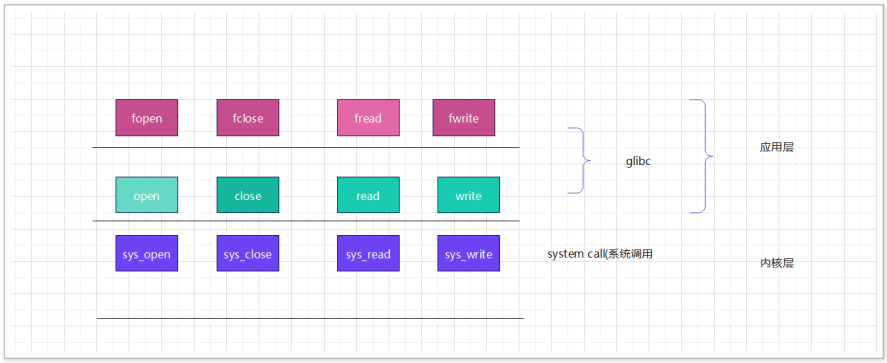
# 3.1 Linux 标准 io 简介_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.1 Linux 标准 io 简介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 标准 io 简介

- 标准 IO 是另外一套 IO 接口，具有如下特点:
  - 标准 I/O 是属于跨平台, 可以在 Linux、windows、mac os 上运行, 文件 IO 只能在 Linux 平台运行
  - 标准 I/O 自带缓冲区，有更高的 IO 效率
  - 标准 IO 提供丰富的操作文本信息接口
- 标准 IO 底层需要依赖于 文件 IO
- 在 Linux 系统下, 标准 I/O 是属于 glibc 库的一部分




- 流（stream）：
  - 流是一串连续不断的传输的数据的集合，就像水管一里的水流，在水管的一端一点一点地供水，而在水管的另一端看到的是一股连续不断的水流
- 一般流可以分为 文本流 与 二进制流
- 文本流:
  - 在流中处理的数据是以字符出现。在文本流中，’\n’被转换成回车符 CR 和换行符 LF 的 ASCII 码 0DH 和 0AH, 而当输出时，0DH 和 0AH 被转换成’\n’
  - 流中处理的是二进制序列。若流中有字符，则用一个字节的二进制 ASCII 码表示；若是数字，则用对应的二进制数表示
- 文件指针
  :
- FILE 指针：每个被使用的文件都在内存中开辟一个区域，用来存放文件的有关信息，这些信息是保存在一个结构体类型的变量中，该结构体类型是由系统定义的，取名为 FILE。
- FILE 结构体定义在 /usr/libio.h 中 struct _IO_FILE

```
struct _IO_FILE;
typedef struct _IO_FILE __FILE;
```

- 标准 I/O 库的所有操作都是围绕流 (stream) 来进行的，在标准 I/O 中，流用 FILE * 来描述
- 标准 I/O 库是由 Dennis Ritchie 在 1975 年左右编写的
- 文件指针关联到数据流的两端, 可以抽象成“水管”
- 标准 I/O 预定义 3 个流对象指针, 在应用程序运行自动被打开.
    - 标准输入：流对象操作的是标准输入设备, 流对象指针的名称为 stdin , 对应的文件描述符为 0
    - 标准输出：流对象操作的是标准输出设备, 流对象指针的名称为 stdout, 对应的文件描述符为 1
    - 标准错误输出: 流对象操作的是标准错误输出设备, 流对象指针的名称为 stderr, 对应的文件描述符为 2
- 对应的 printf , 函数操作的就是 stdout , 由于是默认操作, 一般无需指定具体的流对象参数
- 当在输出时需要指定流对象的类型时, 则需要使用 fprintf 函数

函数原型

int fprintf(FILE *stream, const char *format, ...);

