

文件打包实验

2020.5.20 isszym sysu.edu.cn

保存结构数据到缓冲区

把输入到结构变量的数据拷贝到缓冲区，然后再把缓冲区的数据拷贝到结构变量，最后显示出来。

StructBuffer.cpp

```
struct Person {  
    char username[USER_NAME_LEN];    // 员工名  
    int level;                        // 工资级别  
    char email[EMAIL_LEN];           // email地址  
    DWORD sendtime;                  // 发送时间  
    time_t regtime;                  // 注册时间  
};
```

输入结构数据并保存到缓冲区:

```
struct Person personSent;           // 要发送的员工记录
char inputBuf[100];
int inputNumber;

// 输入员工记录
printf("username: ");
scanf_s("%s", inputBuf, USER_NAME_LEN);
strcpy_s(personSent.username, inputBuf); // 输入用户名
printf("level: ");
scanf_s("%d", & inputNumber);          // 输入工资级别
personSent.level = inputNumber;
printf("email: ");
scanf_s("%s", inputBuf, EMAIL_LEN);
strcpy_s(personSent.email, inputBuf);    // 输入email
personSent.sendtime=(DWORD)now;         // 设置发送时间
personSent.regtime = now;               // 设置注册时间

// 把员工记录保存到缓冲区
memcpy(buf, &personSent, sizeof(Person));
```

从缓冲区读出结构数据并显示

```
/* 读出并显示一个员工记录 */
```

```
struct Person personRecv;  
char regtime[TIME_BUF_LEN];  
char sendtime[TIME_BUF_LEN];
```

```
// 把缓冲区数据拷贝到员工记录
```

```
memcpy(&personRecv,buf,sizeof(Person));
```

```
// 显示员工记录
```

```
printf("用户名: %s\r\n",personRecv.username); // 显示用户名  
printf("级别: %d\r\n",personRecv.level); // 显示工资级别  
printf("Email地址: %s\r\n",personRecv.email); // 显示email
```

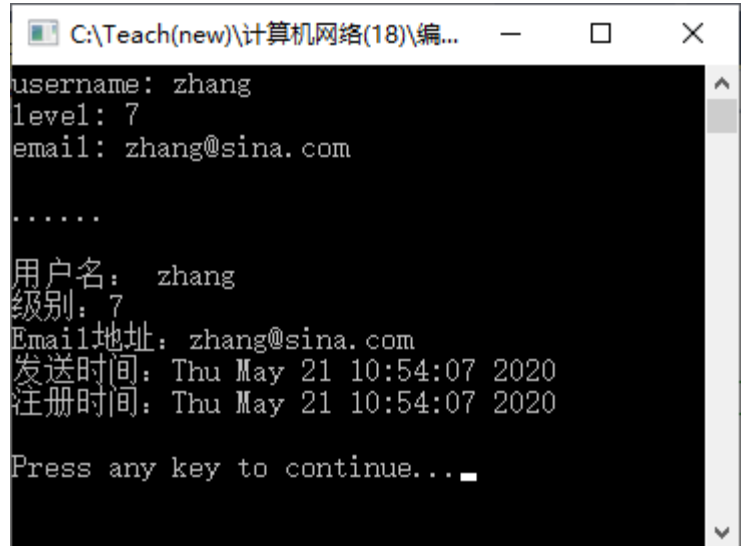
```
time_t t1 = (time_t)personRecv.sendtime;
```

```
ctime_s(sendtime,TIME_BUF_LEN,&t1);
```

```
printf("发送时间: %s",sendtime); // 显示发送时间
```

```
ctime_s(regtime,TIME_BUF_LEN,&personRecv.regtime);
```

```
printf("注册时间: %s",regtime); // 显示注册时间
```



```
C:\Teach(new)\计算机网络(18)\编...  
username: zhang  
level: 7  
email: zhang@sina.com  
.....  
用户名: zhang  
级别: 7  
Email地址: zhang@sina.com  
发送时间: Thu May 21 10:54:07 2020  
注册时间: Thu May 21 10:54:07 2020  
Press any key to continue...
```

