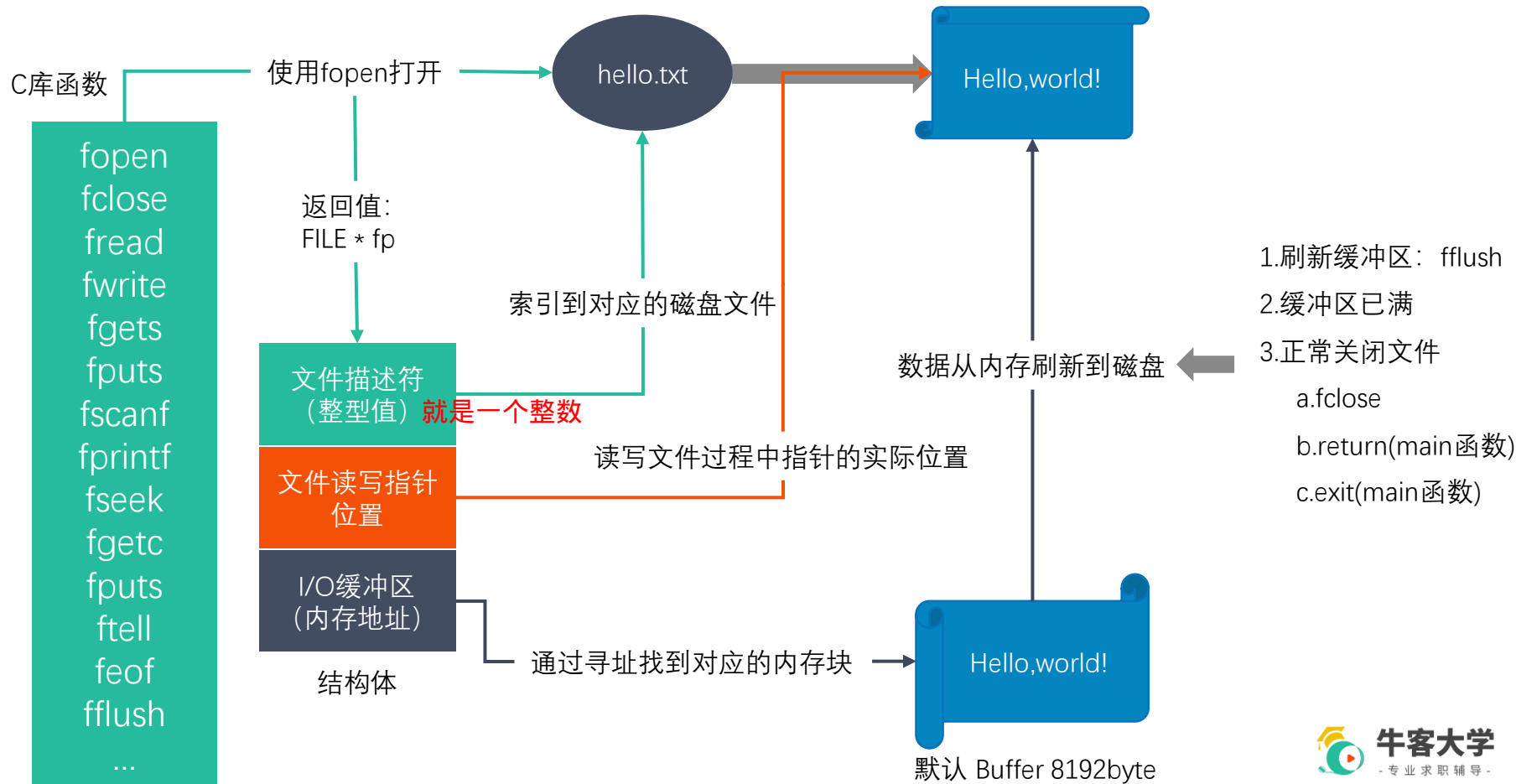


文件IO

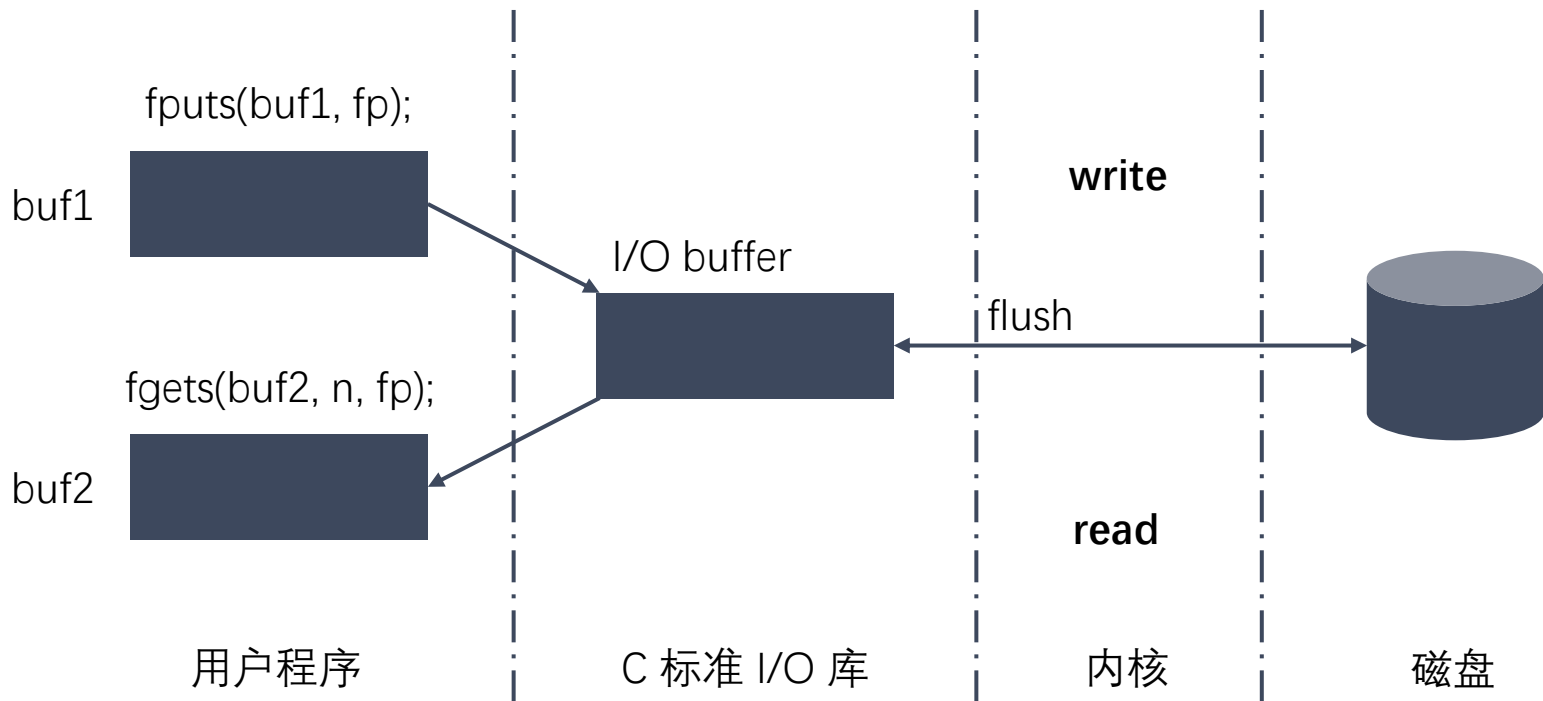
01 / 标准 C 库 IO 函数

=



02 / 标准 C 库 IO 和 Linux 系统 IO 的关系

=



03 / 虚拟地址空间

执行 a.out, sys为其创建虚拟地址空间

二

4G

内核区

Linux kernel
内存管理
进程管理
设备驱动管理
VFS虚拟文件系统

内核空间是受保护的，用户不能对该空间进行读写操作，否则会出现段错误

3G

环境变量

命令行参数

栈空间 (小)



共享库

用户区

堆空间 (大)



.bss (未初始化全局变量)

.data (已初始化全局变量)

.text (代码段, 二进制机器指令)

受保护的地址 (0~4k)

ELF

格式主要包含的三个段
其他段:

1. 只读数据段
2. 符号段等

int main(int argc, char* argv[])

