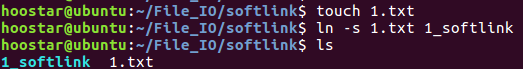
硬链接和软链接：

1. 软链接：

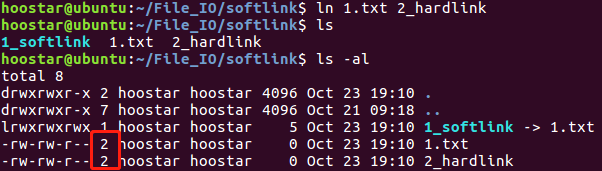
命令：ln -s 1.txt 1\_softlink



1. 硬链接：

命令：ln 1.txt 2\_hardlink

会导致硬链接数+1



1. 区别

在创建好软链接和硬链接后都可以通过这两个链接去打开文件

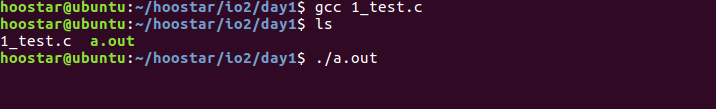
如果此时将原文件1.txt删除，那么将导致无法通过软链接打开原文件了，但是可以通过硬链接打开原文件。



1. 课程内容
2. 标准IO，带缓存
3. 文件IO，不带缓存
4. 时间的编程
5. 文件的目录属性
6. 库的制作
7. 文件io
8. 文件：一组相关数据的有序集合
9. IO：input/output。可以使用（键盘、鼠标、文件、内存）进行输入，可以输入 到屏幕、文件、内存
10. 目的：对文件进行操作
11. 缓存区

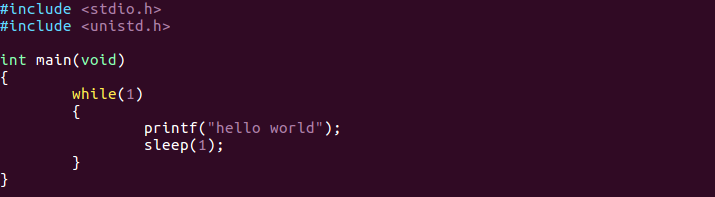
1）

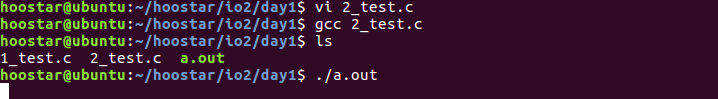




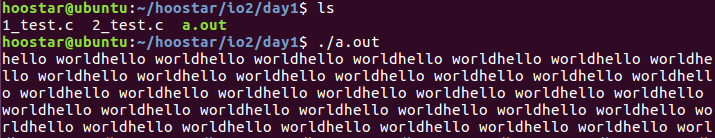
程序执行未打印

2）





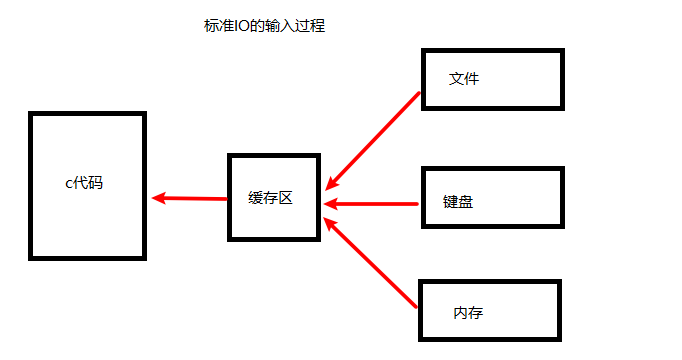
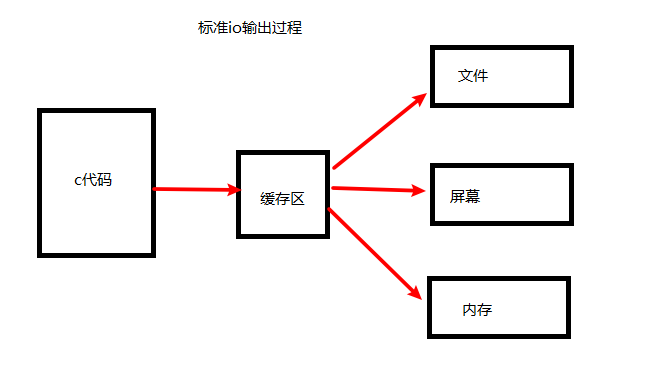
程序执行后，并未直接打印结果



等待缓存区满后，打印出结果

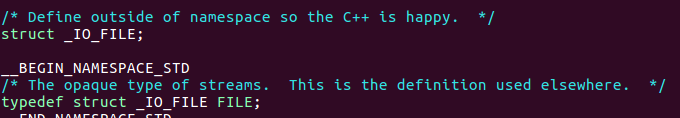
1. 对缓存区的理解

缓存区的本质是内存。由c库实现，不需要自己实现



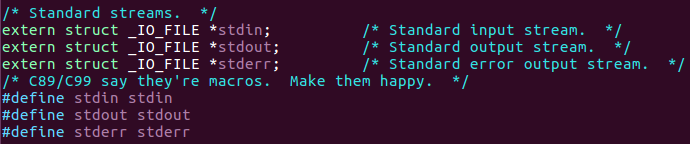
1. 缓存区的分类
   1. 全缓存，必须等到缓存满才会输出/输入；fflush强制进行刷新；程序正常 结束
   2. 行缓存，必须等到回车符才会输出/输入；fflush强制进行刷新；程序正常 结束
   3. 无缓存，直接进行输出/输入；fflush强制进行刷新；程序正常结束
2. 缓存区的意义
   1. 提高效率，cpu 和 硬盘读写速度差距过大
3. 流
4. 流的定义：所有的IO操作仅是数据简单的从程序中移进移出，这种就叫做字节流
5. 流的类型：使用FILE 结构体指针对流进行定义

