

## 【实验题目】数据表示实验

【实验目的】掌握结构数据的保存和读取的方法。

### 【实验说明】

- ♦ 把源程序和可执行文件放在相应的**上交源码**目录中。
- ♦ **截屏**用按键(Ctrl+Alt+PrintScreen)截取当前窗口
- ♦ **把每段具有独立功能的代码单独写入一个函数有助于编码和调试。(新增)**

### 【参考资料】

- ♦ **C 语言函数分类:** [http://msdn.microsoft.com/zh-cn/library/2aza74he\(v=vs.110\).aspx](http://msdn.microsoft.com/zh-cn/library/2aza74he(v=vs.110).aspx)
- ♦ **C 语言字符串函数:** [http://msdn.microsoft.com/zh-cn/library/f0151s4x\(v=vs.110\).aspx](http://msdn.microsoft.com/zh-cn/library/f0151s4x(v=vs.110).aspx)
- ♦ **C 语言程序设计:** <http://www.runoob.com/cprogramming/> (新增)

### 【实验环境】

- ♦ Windows + VS 2012
- ♦ Linux + gcc

### 【实验内容】

#### (1) 结构数据保存和读出 (StructSave.cpp)

##### ▪ 实验要求:

循环输入员工(Person)的信息, 每输入一个员工的信息, 立即写入文件 (Persons.**stru**), 直到输入的姓名为**空**时跳出循环。然后, 读出该文件, 显示每个 Person 的信息。

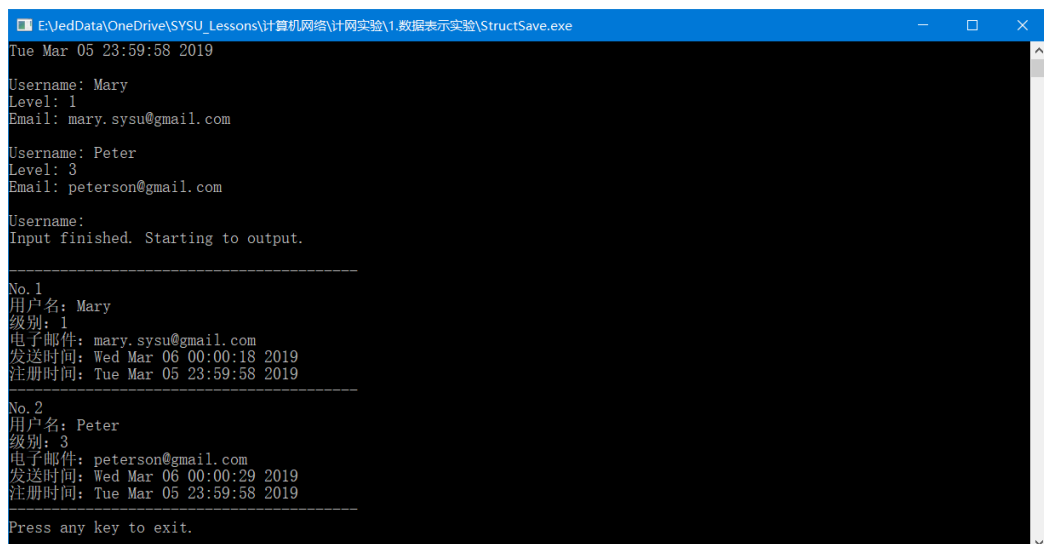
Person 的信息表示:

```
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
};
```

##### ▪ 老师用到的字符串函数和自定义函数(仅作参考): (新增)

```
printf(), scanf(), strcpy()      (VS 2017 要求使用 scanf_s, strcpy_s)
int inputOnePerson(Person *personSent) {...}
```

