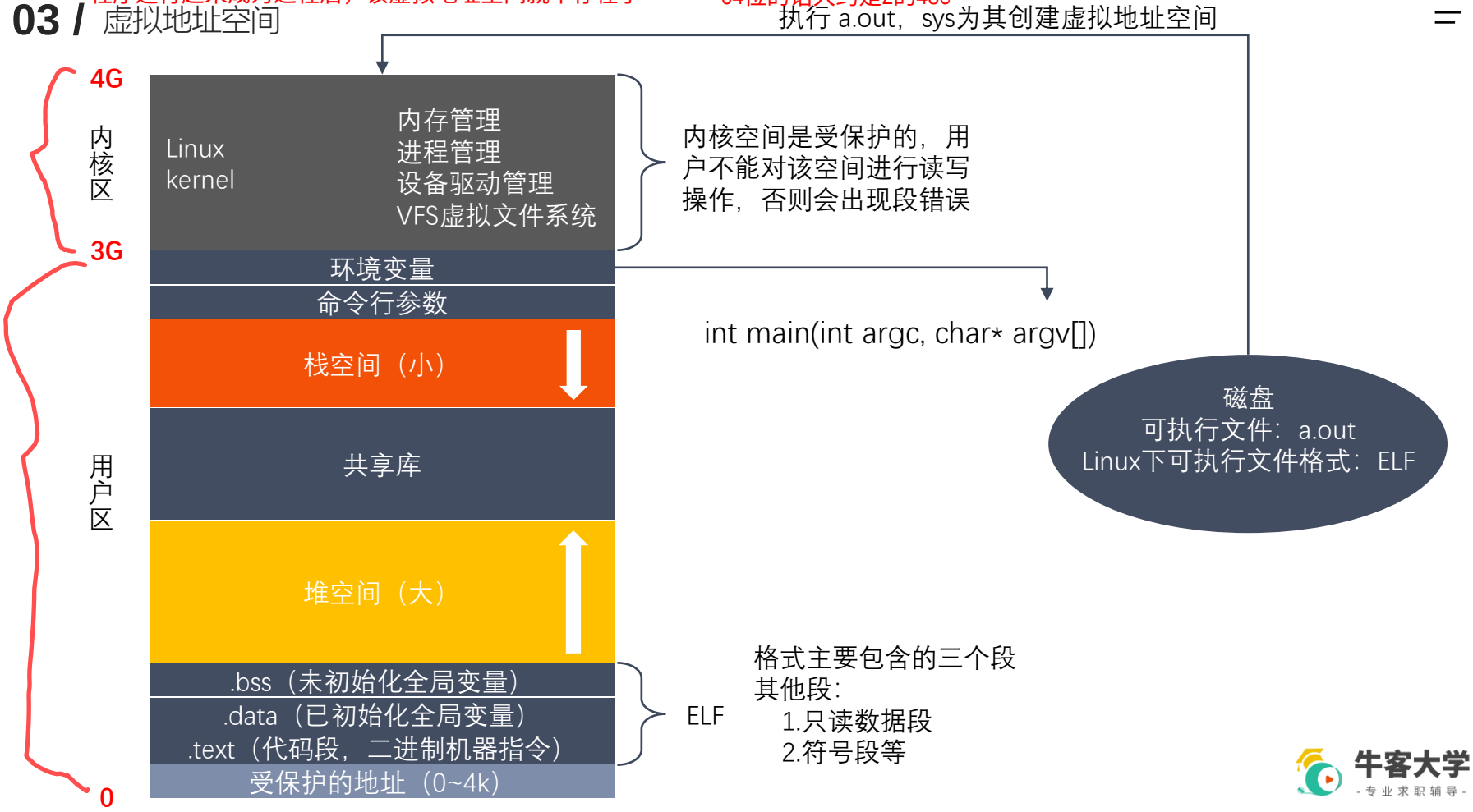


Linux系统下的I/O

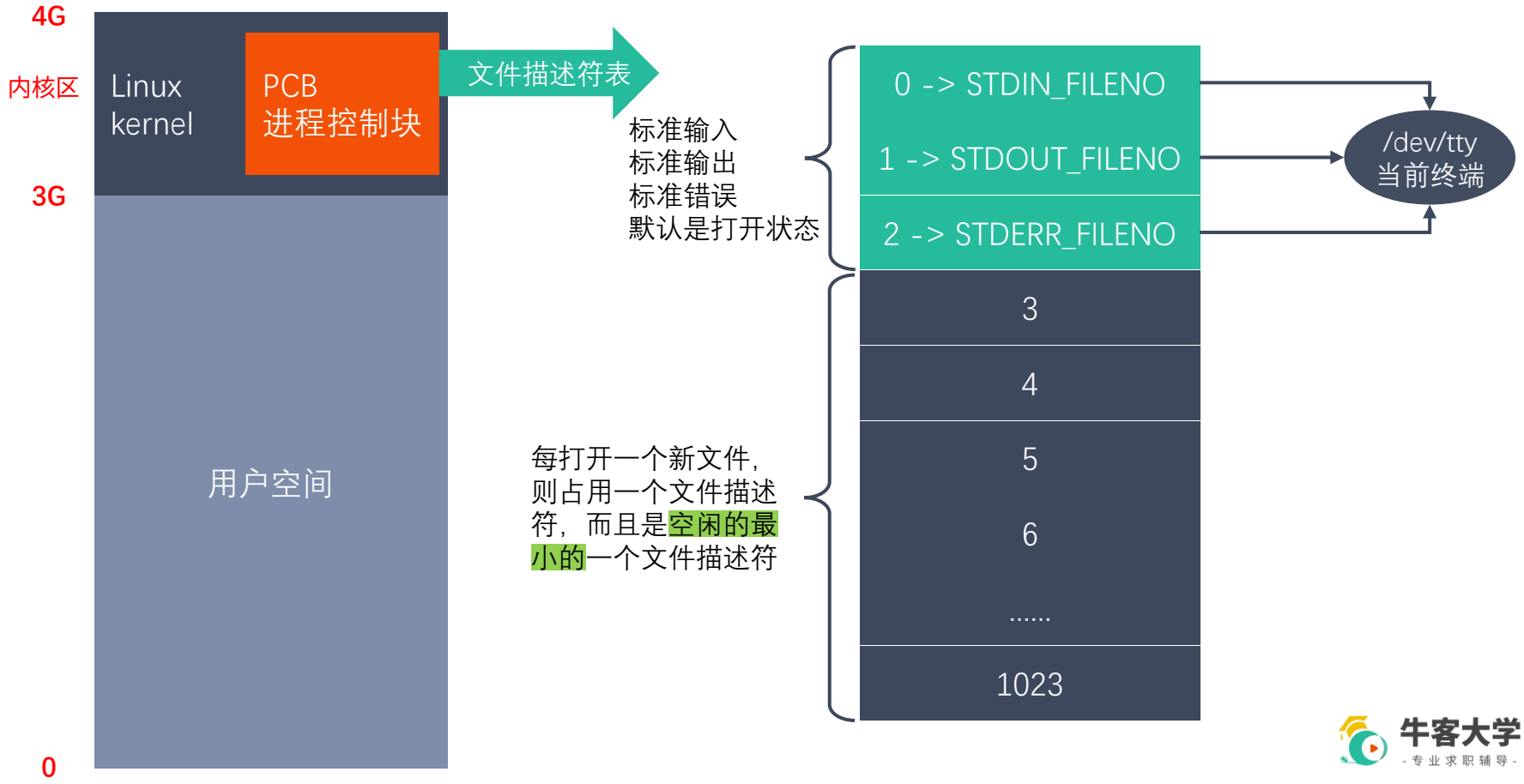
不存在，是我们想象出来的；
程序运行起来成为进程后，该虚拟地址空间就不存在了

大小：由电脑CPU决定，如果是32位的机器就是 2 的32次方，大约是 4 个G；
64位的话大约是2的48G

03 / 虚拟地址空间



04 / 文件描述符



打开一个文件

- `int open(const char *pathname, int flags);`

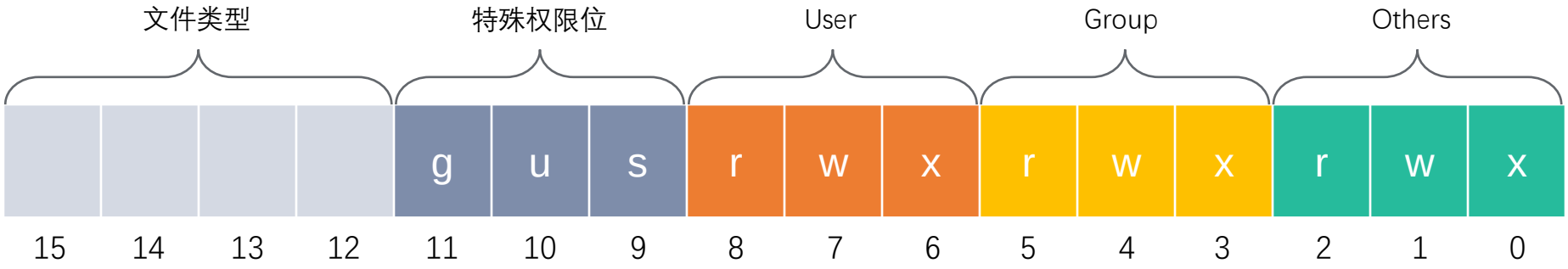
创建一个文件

- `int open(const char *pathname, int flags, mode_t mode);`
- `int close(int fd);`
- `ssize_t read(int fd, void *buf, size_t count);`
- `ssize_t write(int fd, const void *buf, size_t count);`
- `off_t lseek(int fd, off_t offset, int whence);`
- `int stat(const char *pathname, struct stat *statbuf);`
- `int lstat(const char *pathname, struct stat *statbuf);`

```
struct stat {
    dev_t       st_dev;       // 文件的设备编号
    ino_t       st_ino;       // 节点
    mode_t      st_mode;      // 文件的类型和存取的权限
    nlink_t     st_nlink;     // 连到该文件的硬连接数目
    uid_t       st_uid;       // 用户ID
    gid_t       st_gid;       // 组ID
