1. 系统api与库函数的关系

（一）调用库函数的流程

当我们使用一些库函数，如printf(“hello world”)时，调用了文件指针 FILE \*stdout, 该文件指针的组成为：

fd 文件描述符

f\_pos 位置

buffer 8k的缓冲区

如，调用printf(“hello\n”)后，系统会调用write(1, “hello\n”, 6)，接着进入内核调用sys\_write(“hello\n”)，再通过设备驱动显示在显示器中。

（二）内核中的管理

内核中有虚拟文件系统，内存管理，进程管理，设备管理，PCB进程控制块，文件描述符表。

文件描述符表中存的就是文件指针中的文件描述符，如描述符0对应的是FILE \* stdin，1对应的是 FILE \*stdout，2对应的是 FILE \* stderr，因此，上面调用printf(“hello\n”)，系统会自动调用write(1, “hello\n”, 6)中的1指的就是FILE \*stdout，写到标准输出上。

1. 文件IO
2. open函数

通过查看man文档的第二页来了解open函数：man 2 open

该函数用来打开文件。

1. 函数原型

int open(const char \*pathname, int flags);

int open(const char \*pathname, int flags);

1. 函数参数

pathname:文件名

flags:

必选项：

O\_RDONLY:只读

O\_WRONLY:只写

O\_RDWR:读写

可选项：

O\_APPEND:追加

O\_CREATE ; 创建文件

O\_NONBLOCK:非阻塞

1. 函数的返回值

成功时返回最小的可用文件描述符，失败时返回-1.设置errno。

1. close函数
2. 用途

关闭一个文件描述符

1. 函数原型

int close(int fd);

1. 函数返回值

成功时返回0，错误时返回-1、设置errno。

1. read

读取文件

1. 函数原型

size\_t read(int fd, void\* buf, size\_t count);

1. 函数参数

fd : 文件描述符

buf: 缓冲区

count ：缓冲区大小

1. 函数返回值

成功时返回读到的大小

失败时返回-1，并设置errno

返回0表示读到了文件末尾

1. write

向文件中写入。

1. 函数原型

size\_t write(int fd, const void\* buf, size\_t count);

1. 函数参数

fd: 文件描述符

buf: 缓冲区

count: 缓冲区大小

1. 函数返回值

成功时返回写入的字节数

失败时返回-1，并设置errno

返回0代表未写入

1. lseek

移动文件的读写位置。

1. 函数原型

off\_t lseek(int fd, off\_t offset, int whence);

1. 函数参数

fd: 文件描述符

offset: 偏移量

whence:

SEEK\_SET: 文件开始位置

SEEK\_CUR: 当前位置

SEEK\_END: 文件的结尾

1. 函数返回值

成功时返回当前位置到开始的长度

失败时返回-1.并设置errno

1. 函数的作用

移动文件读写位置

计算文件大小

1. lseek实现计算文件大小

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<sys/stat.h

#include<fcntl.h>

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

if(argc != 2){

printf(“./a.out filename\n”);

return -1;

}

int fd = open(argv[1], 0\_RDONLY);

int ret = lseek(fd, 0, SEEK\_END);

printf(“file size is %d\n”, ret);

close(fd);

return 0;

}