

C程序设计语言

第7章 I/O

孙志岗
sun@hit.edu.cn
<http://sunner.cn>

I/O 设备

- 输入设备
 - 键盘、鼠标
 - 软盘、硬盘、光驱（以文件的形式）
 - 串行口、并行口、USB接口、IEEE1394口、网络端口
 - 扫描仪、视频采集卡、电视卡、游戏杆、话筒
 -
- 输出设备
 - 显示器、打印机
 - 软盘、硬盘、CD-RW/DVD-RW（以文件的形式）
 - 串行口、并行口、USB接口、IEEE1394口、网络端口
 - 音箱
 -
- 单纯的输入设备或者单纯的输出设备越来越少

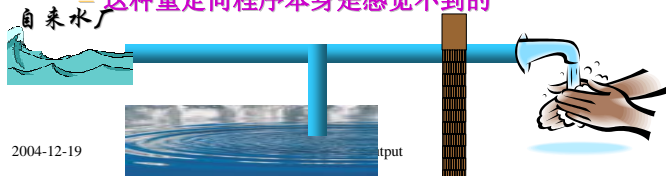
2004-12-19

Input and Output

2

标准输入输出

- 字符界面的操作系统一般都提供标准输入与输出设备
 - DOS、Linux、Unix.....
- 一般情况，标准输入就是键盘，标准输出就是终端显示器
 - 操作系统有能力重定向标准输入与输出，比如让文件作为标准输入，打印机作为标准输出
 - 这种重定向程序本身是感觉不到的



2004-12-19

3

DOS 下的标准输入输出重定向

- 程序prog.c如下

```
main()
{
    char c;
    while ((c=getchar()) != '\n')
        putchar(++c);
}
```
- 输入重定向
 - prog.exe < infile
- 输出重定向
 - prog.exe > outfile

2004-12-19

Input and Output

4

格式化输出——printf

- `int printf(const char *format, 参数1, 参数2, ...);`
- 参数 `format` 是用来控制格式的字符串
 - 具体格式查阅参考书
 - 常用转换字符: `%d`、`%c`、`%s`、`%.2f`、`%u`、`%ld`
- 返回值是最后输出的字符串长度, 出错返回 `EOF`
- `printf` 并不对参数的类型及个数进行检查, 所以一定要确切地把参数和前面的转换字符匹配好
- 这种参数形式不仅在C语言里应用广泛, 很多其它语言、类库也对此进行了模仿、扩充
- 相似的库函数
 - `sprintf`、`fprintf`

2004-12-19

Input and Output

5

格式化输入——scanf

- `int scanf(const char *format, 参数1, 参数2, ...);`
- 参数 `format` 是用来控制格式的字符串
 - 具体格式查阅参考书
 - 常用转换字符: `%d`、`%c`、`%s`、`%f`、`%u`、`%ld`
- 返回值是成功匹配的输入项的个数, 遇到结尾返回 `EOF`
- 所有参数必须是指针, 且类型要与前面的转换字符匹配好
- 相似的库函数
 - `sscanf`、`fscanf`
- 使用不当, 很容易造成恶性后果

2004-12-19

Input and Output

6

外存

- 磁盘 (Magnetic disks)
- 光盘 (CD、DVD)
- U盘 (Flash Memory)
-
- 数据断电后不丢失



2004-12-19

Input and Output

7

外存原理

- 磁盘表面涂有磁性物质
- 磁性单元的N-S极的两种指向表示0-1



2004-12-19

Input and Output

8

外存原理

- 光盘表面有一层特殊介质
- 介质的高低不平的交替表示0-1



光盘表面



读盘原理(CD机)

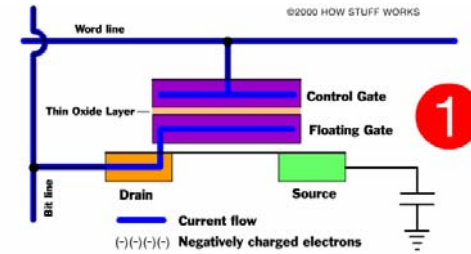
2004-12-19

Input and Output

9

外存原理

- Flash Memory是一种电化学存储介质
- 电流的通断表示0-1



2004-12-19

Input and Output

10

文件 (File)