    dev_t       st_rdev;      // 设备文件的设备编号
    off_t       st_size;      // 文件字节数(文件大小)
    blksize_t   st_blksize;   // 块大小
    blkcnt_t    st_blocks;    // 块数
    time_t      st_atime;     // 最后一次访问时间
    time_t      st_mtime;     // 最后一次修改时间
    time_t      st_ctime;     // 最后一次改变时间(指属性)
};
```

07 / st_mode 变量

setGID – 设置组id
setUID – 设置用户id
Sticky – 粘住位



- S_IFSOCK 0140000 套接字
- S_IFLNK 0120000 符号链接（软链接）
- S_IFREG 0100000 普通文件
- S_IFBLK 0060000 块设备
- S_IFDIR 0040000 目录
- S_IFCHR 0020000 字符设备
- S_IFIFO 0010000 管道
- S_IFMT 0170000 掩码

(st_mode & S_IFMT) == S_IFREG

- S_IRUSR 00400
- S_IWUSR 00200
- S_IXUSR 00100
- S_IRWXU 00700

- S_IRGRP 00040
- S_IWGRP 00020
- S_IXGRP 00010
- S_IRWXG 00070

- S_IROTH 00004
- S_IWOTH 00002
- S_IXOTH 00001
- S_IRWXO 00007

- `int access(const char *pathname, int mode);`
- `int chmod(const char *filename, int mode);`
- `int chown(const char *path, uid_t owner, gid_t group);`
