

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 |
| 1.1 编程语言 | 1 |
| 1.2 程序简介 | 1 |
| 1.2.1 程序名称..... | 1 |
| 1.2.2 程序功能..... | 1 |
| 1.3 具体要求 | 1 |
| 1.3.1 查看 | 1 |
| 1.3.2 记录 | 1 |
| 1.3.3 删除 | 1 |
| 1.3.4 补登 | 1 |
| 1.3.5 加密 | 2 |
| 第 2 章 实验环境..... | 3 |
| 2.1 操作系统 | 3 |
| 2.2 编译环境 | 3 |
| 2.3 编辑器 | 3 |
| 第 3 章 实验设计..... | 5 |
| 3.2 功能设计 | 5 |
| 3.2.1 数据内容..... | 5 |
| 3.2.2 时间 | 5 |
| 3.2.3 数据操作..... | 5 |
| 3.2.4 文件读写..... | 5 |
| 3.2.5 加密 | 6 |
| 3.3 文件结构 | 6 |
| 3.3.1 文件列表..... | 6 |
| 第 4 章 源文件 | 7 |

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

3.2 功能设计

3.2.1 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。

3.2.2 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。

3.2.3 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。

3.2.4 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。

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

3.2.5 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。

3.3 文件结构

3.3.1 文件列表

根据需要使用功能，创建以下文件：

代码 3.1 文件列表

| |
|--|
| <pre>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</pre> |
|--|

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->
27                     money, p->v->text);
28             i++;
29             total += p->v->money;
```

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

```

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 ;
41 void _cb_db_save_items(cb_db_item* items, int n, FILE* fl)
42 ;
43
44 void cb_db_init();
45 void cb_db_read_db();
46 void cb_db_save_db();
47 void cb_db_add_item(const cb_db_item* x);
48 void cb_db_del_item(const int k);
49
50 void add_item(time_t t, int money, const char* s);
51 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
80         (0);
81     _cb_db_save_number(Dat.n, fl);
82     for (node* p = Dat.head; p; p = p -> next)
83     {
84         _cb_db_save_items(p->v, 1, fl);
85     }
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         ");
95     if (!fl) printf("Err: can't open " DATE_FILENAME), exit
96         (0);
97
98     int n = _cb_db_read_number(fl);
99
100     for (int i = 0; i < n; i++)
101     {
102         cb_db_item* t = (cb_db_item*)malloc(sizeof(
103             cb_db_item));
104         _cb_db_read_items(t, 1, fl);
105     }
106 }
```

```
101     cb_db_add_item(t);
102 }
103 }
104
105 /* add item at the tail of the linklist
106 void cb_db_add_item_TAIL(const cb_db_item* x)
107 {
108     if (NULL == x) return;
109     Dat.n++;
110     node* t = (node*)malloc(sizeof(node));
111     t->v = (cb_db_item*)x; t->next = NULL;
112     if (NULL == Dat.head){
113         Dat.head = t;
114     } else {
115         node* p = Dat.head;
116         while(p->next) p = p->next;
117         p->next = t;
118     }
119 }
120 */
121
122 /* add item in the linklist (ACS by `time`, assume the
123    linklist is already ACS) */
124 void cb_db_add_item(const cb_db_item* x)
125 {
126     if (NULL == x) return;
127     Dat.n++;
128     node* t = (node*)malloc(sizeof(node));
129     t->v = (cb_db_item*)x; t->next = NULL;
130     if (NULL == Dat.head)
131     {
132         Dat.head = t;
133     } else
134     {
135         node *pre, *cur;
136         pre = cur = Dat.head;
137         while(cur && cur->v->time < x->time){
138             pre = cur;
139             cur = cur->next;
140         }
141         if (cur == Dat.head)
142         {
143             t->next = cur;
144             Dat.head = t;
145         }
146         else
147         {
148             t->next = cur;
149             pre->next = t;
150         }
151     }
152 }
153
154 /* del k-th item in the linklist (k = 0,1,...) */
155 void cb_db_del_item(const int k)
```

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