中国科学技术大学

实验报告

程序设计 I

CashBox 上机实验报告

作者姓名： 何纪言

作者学号： PB16111447

学科专业：计算机科学与技术学院

导师姓名： 马建辉 老师

完成时间： 二〇一六年十一月

目 录

第 1 章　实验要求

1.1　编程语言

使用 C 语言实现相关功能[1]。

1.2　程序简介

1.2.1　程序名称

CashBox

1.2.2　程序功能

一款有查看、记录、删除、补登、加密的简单记账程序。

1.3　具体要求

1.3.1　查看

按照时间顺序显示每个记录的时间、金额、注释。

1.3.2　记录

输入一个金额和注释，将该记录添加到数据库（程序将自动为该记录添加当前的时间）。

1.3.3　删除

指定一条记录，将它从数据库中删除。

1.3.4　补登

输入一个时间，插入一条该时间点的记录。

[1]本次采用了 C99 标准

1.3.5　加密

对数据文件进行简单的加密。

第 2 章　实验环境

2.1　操作系统

Linux 4.4.0-47-generic Ubuntu x86_64 x86_64 GNU/Linux

2.2　编译环境

GCC: (Ubuntu 5.4.0-6ubuntu1 16.04.4) 5.4.0 20160609

2.3　编辑器

VIM - Vi IMproved 7.4 (2013 Aug 10, compiled Jun 16 2016 10:50:38)
Sublime Text Build 3126

第 3 章　实验设计

3.1　功能设计

3.1.1　数据内容

每一条记录起码应该包含：时间，金额，注释这三个信息。

3.1.2　时间

程序中需要大量用到时间这一物理量，所以时间的储存形式应该使得时间便于比较，排序，能方便的转换为用户友好的格式。

考虑到软件工程习惯用法，使用 `time_t` 类型记录一个时间戳[1]，使得时间便于比较。

同时使用 C 语言中 `<time.h>` 中的函数，可以方便的将时间戳转换为用户友好的显示形式。

3.1.3　数据操作

由于程序需要随机添加，随机删除的功能，选用链表作为数据结构比较方便。

另外由于我们需要的数据记录是有序的，在添加中应该做到能插入到链表的合适位置。

综上所述，选用单向链表作为数据结构较好。

3.1.4　文件读写

程序需要能够读取已经保存的数据，保存修改后的数据，还需要做到在首次打开时创建数据文件。

选用 `<stdio.h>` 中的相关函数，以二进制方式保存文件。

[1]Unix timestamp, 自公元 1970 年 1 月 1 日经过的秒数

3.1.5　加密

由于加密不是本次实验的重点，故选用一个简单的抑或加密算法，即选用一个 KEY，将各个数据与 KEY 抑或后储存。

解密过程和加密过程完全相同。

3.2　文件结构

3.2.1　文件列表

根据需要使用的功能，创建以下文件：

代码 3.1　文件列表

```
1 main.c - 文件主入口
2 main.h
3 cb-db.c - 数据库相关函数
4 cb-db.h
5 cb-time.c - 时间处理相关函数
6 cb-time.h
7 Makefile
```

第 4 章　源文件

代码 4.1　main.h

```
1  /*
2   * @FILENAME main.h
3   * @DESCRIPTION headers for main.c
4   * @AUTHOR  hejiyan
5   * @VERSION 0.1
6   */
7
8  #include <stdio.h>
9  #include <string.h>
10 #include <stdlib.h>
11
12 #include "cb-time.h"
13 #include "cb-db.h"
```