FIEL结构体定义在stdio.h中



1. find查找命令的格式： find [路径] -name [文件名]
2. 常见的流的分类：
   * 1. 标准输入流,stdin
     2. 标准输出流,stdout
     3. 标准错误流,stderr 以上三种程序开始就存在
     4. 自己定义的流
3. 流和缓存

1）FILE \* fp = fopen();此时的fp 就是流，FILE的类型struct \_IO\_FILE



struct \_IO\_FILE \* stdin = fopen();

FILE \*stdin =f open(); stdin是结构体指针类型，指针类型存放的是地址，对应的内存就 是缓存

FILE \* fp = fopen();

1. 流和缓存的关系
   1. 标准输入输出流 <==> 行缓存
   2. 标准错误流 <==> 无缓存
   3. 自己定义流 <==> 全缓存
2. 文件的打开操作
   1. 要对文件操作就必须先打开对应的文件
   2. FILE \*fopen(const char \*path, const char \*mode);

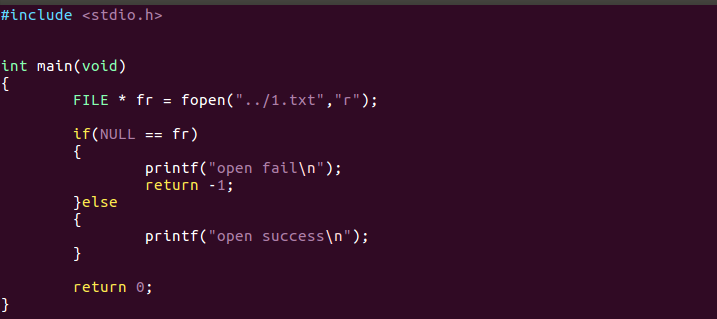
函数名称：fopen

函数功能：打开文件

参数： path：对应的路径； mode : 打开的模式

返回值：成功返回的是一个流，失败返回NULL并且设置错误号

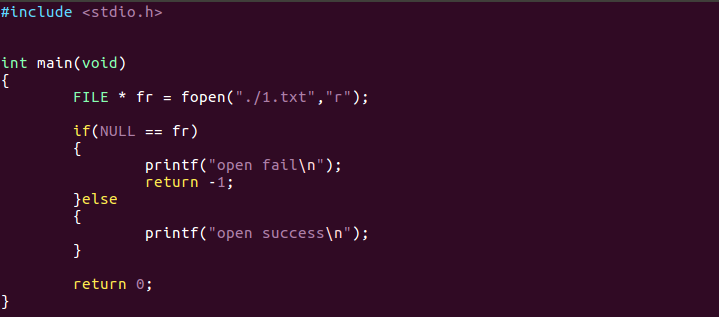
Path:要打开文件的路径

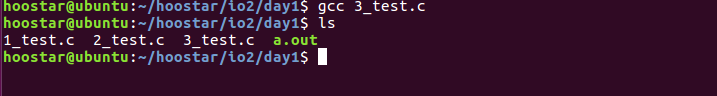




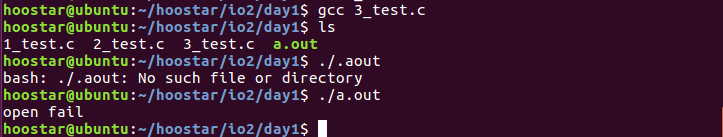
**Mode：打开文件的方式**

**r: 以只读方式打开文件，文件不存在报错，不会创建新文件**

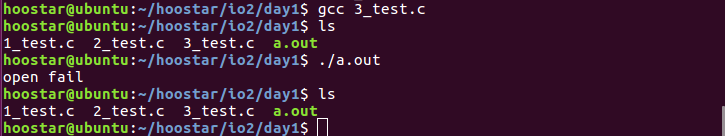
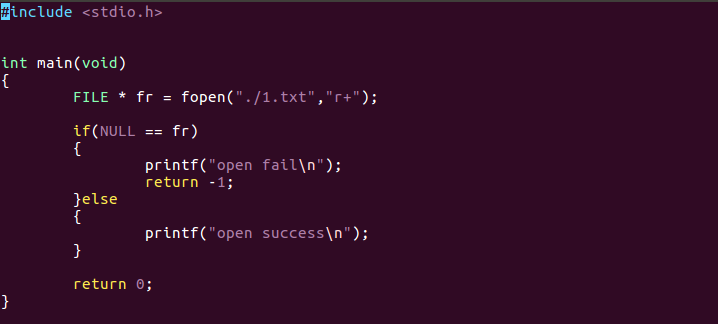




