Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «Операционные системы»
Работа с динамическими библиотеками

Студент: А. П. Шорохов
Преподаватель: А. А. Соколов
Группа: М8О-206Б
Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №5

Задача: Требуется создать динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами:
1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов

В конечном итоге, программа должна состоять из следующих частей:
· Динамическая библиотека, реализующая заданных вариантом интерфейс;
· Тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции;
· Тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс.

Провести анализ между обоими типами использования библиотеки.

Вариант:
1. Работа со списком
1. Целочисленный 32-битный

1 Описание

1. Создадим файлы динамической библиотеки.
2. Соберём библиотеку и получим файл с расширением .so.
3. Напишем 2 программы, одна из которых использует системные вызовы для открытия библиотеки, а другая подключает библиотеку на этапе линковки.
4. Соберём эти программы и проверим корректность их работы.

Сборка и подключение выполняются *make*.
Будем использовать следующие системные вызовы:
1. *dlopen* - загружает динамическую библиотеку и возвращает handle.
2. *dlclose* - уменьшает счётчик ссылок на динамический объект в памяти, если он становится равным нулю, то объект выгружается из памяти.
3. *dlsym* - позволяет получить указатель на функцию, находящуюся в динамической библиотеке.
4. *dlerror* - возвращает читабельную строку, описывающую последнюю возникшую ошибку, возникшую при взаимодействии с динамической библиотекой.

2 Исходный код

list32.h

```
#ifndef D_LIST_H
#define D_LIST_H 1

#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdint.h>

typedef struct _listItem {

  int32_t value;

  //listItem() = default;
  //listItem(int32_t);

  struct _listItem * next;

} listItem;

typedef struct _List {

  listItem * head;
  listItem * tail;

  int size;


  //List();


} List;

extern void listItemInit(listItem * li, int32_t data);

extern void listInit(List * l);
extern void push(List * l, int32_t data);
extern void print(List * l);
extern void destroy(List * l);

extern void erase(List * l, listItem * it);
extern void placeAfter(List * l, listItem * it, int32_t data);

#endif
```

listint32.c

```c
#include "list32.h"

void listItemInit(listItem * li, int32_t data) {

  li -> value = data;
  li -> next = NULL;

}

void listInit(List * l) {

  l -> size = 0;
  l -> head = NULL;
  l -> tail = NULL;

}

void push(List * l, int32_t data) {

  l -> size++;

  listItem * tmp = malloc(sizeof(listItem));
  listItemInit(tmp, data);

  if (l -> head == NULL) {

    l -> head = tmp;
    l -> tail = tmp;

  } else {

    l -> tail -> next = tmp;
    l -> tail = tmp;

  }

}

void print(List * l) {

  for (listItem * it = l -> head; it != NULL; it = it -> next) {

    printf("%d", it -> value);
    printf(" ");

  }

  printf("\n");

```

```
50 }
51
52 void destroy(List * l) {
53
54   if (l -> size == 0) { // * -> * -> * -> * -> * -> 0
55
56     return;
57
58   } else {
59
60     listItem * tmp = l -> head;
61
62     while (tmp != NULL) {
63
64       listItem * buf = tmp;
65       tmp = tmp -> next;
66       free(buf);
67
68     }
69
70     l -> head = NULL;
71     l -> tail = NULL;
72     l -> size = 0;
73
74
75   }
76
77 }
78
79 void erase(List * l, listItem * it) {
80
81   if (it == l -> head) {
82     l -> head = l -> head -> next;
83     free(it);
84     return;
85   }
86
87   listItem * it__ = l -> head;
88
89   while (it__ -> next != it)
90   it__ = it__ -> next;
91
92   it__ -> next = it -> next;
93
94   free(it);
95
96   l -> size--;
97
98 }
```

```
99
100 void placeAfter(List * l, listItem * it, int32_t data) {
101
102   if (l -> size == 0)
103   return;
104
105   listItem * it__ = l -> head;
106
107   while (it__ != it) {
108     it__ = it__ -> next;
109   }
110
111   listItem * tmp = malloc(sizeof(listItem));
112   listItemInit(tmp, data);
113
114   tmp -> next = it__ -> next;
115   it__ -> next = tmp;
116
117   l -> size++;
118
119 }
```

