****Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5**

**По курсу:  
«Операционные системы»**

|  |  |
| --- | --- |
| Студент: | Захаров И.С. |
| Группа: | М8О-208Б-17 |
| Преподаватель: | Миронов Е. С. |
|  |  |
| Оценка: |  |
| Дата: |  |

**Москва, 2018 г.**

## Цель работы

Целью является приобретение практических навыков в:

* Создание динамических библиотек
* Создание программ, которые используют функции динамических библиотек

## Задание

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.

## Структура данных, с которой должна обеспечивать работу библиотека

1. Работа со списком

## Тип данных, используемый структурой

1. Целочисленный 32-битный

## Справочный материал

1. <https://msdn.microsoft.com/en-us/library/ms235636.aspx>
2. <https://msdn.microsoft.com/en-us/library/windows/desktop/ms684175(v=vs.85).aspx>
3. <https://msdn.microsoft.com/en-us/library/windows/desktop/ms683212(v=vs.85).aspx>
4. <http://www.ibm.com/developerworks/library/l-dynamic-libraries/>

## Исходный код

**list.c**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include "list.h"

list\* list\_create(){

list\* l = (list\*)malloc(sizeof(list));

if(l == NULL) return l;

l->first=NULL;

l->last = NULL;

l->size = 0;

return l;

}

size\_t size(list\* l){

return l->size;

}

void list\_destroy(list\* l){

list\_el\* cur = l->first;

while (cur != NULL){

list\_el\* n = cur->next;

free(cur);

cur = n;

}

free(l);

}

iter list\_begin\_iter(list\* l){

iter it;

it.l = l;

it.cur = NULL;

it.prev = NULL;

return it;

}

bool it\_has\_next(iter\* it){

if(it->cur != NULL) return (it->cur->next != NULL);

return (it->l->first != NULL);

}

void it\_move\_next(iter\* it){

if ( it->cur != NULL){

it->prev = it->cur;

it->cur = it->cur->next;

}else{

it->cur = it->l->first;

}

}

int it\_get\_val(iter\* it){

return it->cur->val;

}

void it\_set\_val(iter\* it, int v){

it->cur->val = v;

}

void it\_delete(iter\* it){

if (it->cur == NULL) return;

if (it->prev == NULL){

it->l->first = it->cur->next;

it->prev = it->cur;

it->cur = it->cur->next;

free(it->prev);

it->prev = NULL;

it->l->size -= 1;

return;

}

it->prev->next = it->cur->next;

free(it->cur);

it->cur = it->prev->next;

it->l->size -= 1;

}

bool it\_insert(iter\* it, int v){

list\_el\* el = (list\_el\*)malloc(sizeof(list\_el));

if(el == NULL){

return false;

}

el->val = v;

if(it->cur == NULL){

el->next = it->l->first;

it->l->first = el;

if(it->l->last == NULL){

it->l->last = el;

}

it->l->size++;

}else{

el->next = it->cur->next;

it->cur->next = el;

if (el->next == NULL){

it->l->last=el;

}

it->l->size++;

}

it\_move\_next(it);

return true;

}

bool insert(list\* l, int place, int v){

if (l == NULL) return false;

if (place > l->size + 1) return false;

iter it = list\_begin\_iter(l);

for(int i = 1; i < place; ++i){

it\_move\_next(&it);

}

if (it\_insert(&it,v)) return true;

return false;

}

iter get\_iter\_place(list\* l, int place){

iter it = list\_begin\_iter(l);

for(int i = 1; i <= place; ++i){

it\_move\_next(&it);

}

return it;

}

iter get\_iter(list\* l, int v){

iter it = list\_begin\_iter(l);

for (int i = 0; i < l->size; ++i){

it\_move\_next(&it);

if (v == it\_get\_val(&it)) return it;;

}

(&it)->cur = NULL;

return it;

}

void print(list\* l){

if(l == NULL || l->size == 0){

return;

}

iter it = list\_begin\_iter(l);

if (it\_has\_next(&it)) it\_move\_next(&it);

while (it\_has\_next(&it)){

printf("%d ", it\_get\_val(&it));

it\_move\_next(&it);

}

printf("%d\n", it\_get\_val(&it));

}

**list.h**

#ifndef \_LIST\_H\_

#define \_LIST\_H\_

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

typedef struct list\_el list\_el;

struct list\_el{

int val;

list\_el\* next;

};

typedef struct {

list\_el\* first;

list\_el\* last;

size\_t size;

} list;

typedef struct {

list\* l;

list\_el\* prev;

list\_el\* cur;