目录中未存在1.txt，open fail



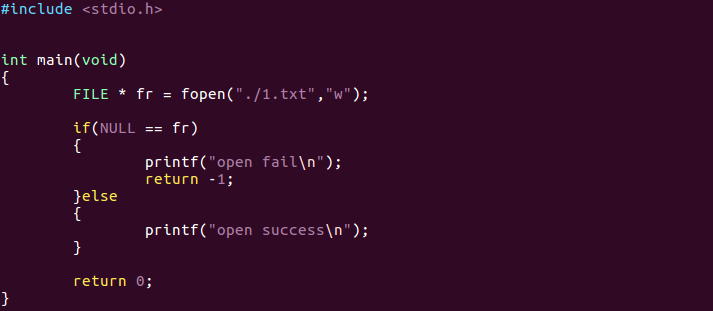
**r+: 以读写方式打开文件，文件不存在报错，不会创建新文件**

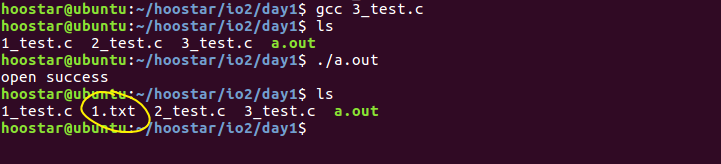


**W:以只写的方式打开**

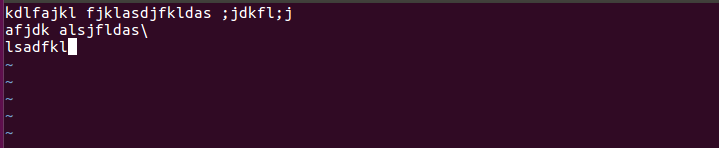
**文件不存在，创建文件**

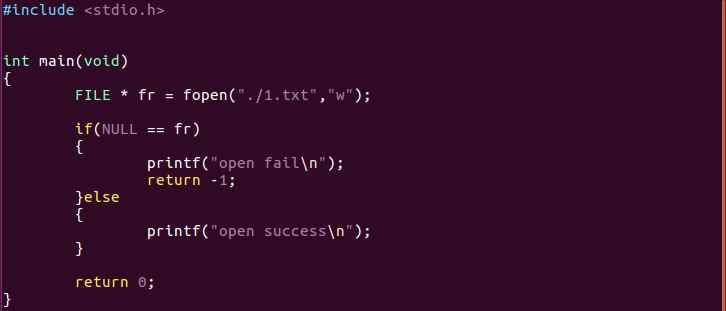


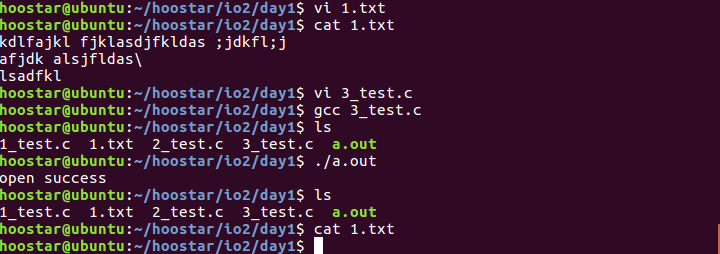




文件存在，清空文件再写







**W+：以读写的方式文件打开文件**

**如果文件存在，则清空再写，如果文件不存在，创建文件**（自己验证）

**a：以只写的方式**

**如果文件存在，则在文件末尾追加**

**如果文件不存在，创建文件**

**a+：以读写的方式**

**如果文件存在，则在文件末尾追加**

**如果文件不存在，创建文件**

1. 直接读写

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

函数名：fread()

功能：从stream流读取 nmemb \*size 个字节大小到ptr中

参数：ptr:缓存；size：每一块的大小；nmemb：需要读的块数；stream：需要读的流

返回值：如果成功，fread()和fwrite()返回已读或已写的nmemb。如果发生错误，或者到 达文件的末尾，返回值是<=0。

size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

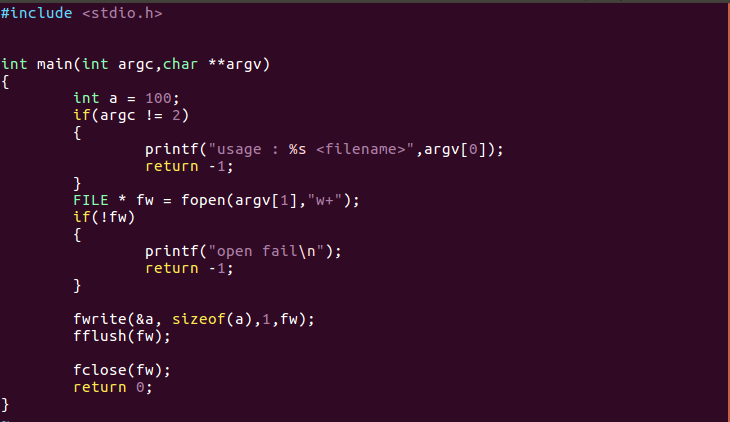
函数名：fread()