##### ▪ 截屏运行结果:



```
E:\JedData\OneDrive\SYSU_Lessons\计算机网络\计网实验\1.数据表示实验\StructSave.exe
Tue Mar 05 23:59:58 2019

Username: Mary
Level: 1
Email: mary.sysu@gmail.com

Username: Peter
Level: 3
Email: peter@gmail.com

Username:
Input finished. Starting to output.

-----
No.1
用户名: Mary
级别: 1
电子邮件: mary.sysu@gmail.com
发送时间: Wed Mar 06 00:00:18 2019
注册时间: Tue Mar 05 23:59:58 2019
-----
No.2
用户名: Peter
级别: 3
电子邮件: peter@gmail.com
发送时间: Wed Mar 06 00:00:29 2019
注册时间: Tue Mar 05 23:59:58 2019
-----
Press any key to exit.
```

▪ 源代码:

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

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

typedef unsigned long DWORD;

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 (填写每个用户信息的时间)
    time_t regtime; // 注册时间 (本程序启动的时间)
};

void inputPerson(time_t regtime); // 接受用户输入并写入 Persons.stru
void printPersons(); // 从读出 Persons.stru 并显示信息

int main()
{
    time_t startup_time = time(NULL); // 启动程序的时间
    printf("ctime(&startup_time));

    inputPerson(startup_time);
    printf("Input finished. Starting to output.\n\n");
    printPersons();

    printf("Press any key to exit.\n");
    getchar();
    return 0;
}

/* 接受用户输入并写入 Persons.stru */
void inputPerson(time_t regtime)
{
    FILE* outfile;
    char input_buf[BUF_LEN]; // 建立输入缓冲区
    struct Person person;

    if ((outfile = fopen(".\\Persons.stru", "wb")) == NULL)
    {
        printf("Can't open the file! Exiting...");
        getchar();
        exit(1);
    }

    // while (~scanf("%s", input_buf)) {
    while (1) {
        printf("\nUsername: ");
        fgets(input_buf, BUF_LEN, stdin);
        if(input_buf[0]=='\n') { break; } // 输入为空时跳出循环
        input_buf[strlen(input_buf)-1] = '\0'; // 将换行符替换为空字符
        strcpy(person.username, input_buf);

        printf("Level: ");
        scanf("%d", &person.level);

        printf("Email: ");
        scanf("%s", input_buf);
        strcpy(person.email, input_buf);

        person.sendtime = (DWORD)time(NULL);
        person.regtime = regtime; // 注册时间: 本程序启动的时间

        if (fwrite(&person, sizeof(struct Person), 1, outfile) != 1) {
            printf("File write error!\n");
        }
        getchar(); // 丢弃最后一个换行符
    }
}
```

```

    }
    fclose(outfile);
}

/* 读出Persons.stru 并显示信息 */
void printPersons()
{
    FILE * infile;
    struct Person person;
    int count = 1;

    if ((infile = fopen(".\\Persons.stru", "rb")) == NULL)
    {
        printf("Can't open the file! Exiting...");
        getchar();
        exit(1);
    }
    printf("-----\n");
    while (fread(&person, sizeof(struct Person), 1, infile) == 1) {
        printf("No.%d\n", count++);
        printf("用户名: %s\n", person.username);
        printf("级别: %d\n", person.level);
        printf("电子邮件: %s\n", person.email);
        time_t sendtime = person.sendtime;
        printf("发送时间: %s", ctime(&sendtime));
        printf("注册时间: %s", ctime(&person.regtime));
        printf("-----\n");
    }
}

```

## (2) 多文件合并保存和读出(FilePack.cpp)

### ■ 实验要求:

循环输入多个文件名（不超过 200MB，可以自己确定），每输入一个，就把该文件的文件名（最多 300 字节）、文件大小(long) 和文件内容写入文件 FileSet.pak 中，输入文件名为空时跳出循环。然后，读 FileSet.pak，每读出一个文件就把它保存起来，有重名文件存在时文件名加上序号（从 2 开始）。

\* 合并时可以先取得文件大小，然后边读边写。

### ■ 老师用到的字符串函数和自定义函数(仅作参考): (新增)

strcpy(), scanf(), printf()

sprintf()—用于多个字符串和整数合并成一个字符串

strrchr()—反向查找字符

