2.1 Linux 文件 io 接口_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 2.1 Linux 文件 io 接口涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

1.Linux 文件 io 接口 – open/close

- 在 Linux 系统下, 用于对文件操作的库函数叫做 文件 I/O
- 主要包括 open()/close()/read()/write() /lseek（） 相应的系统调用 (准确说法是对系统调用的封装的库函数)
- 文件描述符是一个 非负整数 , 当打开一个已存在文件或者创建一个新文件时, 内核向进程返回一个文件描述符
- 每个程序运行后, 操作系统会默认打开三个文件 标准输入 标准输出 标准错误输出, 文件描述符分别为 0 , 1 , 2
- 标准输入对应的设备一般为键盘
- 标准输出与标准错误输出设备一般为显示器
- 示例 : 通过 write 函数 (后面会详细讲解) 使用标准输出来打印 Hello world

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>

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

    write(0,"helloworld",10);
    return 0;
}
```

函数功能

函数原型

- int open(const char *pathname, int flags);
- int open(const char *pathname, int flags, mode_t mode);

头文件说明

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

参数说明

- pathname : 文件路径名

- flags : 打开标志
    - O_RDONLY: 只读方式打开文件
    - O_WRONLY: 可写方式打开文件
    - O_RDWR: 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，并用第三的参数为其设置权限
    - O_EXCL: 如果使用 O_CRATE 时文件存在, open() 报错
    - O_TRUNC: 如果文件已经存在，并且以读 / 写或只写成功打开, 并清零
    - O_APPEND: 以添加的方式打开文件，在打开文件的同时，文件指针指向文件末尾

- mode :
    - 指定创建新的文件的默认权限

返回值:

- 成功： 返回 文件描述符
- 失败 : 返回 –1, 并将错误编码保存到 errno
- 示例 1 通过只读的方式打开一个文件

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>

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

    int fd;


    if (argc != 2){
        fprintf(stderr,"Usage : < %s > < pathname >\n",argv[0]);
        return -1;
    }



    fd = open(argv[1],O_RDONLY);

    if (fd == -1){
        perror("Open(): ");
        return -1;
    }


    close(fd);

    return 0;
}
```

- 练习 : 以只写的方式打开文件, 如果不存在则创建, 如果文件存在则截短

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc,char *argv[])
{
    int fd;
    if(argc != 2)
    {
        printf("Usage : %s <pathname> .\n",argv[0]);
    }

    fd = open(argv[1],O_WRONLY|O_CREAT|O_TRUNC,0644);
    if (fd == -1)
    {
        perror("open()");
```

```
        exit(-1);
    }

    printf("fd = %d\n",fd);

    close(fd);

    return 0;
}
```

- errno 是 Linux 操作系统中用于存储错误编码的全局变量, 错误编码在 Linux 系统中的定义如下:

```
#define EPERM        1
#define ENOENT       2
#define ESRCH        3
#define EINTR        4
#define EIO          5
#define ENXIO        6
#define E2BIG        7
#define ENOEXEC      8
#define EBADF        9
#define ECHILD      10
#define EAGAIN      11
#define ENOMEM      12
#define EACCES      13
#define EFAULT      14
#define ENOTBLK     15
#define EBUSY       16
```

函数头文件

#include <stdio.h>

函数原型

void perror(const char *s)

函数参数

- s：自定义字符串参数
- 错误信息转换主要使用 strerror() 函数, 具体说明如下:

函数头文件

#include <string.h>

函数原型

char *strerror(int errnum)

函数功能

- 将错误编码转换成字符串信息，并返回该字符串的地址
  函数参数
- errnum：错误编码
  函数返回值
- 返回错误码转换之后的字符串 or “Unknown error nnn”

示例：使用 perror 函数打印 出错信息

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(int argc,char *argv[])
```

```
{
    int fd;

    fd = open(argv[1],O_RDONLY,0644);

    if (fd == -1)
    {
        perror("open(): ");
        return -1;
    }

    return 0;
}
```

示例：使用 strerror 函数转换 错误码

```
include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>

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

    fd = open(argv[1],O_RDONLY,0644);

    if (fd == -1)
    {
        fprintf(stderr,"open() : %s\n",strerror(errno));
        return -1;
    }

    return 0;
}
```

函数头文件

#include <unistd.h>

函数原型

int close(int fd);

函数功能

- close 函数用于关闭文件，在 io 操作结束后需要关闭文件，释放相关资源

函数参数

fd：文件描述符

函数返回值

成功：返回 0

失败：返回 –1

示例：将前面已经打开的文件使用 close 函数关闭