



计算机网络实验报告

实验一：数据表示实验

数据科学与计算机学院 17大数据与人工智能

17341015 陈鸿峰

一、实验目的

掌握结构数据的保存和读取方法。

二、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

三、参考资料

- 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++文件流: <http://www.cplusplus.com/doc/tutorial/files/>

四、实验环境

本机为Windows 10 + gcc 7.3.0

五、实验内容

1. 结构数据保存和读出(StructSave.cpp)

(i) 实验要求

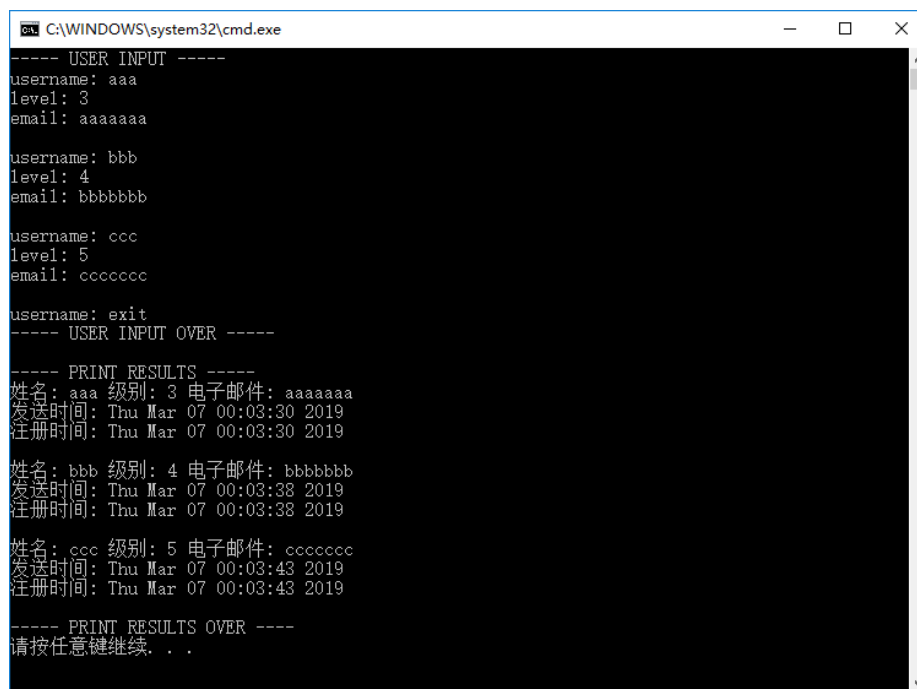
循环输入员工(Person)的信息，每输入一个员工的信息，立即写入文件(Persons.stru)，直到输入的姓名为exit时跳出循环。然后读出该文件，显示每个Person的信息。

Person的信息表示：

```
1 struct Person {  
2     char username[USER_NAME_LEN];  
3     int level;  
4     char email[EMAIL_LEN];  
5     DWORD sendtime;  
6     time_t regtime;  
7 };
```

(ii) 运行结果

控制行运行结果如下，并在当前文件夹中产生文件Persons.stru。注意这里将用户输入和程序输出分开为两个部分。



```
C:\WINDOWS\system32\cmd.exe
----- USER INPUT -----
username: aaa
level: 3
email: aaaaaaa

username: bbb
level: 4
email: bbbbbbb

username: ccc
level: 5
email: ccccccc

username: exit
----- USER INPUT OVER -----

----- PRINT RESULTS -----
姓名: aaa 级别: 3 电子邮件: aaaaaaa
发送时间: Thu Mar 07 00:03:30 2019
注册时间: Thu Mar 07 00:03:30 2019

姓名: bbb 级别: 4 电子邮件: bbbbbbb
发送时间: Thu Mar 07 00:03:38 2019
注册时间: Thu Mar 07 00:03:38 2019

姓名: ccc 级别: 5 电子邮件: ccccccc
发送时间: Thu Mar 07 00:03:43 2019
注册时间: Thu Mar 07 00:03:43 2019

----- PRINT RESULTS OVER -----
请按任意键继续. . .
```

(iii) 源代码

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <time.h>
5
6 #define BUF_LEN 100
7 #define USER_NAME_LEN 20
8 #define EMAIL_LEN 80
9 #define TIME_BUF_LEN 30
10 #define MAX_INT 0x3f3f3f3f
11
12 typedef unsigned long DWORD;
13
14 typedef struct Person {
15     char username[USER_NAME_LEN];
16     int level;
17     char email[EMAIL_LEN];
18     DWORD sendtime;
19     time_t regtime;
```