代码 4.2　main.c

```
1  /*
2   * @FILENAME main.c
3   * @DESCRIPTION the entry of cashbox
4   * @AUTHOR  hejiyan
5   * @VERSION 0.1
6   */
7  #include "main.h"
8
9  extern cb_db_dataset Dat;
10
11 void print_db(int k)
12 {
13    char time_str[MAX_TEXT_LEN];
14    int i = 0;
15    int total = 0;
16    if (k == 0) k = Dat.n;
17
18    puts("*****************************");
19
20    printf("* [id]   [time]     [money] [text]\n");
21    for (node* p = Dat.head; p ; p = p -> next)
22    {
23       if (i >= Dat.n - k)
24       {
25          time_to_str(p->v->time, time_str);
26          printf("* [%2d] %s %d %s\n", i, time_str, p->v->
               money, p->v->text);
27       }
28       i++;
29       total += p->v->money;
```

```
30      }
31
32      printf("* Items: %d, Total money: %d.\n", Dat.n, total)
            ;
33      puts("******************************");
34
35  }
36
37  void print_long_tips()
38  {
39      puts("[cashbox]Welcome!");
40      puts("[cashbox]Enter `p` to show the cashbox(latest 5
            items);");
41      puts("[cashbox]Enter `l` to show the cashbox(all items)
            ;");
42      puts("[cashbox]Enter `a` to add an item (with current
            time);");
43      puts("[cashbox]Enter `t` to add an item (with custom
            time);");
44      puts("[cashbox]Enter `d` to delete an item;");
45      puts("[cashbox]Enter `h` to view help text;");
46      puts("[cashbox]Enter `q` to quit.");
47  }
48
49  void print_tips()
50  {
51      puts("[cashbox]Enter command [p|l|a|t|d|h|q]:");
52  }
53
54  int main()
55  {
56      cb_db_init();
57      char c;
58      int money, id;
59      time_t t;
60      char text[MAX_TEXT_LEN];
61      print_long_tips();
62      while((c = getchar()) != 'q')
63      {
64          switch(c)
65          {
66              case 'p':
67                  print_db(5);
68                  print_tips();
69                  break;
70              case 'l':
71                  print_db(0);
72                  print_tips();
73                  break;
74              case 'a':
75                  printf("[cashbox.add]Enter money:");
76                  scanf("%d", &money);
77                  printf("[cashbox.add]Enter text:");
78                  scanf("%s", text);
79
```

```
80              add_item(get_cur_time(), money, text);
81              print_db(5);
82              print_tips();
83              break;
84          case 't':
85              printf("[cashbox.add]Enter time(eg.
                    1998/04/11-12:34:56) :");
86              scanf("%s", text);
87              t = str_to_time(text);
88
89              printf("[cashbox.add]Enter money:");
90              scanf("%d", &money);
91              printf("[cashbox.add]Enter text:");
92              scanf("%s", text);
93
94              add_item(t, money, text);
95              print_db(5);
96              print_tips();
97              break;
98          case 'd':
99              printf("[cashbox.del]Enter the id :");
100             scanf("%d", &id);
101
102             del_item(id);
103             print_db(5);
104             print_tips();
105             break;
106         case 'h':
107             print_long_tips();
108             break;
109       }
110   }
111   return 0;
112 }
```

代码 4.3　cb-time.h

```
1  /*
2   * @FILENAME cb-time.h
3   * @DESCRIPTION headers for cb-time.c
4   * @AUTHOR  hejiyan
5   * @VERSION 0.1
6   */
7  #include <stdio.h>
8  #include <time.h>
9  #ifndef _CB_TIME_
10 #define _CB_TIME_
11
12 void time_to_str(time_t t, char* s);
13
14 time_t str_to_time(const char* s);
15
16 time_t get_cur_time();
17
18 #endif
```

代码 4.4　cb-time.c

```
1  /*
2   * @FILENAME cb-time.c
3   * @DESCRIPTION functions about time
4   * @AUTHOR  hejiyan
5   * @VERSION 0.1
6   */
7  #include <time.h>
8  #include <string.h>
9  #include "cb-time.h"
10
11 /* return current time */
12 time_t get_cur_time()
13 {
14     return time(NULL);
15 }
16
17 /* convert time_t to string */
18 void time_to_str(time_t t, char* s)
19 {
20     struct tm* now_time;
21     now_time = localtime(&t);
22     sprintf(
23         s,
24         "%04d/%02d/%02d-%02d:%02d:%02d",
25         1900 + now_time->tm_year,
26         now_time->tm_mon,
27         now_time->tm_mday,
28         now_time->tm_hour,
29         now_time->tm_min,
30         now_time->tm_sec
31     );
32 }
33
34 /* convert string to time_t(uint) */
35 time_t str_to_time(const char* s)
36 {
37     /* template: 1998/04/11-12:34:56 */
38     struct tm tmp, *now_time = &tmp;
39     memset(&tmp, 0, sizeof(tmp));
40     sscanf(
41         s,
42         "%d/%d/%d-%d:%d:%d",
43         &now_time->tm_year,
44         &now_time->tm_mon,
45         &now_time->tm_mday,
46         &now_time->tm_hour,
47         &now_time->tm_min,
48         &now_time->tm_sec
49     );
50     now_time->tm_year -= 1900;
51     return mktime(now_time);
52 }
```

