电子科技大学

University of Electronic Science and Technology of China

Linux操作系统编程

实验二 | 文件I/O实验

主讲老师：杨珊

实验二 | 文件I/O实验目的

目的一：掌握Linux应用程序命令行参数传递方法

目的二：掌握POSIX API中文件I/O操作方法

- ✓打开文件　✓创建文件
- ✓关闭文件　✓读写文件
- ✓确定和改变文件当前位置

实验二｜文件I/O实验内容

仿写cp命令的部分功能（编写mycp程序）：

1、将源文件复制到另外一个文件（将test1.text复制成test2.txt)

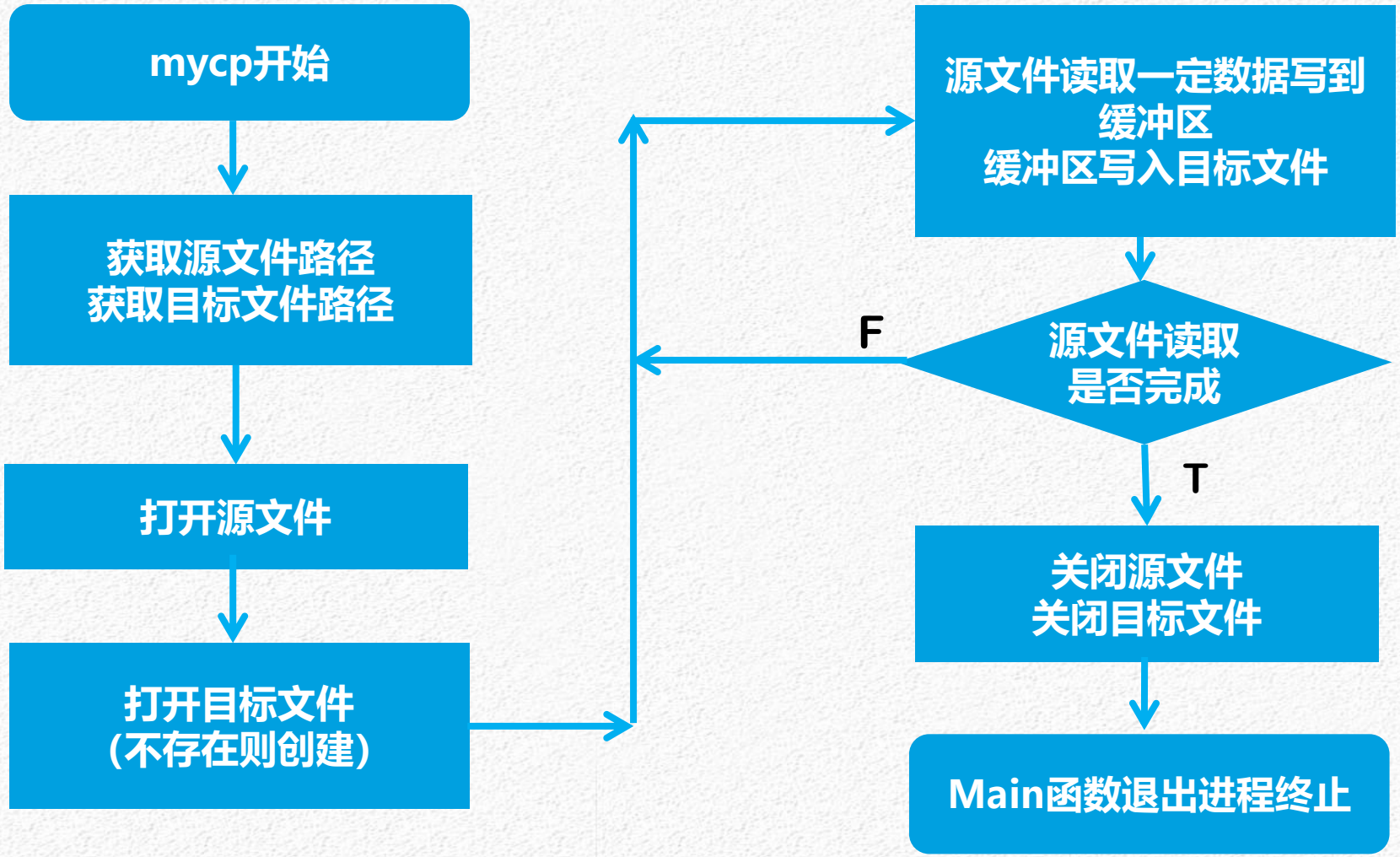
[test@linux test]$ ./mycp /home/test1.txt /usr/test2.txt