```

struct FileStruct {
    char fileName[300];
    __int64 fileSize;
};

__int64 getFileSize(char * fileName) {...}
char * getFileName(char *pathName) {...}
int packFile(char *srcFileName, FILE * destFile) {...}
// 拷贝filePathName中前面长度为len的字符串到fileFullName
int mystrcpy(char * fileFullName, int len, char * filePathName) {...}
void getUniqueName(char *newFileName, char *filePathName) {...}
int unpackFile(FILE *srcFile, char *Path) {...}

```

### ■ 截屏运行结果:

我的测试在“testfolder”文件夹内进行。初始时，文件夹内存在三个用于测试的文件，分别是 a.text、b.pdf、c.jpg。此外，PackFiles.exe 为我的程序。

名称	状态	修改日期	类型	大小
a.text	✓	2019-3-5 星期二 ...	TEXT 文件	1 KB
b.pdf	✓	2019-2-27 星期...	PDF 文档	2,036 KB
c.jpg	✓	2018-4-18 星期...	JPG 文件	4,335 KB
PackFiles.exe	✓	2019-3-5 星期二 ...	应用程序	161 KB

首先，三个测试文件都可以正常打开，如图所示：



下面运行我的程序，随意地进行测试。如下图，一共输入了 7 个有效的文件名，其中包含若干重复项：

```
E:\JedData\OneDrive\SYSU_Lessons\计算机网络\计网实验\1.数据表示实验\testfolder\PackFiles.exe

[+] Name of file 1: a.text
Succeeded on a.text, 29 byte(s) written.
[+] Name of file 2: b.pdf
Succeeded on b.pdf, 2084156 byte(s) written.
[+] Name of file 3: b.pdf
Succeeded on b.pdf, 2084156 byte(s) written.
[+] Name of file 4: a.text
Succeeded on a.text, 29 byte(s) written.
[+] Name of file 5: c.jpg
Succeeded on c.jpg, 4438382 byte(s) written.
[+] Name of file 6: 某个不存在的文件名
[-] Can't open the file! Try again.
[+] Name of file 6: a.text
Succeeded on a.text, 29 byte(s) written.
[+] Name of file 7: b.pdf
Succeeded on b.pdf, 2084156 byte(s) written.
[+] Name of file 8:
7 file(s) have been packed into FileSet.pak.