SourceDY.c

```
1  #include "list32.h"
2  #include <string.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <dlfcn.h>
6
7  void menu() {
8    printf("_____\n");
9    printf("1 : push\n");
10   printf("2 : print\n");
11   printf("3 : destroy\n");
12   printf("4 : erase with iterator\n");
13   printf("5 : place with iterator\n");
14   printf("6 : quit\n");
15   printf("> ");
16 }
17
18
19 int main() {
20
21   void (*listItemInit)(listItem * li, int32_t data);
22
23   void (*listInit)(List * l);
24   void (*push)(List * l, int32_t data);
25   void (*print)(List * l);
26   void (*destroy)(List * l);
```

```
27
28  void (*erase)(List * l, listItem * it);
29  void (*placeAfter)(List * l, listItem * it, int32_t data);
30
31  void* libHandle;
32  libHandle = dlopen("./liblist32dyn.so", RTLD_LAZY);
33  if (!libHandle) {
34    fprintf(stderr, "%s\n", dlerror());
35    exit(-1);
36  }
37
38
39  listInit = dlsym(libHandle, "listInit");
40  listItemInit = dlsym(libHandle, "listItemInit");
41  push = dlsym(libHandle, "push");
42  print = dlsym(libHandle, "print");
43  destroy = dlsym(libHandle, "destroy");
44  erase = dlsym(libHandle, "erase");
45  placeAfter = dlsym(libHandle, "placeAfter");
46
47
48  List list;
49  (*listInit)(&list);
50  int cmd;
51
52  while (true) {
53    menu();
54    scanf("%d", &cmd);
55
56    if (cmd == 1) {
57      int32_t tmp;
58
59      printf("Enter int32_t value > ");
60
61      scanf("%d", &tmp);
62
63      (*push)(&list, tmp);
64
65
66    } else if (cmd == 2) {
67
68      (*print)(&list);
69
70    } else if (cmd == 3) {
71
72      (*destroy)(&list);
73
74    } else if (cmd == 4) {
75
```

```
76        int position;
77
78        printf("Enter position > ");
79        scanf("%d", &position);
80
81        if (position >= 0 && position < list.size) {
82          listItem * it = list.head;
83
84          for (int i = 0; i < position; i++)
85          it = it -> next;
86
87          (*erase)(&list, it);
88        } else {
89          printf("ERROR : wrong position\n");
90        }
91
92      } else if (cmd == 5) {
93
94        int position;
95
96        printf("Enter position > ");
97        scanf("%d", &position);
98
99        if (position >= 0 && position < list.size) {
100         int32_t tmp;
101         printf("Enter int32_t value > ");
102         scanf("%d", &tmp);
103
104         listItem * it = list.head;
105
106         for (int i = 0; i < position; i++)
107         it = it -> next;
108
109         (*placeAfter)(&list, it, tmp);
110       } else {
111         printf("ERROR : wrong position\n");
112       }
113
114     } else if (cmd == 6) {
115       dlclose(libHandle);
116       return 0;
117
118     } else {
119
120       printf("ERROR : wrong command\n");
121
122     }
123
124   }
```

```
125
126
127 }
```

SourceST.c

```
1  #include "list32.h"
2  #include <string.h>
3  #include <stdio.h>
4
5  /*
6  extern void listItemInit(listItem * li, int32_t data);
7
8  extern void listInit(List * l);
9  extern void push(List * l, int32_t data);
10 extern void print(List * l);
11 extern void destroy(List * l);
12
13 extern void erase(List * l, listItem * it);
14 extern void placeAfter(List * l, listItem * it, int32_t data);
15
16 */
17
18 void menu() {
19   printf("_____________________________________________________\n");
20   printf("1 : push\n");
21   printf("2 : print\n");
22   printf("3 : destroy\n");
23   printf("4 : erase with iterator\n");
24   printf("5 : place with iterator\n");
25   printf("6 : quit\n");
26   printf("> ");
27 }
28
29 int main() {
30
31   List list;
32   listInit(&list);
33   int cmd;
34
35   while (true) {
36     menu();
37     scanf("%d", &cmd);
38
39     if (cmd == 1) {
40       int32_t tmp;
41
42       printf("Enter int32_t value > ");
43
44       scanf("%d", &tmp);
```

```
45
46        push(&list, tmp);
47
48
49      } else if (cmd == 2) {
50
51        print(&list);
52
53      } else if (cmd == 3) {
54
55        destroy(&list);
56
57      } else if (cmd == 4) {
58
59        int position;
60
61        printf("Enter position > ");
62        scanf("%d", &position);
63
64        if (position >= 0 && position < list.size) {
65          listItem * it = list.head;
66
67          for (int i = 0; i < position; i++)
68          it = it -> next;
69
70          erase(&list, it);
71        } else {
72          printf("ERROR : wrong position\n");
73        }
74
75      } else if (cmd == 5) {
76
77        int position;
78
79        printf("Enter position > ");
80        scanf("%d", &position);
81
82        if (position >= 0 && position < list.size) {
83          int32_t tmp;
84          printf("Enter int32_t value > ");
85          scanf("%d", &tmp);
86
87          listItem * it = list.head;
88
89          for (int i = 0; i < position; i++)
90          it = it -> next;
91
92          placeAfter(&list, it, tmp);
93        } else {
```

```
94          printf("ERROR : wrong position\n");
95        }
96
97      } else if (cmd == 6) {
98
99        return 0;
100
101     } else {
102
103       printf("ERROR : wrong command\n");
104
105     }
106
107   }
108
109 }
```

3 Консоль

```
dobriy_alien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/lab5/Clib$
./rundy
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>1
Enter int32_t value >12
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>1
Enter int32_t value >13
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>2
12 13
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>6
dobriy_alien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/lab5/Clib$
./runst
```

```
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>1
Enter int32_t value >13
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>1
Enter int32_t value >14
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>2
13 14
-----------------------------------------------------
1 : push
2 : print
3 : destroy
4 : erase with iterator
5 : place with iterator
6 : quit
>6
```

4 Выводы

В результате выполнения данной лабораторной работы я научился создавать статические и динамические библиотеки, ознакомился с тем, как они размещаются в памяти, и принципами их применения на примере работы с линейными структурами данных.