

嵌入式应用程序设计综合教程

教程架构

1、I/O编程

- ◆ 标准I/O编程
- ◆ 文件I/O编程

2、进程编程

- ◆ 进程线程的创建
- ◆ 进程间通信
- ◆ 进程间同步与互斥

3、网络编程

- ◆ 网络通信模型
- ◆ TCP/IP协议簇
- ◆ SOCKET编程

操作系统
提供的功
能

学习方法

基础教程

- ▶ 易于理解，难以使用。
- ▶ 使用起来千变万化，重在程序架构设计。
- ▶ 无需了解操作系统。

聪明+勤奋

综合教程

- ▶ 难以理解，易于使用。
- ▶ 使用有固定的模式、无需创新设计。
- ▶ 和操作系统贴合较近，使用它们需要了解系统的机制。

勤奋+勤奋

I/O编程学习概要

两个I/O操作框架：

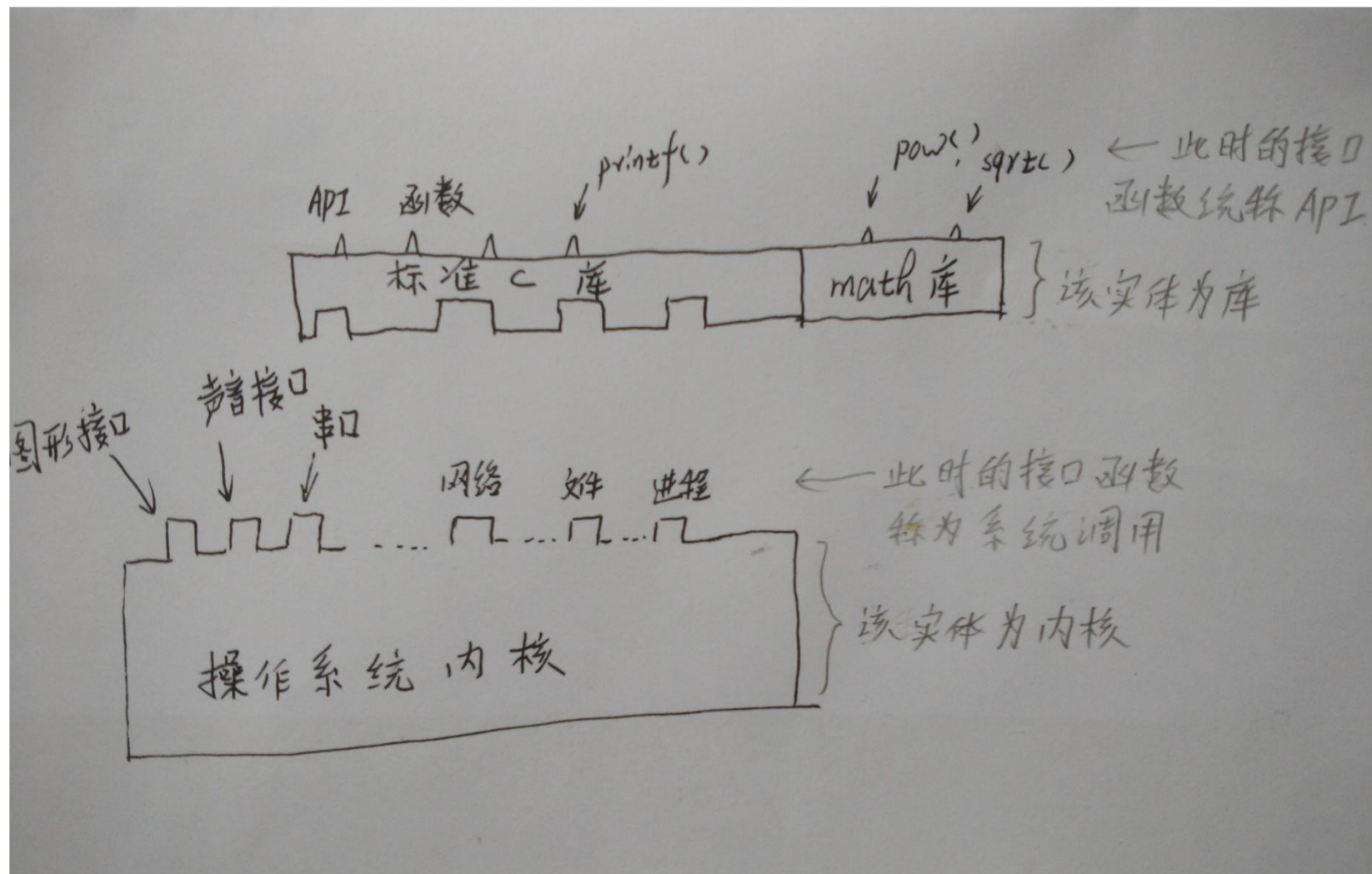
- ▶ **标准I/O** → C标准库提供的I/O框架。
- ▶ **文件I/O** → linux操作系统提供的文件I/O框架。