a(2).text created, sized 29 byte(s).
b(2).pdf created, sized 2084156 byte(s).
b(3).pdf created, sized 2084156 byte(s).
a(3).text created, sized 29 byte(s).
c(2).jpg created, sized 4438382 byte(s).
a(4).text created, sized 29 byte(s).
b(4).pdf created, sized 2084156 byte(s).
All finished!
Press any key to exit.
```

执行程序后，testfolder 文件夹的内容变为下图：

名称	状态	修改日期	类型	大小
a(2).text	✓	2019-3-5 星期二 ...	TEXT 文件	1 KB
a(3).text	✓	2019-3-5 星期二 ...	TEXT 文件	1 KB
a(4).text	✓	2019-3-5 星期二 ...	TEXT 文件	1 KB
a.text	✓	2019-3-5 星期二 ...	TEXT 文件	1 KB
b(2).pdf	✓	2019-3-5 星期二 ...	PDF 文档	2,036 KB
b(3).pdf	✓	2019-3-5 星期二 ...	PDF 文档	2,036 KB
b(4).pdf	✓	2019-3-5 星期二 ...	PDF 文档	2,036 KB
b.pdf	✓	2019-2-27 星期...	PDF 文档	2,036 KB
c(2).jpg	✓	2019-3-5 星期二 ...	JPG 文件	4,335 KB
c.jpg	✓	2018-4-18 星期...	JPG 文件	4,335 KB
FileSet.pak	✓	2019-3-5 星期二 ...	PAK 文件	10,443 KB
PackFiles.exe	✓	2019-3-5 星期二 ...	应用程序	161 KB

观察发现，确实多出来了 7 个新文件，这与我们之前添加的文件数量相同。对于文件名重名的情况，我的程序确实能够在后面添加序号从而避免重名。然后，分别打开这些新增的文件，内容都与源文件相同。测试成功！

■ 源代码:

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

#define BUF_SIZE 1024
char buf[BUF_SIZE]; // 建立全局缓冲区

void packFiles(const char* destfile);
void unpackFiles(const char* destfile);
void validateFilename(char* filename);

int main()
{
    printf("注意: 仅接受文件名, 不支持绝对路径.\n\n");
    packFiles("FileSet.pak"); // 调用打包函数
    unpackFiles("FileSet.pak"); // 调用解包函数

    printf("Press any key to exit.\n");
    getchar();
    return 0;
}

/* 打包 */
void packFiles(const char* destfilename)
{
    FILE *destfile = fopen(destfilename, "wb");
    if (destfile == NULL) {
        printf("[-] Can't open FileSet.pak! Exiting...\n");
        getchar();
        exit(1);
    }
    char filename[300]; // 用于储存文件名的缓冲区
    int count = 1; // 文件计数

    while(1) {
        printf("[+] Name of file %d: ", count);
        fgets(filename, 300, stdin);
        if(filename[0]=='\n') { break; } // 输入为空时跳出循环
        filename[strlen(filename)-1] = '\0'; // 将换行符替换为空字符

        FILE *srcfile = fopen(filename, "rb");
        if (srcfile == NULL)
        {
            printf("[-] Can't open the file! Try again.\n");
            continue;
        }
        count++;

        /* 写入文件名 (300 个字节) */
        fwrite(filename, 300, 1, destfile);

        /* 写入文件大小 (__int64) */
        fseek(srcfile, 0, SEEK_END);
        __int64 filesize = ftell(srcfile);
        fseek(srcfile, 0, SEEK_SET);
        fwrite(&filesize, sizeof(__int64), 1, destfile);

        /* 写入文件内容 */
        int remain_bytes = 0;
        while((remain_bytes = fread(buf, 1, BUF_SIZE, srcfile)) >= BUF_SIZE) {
            fwrite(buf, 1, BUF_SIZE, destfile);
        }
        fwrite(buf, 1, remain_bytes, destfile);
        fclose(srcfile);
        printf("Succeeded on %s, %I64d byte(s) written.\n", filename, filesize);
    }
    fclose(destfile); // 打包完成, 关闭文件
    printf("%d file(s) have been packed into FileSet.pak.\n", count-1);
    printf("-----\n");
}

/* 解包 */
void unpackFiles(const char* srcfilename)
{
    FILE *srcfile = fopen(srcfilename, "rb");
```

```

if (srcfile == NULL) {
    printf("[-] Can't open FileSet.pak! Exiting...");
    getchar();
    exit(1);
}

while(1) {
    /* 读文件名 */
    char filename[300];
    if(fread(filename, 300, 1, srcfile) != 1) { break; }

    validateFilename(filename); // 构造有效的（不重名）的文件名

    FILE* destfile = fopen(filename, "wb");
    if (destfile == NULL) {
        printf("[-] Can't open %s! Exiting...", filename);
        getchar();
        exit(1);
    }

    /* 读文件大小 */
    __int64 filesize;
    fread(&filesize, sizeof(__int64), 1, srcfile);

    /* 读文件内容 */
    int currentsize = 0;
    while(currentsize < filesize) {
        int writesize = (BUF_SIZE>filesize-currentsize) ? filesize-currentsize :
BUF_SIZE;
        fread(buf, 1, writesize, srcfile);
        currentsize += writesize;
        fwrite(buf, 1, writesize, destfile);
    }
    fclose(destfile);
    printf("%s created, sized %I64d byte(s).\n", filename, filesize);
}
printf("All finished!\n");
}

/* 构造有效的文件名（即为重名文件添加序号） */
void validateFilename(char* filename)
{
    char temp[300];
    char numstr[10];
    int num = 1;
    while(access(filename, 0)==0) { // 文件已存在（重名）
        int dotpos = strrchr(filename, '.') - filename; // 从右往左找到小数点的位置
        if(strrchr(filename, '(')!=NULL && strrchr(filename, ')')!=NULL) {
            int leftpos = strrchr(filename, '(') - filename; // 左括号的位置
            strncpy(temp, filename+dotpos, 300);
            filename[leftpos] = '\0';
            strcat(filename, temp, 300); // 恢复不带序号的文件名
        }
        dotpos = strrchr(filename, '.') - filename; // 更新小数点的位置
        num++; // 递增序号
        numstr[0] = '('; // 添加左括号
        itoa(num, numstr+1, 10); // 添加序号
        numstr[strlen(numstr)+1] = '\0';
        numstr[strlen(numstr)] = ')'; // 添加右括号
        strncpy(temp, filename, dotpos);
        temp[dotpos] = '\0';
        strcat(temp, numstr);
        strcat(temp, filename + dotpos);
        strncpy(filename, temp, 300); // 把添加了序号的文件名拷贝回filename
    }
}