```
20 } Person;
21
22 int inputOnePerson(FILE* pfile)
23 {
24     Person person;
25
26     fflush(stdin);
27     char name[USER_NAME_LEN];
28     printf("username: ");
29     gets(name);
30     if (strcmp(name,"exit") == 0)
31         return 0;
32     strcpy(person.username,name);
33     fprintf(pfile, "%s\n", name);
34
35     int l;
36     printf("level: ");
37     scanf("%d",&l);
38     person.level = l;
39     fprintf(pfile, "%d\n", l);
40
41     char email[EMAIL_LEN];
42     printf("email: ");
43     scanf("%s",&email);
44     strcpy(person.email,email);
45     fprintf(pfile, "%s\n", email);
46
47     time_t now; // current time
48     time (&now); // get now time
49     struct tm* lt = localtime (&now);
50     person.sendtime = (DWORD)now;
51     person.regtime = now;
52     fprintf(pfile, "%ld\n", person.sendtime);
53     fprintf(pfile, "%ld\n", person.regtime);
54
55     printf("\n");
56     return 1;
57 }
58
59 int main()
60 {
61     FILE* pfile;
62     int i;
63     // Input
64     pfile = fopen("./Persons.stru","wb");
```

```
65     printf("----- USER INPUT -----\\n");
66     for (i = 0; i < MAX_INT; ++i)
67         if (!inputOnePerson(pfile))
68             break;
69     fclose(pfile);
70     printf("----- USER INPUT OVER -----\\n\\n");
71
72     // Read the file
73     pfile = fopen("Persons.stru", "r");
74     if (pfile == NULL)
75         exit(EXIT_FAILURE);
76     printf("----- PRINT RESULTS -----\\n");
77     for (i = 0; i < MAX_INT; ++i){
78         char name[USER_NAME_LEN];
79         if (fscanf(pfile, "%s", name) != 1)
80             break;
81         printf("姓名: %s ", name);
82
83         int l;
84         fscanf(pfile, "%d", &l);
85         printf("级别: %d ", l);