相关概念：

1、库、API、系统调用。

标准I/O	库提供的API
文件I/O	系统调用

库、函数和系统调用



IO常见操作

- ▶ 创建文件
 - ▶ 删除文件
 - ▶ 读文件
 - ▶ 写文件
 - ▶ 修改文件局部内容
 - ▶ 追加文件
 - ▶ 截短文件
 - ▶ 检查文件是否存在
 - ▶ 得到文件大小
 - ▶ ...
- ▶ 创建目录
 - ▶ 删除目录
 - ▶ 枚举目录项
 - ▶ 检查目录是否存在
 - ▶ 判断目录项是文件还是目录
 - ▶ 切换当前目录
 - ▶ 改变文件权限
 - ▶ **link**文件创建和删除
 - ▶ ...

I/O操作之

标准I/O库

课程目标

- ▶ 会使用标准I/O相关函数
- ▶ 编程实现创建文件
- ▶ 编程实现读写文件内容
- ▶ 编程实现复制文件
- ▶ 编程实现加解密文件

标准I/O相关函数

- ▶ 文件打开，关闭
- ▶ 文件读写
 - ▶ fopen, fclose
 - ▶ fread, fwrite
 - ▶ fgetc, fputc
 - ▶ fgets, fputs
- ▶ 文件定位，结束
- ▶ 文件缓存
- ▶ 文件格式化操作
 - ▶ fseek, ftell, feof
 - ▶ fflush
 - ▶ fprintf, fscanf

其他常用函数

- ▶ 出错处理
- ▶ 休眠
- ▶ 时间操作
- ▶ perror, strerror
- ▶ sleep, usleep
- ▶ time, ctime

fopen

▶ `FILE *fopen (const char *path, const char *mode);`

功能： 打开由`path`指定的一个文件。

`path` : 字符串常量，标识文件的路径（相对、绝对）

`mode` : 文件的打开方式、标识打开权限及特殊情况下的处理方式。

返回：

成功，返回操作文件的指针

失败，返回NULL



标准I/O - fopen() - mode参数

打开标准I/O流的mode参数:

r或rb	打开只读文件，该文件必须存在。
r+或r+b	打开可读写的文件，该文件必须存在。
w或wb	打开只写文件，若文件存在则文件长度清为0，即会擦些文件以前内容。若文件不存在则建立该文件。
w+或w+b或wb+	打开可读写文件，若文件存在则文件长度清为零，即会擦些文件以前内容。若文件不存在则建立该文件。
a或ab	以附加的方式打开只写文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。
a+或a+b或ab+	以附加方式打开可读写的文件。若文件不存在，则会建立该文件，如果文件存在，写入的数据会被加到文件尾后，即文件原先的内容会被保留。

* 当给定“b”参数时，表示以二进制方式打开文件。



标准I/O - fopen() - mode参数

打开一个标准I/O流的六种不同的方式

限 制	r	w	a	r+	w+	a+
文件必须已存在	•			•		
擦除文件以前的内容		•			•	
流可以读	•			•	•	•
流可以写		•	•	•	•	•
流只可在尾端处写			•			•