代码 4.5　cb-db.h

```
1  /*
2   * @FILENAME cb-db.h
3   * @DESCRIPTION headers for cb-db.c
4   * @AUTHOR  hejiyan
5   * @VERSION 0.1
6   */
7
8  #ifndef _CB_DB_
9  #define _CB_DB_
10
11 #define MAX_TEXT_LEN 100 /* max len of text in cb_db_item
       */
12 #define DATE_FILENAME "db.dat" /* filename of db file */
13
14 /* WARNING: OLD DATA CANT BE DECRYPTED BY NEW KEY*/
15
16 #define CB_DB_KEY 2333333 /* the key for encrypt & decrypt
       */
17
18 typedef struct cb_db_item{
19    time_t time;
20    int money;
21    char text[MAX_TEXT_LEN];
22 } cb_db_item;
23
24
25 typedef struct node{
26    cb_db_item* v;
27    struct node* next;
28 } node;
29
30 typedef struct cb_db_dataset{
31    int n;
32    node* head;
33 } cb_db_dataset;
34
35 int _cb_db_read_number(FILE* fl);
36 void _cb_db_encrypt(cb_db_item* x);
37 void _cb_db_decrypt(cb_db_item* x);
38 void _cb_db_save_number(int n, FILE* fl);
39 void _cb_db_read_items(cb_db_item* items, int n, FILE* fl)
       ;
40 void _cb_db_save_items(cb_db_item* items, int n, FILE* fl)
       ;
41
42
43 void cb_db_init();
44 void cb_db_read_db();
45 void cb_db_save_db();
46 void cb_db_add_item(const cb_db_item* x);
47 void cb_db_del_item(const int k);
48
49 void add_item(time_t t, int money, const char* s);
50 void del_item(const int k);
```

```
51 void init_db();
52 void save_db();
53 #endif
```

