CH13 文件I/O_FILE I/O

摘要: 文件的形式有程序, 文档, 数据, 书信, 表格, 图形, 照片, 视频和许多其它种类的信息, 可以把文件看成这些内容的一个具体形式; **函数**: fopen, getc, putc, exit, fclose, fprintf, fscanf, fgets, fputs, rewind, fseek, ftell, fflush, fgetpos, fsetpos, feof, ferror, ungetc, setvbuf, fread, fwrite, (太多了); **其它**: 系统I/O和C标准I/O, 文件模式/二进制模式, 文本/二进制格式, 缓冲/无缓冲I/O, 顺序访问文件/随机访问文件的函数.

13.1 与文件通信

所谓与文件通信, 其实就是与文件进行数据的读取和写入. 文件(file)通常是磁盘上的一段已命名的存储区. 相较于C函数而言, 对操作系统而言, 文件会更复杂一些.

C提供两种文件模式: 文本模式, 二进制模式.

13.1.2 文本模式和二进制模式:

1.所有文件都以二进制形式(0和1)储存, 如果文件最初使用二进制编码的字符(ASCII或Unicode)表示文本, 那么该文件就是文本文件, 内容是文本内容. 如果文件中的二进制值代表机器语言代码或数值数据(int等)或图片或音乐编码, 该文件就是二进制文件, 内容是二进制内容.

(其实, 这个命名法有点晕, 我看是所有的文件都是二进制文件, 而原先的文本文件是二进制文本文件, 原先的二进制文件是二进制非文本文件; 二进制非文本文件之间的区别是解码不一样, 而文本文件的解码方式相较来说是固定的).

2.C提供两种方式访问文件: 二进制模式和文本模式, 它们的区别是在二进制模式中, 程序可以访问文件的每个字节, 而文本模式中, 程序所见的内容和文件的实际内容是不同的.

3.C提供了二进制模式和文本模式, 但因为UNIX只使用一种文件格式, 故这两种模式对于UNIX的实现而言, 完全相同. (C是标准C库的, 而UNIX是系统层次的)

13.1.3 I/O级别:

除了可以选择文件模式外, 还可以选择I/O级别(处理文件访问的两个级别): 底层I/O(low-level I/O), 标准高级I/O(standard high-level I/O).

13.1.4 标准文件:

C程序会自动打开3个文件: 标准输入, 标准输出, 标准错误输出, 默认情况下对应键盘, 显示屏, 显示屏.

标准输入为程序提供输入, 是getchar()和scanf()使用的文件; 程序的输出通常输出到标准输出文件, 也是putchar(), puts()和printf()使用的文件.

(满满的都是套路: C库(C标准库)也是用到UNIX系统调用的, 要不然呢?! C自己就是系统了, 就能读键盘输出到屏幕了?!)

```
键盘            -> 标准输入文件    -> getchar()
putchar()    -> 标准输出文件    -> 显示器
```

13.2 标准I/O

13.2.1 检查命令行参数

13.2.2 fopen()函数

13.2.3 getc()和putc()函数

13.2.4 文件结尾

13.2.5 fclose()函数

13.2.6 指向标准文件的指针

13.3 一个文件压缩程序

略.

13.4 文件I/O: fprintf(), fscanf(), fgets()和fputs()

13.4.1 fprintf()和fscanf()函数

13.4.2 fgets()和fputs()函数

13.5 随机访问: fseek()和ftell()

13.5.1 fseek()和ftell()工作原理: 略;

13.5.2 二进制模式和文本模式

13.5.3 可移植文件

13.5.4 fgetpos()和fsetpot()函数

13.6 标准I/O的机理

略.

13.7 其它标准I/O函数

13.7.1 int ungetc(int c, FILE *fp)函数

13.7.2 int fflush()函数

13.7.3 int setvbuf()函数

13.7.4 二进制IO: fread()和fwrite()

13.7.5 size_t fwrite()函数

13.7.6 size_t fread()函数

13.7.7 int feof(FILE *fp)和int ferror(FILE* fp)函数

13.7.8 一个示例程序

13.7.9 用二进制IO进行随机访问

13.8 关键概念

略.

13.9 小结

略.