**IO课程**

文件管理

进程管理

设备管理

内存管理

网络管理

unistd.h stdio.h

write();<unistd.h>

printf()调用write(),write()为内核调用函数

一：IO课程

1.文件IO

2.标准IO

3.目录IO

一：文件IO

open函数

内核层输入应用层input read函数

应用层输出到内核层output write函数

close函数

umask 掩码

1：int open—打开或创建一个文件

open（char\*，flag，mode）<fcntl.h>

char\*—文件名和路径。

Flag—打开文件的方式

Mode—创建文件的权限

|  |  |
| --- | --- |
| Flag | 功能 |
| O\_RDONLY | 只读 |
| O\_WRONLY | 只写 |
| O\_RDWR | 读写 |
| O\_CREAT | 创建一个文件 |
| O\_TRUNC | 打开文件,删除已经存在的内容。 |
| O\_EXCL | 使用O\_CREAT是文件存在则返回错误，可测试文件是否存在 |
| O\_APPEND | 追加方式打开文件。 |

返回值—文件描述符。

2:ssize\_t read(int fd,void \*buf,size\_t count)<unistd.h>

3:ssize\_t write(int fd,void \*buf,size\_t count)<unistd.h>

4:off\_t lseek(int handle,off\_t offset,int from where);<unistd.h> <sys/types.h>

SEEK\_SET 从文件开头读

SEEK\_CUR 从当前位置读

DEEK\_END 从文件结尾读

返回值 成功则为距离开头的字节数。

失败为-1

二：标准IO <stdio.h>

getchar(),putchar() 一个字符。

gets(buf),puts(buf) 一串字符。

scanf(),printf() 一个字符，一串字符都可以。

三个缓存的概念。

用户空间的换成【数组】 user buffer

库缓存 lib buffer

内核缓存 kernel buffer

printf 遇到\n的时候lib buffer写到kernel buffer

lib buffer写满时，写到kernel buffer

三类库函数

全缓存

读 fread

写 fwrite

读 fgetc getc getchar

写 fputc putc putchar

行缓存

读 fgets gets printf fpringf sprint

写 fputs puts scanf

无缓存

stderr

FILE \*fopen(const char \*path,const char \*mode)

FILE: struct\_IO\_FILE,在/usr/include/libio.h

包含读写缓存的首地址、大小、位置指针

标准的输入流：stdin 0

标准的输出流：stdout 1

标准的出错流：stderr 2

mode

b 二进制文件

r 只读方式打开文件，文件必须存在。

w 以只写方式打开，不存在则创建。原内容清零。

a 以只写方式打开，不存在则创建。原内容追加。

+ 以读写方式打开。

默认权限0666

fclose

fflush

fseek(FILE \*fd,offset,where)返回值成功0，失败-1

rewind(FILE \*fd)只想文件开头。

ftell(FILE \*fd)去的文件当前位置，返回值为当前位置。

char \*fget(char \*s,int size,FILE \*stream)

返回值是缓存地址，读完为NULL

int fputs(const char \*s,FILE \*stream)

char \*gets(char \*s);

gets()不将新行符写入

int puts(const char \*s);

puts()加入新行符

int fprintf(FILE \*stream,””,…);

int sprint(str \*,””,…);

fputc(char,FILE \*fd);

int fgetc(FILE \*fd);