操作系统实验报告

171491125 吴俊

1. **文件和流的关系**

C将每个文件简单地作为顺序字节流。每个文件用文件结束符结束，或者在特定字节数的地方结束，这个特定的字节数可以存储在系统维护的管理数据结构中。当打开文件时，就建立了和文件的关系。

　　在开始执行程序的时候，将自动打开3个文件和相关的流：标准输入流、标准输出流和标准错误。流提供了文件和程序的通信通道。

**2. C语言文件操作的底层实现简介**

**2.1 FILE结构体**

C语言的stdio.h头文件中，定义了用于文件操作的结构体FILE。这样，我们通过fopen返回一个文件指针(指向FILE结构体的指针)来进行文件操作。可以在stdio.h(位于visual studio安装目录下的include文件夹下)头文件中查看FILE结构体的定义，如下：

VC6.0中：

#ifndef \_FILE\_DEFINED

struct \_iobuf {

　　　　char \*\_ptr; //文件输入的下一个位置

　　　　int \_cnt; //当前缓冲区的相对位置

　　　　char \*\_base; //指基础位置(即是文件的其始位置)

　　　　int \_flag; //文件标志

　　　　int \_file; //文件的有效性验证

　　　　int \_charbuf; //检查缓冲区状况,如果无缓冲区则不读取

　　　　int \_bufsiz; //

　　　　char \*\_tmpfname; //临时文件名

};

typedef struct \_iobuf FILE;

#define \_FILE\_DEFINED

#endif

**2.2 C语言文件管理的实现**

C程序用不同的FILE结构管理每个文件。程序员可以使用文件，但是不需要知道FILE结构的细节。实际上，FILE结构是间接地操作系统的文件控制块。

**2.3 操作系统文件管理简介**

文件是存放在物理磁盘上的，包括文件控制块(FCB)和数据块。文件控制块通常包括文件权限、日期（创建、读取、修改）、拥有者、文件大小、数据块信息。数据块用来存储实际的内容。系统级打开文件表复制了文件控制块的信息等；进程级打开文件表保存了指向系统级文件表的指针及其他信息。

系统级文件表每一项都保存一个计数器，即该文件打开的次数。我们初次打开一个文件时，系统首先查看该文件是否已在系统级文件表中，如果不在，则创建该项信息，否则，计数器加1。当我们关闭一个文件时，相应的计数也会减1，当减到0时，系统将系统级文件表中的项删除。

进程打开一个文件时，会在进程级文件表中添加一项。每项的信息包括当前文件偏移量（读写文件的位置）、存取权限、和一个指向系统级文件表中对应文件项的指针。系统级文件表中的每一项通过文件描述符（一个非负整数）来标识。

**3.顺序访问文件**

**3.1 顺序写入文件**

写入文件大致需两步：定义文件指针和打开文件。

**3.2 文件打开模式**

|  |  |
| --- | --- |
| 模式 | 说明 |
| r | 打开文件，进行读取。 |
| w | 创建文件，以进行写入。如果文件已经存在，则删除当前内容。 |
| a | 追加，打开或创建文件以在文件尾部写入。 |
| r+ | 打开文件以进行更新(读取和写入)。 |
| w+ | 创建文件以进行更新。如果文件已经存在，则删除当前内容。 |
| a+ | 追加，打开或者创建文件以进行更新，在文件尾部写入。 |

**3.3 顺序读取文件**

**4.随机访问文件**

文件中用格式化输入函数fprintf所创建的记录的长度并不是完全一致的。然而，在随机访问文件中，单个记录的长度通常是固定的，而且可以直接访问(这样速度更快)而无需通过其他记录来查找。这使得随机文件访问适合飞机订票系统，银行系统，销售点系统和其他需要快速访问特定数据的事务处理系统。我们可以有很多方法来实现随机访问文件，但是这里我们将把讨论的范围限制在使用固定长度记录的简单方法上。文件处理程序很少向文件中写入字段。通常情况下，它们一次写入一个struct。

4.1 创建随机访问的文件

#include<stdio.h>

struct clientData

{

int acctNum;

char lastName[15];

char firstName[10];

double balance;

};

int main()

{

int i;

struct clientData blankClient={0,"","",0.0};

FILE \*cfPtr;

if ((cfPtr = fopen("credit.dat","wb"))== NULL)

{

printf("File could not be opened.\n");

}

else

{

for (i=1;i<=100;i++)

{

fwrite(&blankClient,sizeof(struct clientData),1,cfPtr);

}

fclose(cfPtr);

}

return 0;

}

fwrite(&blankClient,sizeof(struct clientData),1,cfPtr);用于向文件中写入一个数据块，其会在cfPtr指向的文件中写入大小为sizeof(struct clientData)的结构blankClient。当然，也可以写入对象数组的多个元素，只需把数组名传给第一个参数，把要写入的元素个数写入第三个参数即可。

**4.2 随机向随机访问文件中写入数据**

#include<stdio.h>

struct clientData

{

int acctNum;

char lastName[15];

char firstName[10];

double balance;

};

int main()

{

int i;

struct clientData client={0,"","",0.0};

FILE \*cfPtr;

if ((cfPtr = fopen("credit.dat","rb+"))== NULL)

{

printf("File could not be opened.\n");

}

else

{

printf("Enter account number(1 to 100, 0 to end input\):\n");

scanf("%d",&client.acctNum);

while (client.acctNum!=0)

{

printf("Enter lastname, firstname, balance\n");

fscanf(stdin,"%s%s%lf",client.lastName,client.firstName,&client.balance);

//在文件中定位用户指定的记录

fseek(cfPtr,(client.acctNum-1)\*sizeof(struct clientData),SEEK\_SET);

//将用户指定的信息写入文件

fwrite(&client,sizeof(struct clientData),1,cfPtr);

//输入下一个账号

printf("Enter account number:\n");

scanf("%d",&client.acctNum);

}

fclose(cfPtr);

}

return 0;

}  
fseek(cfPtr,(client.acctNum-1)\*sizeof(struct clientData),SEEK\_SET);将cfPtr所引用文件的位置指针移动到由(client.acctNum-1)\*sizeof(struct clientData)计算所得到的字节位置处，这个表达式的值称为偏移量或者位移。负号常量SEEK\_SET说明，文件位置指针指向的位置是相对于文件开头的偏移量。  
**4.2 从随机访问文件中读取数据**

#include<stdio.h>

struct clientData

{

int acctNum;

char lastName[15];

char firstName[10];

double balance;

};

int main()

{

struct clientData client={0,"","",0.0};

FILE \*cfPtr;

if ((cfPtr = fopen("credit.dat","rb"))== NULL)

{

printf("File could not be opened.\n");

}

else

{

printf("%-6s%-16s%-11s%10s\n","Acct","Last name","First name","Balance");

while(!feof(cfPtr))

{

fread(&client,sizeof(struct clientData),1,cfPtr);

if (client.acctNum!=0)

{

printf("%-6d%-16s%-11s%10.2f\n",client.acctNum,client.lastName,client.firstName,client.balance);

}

}

fclose(cfPtr);

}

return 0;

}