86
87         char email[EMAIL_LEN];
88         fscanf(pfile, "%s", email);
89         printf("电子邮件: %s\\n", email);
90
91         char buf[TIME_BUF_LEN];
92         time_t sendtime;
93         fscanf(pfile, "%ld", &sendtime);
94         struct tm* lt = localtime (&sendtime);
95         // Www Mmm dd hh:mm:ss yyyy\\n
96         strftime(buf, TIME_BUF_LEN, "%a %b %d %H:%m:%S %Y", lt);
97         printf("发送时间: %s\\n", buf);
98
99         time_t regtime;
100        fscanf(pfile, "%ld", &regtime);
101        lt = localtime (&regtime);
102        strftime(buf, TIME_BUF_LEN, "%a %b %d %H:%m:%S %Y", lt);
103        printf("注册时间: %s\\n", buf);
104
105        printf("\\n");
106    }
107    printf("----- PRINT RESULTS OVER -----\\n");
108    fclose(pfile);
109    return 0;
```

2. 多文件合并保存和读出(FilePack.cpp)

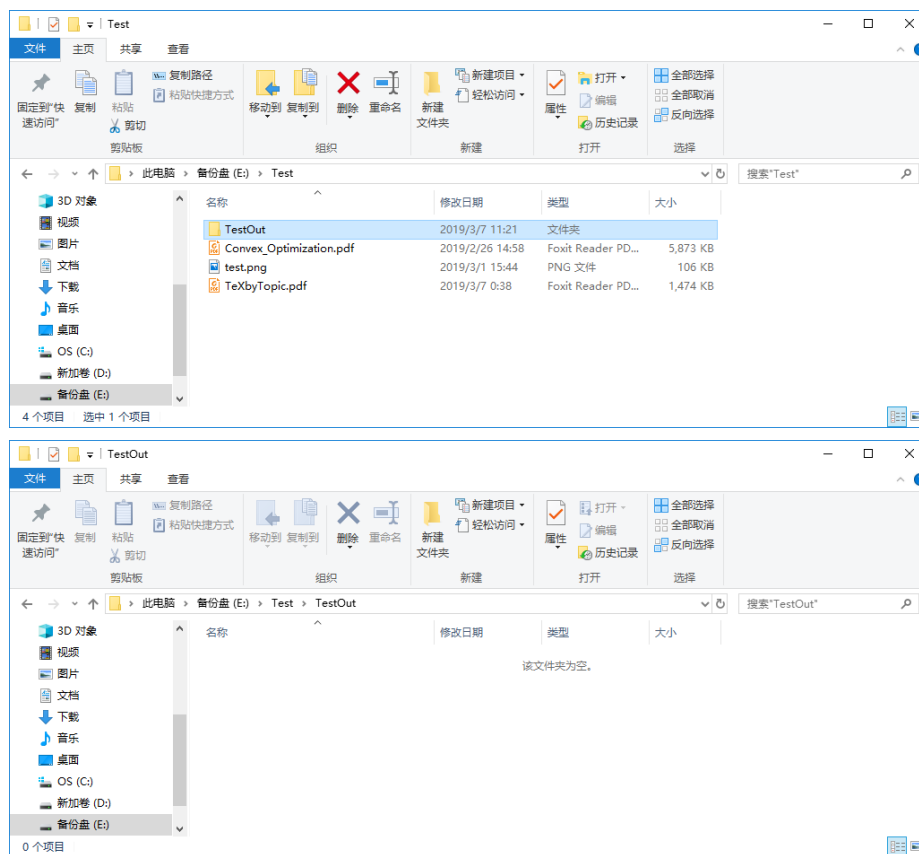
(i) 实验要求

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exit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）。

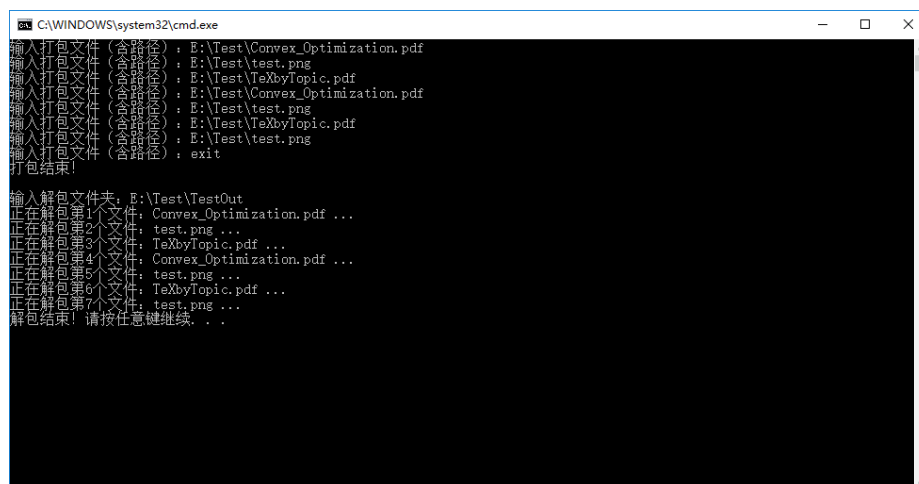
注意：合并时可以先取得文件大小，然后边读边写。

(ii) 运行结果

E:\Test下的三个文件：Convex_Optimization.pdf、test.PNG、TeXbyTopic.pdf，并有一个空文件夹E:\Test\TestOut

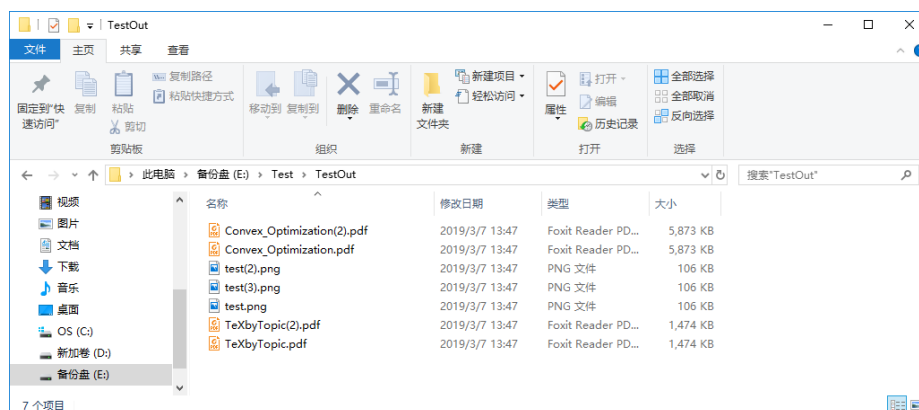


运行程序控制行结果如下



注意在我的程序中FileSet.pak生成位置与执行文件FilePack.exe的位置相同，在这里没有显示出来，但文件大小与所有打包文件大小之和相同。

文件生成展示如下，见最后一列可见生成文件与原文件大小相同。

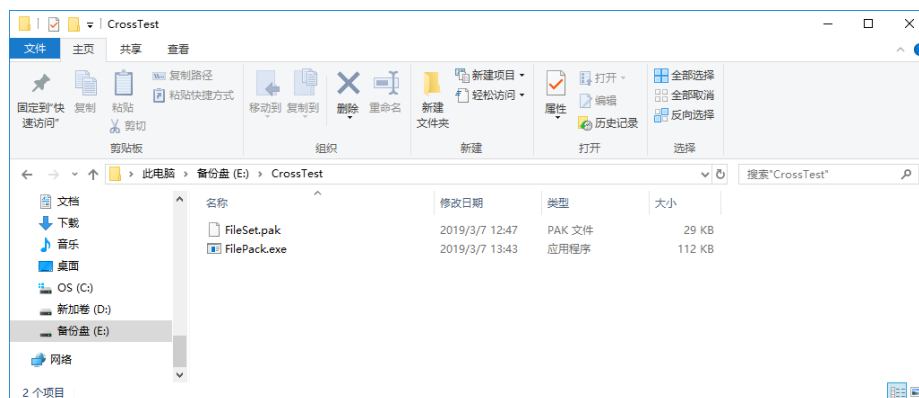


(iii) 与同学互测并截屏运行结果

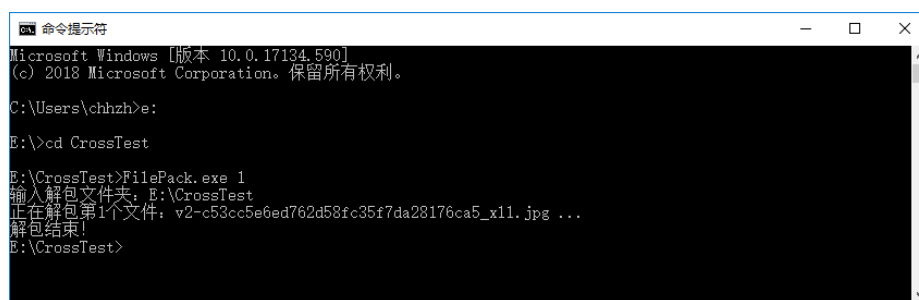
把保存的文件(FileSet.pak)给同学，看他是否可以取出其中文件，同样可以测试是否可以读出并保存同学的文件。注意结构要相同。

为方便测评，在我的程序中添加了从命令行读入指令的操作（源程序121行），如果读入指令为1，则直接进行解压操作。

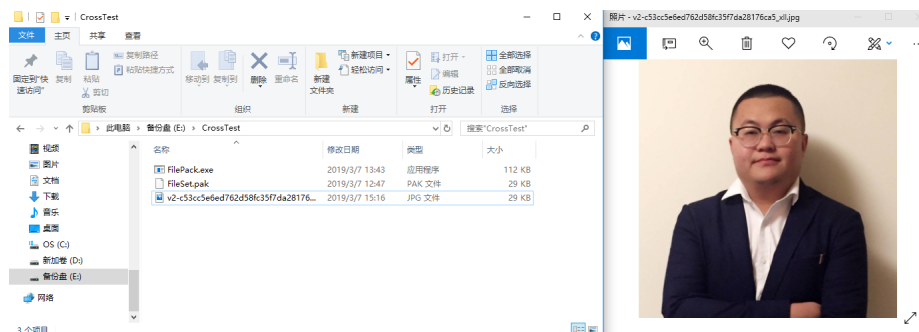
原来的文件夹E:\CrossTest中只有执行文件和pak文件。



命令行执行



从下面的结果可以看出我的程序可以成功解压同学的FileSet.pak文件，并生成对应图片且可以正常打开。



(iv) 源代码