思考1: 多一个“+”代表可读写, 为什么不写成rw?

fclose

▶ `int fclose(FILE *fp);`

功能： 关闭文件指针，释放资源。

fp: fopen等返回的文件指针


返回：

成功，0

失败，-1（异常，崩溃）

标准输入输出

系统为每个应用程序默认打开了三个文件指针，方便开发者使用。



stderr	标准出错，perror打印
stdout	标准输出，printf打印
stdin	标准输入，scanf获取内容
shell重定向符号： 1> 例如：ls -lt 1> logfile	
shell重定向符号： 2> 例如：ls -lt noexistfile 2> logfile	

标准输入输出

- ▶ `int printf (const char * format, ...);`
- ▶ `int fprintf (FILE *fp, const char *format, ...);`
- ▶ `int printf (const char * format, ...)`等价于
- ▶ `int fprintf (stdout, const char *format, ...);`
- ▶ 前者把输出打印到标准输出设备上，后者可以将输出打印到任何可写的文件里面。

实验1

- ▶ 1、创建一个新文件。
- ▶ 2、打开一个已存在的文件，不改变内容。
- ▶ 3、打开一个已存在的文件，清空其内容。
- ▶ 4、用**scanf**获取标准输入，再将内容打印到标准输出和标准出错。

fgetc/fputc

◆ `int fgetc(FILE *fp);`

- ▶ 功能：从文件中读一个字符
- ▶ **fp**：文件指针（要有可读权限）
- ▶ 返回值
 - ▶ 成功：返回读出的字符
 - ▶ 失败：EOF

◆ `int fputc(int c, FILE *fp);`

- ▶ 功能：写一个字符到文件中
- ▶ **c**：要写入的字符
- ▶ **fp**：文件指针（要有可写权限）
- ▶ 返回值
 - ▶ 成功：字符**c**
 - ▶ 失败：EOF

思考2：fgetc成功的话返回值一个char类型字符，为什么返回值类型却设计成int类型？fgetc如何判断文件是否结尾？

feof

- ▶ `int feof(FILE *fp);`
- ▶ 功能：检查文件是否读取结束。
- ▶ `fp`：指向待检查文件的指针
- ▶ 返回值：
 - ▶ 文件未结束：0
 - ▶ 文件已结束：1

feof在读到文件的最后一个字符时，并不会认为文件结束，而是尝试读超过文件长度时才返回文件结束。即如果**feof**返回1，则**feof**上一次的读操作是失败的。

feof危险错误

▶ 错误:

```
while(!feof(fp))
{
    c = fgetc(fp);
    printf("%X\n", c);
}
```

▶ 正确:

```
int c;
c = fgetc(fp);
while(!feof(fp))
{
    printf("%X\n", c);
    c = fgetc(fp); // 最后一个c的值为-1
}
```

实验2

- ▶ 1、创建一个新文件case2.ppp，用fputc向文件中写入999个2，最后写入一个1。
- ▶ 2、用fgetc、fputc完成复制文件（有两种办法判断文件复制完成）。
- ▶ 3、打开case2.ppp，用fgetc把文件中内容读出，每5ms读一个，并打印。