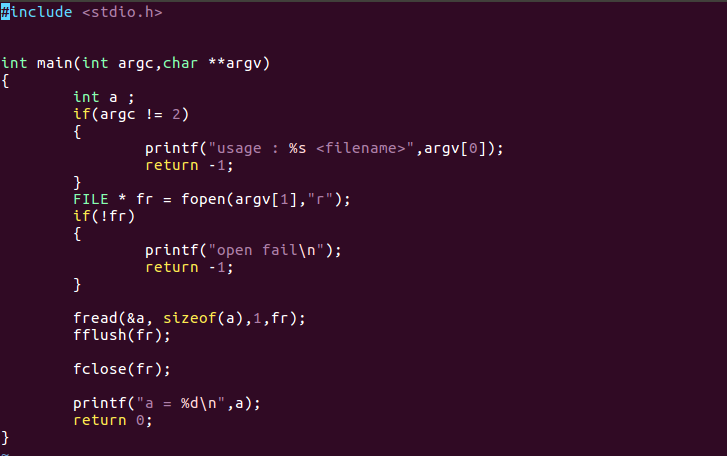
功能：从stream流读取 nmemb \*size 个字节大小到ptr中

参数：ptr:缓存；size：每一块的大小；nmemb：需要读的块数；stream：需要读的流

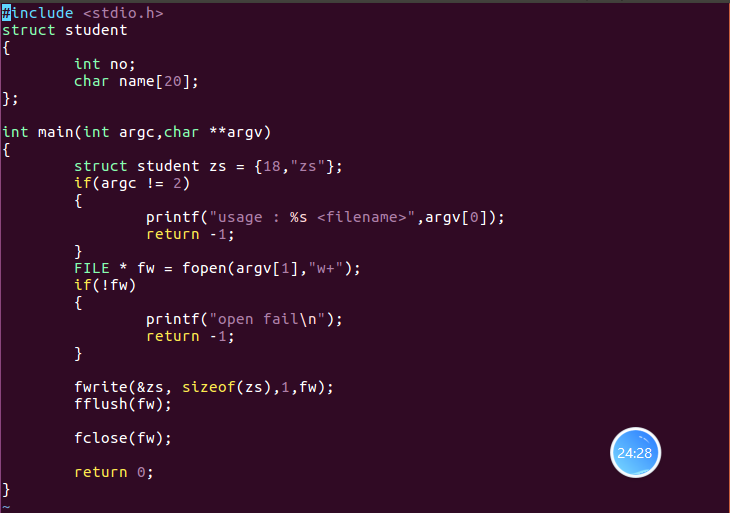
返回值：如果成功，fread()和fwrite()返回已读或已写的nmemb。如果发生错误，或者 到达文件的末尾，返回值是<=0。