```

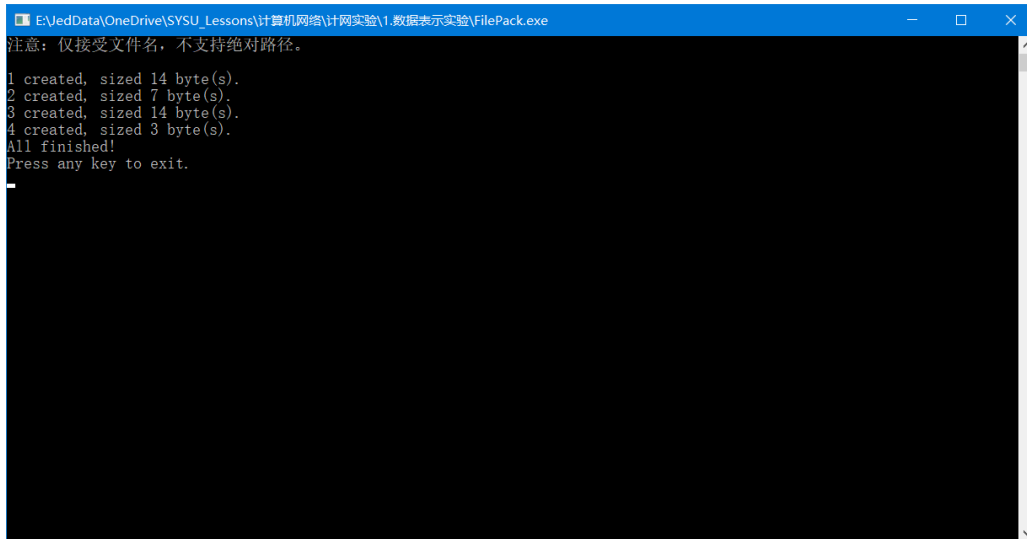
- 与同学互测并截屏运行结果：

把打包的文件给同学，看他是否可以取出其中文件，同样测试是否可以读出并取出同学的打包文件。

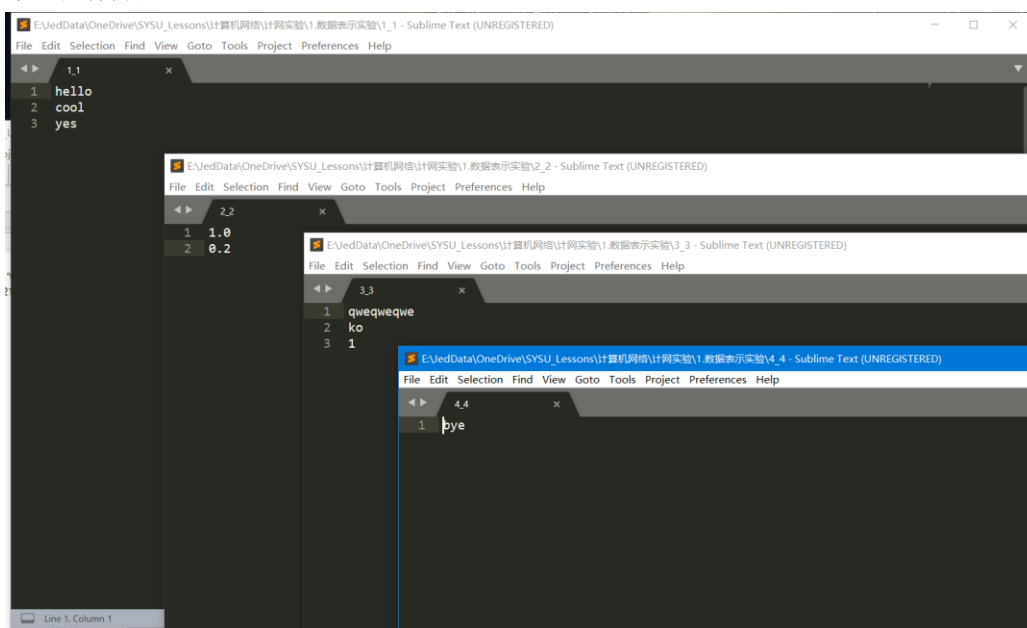
\* 注意结构要相同

在和同学统一了文件结构后，我们可以相互读取对方创建的 FileSet.pak 文件并恢复其中的内容。

执行程序进行解包：



检查文件内容：



经检查，内容和同学自己打包的文件内容完全相同。测试成功。

### 【实验体会】

1. **关于C语言。**系统地学习C语言是在大的一学期，自从学了C++后，写纯C语言程序的次数非常少，因此我希望借此实验来复习一下C语言的格式化输入输出、C风格字符串函数、文件读写、结构体等。因此，我使用纯C语言来完成本次实验，没有用到任何C++特性。
2. **输入为空时跳出循环。**众所周知，普通的scanf函数用%s读取字符串是会吞掉换行符\n的，因此不能使用传统的“while(~scanf("%s",buf))”作为循环条件，否则必须使用EOF（Windows下为Ctrl+Z），不符合实验要求。一种解决方法是使用fgets从stdin读入，值得注意的是，该函数读取并保留换行符\n，因此必须手动地将\n替换为\0作为字符串的正确结尾。每次读入后，判断第一个字符是否为换行符，如果是，则break出循环。