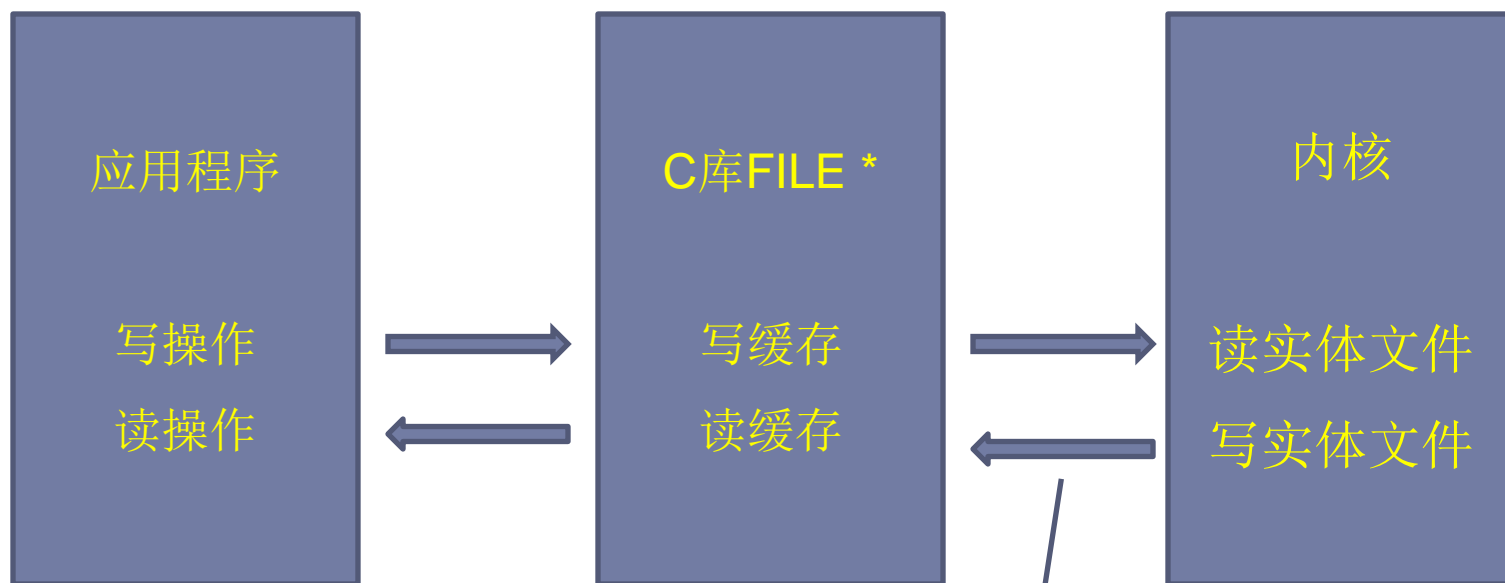
```
usleep(5000)  
printf("%c", c)
```

观察：打印的内容会一个一个的显示出来吗？

fflush

- ▶ `int fflush(FILE *fp);`
- ▶ 功能：可强制刷新一个流的缓存。此函数使该流所有未写的数据都被传递至内核。
- ▶ `fp`：要刷新的文件指针
如果为0，则刷新stdout（注：stderr不会缓存）
- ▶ 概念：
- ▶ 缓存、cache，缓冲区

缓存图示



不缓存：立即触发

行缓存：遇到换行符和缓存满触发

全缓存：缓存满触发

程序正常结束也会触发缓存刷新。

什么时机触发交互？

实验3

- ▶ 1、使用fflush完成实验2中的程序。
- ▶ 2、设计一个程序，确定系统默认的缓存大小。

标准I/O	库提供的API	有缓存
文件I/O	系统调用	无缓存

fgets/fputs

int fputs(const char *s, FILE * fp)

功能：将字符串**s**写入**fp**指向的文件中。

s: 要写入文件的缓冲区指针。

fp: 要写入的目标文件的流指针。

返回值:

成功: 写入的长度

失败: EOF

char *fgets(char *s, int size, FILE * fp)

功能：从**fp**指向的文件中读出一行（最多**size-1**个字符），写入**s**指向的缓冲区。

s: 读取文件内容的缓冲区指针。

size: 缓冲区的长度。

fp: 要读取的目标文件的流指针。

返回值:

成功: **s**

失败: **NULL**

fgets注意

`char *fgets(char *s, int size, FILE * fp)`

功能：从**fp**指向的文件中读出一行（最多**size-1**个字符），写入**s**指向的缓冲区。

如果**fp**文件中一行很长，超过**s**存储能力，那么只会读**size-1**个字符，**s**的最后一个字符必须为0。

如果读了一行，字符串倒数第二个字符一定是'\n'，否则就只读了一行的一部分或者到文件末尾。

Linux中换行符是'\n'，即0x0a。

fgets注意

函数原型 : `char * fgets(char *buf, int size, FILE *fp);`

功能 : 读取字符串, 优先读取一行信息

`buf` : 存放缓冲区, 返回的字符串格式为“读到的字符集合\0”

`size` : 要读取字符串的长度, 一般为`buf`的大小

读取过程中 遇到 '`\n`' 读取结束, 得到的字符串 "字符集合\0"

读取`size-1`个字符 没有 '`\n`' , 读取结束 得到的字符串 "字符集合\0"

`fp` : 流

返回值 : 成功: 字符串的首地址 `buf`

失败: `NULL`

fgets注意

- ▶ fgets遇到'\n'，停止读取数据。
- ▶ fgets可以读取可见字符，不可见字符，包括0。类似与

```
while(1){  
    c = fgetc(fp);  
    if(c == EOF || c == '\n' || i < sizeof(buf) - 1)break;  
    buf[i++] = c;  
}  
buf[i] = 0;
```

- ▶ fputs写入的长度是字符串的长度，如果字符串第一个字符为0，则不会写入。
- ▶ fgets、fputs一般用于读写字符文件，如果文件中包含不可打印的ASCII字符，请尽量避免使用。

实验4

- ▶ 1、创建一个名称为biaobai.ppp文件，**5**秒内持续fputs “I love you 你心梦中女（男）神的名字 \n”，然后用fgets统计下行数。
- ▶ 2、用fputs、 fgets完成复制实验1产生的文件。

fread/fwrite

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *fp);
```

```
size_t fwrite(void *ptr, size_t size, size_t nmemb, FILE *fp);
```

功能：读写文件

ptr: 要读入或写出的缓冲区指针，提前申请好的。

size: 要读出或写入的信息单元的大小

nmemb: 要读出或写入的信息单元的个数， $\text{size} * \text{nmemb}$ 不大于ptr大小

fp: 要读或写的文件指针，提前用fopen打开的

返回值:

成功：读取或写入的信息单元的个数

失败：EOF

fread/fwrite

`size_t fread(void *ptr, size_t size, size_t nmemb, FILE *fp);`

`size_t fwrite(void *ptr, size_t size, size_t nmemb, FILE *fp);`

- 一般情况下， **size** 设为1， **nmemb** 设为想要读取的字节数或者 **ptr** 指向 **buf** 的大小，返回值为实际读写的字节数。

```
while(!feof(fp))
{
    ret = fread(buf, 1, sizeof(buf), fpr);
    if(ret > 0){
        fwrite(buf, 1, ret, fpw);
    }
}
```

fread/fwrite易犯错误

- ▶ **fread**返回读出的长度即是**buf**中的有效数据长度，**buf**中下一个字符是否为0和**fread**没有任何关系。
(之前学的**fgets**才会将最后一个字符置0)

实验5

▶ 1、用fread/fwrite复制文件。

▶ 2、将诗

我住长江头，君住长江尾。

日日思君不见君，共饮长江水。

此水几时休？此恨何时已？

只愿君心似我心，定不负相思意。

写入文件。

3、逐字节异或0x33加(解)密文件，并将加(解)密内容写道一个新的文件中。

fseek

```
int fseek(FILE *fp, long offset, int whence);
```

fp: 文件指针，提前用fopen打开的。

offset: 移动的指针的相对偏移，

offset > 0, 向后移动读写文件的指针

offset < 0, 向前移动读写文件的指针，

whence: offset相对偏移的基准点

SEEK_SET: 文件开头

SEEK_CUR: 文件当前位置

SEEK_END: 文件末尾

返回值: 成功 0

失败 -1

ftell/rewind

ftell()/rewind()函数原型:

```
#include <stdio.h>
```

```
long ftell(FILE *fp);
```

```
void rewind(FILE *fp);
```

ftell()用于取得当前的文件位置，调用成功则为当前文件位置指示，若出错则为-1L

rewind()用于设定流的文件位置指示为文件开始，该函数调用成功无返回值。rewind()等价于：

fseek(stream, 0L, SEEK_SET)

实验6

- ▶ 1、得到文件的大小。
- ▶ 2、16进制打印可执行文件本身的第1、2、3，100和最后一个字符(开头的为第0个字符)。
- ▶ 3、倒叙打印一个文件。
- ▶ 4、将实验中5-3的输出结果写到源文件中，（加密后的数据写到原来的明文数据位置）。

格式化输出

- ▶ `int printf (const char * format, ...);`
- ▶ `int sprintf(char * buf, const char *format, ...);`
- ▶ `int fprintf (FILE *fp, const char *format, ...);`
- ▶ format格式（包含 %d %s %c....）
- ▶ **printf**格式化输出到**stdout**
- ▶ **sprintf**格式化得到的字符串保存到**buf**里面
- ▶ **fprintf**格式化得到的字符串写到**fp**指向的文件里面

格式化输出例子

```
int i = 9;  
char buf[1024];  
FILE * fp = fopen("file1", "w");  
  
printf("i = %d\n", i);  
sprintf(buf, "i = %d\n", i);  
fprintf(fp, "i = %d\n", i);
```

格式化输入

- ▶ `int scanf (const char * format, ...);`
 - ▶ `int sscanf (char * s, const char * format, ...);`
 - ▶ `int fscanf (FILE * fp, const char * format, ...);`
-
- ▶ `scanf` 从`stdin`中格式化输入
 - ▶ `sscanf`从字符串`s`中格式化输入
 - ▶ `fscanf`从`fp`中格式化输入

格式化输入注意

- ▶ **scanf**匹配一个单元的时候，遇到空格和换行，匹配停止。
- ▶ 在空格前停止匹配，匹配内容会忽略空格，匹配格式不会。
- ▶ **fscanf**返回值为匹配成功的个数，如果一个也没有匹配成功，**fp**不会前进！

格式化输入

- ▶ **fprintf**, 从 **fp** 文件流中取出一行进行匹配，以给定的格式取出相应位置的数据，存放在自定义的变量中。
- ▶ 格式化中以空格和换行结束。
- ▶ 如果是循环的话，要保证每一行都被匹配完。

实验7

- ▶ 1、输入一个二元四则算术题，输出结果。
- ▶ 2、将作业中的二元四则算术题做一遍。

总结&要求

- ▶ 1、熟练掌握fopen打开文件的方式
- ▶ 2、熟练运用fread、fwrite、fgets、fputs, fseek、fprintf等函数，可得到文件大小，随机读写，格式化输出文件。
- ▶ 3、能将文件读写和文件处理结合起来，能独立设计程序。
- ▶ 4、了解文件概念、对linux系统的常接触的特殊文件慢慢开始积累。
- ▶ 5、了解所学的知识在整个体系中的位置，清楚自己懂什么，不懂什么。

扩展自学

- ▶ tmpfile
- ▶ strerror
- ▶ getc/getchar
- ▶ putc/putchar
- ▶ gets
- ▶ puts
- ▶ access
- ▶ truncate

编程习惯

- ▶ 嵌入式应用程序**设计**。
 - ▶ 理解API的功能，清楚什么时候该用它。
 - ▶ 明白API调用一般流程，先后顺序，这个不能靠记忆、靠教训。
 - ▶ 对函数的输入输出有一定的记忆，并能通过man快速恢复记忆。

编程工具

- ▶ 1、VI、VIM属于石器时代。
- ▶ 2、notepad++为封建社会。
- ▶ 3、建议sublime，尽量在linux主机里面完成所有的编程任务。



课后

- ▶ 1、安装**sublime**，使用其完成代码编写、编译的工作(下载、安装、注册、配置**gcc**、配置输入中文)。