函数功能

将格式化数据写入到指定文件中

函数参数

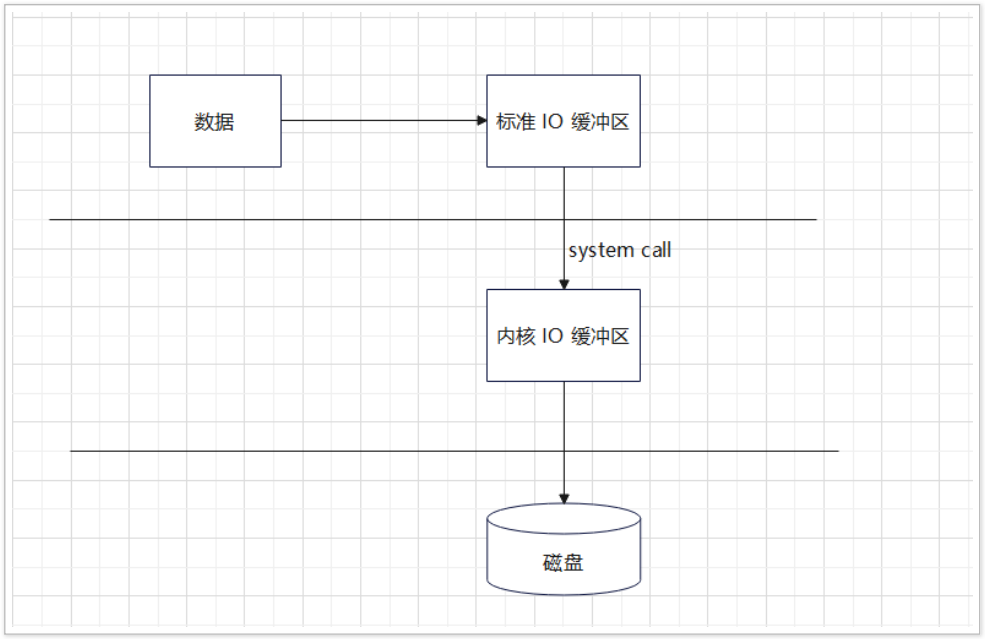
- stream：流对象指针
- format：格式字符串

示例：通过 stdout 与 stderr 进行输出

```
int main(void)
{
    fprintf(stdout,"Linux std io .\n");
    fprintf(stderr,"can't open file.\n");

    while(1){}
    return 0;
}
```

- 注意: 在上述程序中, 将‘\n’去掉之后, 在添加一个死循环后, 则程序运行的结果则不同, 这里是与标准 I/O 的缓冲区有关系.
- 缓冲文件系统
    - 尽量减少使用 read/write 的调用次数, 来提高效率, 每次进行系统调用都会涉及到从用户空间到内核空间的切换以及内核进行系统调用所产生的开销
- 系统自动的在内存中为每一个正在使用的文件开辟一个缓冲区，从内存向磁盘输出数据必须先送到内存缓冲区，装满缓冲区在一起送到磁盘中去.
- 从磁盘中读数据，则一次从磁盘文件将一批数据读入到内存缓冲区中，然后再从缓冲区逐个的将数据送到程序的数据区




- 标准 I/O 的缓存大小为 8192, 在系统中定义如下 (stdio.h):
- 一般标准 I/O 的分类为:
  - **全缓存**：当相应的缓冲区已经装满数据时, 才进行一次 I/O 操作
  - **行缓存**：当相应的缓冲区存储一行时，则进行一次 I/O 操作, stdout 就是行缓存
  - **不缓存**：直接进行 I/O 操作, 不进行缓存, stderr 就是不缓存
- 一般情况下, 程序在结束时会 **自动刷新缓冲区, 但是当程序还未结束时**, 刷新缓冲区则需要调用 fflush() 函数

函数原型

int fflush(FILE *stream);

函数功能

强制刷新缓冲区

函数参数

函数返回值

示例：使用 fflush 函数刷新缓冲区的数据

```
#include <stdio.h>


int main(void)
{

    printf("hello.");
    fflush(stdout);


    while(1){}

    return 0;
}
```

练习：使用 fprintf 函数 "Hello,Linux io" 到 标准输出，并使用 fflush 函数进行强制刷新

全文完