} iter;

list\* list\_create();

size\_t size(list\* l);

void list\_destroy(list\* l);

iter list\_begin\_iter(list\* l);

bool it\_has\_next(iter\* it);

void it\_move\_next(iter\* it);

int it\_get\_val(iter\* it);

void it\_set\_val(iter\* it, int v);

void it\_delete(iter\* it);

bool it\_insert(iter\* it, int v);

bool insert(list\* l, int place, int v);

iter get\_iter\_place(list\* l, int place);

iter get\_iter(list\* l, int v);

void print(list\* l);

#endif

**main.Dyn.c**

#include <stdio.h>

#include <stdlib.h>

#include "list.h"

#include <dlfcn.h>

#define FAILURE 1

int main() {

list \*(\*n\_list\_create)();

bool (\*list\_insert)(list\* l, int place, int v);

iter (\*list\_get\_iter\_place)(list\* l, int place);

void (\*list\_print)(list\* l);

void (\*n\_list\_destroy)(list\* l);

void (\*list\_it\_delete)(iter\* it);

void \*libHandle;

libHandle = dlopen("./liblist.so", RTLD\_LAZY);

if (!libHandle) {

fprintf(stderr, "%s\n", dlerror());

exit(FAILURE);

}

n\_list\_create = dlsym(libHandle, "list\_create");

list\_insert= dlsym(libHandle,"insert");

list\_get\_iter\_place=dlsym(libHandle, "get\_iter\_place");

list\_print=dlsym(libHandle,"print");

n\_list\_destroy=dlsym(libHandle,"list\_destroy");

list\_it\_delete=dlsym(libHandle,"it\_delete");

list\* l;

l = n\_list\_create();

int i = 1;

int n;

do {

printf("\n выберите действие\n");

printf(" 1- вставить элемент в список\n");

printf(" 2- удалить по индексу\n");

printf(" 3- распечатать список\n");

printf(" 4- удалить список\n");

printf(" 0- завершение программы\n");

scanf("%d",&n);

switch (n) {

case 1: {

int val;

printf("введите значение: ");

scanf("%d",&val);

list\_insert(l, i, val);

i++;

break;

}

case 2: {

int place;

printf("введите индекс: ");

scanf("%d",&place);

iter it;

it = list\_get\_iter\_place(l, place);

list\_it\_delete(&it);

break;

}

case 3: {

list\_print(l);

break;

}

case 4: {

n\_list\_destroy(l);

break;

}

}

} while(n != 0);

return 0;

}

**mainStat.c**

#include <stdio.h>

#include <stdlib.h>

#include "list.h"

int main() {

list\* l;

l = list\_create();

int i = 1;

int n;

do {

printf("\n выберите действие\n");

printf(" 1- вставить элемент в список\n");

printf(" 2- удалить по индексу\n");

printf(" 3- распечатать список\n");

printf(" 4- удалить список\n");

printf(" 0- завершение программы\n");

scanf("%d",&n);

switch (n) {

case 1: {

int val;

printf("введите значение: ");

scanf("%d",&val);

insert(l, i, val);

i++;

break;

}

case 2: {

int place;

printf("введите индекс: ");

scanf("%d",&place);

iter it;

it = get\_iter\_place(l, place);

it\_delete(&it);

break;

}

case 3: {

print(l);

break;

}

case 4: {

list\_destroy(l);

break;

}

}

} while(n != 0);

return 0;

}

**Makefile**

CC = gcc

FLAGS = -std=c99 -pthread -w -pipe -O2 -Wextra -Werror -Wall -Wno-sign-compare -pedantic -lm

all: run

run: liblist.so mainStat.o mainDyn.o

$(CC) $(FLAGS) -o run-stat mainStat.o -L. -llist -Wl,-rpath,.

$(CC) $(FLAGS) -o run-dyn mainDyn.o -ldl

mainStat.o: mainStat.c

$(CC) -c $(FLAGS) mainStat.c

mainDyn.o: mainDyn.c

$(CC) -c $(FLAGS) mainDyn.c

list.o: list.c

$(CC) -c -fPIC $(FLAGS) list.c

liblist.so: list.o

$(CC) $(FLAGS) -shared -o liblist.so list.o

clean:

rm -f \*.o run-stat run-dyn \*.so

**Отчет Strace (подключение динамической библиотеки)**

openat(AT\_FDCWD, "./liblist.so", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\t\0\0\0\0\0\0"..., 832) = 832

## Вывод

В ходе данной лабораторной работы я приобрел навык создания и использования динамических библиотек. Созданную динамическую структуру данных я подключал к использованию основной программой двумя способами: линковщиком, с переопределением внутри программы необходимых мне функций, и стандартным подключением.