- `int truncate(const char *path, off_t length);`

- `int rename(const char *oldpath, const char *newpath);`
- `int chdir(const char *path);`
- `char *getcwd(char *buf, size_t size);`
- `int mkdir(const char *pathname, mode_t mode);`
- `int rmdir(const char *pathname);`

- `DIR *opendir(const char *name);`
- `struct dirent *readdir(DIR *dirp);`
- `int closedir(DIR *dirp);`

```
struct dirent
{
    // 此目录进入点的inode
    ino_t d_ino;
    // 目录文件开头至此目录进入点的位移
    off_t d_off;
    // d_name 的长度, 不包含NULL字符
    unsigned short int d_reclen;
    // d_name 所指的文件类型
    unsigned char d_type;
    // 文件名
    char d_name[256];
};
```

d_type

- DT_BLK - 块设备
- DT_CHR - 字符设备
- DT_DIR - 目录
- DT_LNK - 软连接
- DT_FIFO - 管道
- DT_REG - 普通文件
- DT_SOCK - 套接字
- DT_UNKNOWN - 未知

- `int dup(int oldfd);`

  复制文件描述符

- `int dup2(int oldfd, int newfd);`

  重定向文件描述符

- `int fcntl(int fd, int cmd, ... /* arg */ );`

  复制文件描述符

  设置/获取文件的状态标志

牛客大学
-专业求职辅导-

THANKS

关注【牛客大学】公众号
回复“牛客大学”获取更多求职资料