武汉纺织大学
WUHAN TEXTILE UNIVERSITY

崇真尚美

Linux系统编程

数学与计算机学院

教师：朱萍

本课程的主要内容

- Linux基础知识（4学时）
- Linux文件系统（6学时）
- Linux系统管理（4学时）
- Linux网络管理及应用（6学时）
- Linux Shell编程（6学时）
- Linux下C编程（4学时）
- Linux下进程控制（6学时）
- Linux下线程控制（4学时）
- Linux文件接口（4学时）

Linux文件IO

Linux文件IO函数简介

文件IO函数：新建creat

文件IO函数：打开open

文件IO函数：关闭close

文件IO函数：读read

文件IO函数：写write

文件IO函数：定位lseek

Linux文件IO函数简介

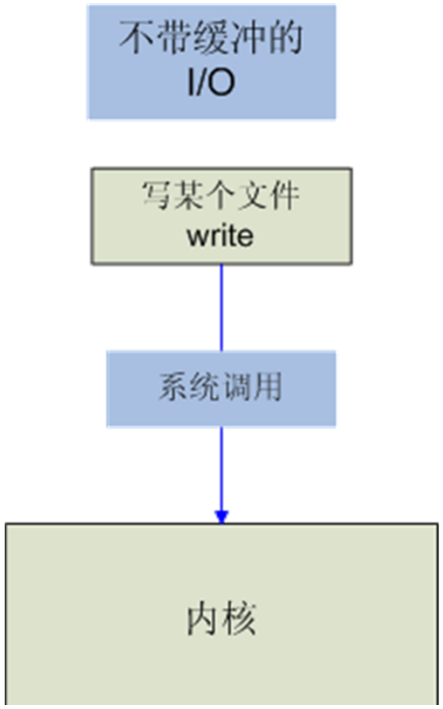
- “一切都是文件”是linux的基本哲学思想；
- 这里的“一切”确确实实意味着一切。硬盘，硬盘分区，设备接口，网络连接，所有这些都是文件。甚至目录也是文件。
- 除了标准的文件和目录，Linux 还可以辨识很多别的类型的文件。注意这里说的文件类型不是文件内容的类型：不论是 PNG 图象，二进制文件还是其他什么，文件就是一串数据流。通过内容区分文件类型是留给应用程序的任务。

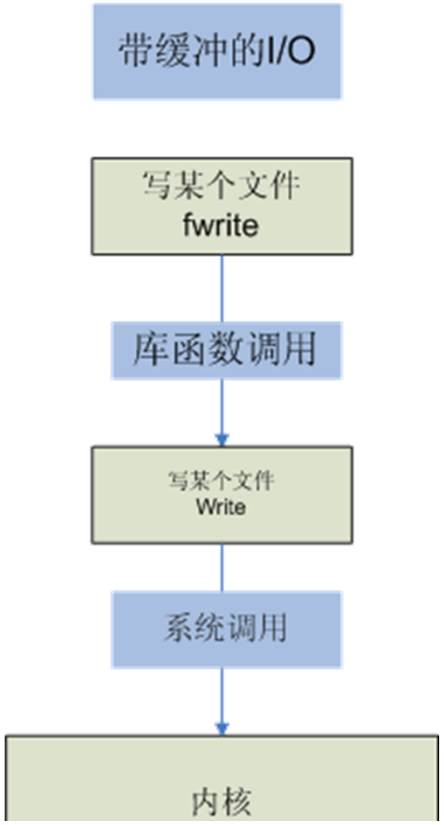
Linux文件IO函数简介

- 文件描述符：当打开一个文件时，内核就会分配给你一个非负整数，通过这个整数便可读写文件；
- 当一个进程启动时，系统会自动分配三个文件描述符，分别是：
  - 标准输入：文件描述为0
  - 标准输出：文件描述为1
  - 标准错误输出：文件描述符为2

Linux文件IO函数简介

- Linux中文件访问的IO函数分为两类：
  - 不带缓冲的I/O,直接去找系统调用，速度快。（open read write lseek close），这些函数不是ANSI C的组成部分，是POSIX的一部分。
  - 带缓冲的I/O,在系统调用前采用一定的策略，速度慢，比不带缓冲的I/O安全。(fopen fread fwrite)





Linux文件IO

Linux文件IO函数简介

文件IO函数：新建creat

文件IO函数：打开open，关闭close

文件IO函数：读read

文件IO函数：写write

文件IO函数：定位lseek

新建文件：creat

- 语法：

```
#include <fcntl.h>

int creat(const char *pathname, mode_t mode);
```

结尾
没有e

- 成功，返回文件描述符，不成功返回-1
- pathname：文件名，可以是相对路径名也可以是绝对路径名

新建文件：creat

- mode指定新建文件的存取权限，见P223/表7.8

- 更常用的用法是用数字组合，含义与chmod命令参数同
  - creat (“file1.txt”, 0755);
  - creat (“/home/jkx/work/chapt8/file2.txt”,0644);

- 在/home/jkx目录下创建一个文件主可读可写，其它人只读的文件myfile.txt，不具有可执行权限

Linux文件IO

Linux文件IO函数简介

文件IO函数：新建creat

文件IO函数：打开open，关闭close

文件IO函数：读read

文件IO函数：写write

文件IO函数：定位lseek

打开文件：open

- open()函数用于打开或创建文件，在打开或创建文件时可以指定文件的属性及用户的权限等各种参数。
- 语法：
  - open(const char *pathname, int flags);
  - open(const char *pathname, int flags, mode_t mode);
  - 当打开存在的文件时，用上面的，打开不存在（也就是新建并打开）文件时，用下面的。