```

1 #include <iostream>
2 #include <fstream>
3 #include <cstdlib>
4 #include <unistd.h>
5 #include <string>
6 using namespace std;
7
8 #define PAK_FILE_PATH "FileSet.pak"
    
```

```
9
10 class FileClass
11 {
12 public:
13     FileClass(const string _filepath):
14         filepath(_filepath){};
15
16     string getFileName()
17     {
18         if (filename != "")
19             return filename;
20         for (int i = filepath.size() - 1; i >= 0; --i)
21             if (filepath[i] != '\\')
22                 filename = filepath[i] + filename;
23             else
24                 break;
25         return filename;
26     }
27
28     ifstream::pos_type getSize()
29     {
30         ifstream in(filepath, ifstream::ate | ifstream::binary); // at the end of
31         ↪ file
32         return in.tellg();
33     }
34
35     ifstream getStream()
36     {
37         ifstream input(filepath, ios::binary);
38         return input;
39     }
40 private:
41     string filepath;
42     string filename;
43 };
44
45 bool packFile(const string src, const string dst)
46 {
47     FileClass infile(src);
48     ofstream outfile(dst, ios::app | ios::binary); // append
49
50     outfile << infile.getFileName() << endl;
51     outfile << infile.getSize() << endl;
52     ifstream input = infile.getStream();
```