- 为什么需要文件?
 - 我们需要数据在断电后依然存在, 需要外存
 - 外存原理迥异, 如果必须按其特性使用, 悲惨世界
- 文件哪里来?
 - 操作系统有一个模块, 叫“文件系统”
 - 文件系统封装了外存的特性, 让我们可以用同一个接口使用不同外存
 - 这个接口就是“文件”
- 原理
 - 操作系统决定文件在介质上的保存位置
- 使用文件需要知道
 - 目录结构, 文件结构, 命令接口, 开发接口

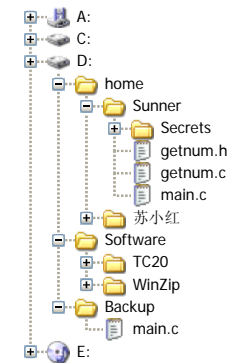
2004-12-19

Input and Output

11

目录结构 (Directory Tree)

- 树形的目录结构是文件系统的事实标准
- 每个文件都可以被唯一的“绝对路径 (Absolute Path)”表示
- 在DOS/Windows下:
 - D:\home\Sunner\main.c
- 在Unix/Linux下
 - /D/home/Sunner/main.c
- 相对路径 (Relative Path):
 - main.c
 - 在不同的当前路径指代不同的文件



2004-12-19

Input and Output

12

文件结构 (File Structure)

- 文件在外存存储就和数据在内存保存一样
 - 都是数据二进制形式的保存
 - 介质不关心数据类型
 - 你把数据当作是什么类型, 它就是什么类型
 - 如果以类型A的二进制形式存入, 却当作类型B读出, 就乱套了
- 所以, 读文件前, 必须确切知道文件每一个字节的确切类型和含义
 - 所以才有标准的mp3、bmp、jpg等文件
- 两种文件读写方式
 - 二进制方式和文本方式

2004-12-19

Input and Output

13

文本文件与二进制文件

- 文件都是二进制的
- 为方便使用, 硬性定义一种“文本文件 (Text File)”
 - 所有内容都是可打印字符的文件
 - .txt、.c、.h、.htm.....
 - 与之相对的就是所谓“二进制文件 (Binary File)”
 - .exe、.bmp、.mp3、.jpg、.doc、.swf.....
- 按二进制方式读写文件, 保持数据原状
 - 平实地反映数据和文件的真面貌, 不做任何处理

2004-12-19

Input and Output

14

按文本方式读写文件

- | | |
|----------------------------|--------------------------|
| DOS/Windows下: | Unix/Linux下: |
| ■ 用"\r\n"表示换行 | ■ 用"\n"表示换行 |
| ■ 按文本文件写入"\n", 会自动写入"\r\n" | ■ 文本方式和二进制方式没有区别 |
| ■ 假设文件中有"\r\n"这样的连续字符 | |
| ■ 按文本文件读, 只能得到"\n" | Mac下: |
| | ■ 用"\r"表示换行 |
| | ■ 写入"\n", 自动改为"\r" (未验证) |
| | ■ 读入"\r", 自动转为"\n" (未验证) |

2004-12-19

Input and Output

15

命令接口

- 操作系统界面 (Shell) 提供的文件操作命令
- DOS下:
 - copy、del、move、type.....
- Windows下:
 - 鼠标拖拽、ctrl+c,v、ctrl+x,v.....
- Unix/Linux下:
 - cp、rm、mv、cat.....
- 命令接口都是一些用开发接口编写的程序

2004-12-19

Input and Output

16

开发接口

- 给程序员在程序中使用的接口
- DOS下:
 - INT 21H.....
- Windows下:
 - CreateFile()、ReadFile()、WriteFile()、CloseHandle().....
- Unix/Linux下:
 - open、read、write、close.....
- 很多语言提供了自己的访问文件接口，它们都是基于操作系统提供的接口实现的