打开文件：open

- 第二个参数flags可以是下表的组合。
- 第三个参数同creat的参数。

- 函数返回：成功，返回文件fd，失败返回-1

所需头文件	#include <sys/types.h> /* 提供类型 pid_t 的定义 */ #include <sys/stat.h> #include <fcntl.h>	
函数原型	int open(const char *pathname, int flags, int perms)	
函数传入值	pathname	被打开的文件名（可包括路径名）
	flag：文件打开的方式	O_RDONLY：以只读方式打开文件
		O_WRONLY：以只写方式打开文件
		O_RDWR：以读写方式打开文件
		O_CREAT：如果该文件不存在，就创建一个新的文件，并用第三个参数为其设置权限
		O_EXCL：如果使用 O_CREAT 时文件存在，则可返回错误消息。这一参数可测试文件是否存在。此时 open 是原子操作，防止多个进程同时创建同一个文件。
		O_NOCTTY：使用本参数时，若文件为终端，该终端不会成为调用 open()的那个进程的控制终端
		O_TRUNC：若文件已经存在，那么会删除文件中的数据并且设置文件大小为 0。
		O_APPEND：以添加方式打开文件，即将写入的数据添加到文件的末尾，即将写入的数据添加
	perms	被打开文件的存取权限。 可以用一组宏定义：S_I(R/W/X)(USR/GRP/OTH) 其中 R/W/X 分别表示读/写/执行权限 USR/GRP/OTH 分别表示文件所有者/文件所属组/其他用户。 例如 S_IRUSR|S_IWUSR 表示设置文件所有者的可读可写属性。八进制表示法中 600 也表示同样的权限。
函数返回值	成功：返回文件描述符 失败：-1	

第三个参数是在第二个参数中有**O_CREAT**时才用作用。若没有，则第三个参数可以不写。

关闭文件close

- 语法
  - close(int fd);

- 返回：成功返回0，失败返回-1

- 注意：文件的打开和关闭通常是成对，文件打开后可以对文件处理，处理完毕后需要对文件关闭，不关闭容易造成文件错误。

Linux文件IO

Linux文件IO函数简介

文件IO函数：新建creat

文件IO函数：打开open，关闭close

文件IO函数：读read

文件IO函数：写write

文件IO函数：定位lseek

文件IO函数：读read

- 语法
  - sisze_t read(int fd, void *buf, size_t count)
  - 从文件fd中读count个字节数据放到缓冲区buf中
- 返回：成功返回读取到的字节数，失败返回-1
- 课堂练习：
  - 自己先创建一个文件myfile.txt，里面写入内容，假设不超过512字节。
  - 编程readtest.c，实现：以只读方式打开文件myfile.txt，并将其内容输出到屏幕上。

readtest.c

```c
void main()
{
        char buf[512];
        int fd;
        int count;

        fd = open(     );
        count=read(  );
        printf("myfile.txt content:\n");
        printf(    );
        close(fd);
}
```

Linux文件IO

Linux文件IO函数简介

文件IO函数：新建creat

文件IO函数：打开open，关闭close

文件IO函数：读read

文件IO函数：写write

文件IO函数：定位lseek

文件IO函数：写write

- 语法
  - ssize_t write(int fd, void *buf, size_t count)
  - 将缓冲区buf中的count个字节的数据写入文件fd中
- 返回：成功返回写入的字节数，失败返回-1
- 课堂练习：
  - 编程writetest.c，实现：以创建新文件方式和可写方式打开myfile2.txt，并将myfile.txt的内容复制到文件myfile2.txt中。

writetest.c

```
void main()
{
        char buf[512];
        int fd1, fd2;
        int count;

        fd1 = open(  );
        fd2 = open(  );
        count = read(  );
        write(  );
        close(fd1);
        close(fd2);
}
```

Linux文件IO

Linux文件IO函数简介

文件IO函数：新建creat

文件IO函数：打开open，关闭close

文件IO函数：读read

文件IO函数：写write

文件IO函数：定位lseek

文件IO函数：定位lseek

- 语法
  - off_t lseek(int fd, off_t offset, int whence)
  - 对文件fd的读写指针进行设置。
  - whence可以是以下三种宏之一：
    - SEEK_SET:offset就是新的读写指针位置
    - SEEK_CUR：新的位置为当前读写位置后增加offset
    - SEEK_END:新的读写位置为文件尾后再增加offset
- 返回：成功返回当前读写位置，也就是距离文件头的偏移量，失败返回-1

lseek的应用

- lseek(fd, 0, SEEK_SET)
  - 读写位置移到文件开头
- lseek(fd, 0, SEEK_END)
  - 读写位置移到文件开头
  - 通过该返回值可以获得文件的大小

课堂综合作业

- 编写一个程序，任意打开一个文件，获取该文件的大小，然后将该文件切分成三等份，分别存到三个文件中。fd -> fd1, fd2, fd3
  - Len=lseek(fd,0,SEEK_END);
  - Lseek(fd,0,SEEK_SET)
  - Read(fd, buf , len/3);
  - Write(fd1)
  - Read(fd,buf,len/3)
  - Write(fd2)
  - Read(fd,buf,len/3)
  - Write(fd3)

武漢紡織大學
WUHAN TEXTILE UNIVERSITY

崇真尚美

Thank You !