3. **fgets和scanf混用。**由于fgets接受\n，scanf不接受\n，因此混用时可能会出现缓冲区内残留的\n字符被fgets读取的情况，这是不希望发生的。所以，要在每次循环的最后，也即调用fgets之前使用getchar()吞掉多余的换行符。
4. **日期时间<time.h>头文件。**个人认为此模块总体来说不是很好用，不过其中的time()函数、ctime()函数用起来还是很方便的。老师的示例代码中是调用time(&now)获取当前系统时间的，而我更喜欢用now=time(NULL)这种形式。两种方法的效果是一样的。

5. **文件读写完毕后应关闭文件。**否则可能会出现执行了写入动作但却没有写入文件的情况。
6. **实验体会：**通过本次实验，我收获最大的地方在于熟悉了 C 语言的文件读写，主要用到了 `fopen`、`fread`、`fwrite` 函数，此外还用了 `fseek` 和 `ftell` 函数作为辅助。在实现为重名文件添加序号的功能时，我复习了 `strcpy`、`strlen`、`strcat` 等 C 风格字符串函数，发现它们比起 C++ STL 的 `string` 类更接近底层，使用起来也更加复杂。然而，事实上我的代码完全没有考虑鲁棒性和安全性，我知道我的程序中存在严重的缓冲区溢出漏洞，而且程序也无法处理错误的用户输入。不过，我认为本次实验的重点不在于这些地方，因此也就没有花时间专门去处理了。