2、将源文件复制到另外一个目录（将test1.txt复制到/tmp目录）

[test@linux test]$ ./mycp /home/test1.txt /tmp

源文件路径和目标文件路径通过命令行参数来指定

实验二 | 程序流程


mycp开始
获取源文件路径
获取目标文件路径
打开源文件
打开目标文件
(不存在则创建)
源文件读取一定数据写到
缓冲区
缓冲区写入目标文件
源文件读取
是否完成
F
T
关闭源文件
关闭目标文件
Main函数退出进程终止

命令行参数：在命令（也是可执行文件名）后，传递进入程序的参数（字符串形式）

cp命令：根据接收的命令行参数进行复制

- ✓ cp /usr/local/src/main.c /r
- ✓ cp /usr/local/src/mai
- ✓ cp –r /usr/local/src /root (-

权限不够的情况，采用sudo cp XX XX的方式。sudo，在linux中输入sodu就是调用这个程序提升权限

实验二｜实验原理｜命令行参数获取

程序命令行参数： 操作系统启动C程序->调用C启动例程->例程获取命令行参数->main函数接收

实验二｜实验原理｜命令行参数获取示例

```
#include <stdio.h>
int main(int argc,char *argv[])
{
  int i;
  for(i=0;i<argc;i++)
    printf("Argv %d is %s.\n",
           i,argv[i]);
  return 0;
}
```

将左边代码编译为hello.o

命令行中执行两种命令：

./hello.o

./hello.o aaa bbb ccc ddd eee

实验二｜实验原理｜命令行参数获取示例

```
root@yan934: /home/dir1
root@yan934:/home/dir1# ./hello.o
Argv 0 is ./hello.o
root@yan934:/home/dir1# ./hello.o aaa bbb ccc ddd eee
Argv 0 is ./hello.o
Argv 1 is aaa
Argv 2 is bbb
Argv 3 is ccc
Argv 4 is ddd
Argv 5 is eee
root@yan934:/home/dir1#
```

argv[0]->可执行文件名; argv[1]-argv[5] ->命令行参数

以字符串的形式传递（空格作为命令行参数之间的分隔）

实验二｜实验原理｜打开文件

头文件： fcntl.h

函数： int open(const char *pathname, int oflag, ...);

文件的路径

...：变参创建新文件时使用

作用： 打开或者创建一个文件，返回值是打开文件的**文件描述符**

Oflag值	含义
O_RDONLY	只读打开
O_WRONLY	只写打开
O_RDWR	读、写打开

Oflag值	含义
O_APPEND	每次写都加到文件尾
O_CREAT	若此文件不存在则创建它，此时需要第三个参数mode，该参数约定了所创建文件的权限，计算方法 mode&~umask
O_EXCL	如同时指定了O_CREAT，此指令会检查文件是否存在，若不存在则建立此文件；若文件存在，此时将出错
O_TRUNC	如果此文件存在，并以读写或只写打开，则文件长度0

实验二｜实验原理｜读取文件

头文件：unistd.h

函数：ssize_t read(int filedes, void *buf, size_t nbytes);

作用：从打开的文件中读数据

返回值：

- ✓ **成功**：返回实际读取的字节数（例外情况：读普通文件，未到要求的字节数前就到达文件末尾，此时返回字节数不等于要求字节数）
- ✓ **出错**：返回-1，**原因**：磁盘满、权限问题、超文件长度限制

注：读操作完成后，文件的当前位置**变为之前位置+实际读取字节数**

实验二｜实验原理｜写文件

头文件： unistd.h

函数： ssize_t write(int filedes, const void *buf, size_t nbytes);

