系统IO 10.1 -10.3

1. 文件读写

   Linux shell 创建的**每个进程开始时都有三个打开的文件：标准输入（描述符为 0），标准输出（描述符为 1），标准错误（描述符为 2）**read 函数从描述符为 fd 的当前文件位置复制最多 n 个字节到内存位置 buf。write 函数从内存位置 buf 复制最多 n 个字节到描述符 fd 的当前文件位置。

C

```c
//文件读写的一些操作
int open(char *filename, int flags, mode_t mode); //若成功则返回文件描述符，若出错返回 -1
int close(int fd); // fd 是一个文件描述符
ssize_t read(int fd, void *buf, size_t n);          //若成功则返回读的字节数，若 EOF 则返回 0，若出错返回 -1
ssize_t write(int fd, const void *buf, size_t n);   //若成功返回写的字节数，若出错返回 -1
int stat(const char *filename, struct stat *buf);
int fstat(int fd, struct stat *buf);
lseek(); //更改当前文件位置

```

2. 文件

1. **普通文件**：包含任意数据。应用程序常会将普通文件进一步分为文本文件（只包含 ASCII 或 Unicode 字符的文件）和二进制文件（其他所有文件）。对内核而言，两者没有区别。
2. **目录：**包含一组链接的文件，其中每一个链接都是一个文件名。**每个目录至少含有两个条目：“.” 表示到该目录自身的链接，“..” 表示到目录层次结构中父目录的链接。**
3. **套接字**：用来与另一个进程进行网络通信的文件。

Linux 将所有的文件组织成一个目录层次结构，由**根目录 (/)** 确定

3. 文件分类

1. **普通文件**：包含任意数据。应用程序常会将普通文件进一步分为文本文件（只包含 ASCII 或 Unicode 字符的文件）和二进制文件（其他所有文件）。对内核而言，两者没有区别。
2. **目录：**包含一组链接的文件，其中每一个链接都是一个文件名。**每个目录至少含有两个条目：“.” 表示到该目录自身的链接，“..” 表示到目录层次结构中父目录的链接。**
3. **套接字**：用来与另一个进程进行网络通信的文件。

Linux 将所有的文件组织成一个目录层次结构，由**根目录 (/)** 确定

10.4 共享文件

内核用三个相关的数据结构来表示打开的文件

- 描述符表（descriptor table）

  每个进程都有它独立的描述符表，它的表项是由进程打开的文件描述符来索引的。每个打开的描述符表项指向文件表中的一个表项。内核（kernel）利用文件描述符（file descriptor）来访问文件。文件描述符是非负整数。打开现存文件或新建文件时，内核会返回一个文件描述符。读写文件也需要使用文件描述符来指定待读写的文件。

  1.进程级的文件描述符表；

  2.系统级的打开文件描述符表；

  3.文件系统的i-node表。

- 文件表（file table）

  打开文件的集合是由一张文件表来表示的，所有的进程共享这张表。每个文件表的表项组成包括当前的文件位置、引用计数（reference count）（即当前指向该表项的描述符表项数），以及一个指向v-node表中对应表项的指针。关闭一个描述符会减少相应的文件表表项中的引用计数。内核不会删除这个文件表表项，直到它的引用计数为零。

Linux 文件系统概念区分

文件描述符

内核为了高效管理已被打开的文件所创建的索引，其是一个非负整数（通常是小整数），用于指代被打开的文件，所有执行I/O操作的系统调用都通过文件描述符。程序刚刚启动的时候，0是标准输入，1是标准输出，2是标准错误。如果此时去打开一个新的文件，它的文件描述符会是3。

文件表项

文件表项的数据结构就是file结构体，而在实际上内核中也并不存在这样一张文件表，只是每个打开的文件都对应一个file结构体，也就是一个文件表项，**打开文件描述符表struct file * fd_array[NR_OPEN_DEFAULT]**数组中的每一项都会指向这样一个文件表项