完整的程序

StructBuffer.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define BUF_LEN 100
#define USER_NAME_LEN 20
#define EMAIL_LEN 80
#define TIME_BUF_LEN 30

typedef unsigned long DWORD;

int main(){
    char pts[TIME_BUF_LEN];    /* pointer to time string */
    time_t now;                /* current time */

    struct Person {
        char username[USER_NAME_LEN]; // 用户名
        int level;                     // 工资级别
        char email[EMAIL_LEN];         // email地址
        DWORD sendtime;                // 发送时间
        time_t regtime;                // 注册时间
    };

    (void)time(&now);              // 取得系统时间
    ctime_s(pts, TIME_BUF_LEN, &now); // 把时间转换为字符串

    struct Person personSent;
    char inputBuf[100];
    int inputNumber;
```

```

/* 输入员工记录 */
char buf[sizeof(Person)];
printf("username: ");
scanf_s("%s", inputBuf, USER_NAME_LEN); // 输入用户名
strcpy_s(personSent.username, inputBuf);
printf("level: ");
scanf_s("%d", &inputNumber); // 输入工资级别
personSent.level = inputNumber;
printf("email: ");
scanf_s("%s", inputBuf, EMAIL_LEN);
strcpy_s(personSent.email, inputBuf); // 输入电子邮件
personSent.sendtime = (DWORD)now; // 设置发送时间
personSent.regtime = now; // 设置注册时间

// 把员工记录保存到缓冲区
memcpy(buf, &personSent, sizeof(Person));

printf("\r\n.....\r\n\r\n");

/* 读出并显示一个员工记录 */
struct Person personRecv;
char regtime[TIME_BUF_LEN];
char sendtime[TIME_BUF_LEN];

```

```

// 把缓冲区数据拷贝到员工记录
memcpy(&personRecv, buf, sizeof(Person));

// 显示员工记录
printf("用户名: %s\r\n", personRecv.username); // 显示用户名
printf("级别: %d\r\n", personRecv.level); // 显示工资级别
printf("Email地址: %s\r\n", personRecv.email); // 显示email
time_t t1 = (time_t)personRecv.sendtime;
ctime_s(sendtime, TIME_BUF_LEN, &t1);
printf("发送时间: %s", sendtime); // 显示发送时间
ctime_s(regtime, TIME_BUF_LEN, &personRecv.regtime);
printf("注册时间: %s", regtime); // 显示注册时间

printf("\r\nPress any key to continue...");
getchar();
getchar();
return 0;
}

```

保存结构数据到文件

把结构数据保存到文件(c:\temp\aaa.stru)中，然后从文件中读出并显示出来。

/* 本文件把三个结构变量值写入文件，然后再从该文件中读出这些值并显示出来 */

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

StructSave.cpp

```
#define NAME_LEN 30
typedef struct {
    char name[NAME_LEN];
    int age;
} people;

int main() {
    FILE * pFile;
    int i;
    people per[3];

    strcpy_s(per[0].name, "li");
    per[0].age = 20;
    strcpy_s(per[1].name, "wang");
    per[1].age = 18;
    strcpy_s(per[2].name, "zhang");
    per[2].age = 21;
```

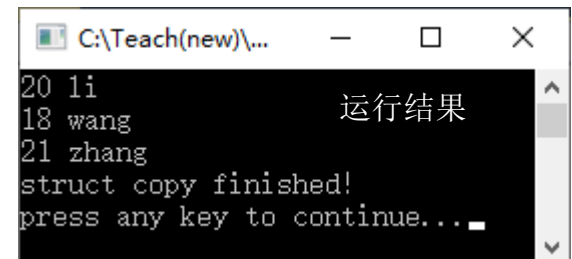


```

// 打开要写的二进制文件(w-write b-binary), 没有则创建, 有则覆盖
if ((fopen_s(&pFile, "c:\\temp\\aaa.stru", "wb"))!=NULL) {
    printf("cant open the file!\n");
    exit(0);
}
for (i = 0; i < 3; i++) {
    // fwrite的参数: 数据的起始地址, 每个数据的所占字节数, 要写入的数据个数, 文件句柄
    //          返回值: 实际写入的数据个数
    if (fwrite(&per[i], sizeof(people), 1, pFile) != 1) {
        printf("file write error!\n");
    }
}
fclose(pFile);