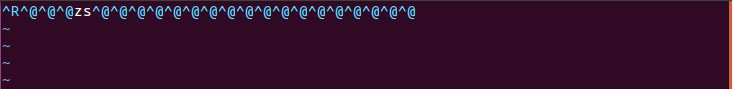
对数字的读写



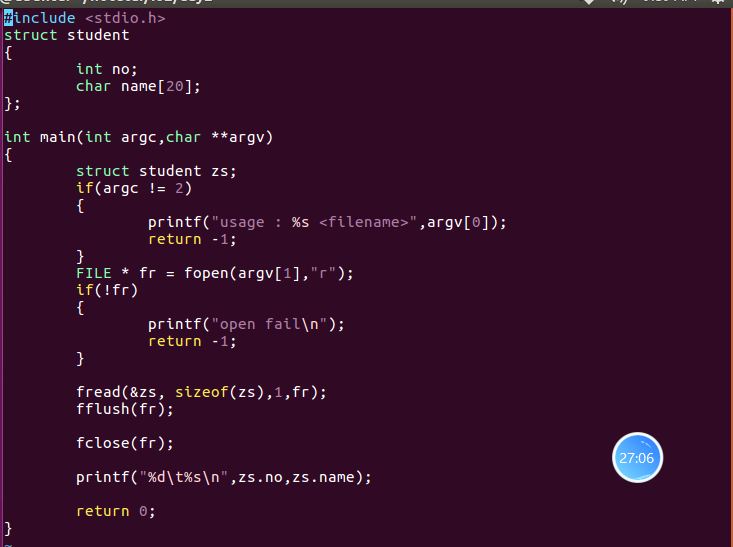


对结构体进行写

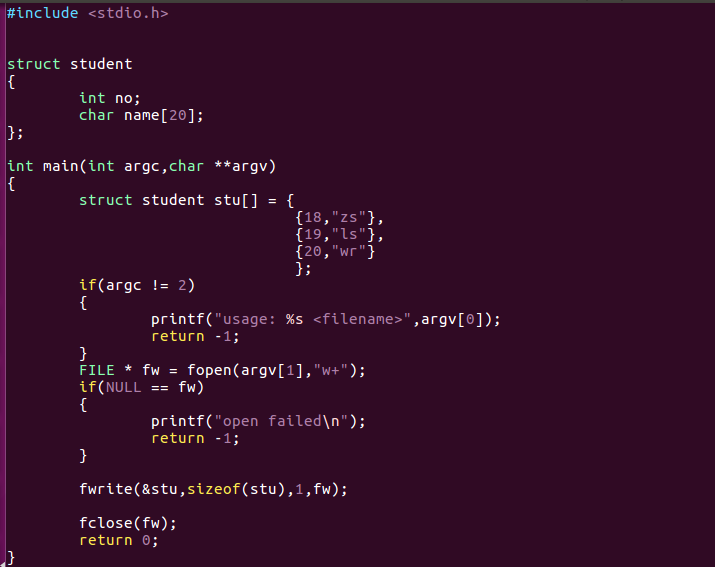




对结构体进行读



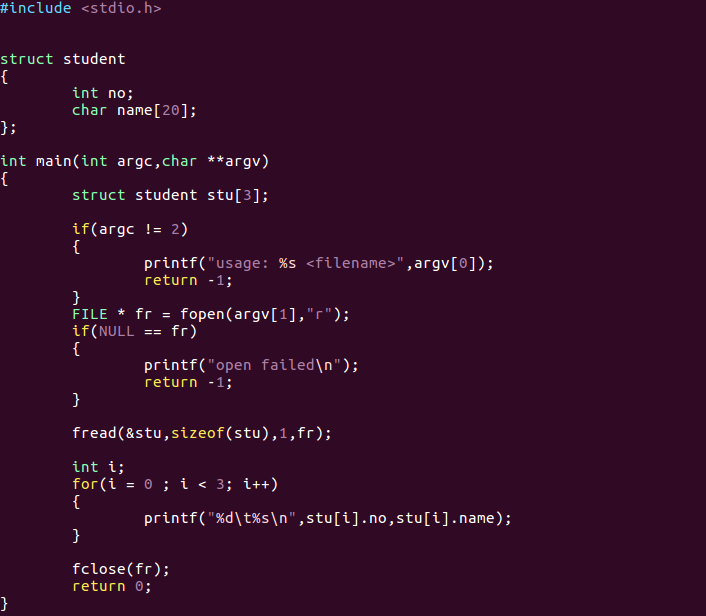
对结构体数组的写

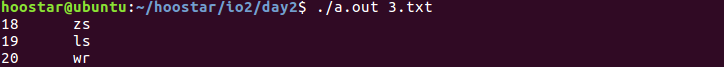




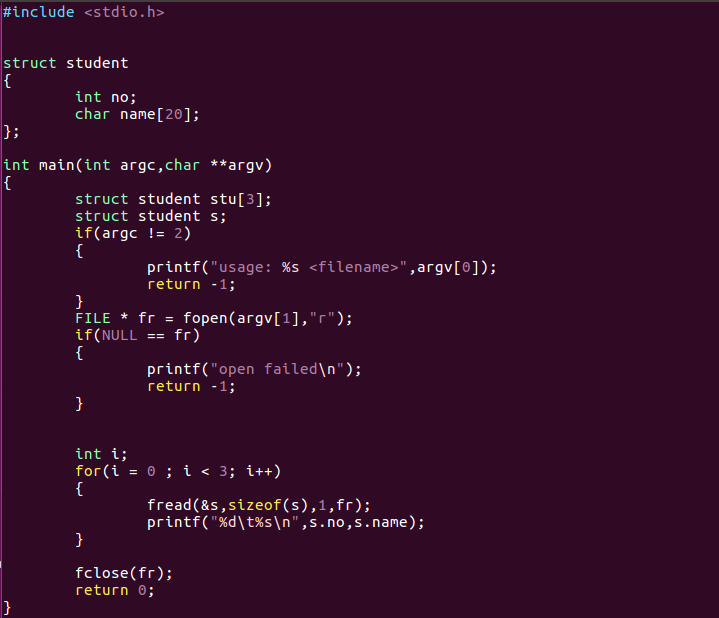


对结构体的读



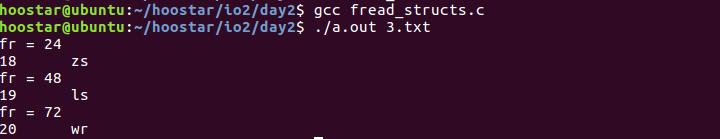


通过单个结构体读写结构体数组

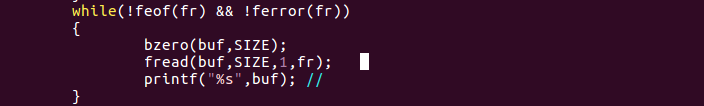


原理是通过文件指针的偏移获取下一个结构体的内容

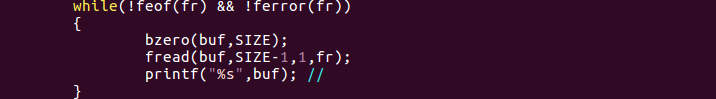




SIZE的影响

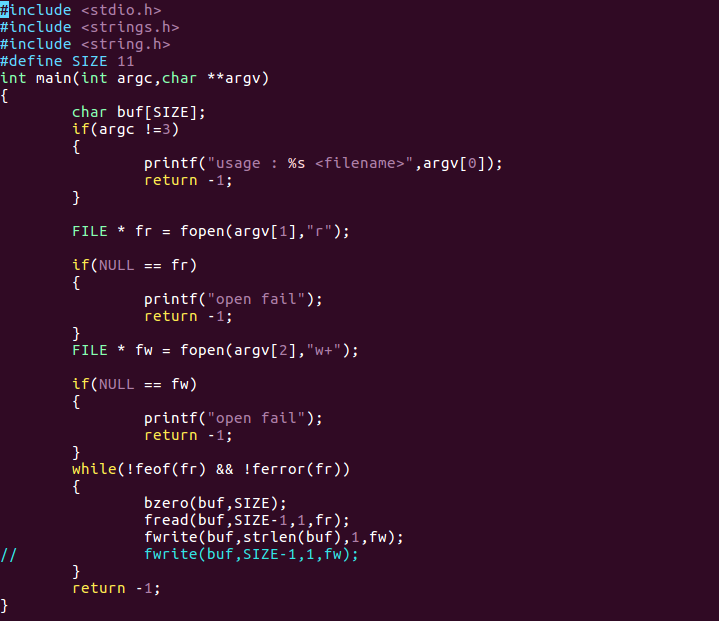