代码 4.6　cb-db.c

```
1  /*
2   * @FILENAME cb-db.c
3   * @DESCRIPTION functions about database
4   * @AUTHOR  hejiyan
5   * @VERSION 0.1
6   */
7
8  #include <stdio.h>
9  #include <string.h>
10 #include <stdlib.h>
11 #include "cb-db.h"
12
13 cb_db_dataset Dat;
14
15
16 /* read 1 int from fl */
17 int _cb_db_read_number(FILE* fl)
18 {
19    int n;
20    if (1 == fread(&n, sizeof(int), 1, fl))
21       return n;
22    return 0;
23 }
24
25 /* encrypt cb_db_item */
26 void _cb_db_encrypt(cb_db_item* x)
27 {
28    x->time ^= CB_DB_KEY;
29    x->money ^= CB_DB_KEY;
30    int len = strlen(x->text);
31    for (int i = 0; i < len; i++)
32    {
33       x->text[i] ^= CB_DB_KEY;
34    }
35 }
36
37 /* decrypt cb_db_item */
38 void _cb_db_decrypt(cb_db_item* x)
39 {
40    _cb_db_encrypt(x);
41 }
42
43 /* write 1 int into fl */
44 void _cb_db_save_number(int n, FILE* fl)
45 {
46    fwrite(&n, sizeof(int), 1, fl);
47 }
48
49 /* read n cb_db_item from fl */
```

```
50  void _cb_db_read_items(cb_db_item* items, int n, FILE* fl)
51  {
52     fread(items, sizeof(cb_db_item), n, fl);
53     for (cb_db_item* p = items; p < items + n; p++)
54        _cb_db_decrypt(p);
55  }
56
57  /* write n cb_db_item into fl */
58  void _cb_db_save_items(cb_db_item* items, int n, FILE* fl)
59  {
60     for (cb_db_item* p = items; p < items + n; p++)
61        _cb_db_encrypt(p);
62
63     fwrite(items, sizeof(cb_db_item), n, fl);
64
65     for (cb_db_item* p = items; p < items + n; p++)
66        _cb_db_decrypt(p);
67  }
68
69  /* call this first if you want use db */
70  void cb_db_init()
71  {
72     cb_db_read_db();
73  }
74
75  /* read data file into Dat */
76  void cb_db_save_db()
77  {
78     FILE* fl = fopen(DATE_FILENAME, "wb");
79     if (!fl) printf("Err: can't open " DATE_FILENAME), exit
          (0);
80     _cb_db_save_number(Dat.n, fl);
81     for (node* p = Dat.head; p; p = p -> next)
82     {
83        _cb_db_save_items(p->v, 1, fl);
84     }
85     fclose(fl);
86  }
87
88  /* save Dat into data file */
89  void cb_db_read_db()
90  {
91     memset(&Dat, 0, sizeof(cb_db_dataset));
92     FILE* fl = fopen(DATE_FILENAME, "rb");
93     if (!fl) cb_db_save_db(), fl = fopen(DATE_FILENAME, "rb
          ");
94     if (!fl) printf("Err: can't open " DATE_FILENAME), exit
          (0);
95
96     int n = _cb_db_read_number(fl);
97
98     for (int i = 0; i < n; i++)
99     {
100       cb_db_item* t = (cb_db_item*)malloc(sizeof(
             cb_db_item));
```

```
101         _cb_db_read_items(t, 1, fl);
102         cb_db_add_item(t);
103     }
104     fclose(fl);
105 }
106 
107 /* add item at the tail of the linklist
108 void cb_db_add_item_TAIL(const cb_db_item* x)
109 {
110     if (NULL == x) return;
111     Dat.n++;
112     node* t = (node*)malloc(sizeof(node));
113     t->v = (cb_db_item*)x; t->next = NULL;
114     if (NULL == Dat.head){
115         Dat.head = t;
116     } else {
117         node* p = Dat.head;
118         while(p->next) p = p->next;
119         p->next = t;
120     }
121 }
122 */
123 
124 /* add item in the linklist (ACS by `time`, assume the
        linklist is already ACS) */
125 void cb_db_add_item(const cb_db_item* x)
126 {
127     if (NULL == x) return;
128     Dat.n++;
129     node* t = (node*)malloc(sizeof(node));
130     t->v = (cb_db_item*)x; t->next= NULL;
131     if (NULL == Dat.head)
132     {
133         Dat.head = t;
134     } else
135     {
136         node *pre, *cur;
137         pre = cur = Dat.head;
138         while(cur && cur->v->time < x->time){
139             pre = cur;
140             cur = cur->next;
141         }
142         if (cur == Dat.head)
143         {
144             t->next = cur;
145             Dat.head = t;
146         }
147         else
148         {
149             t->next = cur;
150             pre->next = t;
151         }
152     }
153 }
154 
```

```
155 /* del k-th item in the linklist (k = 0,1,...) */
156 void cb_db_del_item(const int k)
157 {
158    int i = 0;
159    node *pre, *cur;
160    pre = cur = Dat.head;
161    while (cur)
162    {
163       if (i++ == k) break;
164       pre = cur;
165       cur = cur->next;
166    }
167    if (NULL == cur) return;
168    if (cur == Dat.head) Dat.head = cur->next;
169    pre->next = cur->next;
170    free(cur->v);
171    free(cur);
172    Dat.n--;
173 }
174
175 /* add item then save database */
176 void add_item(time_t t, int money, const char* s)
177 {
178    cb_db_item* x = (cb_db_item*)malloc(sizeof(cb_db_item))
          ;
179    x->time = t; x->money = money;
180    strcpy(x->text, s);
181    cb_db_add_item(x);
182    cb_db_save_db();
183 }
184
185 /* del item then save database */
186 void del_item(const int k)
187 {
188    cb_db_del_item(k);
189    cb_db_save_db();
190 }
191
192 /* init database */
193 void init_db()
194 {
195    cb_db_init();
196 }
197
198 /* init database */
199 void save_db()
200 {
201    cb_db_save_db();
202 }
```