```
53     string str;
54     while (getline(input,str))
55         outfile << str << endl;
56
57     outfile.close();
58     return true;
59 }
60
61 bool unpackFile(const string srcFile, const string dstPath)
62 {
63     ifstream input(srcFile,ios::binary);
64     string dst = dstPath;
65     for (int i = 1; true ; ++i){
66         string str, filename;
67         if (!getline(input,filename))
68             break;
69         if (filename == "")
70             if (!getline(input,filename))
71                 break;
72         if (filename.find("/") != -1 || filename.find("\\") != -1)
73             for (int i = filename.length() -1; i >= 0; --i)
74                 if (filename[i] == '/' || filename[i] == '\\'){
75                     filename = filename.substr(i+1,filename.length()-i);
76                     break;
77                 }
78         cout << "正在解包第" << i << "个文件: " << filename << " ..." << endl;
79         getline(input,str); // size
80         streampos size = stol(str);
81
82         // cout << "FileName: " << filename << endl;
83         // cout << "Size: " << exp_size << endl;
84
85         fstream output_file;
86         // get output file name
87         string path;
88         if (dst[dst.length()-1] != '\\')
89             dst += "\\";
90         int cnt = 1;
91         while (true){
92             if (cnt == 1)
93                 path = dst + filename;
94             else{
95                 int index = filename.find(".");
96                 if (index != -1){
97                     string suffix = filename.substr(index,filename.size()-index);
```

```
98         path = dst + filename.substr(0,index)
99         + "(" + to_string(cnt) + ")" + suffix;
100     } else {
101         path = dst + filename + "(" + to_string(cnt) + ")";
102     }
103 }
104 if (access(path.c_str(), F_OK) == -1) // should NOT use fstream
105     break;
106 cnt++;
107 }
108
109 ofstream output(path, ios::out|ios::binary);
110 char* memblock = new char [size];
111 input.read(memblock,size);
112 output.write(memblock,size);
113 output.close();
114
115 delete [] memblock;
116 }
117 }
118
119 int main(int argc, char *argv[])
120 {
121     if (argc == 1){
122         ofstream output(PAK_FILE_PATH); // initialization
123         output.close();
124         while (true){
125             cout << "输入打包文件 (含路径): ";
126             string src_file_path;
127             getline(cin,src_file_path);
128             if (src_file_path == "exit")
129                 break;
130
131             packFile(src_file_path,PAK_FILE_PATH);
132         }
133         cout << "打包结束! \n" << endl;
134     }
135
136     cout << "输入解包文件夹: ";
137     string output_path;
138     cin >> output_path;
139
140     unpackFile(PAK_FILE_PATH,output_path);
141
142     cout << "解包结束! ";
```

```
143     return 0;  
144 }
```

六、完成情况

- 是否完成以下步骤？(✓完成 ✕未做)

1. [✓] 2. [✓]

- 是否与同学进行了互测？ [✓]

互评同学学号姓名：17341059黄杨峻

七、实验体会

虽然该实验比较简单，但还是遇到了比较多的问题。

因为平常经常写C++程序，C的文件流操作很多都已经忘了，因而**第一个实验**对于**字符串的操作处理**并不得心应手。查了很多C的字符串函数，才顺利完成任务。特别要注意判断空行和判断文件末(EOF)的方法。

同时如果要输出中文字符，**cpp**文件注意要保存成GBK编码，否则保存为UTF-8都无法正常输出中文。

对于**第二个实验**，最大的难点则是判断每一个文件到**哪里终止**，因为在FileSet.pak中，所有文件都存储在一起，又没有行号标识就很难分开。

因而要通过文件大小来判断。一开始的方法是每写入一行就获取一次文件大小，但这样子的速度非常非常慢，解码5M的文件竟然用了五分多钟。后来采用C的buffer方法（见程序111行），解码几十M的文件都能在1秒内完成。

还有要注意文件的命名，需要判重，如果出现重名文件，加序号区分。这一部分不复杂，但也消耗了几十行的代码（见程序91~107行）。

同样，对于C++二进制文件ios::binary的读写操作也查了很多资料才明白其中原理。

最后一点则是要考虑各种极端情况，增强程序的鲁棒性，这样才能适应各种问题。比如在互测过程中，同学保存的文件名还包括了文件路径，而我的原始程序中并没有处理这一点，因而需要进行预处理后（程序72~77行）才能正常运行。

总的来说，第一次计网实验自己实现了文件单机传输任务，重新熟悉了C语言的使用，学到了很多的东西。