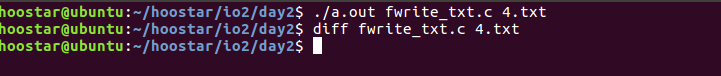
每次都清空为0，但是每次都填满buf，找不到’\0’的位置，printf就有可能打印出其他 内容



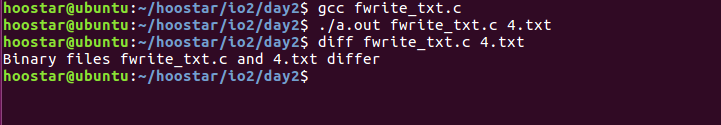
建议使用SIZE-1;

使用fwrite写文本文件时，采用strlen(buf);原因是当最后一次写时，实际内容比SIZE-1还小时，仍然使用SIZE-1,会造将0写进文件的情况，当使用strlen可以避免该情况。









1. 格式化标准输出

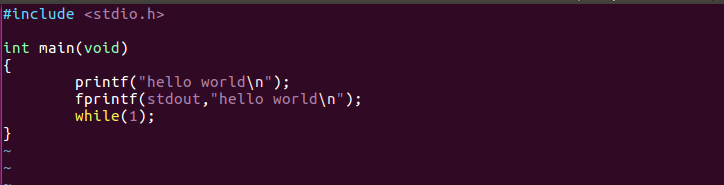


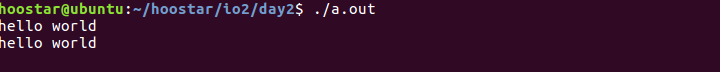
函数：fprintf

功能：将格式化的format输出到stream中

参数：stram: 流；format:格式化的数据

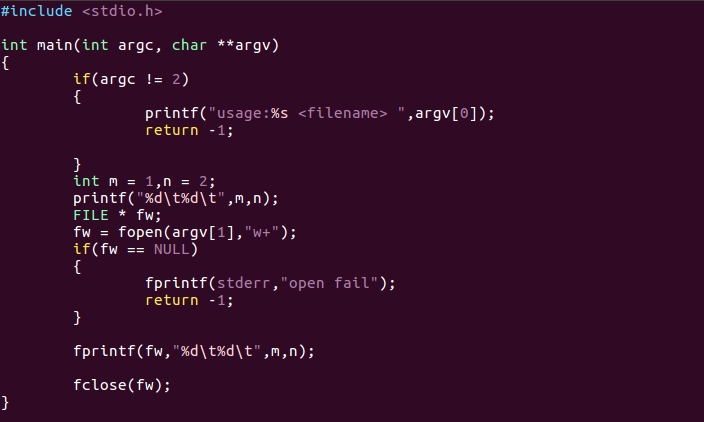
fprintf(stdout,format) == printf(format);