2004-12-19

Input and Output

17

文件访问的基本模式

- **open**: 打开文件，获得对此文件的指针、引用和句柄等，以后通过它们使用此文件
- **read**: 读文件。参数一般指明要读多少字节，读到哪块内存
- **write**: 写文件。参数一般指明把哪块内存的内容写入文件，要写多少字节
- **close**: 关闭文件，表明操作结束，本程序不再使用此文件。open后要及时关闭。
- **file position**:
 - 一个变量，其值表示距离文件头部的偏移量
 - 每次read或write都从这里开始
 - read/write结束后，变量自动加上读/写的字节数
- **seek**: 随时调整file position

2004-12-19

Input and Output

18

TC中的打开文件

- 下面介绍的函数均定义在TC的<io.h>中
- `int open(const char *pathname, int access);`
 - `int fd = open("C:\\\\AUTOEXEC.BAT", O_RDWR | O_CREAT | O_TEXT);`
- **filename**是文件名，包含路径。如果不含路径，表示打开当前目录下的文件
- **access**是打开方式，常用为O_RDONLY、O_WRONLY、O_RDWR与O_CREAT、O_TRUNC、O_APPEND以及O_TEXT、O_BINARY的或运算
- 返回值为文件句柄（File Handle），留待以后使用。如果打开失败，返回值为-1

2004-12-19

Input and Output

19

TC中的读文件

- `int read(int handle, void *buf, unsigned len);`
 - `int n_read = read(fd, buf, BUFSIZE);`
- **handle**是open获得的文件句柄
- 读入的数据将保存在buf指向的内存
- **len**是最大允许读入的字节数
- 返回值为实际读入的字节数，可能小于len。返回0表示读到了末尾，返回-1表示出错

2004-12-19

Input and Output

20

TC 中的写文件

- `int write(int handle, const void *buf, unsigned nbyte);`
 - `int n_write = write(fd, buf, BUFSIZ);`
- `handle`是`open`获得的文件句柄
- `buf`指向的数据被写入文件
- `nbyte`是写入多少字节
- 返回值为实际写入的字节数，可能小于`nbyte`。返回-1表示出错

2004-12-19

Input and Output

21

TC 中的关闭文件

- `int close(int handle);`
 - `int ret = close(fd);`
- `handle`是`open`获得的文件句柄
- 关闭成功返回0，否则返回-1



2004-12-19

Input and Output

22

TC 中的文件随机访问

- `long lseek(int handle, long offset, int fromwhere);`
 - `int pos = lseek(fd, 100L, SEEK_CUR);`
- `handle`是`open`获得的文件句柄
- `offset`是相对`fromwhere`的位置偏移多少，可以为负数
- `fromwhere`可以是`SEEK_SET`、`SEEK_CUR`或`SEEK_END`中的一个，分别表示文件头部、当前位置和文件末尾
- 成功返回移位后的当前位置，从文件头算起；否则返回-1L

2004-12-19

Input and Output

23

标准的文件操作

- `open()`、`read()`、`write()`、`close()`和`lseek()`并不是ANSI C支持的
- 它们定义在POSIX标准中
 - POSIX是Unix/Linux等众多操作系统遵循的标准，Windows下也有支持POSIX的编译器（Cygwin）
 - 在POSIX的世界里具有可移植性（TC并没有完全遵守这个标准）
- ANSI C有自己的标准文件操作函数
 - 它们包装了操作系统提供的文件操作接口
 - 在C语言的世界里具有可移植性

2004-12-19

Input and Output

24

ANSI C的打开文件

- 下面介绍的函数均定义在<stdio.h>中
- FILE* fopen(const char *filename, const char *mode);
 - FILE *fp = fopen("C:\\\\AUTOEXEC.BAT", "r+");
- filename是文件名, 包含路径。如果不含路径, 表示打开当前目录下的文件
- mode是打开方式, 常用为
 - "r"、"w"、"a", 分别表示只读、只写(如文件已存在, 则覆盖, 否则建立新文件)和添加(文件位置指针指向末尾。如文件不存在, 就建立新文件)
 - "+"用在"r"、"w"、"a"之后, 表示按读写方式打开
 - "b"和上面的各种组合合用(不可做第一个字符), 表示按二进制方式打开, 否则就是文本方式(Unix/Linux下忽略)
- 返回值是什么?