作用： 向打开的文件中写数据

返回值：

- ✓ **成功：** 写入成功返回实际写入字节数
- ✓ **出错：** 返回-1，**原因：** 磁盘满、权限问题、超文件长度限制

注： 写操作完成后，文件的当前位置**变为之前位置+实际写入字节数**

实验二 | 实验原理 | 关闭文件

头文件： unistd.h

函数： int close(int filedes)

作用： 关闭打开的文件

返回值：

- ✓ **成功：** 返回0
- ✓ **出错：** 返回-1

注： 关闭后，不能通过该文件描述符操作该文件

实验二｜实验原理｜文件定位

头文件： unistd.h

函数： off_t lseek(int filedes, off_t offset, int whence);

作用：设置或查询文件当前位置

- ✓每个打开文件都有一个“**文件当前位置**”
- ✓打开文件时默认当前位置为**文件头（0）**
- ✓打开时如指定**O_APPEND**选项，则当前位置变为**文件尾（文件长度）**
- ✓回忆：读写操作完成后文件的当前位置**变为之前位置+实际读写字节数**

实验二｜实验原理｜文件定位

文件定位lseek函数的参数whence及offset(offset可正可负)

- ✓ whence=SEEK_SET, 文件当前位置=文件头+offset（字节）
- ✓ whence=SEEK_CUR, 文件当前位置=文件当前位置+offset（字节）
- ✓ whence=SEEK_END, 文件当前位置=文件尾+offset（字节）

```
//come from /usr/include/unistd.h
/* Values for the WHENCE argument to lseek.    */
#ifndef    _STDIO_H        /* <stdio.h> has the same definitions.*/
# define SEEK_SET     0    /* Seek from beginning of file.    */        //文件起始位置
# define SEEK_CUR     1    /* Seek from current position.    */        //当前位置
# define SEEK_END     2    /* Seek from end of file.    */             //文件结束位置
```

调用POSIX API会发生**错误**，需要定位产生问题的更具体**原因**

系统错误码： POSIX规范定义的记录系统错误的标准码，

使用方式：通过全局变量errorno获取该错误码，记录最后一次错误

- ✓ extern errorno
- ✓ **头文件：**errorno.h

```
#define EPERM 1 /* Operation not permitted */
#define ENOENT 2 /* No such file or directory */
#define ESRCH 3 /* No such process */
#define EINTR 4 /* Interrupted system call */
#define EIO 5 /* I/O error */
#define ENXIO 6 /* No such device or address */
#define E2BIG 7 /* Argument list too long */
```

```
#define ENOEXEC 8 /* Exec format error */
#define EBADF 9 /* Bad file number */
#define ECHILD 10 /* No child processes */
#define EAGAIN 11 /* Try again */
#define ENOMEM 12 /* Out of memory */
#define EACCES 13 /* Permission denied */
#define EFAULT 14 /* Bad address */
```

将错误码**转换**为易读的错误信息：

函数1： char *strerror(int errnum);

头文件： string.h

作用： 将errnum转换为对应错误信息（字符串）

函数2： void perror(const char * msg);

头文件： stdio.h

作用： 先输出msg字符串，然后输出当前errorno对应错误信息

实验二 | 实验技巧 | 演示程序

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <string.h>
#include <errno.h>
extern errorno;
int main(int argc,char *argv[])
{
  int i=open("/usr/src/s.t",0);
  perror(argv[0]);
  printf(strerror(errno));
  printf("\n");
  return 0;}
```

将左边代码编译为error.o

使用./error.o运行

得到如下结果：

error.o运行结果

```
root@yan934:/home/dir1# ./error.o
./error.o: No such file or directory
No such file or directory
root@yan934:/home/dir1#
```

实验二 | 文件I/O实验内容| 扩展内容

实现基本功能基础上，针对特殊情况及边界条件，进行流程完善与优化：

✓目标文件存在时，提示“是否覆盖”or“是否合并”等提示信息，并实现覆盖或者目标文件与源文件合并的功能（目标文件尾部追加写入）

✓源文件不存在时给出错误提示信息

✓源文件是目录时给出错误提示信息

电子科技大学

University of Electronic Science and Technology of China

Linux操作系统编程

感谢观看

主讲老师：杨珊