程序设计语言与方法(C语言)

第十三章 文件

通讯薄

- 数据描述
  - 好友
    - 姓名
    - 性别
    - 出生日期
    - 电话
    - QQ
    - 通信地址
    - ……
- 操作（处理或计算）
    - 添加一个好友
    - 修改好友信息
    - 删除一个好友

问题

- 程序退出后，数据丢失
- 第次使用都要重新输入数据?
- 如何把我的通讯薄给其他好友（共享）？
- 打印到纸上?
  - 修改和添加怎么办?

文件

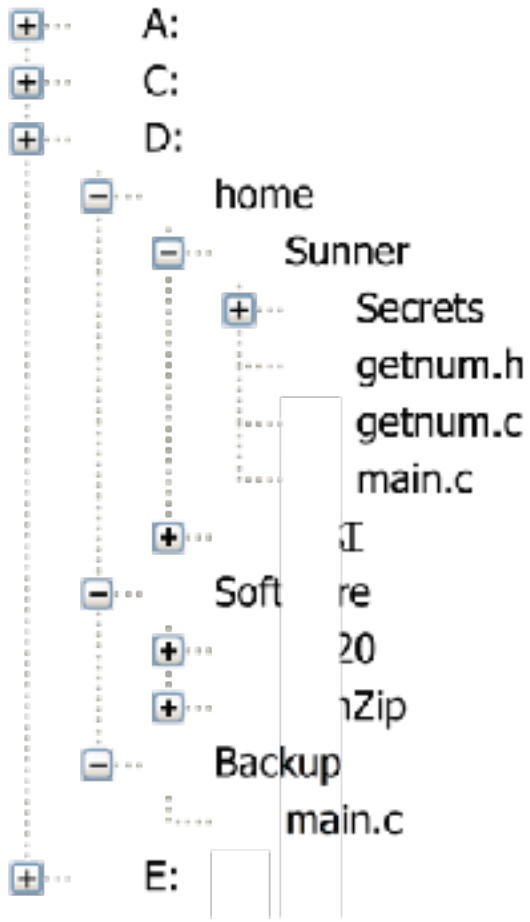
- 一般指存储在外部介质上有名字的一组相关数据的集合
- 用文件可长期保存数据，实现数据共享
- 在C语言中，文件可泛指磁盘文件、终端显示器或打印机……

程序中的文件

- 在程序运行时由程序在磁盘上建立一个文件，通过写操作将数据存入该文件
- 由程序打开磁盘上的某个已有文件，并通过读操作将文件中的数据读入内存供程序使用

计算机中的文件

- 硬盘
- 驱动器（文件存储的起点，根）
  - 文件夹（目录）
    - 子文件夹（子目录）
      - 文件（文件名.扩展名）



- 存储路径
- 存储文件
- 相对路径
- 绝对路径

- 资源管理器
- DOS（dir），MAC(ls)，等等

二进制文件和文本文件

- 二进制文件
  - 是一种字节序列，没有字符变换
  - 按照数据在内存中的存储形式（二进制）存储到文件
    - 短整数123，在内存中占几个字节，文件中也用几个字节来存储，直接存储与123对应的二进制序列（与内存中的存储形式对应）
- 文本文件
  - 是一种字符序列，文件中存储每个字符的ASCII码
    - 如整数123在文件中占3个字节，分别存放这3个字符的ASCII码
- 怎么存的就怎么读，这样才能保证数据的本来面貌
- 字节流与流式文件
  - 读写文件只受程序控制，对C语言来说，都是字节流

缓冲文件与非缓冲文件

- 缓冲型文件系统
  - 指系统自动在内存中为每一个正在使用的文件开辟一个缓冲区，在读写文件时，数据先送到缓冲区，再传给C程序或外存上
  - 缓冲文型件系统利用文件指针标识文件
  - 缓冲型文件系统中的文件操作，也称高级文件操作
  - 高级文件操作函数是ANSI C定义的文件操作函数，具有跨平台和可移植的能力
- 非缓冲型文件系统
  - 不会自动设置文件缓冲区，缓冲区需由程序员自己设定
  - 非缓冲型文件系统没有文件指针，它使用称为文件号的整数来标识文件

文件的操作流程

- 打开
  - fopen
- 读写
  - 字符与字节读写
  - 字符串读写
  - 格式化读写
- 关闭
  - fclose

打开文件

- FILE * fopen(const char * filename, const char * mode);
  - 按指定方式打开由文件名指定的文件，返回一个指向文件的指针，打开失败，返回空指针
  - 文件名（filename参数）
    - 文件名、含全路径的文件名、含相对路径的文件名
    - 符号“\”的使用
  - 文件打开方式（mode 参数）
    - r、w、a、b、+
    - 只读、只写、追加、二进制、组合
    - r, w, a, r+，w+，a+，rb、wb，ab，rb+，wb+，ab+
    - 读时文件必须已存在
    - 写文件时，若指定文件不存在，以指定名称创建文件
      - 写文件可能会覆盖文件原有的数据

读写文件

- 读写字符
  - fgetc, fputc　　P.384
- 读写字符串
  - fgets, fputs　　P.390
- 按格式读写
  - fscanf，fprintf P.391
- 按数据块读写
  - fread，fwrite　P.398
- 读写的时候有没有出现错误?
  - ferror，函数返回0值表示读写正确，非0值表示错误
- 啥时读完文件中的数据啊?
  - 读完最后一个数据
  - feof()

按字符读写文件示例

- 从键盘输入一串字符，转存到磁盘文件上

为什么要判断文件打开是否成功呢?

```
#include <stdio.h>
#include <stdlib.h>
int  main()
{
    FILE *fp;
    char ch;
    if ((fp = fopen("demo.txt","w")) == NULL) /* 判断文件是否成功打开 */
    {
        printf("Failure to open demo.txt!\n");
        exit(0);
    }
    ch = getchar();
    while (ch != '\n')   /* 若键入回车换行符则结束键盘输入和文件写入 */
    {
       fputc(ch, fp);
       ch = getchar();
    }
    fclose(fp);          /* 关闭由函数 fopen()打开的文件 demo.txt */
    return 0;
}
```

示例2

- 将0~127之间的ASCII字符写到文件中，然后从文件中读出并显示到屏幕上

- 如何判断字符是否
- 打印、控制字符?
  - isprint(ch)
  - iscntrl(ch)
  - char ch;/*一个字

```
int  main()
{
    FILE *fp;
    char ch;
    int i;
    if ((fp = fopen("demo.bin","wb")) == NULL) /* 以二进制写方式打开文件 */
    {
        printf("Failure to open demo.bin!\n");
        exit(0);
    }
    for (i=0; i<128; i++)
    {
        fputc(i, fp);     /* 将ASCII码值在 0~127 之间的所有字符写入文件 */
    }
    fclose(fp);
    if ((fp = fopen("demo.bin","rb")) == NULL) /* 以二进制读方式打开文件 */
    {
        printf("Failure to open demo.bin!\n");
        exit(0);
    }
    while ((ch = fgetc(fp)) != EOF)     /* 从文件中读取字符直到文件末尾 */
    {
        putchar(ch);          /* 在显示器上显示从文件读出的所有字符 */
    }
    fclose(fp);
    return 0;
}
```

随机读写

- Fseek
- Rewind
- Ftell
- fflush

输入输出重定向

- 输入重定向“<”：变更数据输入源
- 输出重定向“>”：变更数据输出设备