不加\n无法输出内容。

将变量按照指定格式打印到文件流中





函数：sprintf

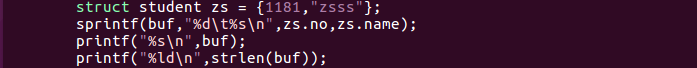
函数功能：将数据按照一定的格式输入str中

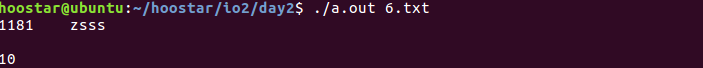
参数：str:缓存；format:格式化数据

将结构体转为字符串存储到文件中



Sprintf转换后的长度：将所有数据转换成字符，数字符个数



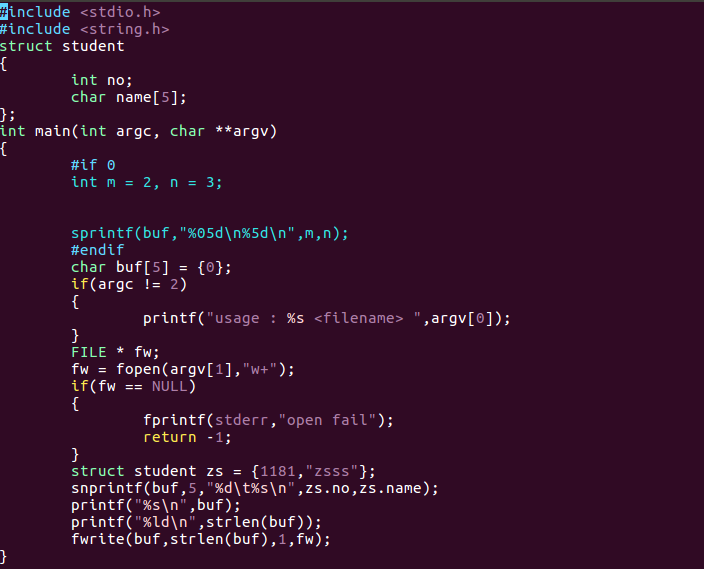




函数：snprintf

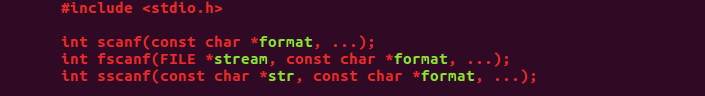
功能：将SIZE-1大小的数据写入str中

参数：str：缓存； size：要写的字节大小；format：格式化后的数据





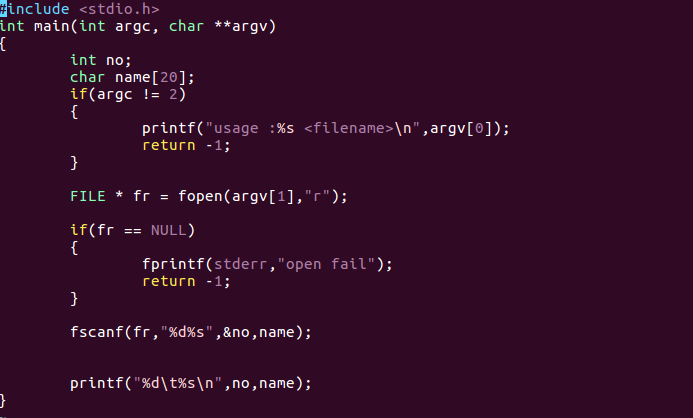
1. 格式化标准输入



函数名：fscanf

功能：从文件流输入数据

参数：stream:文件流；format：数据输入到变量中



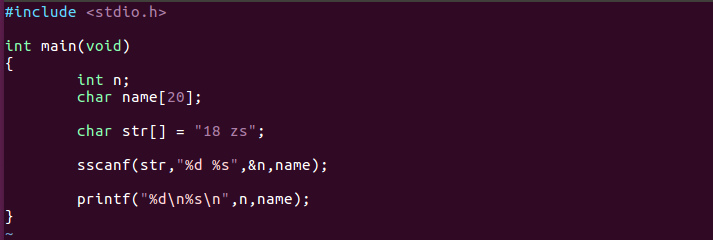


函数名：sscanf

功能：从str中输入数据，数据输入到变量中（字符串转成其他数据格式）

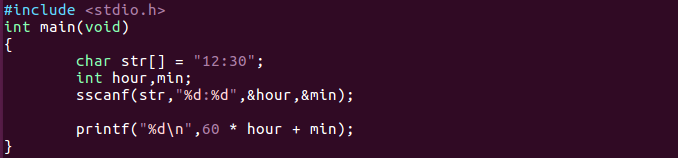
参数：stream:文件流；format：数据输入到变量中







Char str[] = “ 12:30” 时间转换成多少分钟？



注意：格式转化符必须和字符串中的格式一模一样。

1. ftell、fseek

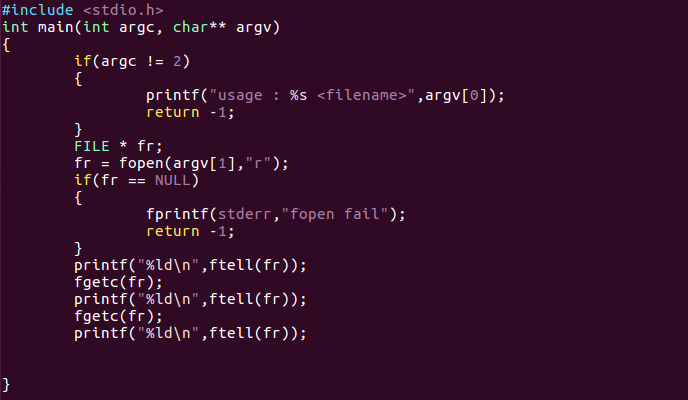


函数名称：ftell