磁盘

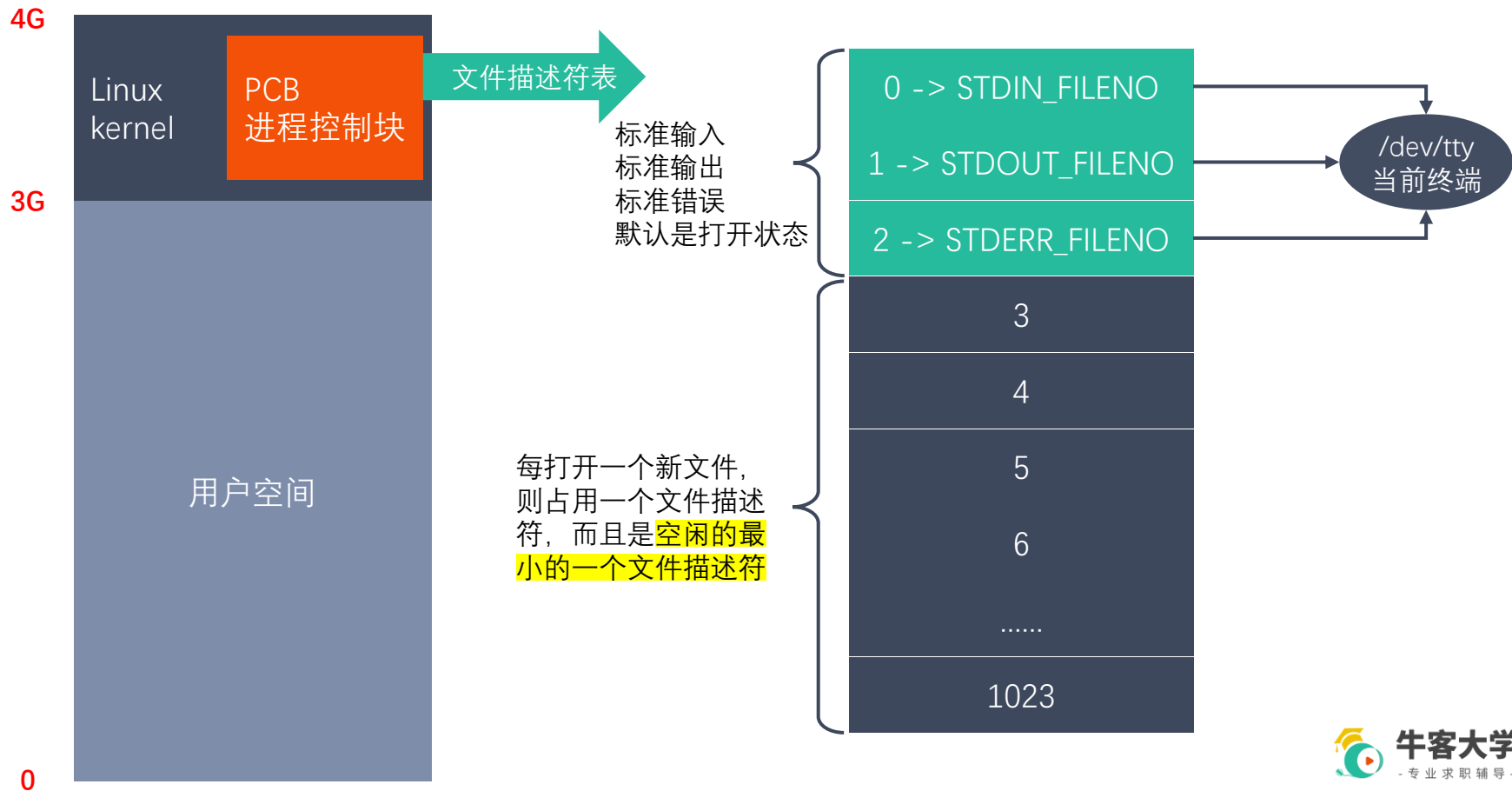
可执行文件: a.out

Linux下可执行文件格式: ELF

0

04 / 文件描述符 文件描述符 = 文件句柄 = 文件指针 = 文件引用

=



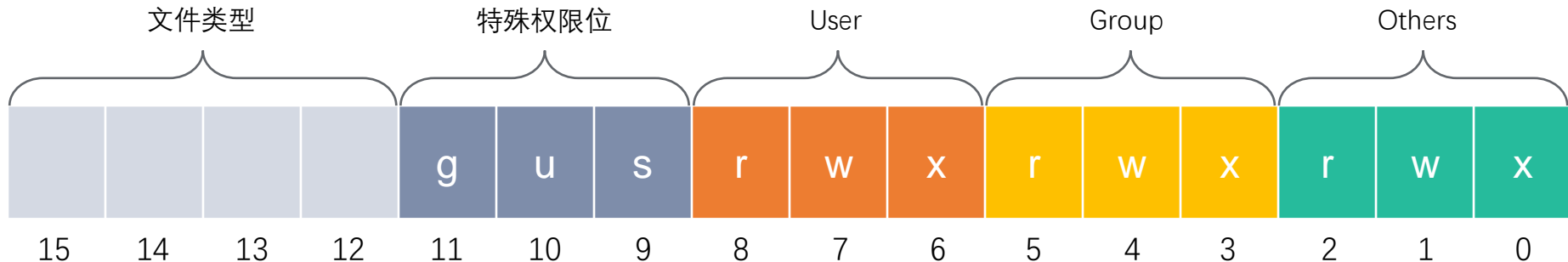
- `int open(const char *pathname, int flags);`
- `int open(const char *pathname, int flags, mode_t mode);`
- `int close(int fd);`
- `ssize_t read(int fd, void *buf, size_t count);`
- `ssize_t write(int fd, const void *buf, size_t count);`
- `off_t lseek(int fd, off_t offset, int whence);`
- `int stat(const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

- 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

(`st_mode & S_IFMT`) == `S_IFREG`

- `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