2004-12-19

Input and Output

25

fopen的返回值

- FILE是C语言定义的一个结构类型
 - 不同的库在成员的定义上会有所不同
 - stdin、stdout和stderr的类型是FILE*
- fopen成功后
 - 建立一个FILE结构的全局变量, 该变量里记录了访问文件的重要信息
 - 文件位置标记, 缓冲区指针, 文件句柄等
 - 该变量的地址作为返回值返回
- fopen失败后
 - 返回NULL

2004-12-19

Input and Output

26

ANSI C的文件操作

- int fgetc(FILE *fp);
- int fputc(int c, FILE *fp);
- char *fgets(char *s, int n, FILE *fp);
- int fputs(const char *s, FILE *fp);
- int fscanf(FILE *fp, const char *format, ...);
- int fprintf(FILE *fp, const char *format, ...);

2004-12-19

Input and Output

27

ANSI C的文件操作

- size_t fread(void *ptr, size_t size, size_t nmem, FILE *fp);
- size_t fwrite(const void *ptr, size_t size, size_t nmem, FILE *fp);
- int feof(FILE *fp);
- int fseek(FILE *fp, long offset, int whence);
- long ftell(FILE *fp);
- int fflush(FILE *fp);
- int fclose(FILE *fp);

copy_err.c

2004-12-19

Input and Output

28

错误处理

- 文件操作是事故多发区
 - 文件不存在
 - 磁盘满
 - 写一个只读文件，读一个只写文件
 - 设备损坏
 -
- 在事故发生时，程序必须能自如应对，这就需要错误处理技术

2004-12-19

Input and Output

29

错误处理技术

- **assert**
 - 一个宏，处理由程序bug引起的错误
 - bug解除，就不会abort了
 - 最终的执行文件中不应该包含assert
 - bug已经都消灭了
 - 在#include<assert.h>之前，#define NDEBUG，所有assert失效
- **分支**
 - 处理由外界引起的错误
 - 用户输入、文件、内存、系统.....
 - 需要仔细规划流程，用简单、直接的方法处理复杂的错误

2004-12-19

Input and Output

30

针对文件的错误处理策略

函数	当返回值	常用策略
fopen()	==NULL	转入错误处理流程
fprintf()	<0	
fputc()、putc()	==EOF	
fwrite()	<nmemb	
fputs()	==EOF	
fgets()	==NULL	停止再次调用该函数，继续下一步
fscanf()	==0 或 ==EOF	
fgetc()、getc()	==EOF	
fread()	<nmemb	
fclose()、fflush()	==EOF	忽略

2004-12-19

Input and Output

31

错误处理流程

- 根据errno，做适当的处理
 - extern int errno;
 - stdio模块中定义的全局变量
 - 它的值说明错误类型
- 用perror告诉用户错误的信息，使其可以协助排除错误
 - void perror(const char *string);
 - 向stderr输出string，并在其后附加一个':', 再输出错误对应的文字信息
- if (fp != NULL) fclose(fp);
- 回正常流程
 - 从头再来、转做其它事情或退出程序等

copy.c sorttext.c sortbin.c

2004-12-19

Input and Output

32

流 (Stream)

- “子在川上曰：逝者如斯夫”
- 时间、水、过客.....
你或许会记下她的样子，但她将永远不再出现
- 计算机中的流，数据流
 - 也有叫做字节流、比特流的，还有很具体的文件流、网络流、视频流、音频流等
- C语言中的输入与输出都是以流的方式处理的
- 计算机中的输入与输出也几乎都是以流的方式处理

2004-12-19

Input and Output

33

从文件说开去

- **open、read、write、close**这种模式成为使用数据流的经典模式
 - 比如网络数据流操作就是使用此模式，甚至在一些平台上可以和文件采用完全相同的函数操作
- **seek**并不是所有的流都支持
 - 网络流就不支持
- 对于不能进行“流控”的流
 - 如果当前数据不立即处理
 - 后来的数据就可能冲走当前数据
 - 于是当前数据丢失
- 流控成为流处理中的一个专门技术
 - 缓冲、握手协议

2004-12-19

Input and Output

34