函数功能：获取文件位置指示符的当前值。

参数：stream:流

返回值：文件流所指向的 位置





函数名字：fseek

函数功能：为文件流设置指向的地方

函数参数：stream:流； offset:偏移量；whence：文件指针的位置

函数返回值：成功返回0，失败返回-1并且设置错误号

SEEK\_SET:文件起始位置；

SEEK\_CUR:文件指针当前位置

SEEK\_END:文件末尾

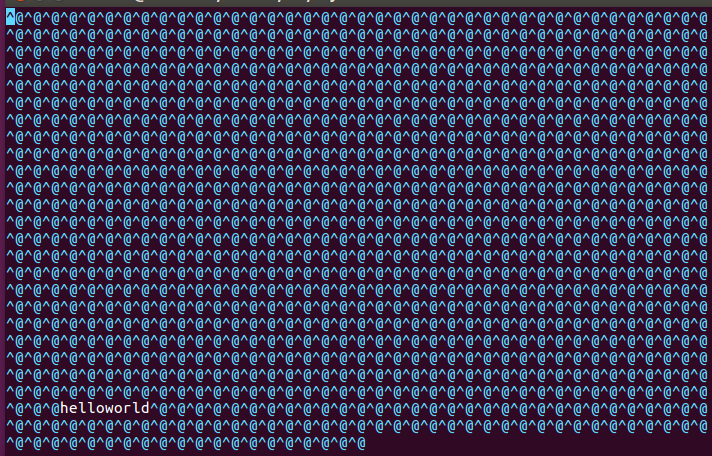
从距离文件头100的位置开始写文件





从末尾向前偏移100个字节进行写



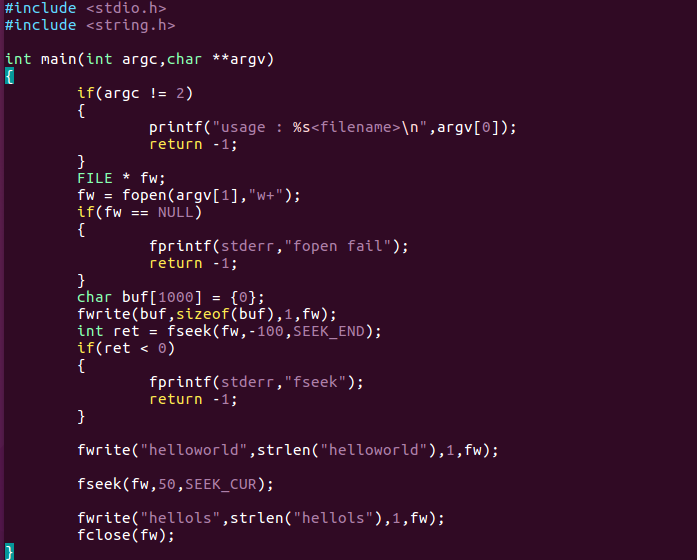


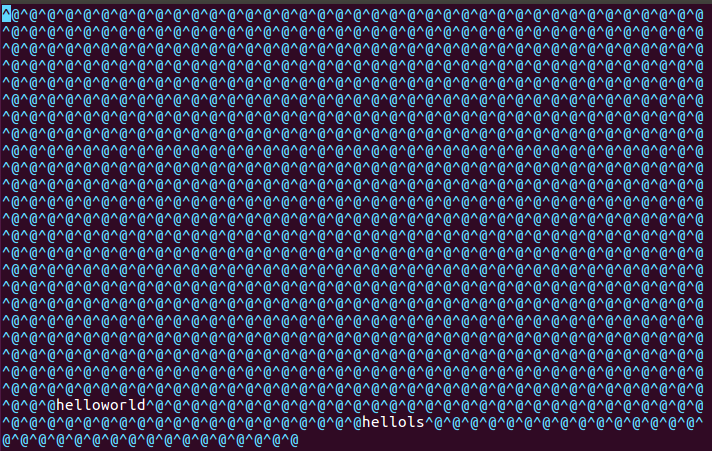
从末尾向后偏移字节





从当前位置开始偏移





从当前位置向前偏移







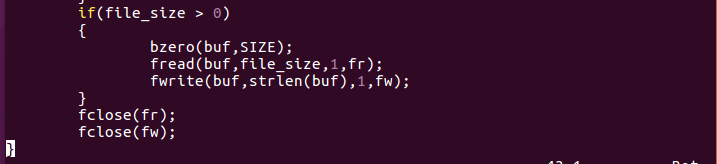
函数：rewind

函数功能：将文件指针指向头部

参数：流

Fseek和ftell完成fread和fwrite的复制





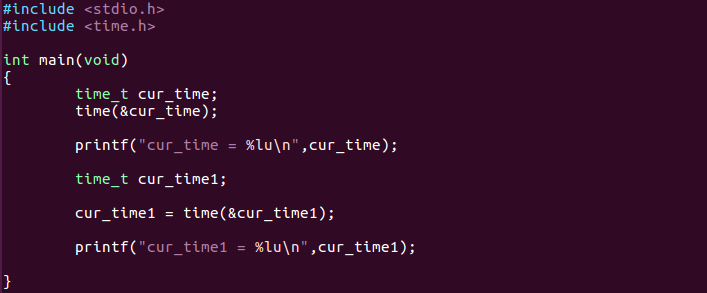
13、时间编程



函数名称：time

函数功能：获取从1970-01-01 00：00:00 到目前为止的秒数

参数：tloc用于获取时间



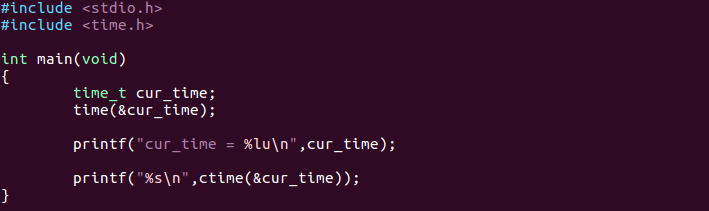




函数ctime

参数：秒数

功能：将秒数转换成字符串







函数：localtime;

函数功能：

函数参数：秒数

函数返回值： 结构体指针类型： struct tm \*

通过find 命令查找到time.h，tm结构体在time.h中定义