FILE * fp;
people perBuf;
// 打开要读出的二进制文件(r-read b-binary)
if (fopen_s(&fp, "c:\\temp\\aaa.stru", "rb") != NULL) {
    printf("can't open the file!\n");
    exit(0);
}
// fread的参数: 缓冲区起始地址, 每个数据的字节数, 可缓存的数据个数, 文件句柄
//          返回值: 实际读出的数据个数
while (fread(&perBuf, sizeof(people), 1, fp) == 1) {
    printf("%d %s\r\n", perBuf.age, perBuf.name);
}
printf("struct copy finished!\n");
printf("press any key to continue...");
return 0;
}

```



```

C:\Teach(new)\...
20 li
18 wang
21 zhang
struct copy finished!
press any key to continue...

```

运行结果

`size_t fread (void * ptr, size_t size, size_t count, FILE * stream);`

`ptr:` 指向保存结果的指针

`size:` 每个数据类型的大小

`count:` 数据的个数

`stream:` 文件指针

返回读取数据的个数

`size_t fwrite (const void * ptr, size_t size, size_t count, FILE * stream);`

`ptr:` 指向保存数据的指针

`size:` 数据类型的大小

`count:` 数据的个数

`stream:` 文件指针

返回写入数据的个数

* 写操作**`fwrite()`**后必须关闭流**`fclose()`**。

* 不关闭流的情况下，每次读或写数据后，文件指针都会指向下一个待写或者待读数据位置的指针。

文件拷贝

输入源文件名和目标文件名，然后把源文件名拷贝到目标文件。如果目标文件不存在，则创建它，如果源文件不存在则出错。

```
#include <stdio.h>
#include <stdlib.h>
```

FileCopy.cpp

```
#define BUF_LEN 20
#define FNAME_LEN 300
```

```
int main(){
    FILE *srcfile=NULL;
    FILE *destfile=NULL;
    char buf[20];
    char srcfname[FNAME_LEN];
    char destfname[FNAME_LEN];

    printf("源文件名: ");
    scanf_s("%s",srcfname,FNAME_LEN);

    printf("目标文件名: ");
    scanf_s("%s",destfname, FNAME_LEN);
```

```
//打开要读取的二进制文件(r-read b-binary)
if (fopen_s(&srcfile, srcfname, "rb")!=NULL){
    exit(1);
}

// 打开要写入的二进制文件(w-write b-binary), 没有则创建, 有则覆盖
if (fopen_s(&destfile, destfname, "wb")!= NULL){
    fclose(srcfile);
    exit(1);
}

int len = 0;
// fread的参数: 缓冲区起始地址, 每个数据(char类型)的字节数, 最多读出的数据个数, 文件句柄
//          返回值: 实际读出的数据个数
while ((len = fread(buf, 1, BUF_LEN, srcfile)) >= BUF_LEN) {
    //fwrite的参数: 数据起始地址, 每个数据(char类型)的字节数, 写入数据的个数, 文件句柄
    //          返回值: 实际写入的数据个数
    fwrite(buf, 1, BUF_LEN, destfile);
}
fwrite(buf, 1, len, destfile); // 写入剩余部分
fclose(srcfile);
fclose(destfile);
printf("file copy finished!\r\n");
printf("press any key to continue...");
getchar();
return 0;
}
```

实验4、文件打包实验

- 1、把输入的结构数据保存到文件中：循环输入员工(Person)的信息，每输入一个员工的信息，立即写入一个文件，直到输入的姓名为exit时跳出循环。

```
struct Person {  
    char userName[USER_NAME_LEN];    // 员工名  
    int level;                        // 工资级别  
    char email[EMAIL_LEN];           // email地址  
    DWORD sendtime;                  // 发送时间  
    time_t regtime;                  // 注册时间  
};
```

- 2、读出上面文件中保存的员工数据，并显示出来。
- 3、输入多个文件名，每输入一个，就把该文件写入一个打包文件中，当输入的文件名为exit时跳出循环。
- 4、解包上面得到的打包文件。如果有重名，则加序号进行保存。
- 5、把自己打包的文件发给同学进行互测。