第 5 章　实验问题

5.1　文件操作

实验过程中，发现有时关闭程序后，部分数据未能保存，代码如下：

```
1  void cb_db_save_db()
2  {
3      FILE* fl = fopen(DATE_FILENAME, "wb");
4      if (!fl) printf("Err: can't open " DATE_FILENAME), exit
           (0);
5      _cb_db_save_number(Dat.n, fl);
6      for (node* p = Dat.head; p; p = p -> next)
7      {
8          _cb_db_save_items(p->v, 1, fl);
9      }
10 }
```

经过检查，发现缺失了 `fclose` 操作，改为以下代码：

```
1  void cb_db_save_db()
2  {
3      FILE* fl = fopen(DATE_FILENAME, "wb");
4      if (!fl) printf("Err: can't open " DATE_FILENAME), exit
           (0);
5      _cb_db_save_number(Dat.n, fl);
6      for (node* p = Dat.head; p; p = p -> next)
7      {
8          _cb_db_save_items(p->v, 1, fl);
9      }
10     fclose(fl);
11 }
```

5.2　时间库函数

C 语言 `time.h` 中提供了一个时间的结构体 `struct tm`，可以用相关函数将时间戳转换为该结构体，获得年月日等信息。

在使用中，以下代码表现出了一定的随机性错误（与预期结果相差一个小时整）：

```
1  time_t str_to_time(const char* s)
2  {
3      /* template: 1998/04/11-12:34:56 */
4      struct tm tmp, *now_time = &tmp;
5      sscanf(
```

```
        s,
        "%d/%d/%d-%d:%d:%d",
        &now_time->tm_year,
        &now_time->tm_mon,
        &now_time->tm_mday,
        &now_time->tm_hour,
        &now_time->tm_min,
        &now_time->tm_sec
    );
    now_time->tm_year -= 1900;
    return mktime(now_time);
}
```

经过检查，发现 `struct tm` 中有一标志变量代表是否采用夏令时[1]，由于 `tmp` 变量未初始化，使得该标志变量为随机值。

修改后如下：

```
time_t str_to_time(const char* s)
{
    /* template: 1998/04/11-12:34:56 */
    struct tm tmp, *now_time = &tmp;
    memset(&tmp, 0, sizeof(tmp));
    sscanf(
        s,
        "%d/%d/%d-%d:%d:%d",
        &now_time->tm_year,
        &now_time->tm_mon,
        &now_time->tm_mday,
        &now_time->tm_hour,
        &now_time->tm_min,
        &now_time->tm_sec
    );
    now_time->tm_year -= 1900;
    return mktime(now_time);
}
```

[1]夏时制，另译夏令时间（英语：Summer time），又称日光节约时制、日光节约时间（英语：Daylight saving time），是一种为节约能源而人为规定地方时间的制度，在这一制度实行期间所采用的统一时间称为“夏令时间”。一般在天亮较早的夏季人为将时间调快一小时，可以使人早起早睡，减少照明量，以充分利用光照资源，从而节约照明用电。各个采纳夏时制的国家具体规定不同。（来源：wikipedia）

第 6 章　实验总结

6.1　实验总结

通过本次实验，我对 C 语言的控制结构，语法特性更加熟悉了，也加强了对函数，指针，结构体等地运用，学习了文件操作的各种方法，

同时，还熟悉了一下库的库函数用法：

```
<stdio.h>
<stdlib.h
<string.h>
<time.h>
<limit.h>
<ctype.h>
```

本次实验采用链表作为数据结构，提高了我对链表使用的熟练度。

本次实验使用多个文件共同编译，使我了解了软件工程中多文件编译的方法，了解了包含守卫等工程方法。

同时本次实验还提高了我使用 Linux，Vim 等的熟练度。