Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Динамические библиотеки**

Студент: Камеш Михаил Алексеевич

Группа: М8О–207Б–18

Вариант: 28

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Примеры работы
6. Вывод

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.

Вариант задания.

Работа с деком

Общие сведения о программе

Программа компилируется из нескольких файлов с помощью Makefile. В программе используются следующие системные вызовы:

1. dlclose - уменьшает количество открытых ссылок на библиотеку.

2. dlopen - открывает библиотеку и увеличивает количество открытых ссылок библиотеки.

3. dlsym - ищет в библиотеке символ (например, функцию) и возвращает его адрес.

4. dlerror - возвращает указатель на строку сообщения после возврата ошибки из dlopen (), dlsym () или dlclose ().

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Создание deque.h/deque.c файлов, описывающих функционал дека

2. Создание stat.c - программа, которая используют библиотеку, используя знания полученные на этапе компиляции

3. Создание dynam.c - программа, которая используют библиотеку, используя знания полученные на этапе компиляции

4. Подключение динамической библиотеки и сбор программы с помощью Makefile

Файлы

Deque.h

#ifndef DEQUE\_H

#define DEQUE\_H

#include <stdbool.h>

typedef struct deque\_struct deque\_type;

typedef char\* deque\_val\_type;

deque\_type \* deque\_alloc();

void deque\_free(deque\_type \*d);

bool deque\_is\_empty(deque\_type \*d);

void deque\_push\_front(deque\_type \*d, deque\_val\_type v);

void deque\_push\_back(deque\_type \*d, deque\_val\_type v);

deque\_val\_type deque\_pop\_front(deque\_type \*d);

deque\_val\_type deque\_pop\_back(deque\_type \*d);

deque\_val\_type deque\_peek\_front(deque\_type \*d);

deque\_val\_type deque\_peek\_back(deque\_type \*d);

#endif

Deque.c

#include <stdlib.h>

#include <assert.h>

#include "deque.h"

struct node\_struct {

struct node\_struct \*next;

struct node\_struct \*prev;

deque\_val\_type val;

};

struct deque\_struct {

struct node\_struct \*head;

struct node\_struct \*tail;

};

deque\_type \* deque\_alloc() {

deque\_type \*d = malloc(sizeof(deque\_type));

if (d != NULL)

d->head = d->tail = NULL;

return d;

}

void deque\_free(deque\_type \*d) {

free(d);

}

bool deque\_is\_empty(deque\_type \*d) {

return d->head == NULL;

}

void deque\_push\_front(deque\_type \*d, deque\_val\_type v) {

struct node\_struct \*n = malloc(sizeof(struct node\_struct));

assert(n != NULL);

n->val = v;

n->next = d->head;

n->prev = NULL;

if (d->tail == NULL) {

d->head = d->tail = n;

} else {

d->head->prev = n;

d->head = n;

}

}

void deque\_push\_back(deque\_type \*d, deque\_val\_type v) {

struct node\_struct \*n = malloc(sizeof(struct node\_struct));

assert(n != NULL);

n->val = v;

n->prev = d->tail;

n->next = NULL;

if (d->head == NULL) {

d->head = d->tail = n;

} else {

d->tail->next = n;

d->tail = n;

}

}

deque\_val\_type deque\_pop\_front(deque\_type \*d) {

deque\_val\_type v = d->head->val;

struct node\_struct \*n = d->head;

if (d->head == d->tail)

d->head = d->tail = NULL;

else

d->head = n->next;

free(n);

return v;

}

deque\_val\_type deque\_pop\_back(deque\_type \*d) {

deque\_val\_type v = d->tail->val;

struct node\_struct \*n = d->tail;

if (d->head == d->tail)

d->head = d->tail = NULL;

else

d->tail = n->prev;

free(n);

return v;

}

deque\_val\_type deque\_peek\_front(deque\_type \*d) {

return d->head->val;

}

deque\_val\_type deque\_peek\_back(deque\_type \*d) {

return d->tail->val;

}

Dynam.c

#include <stdio.h>

#include <assert.h>

#include "deque.h"

#include <dlfcn.h>

int main() {

deque\_type\* (\*deque\_alloc)();

void (\*deque\_free)(deque\_type \*d);

bool (\*deque\_is\_empty)(deque\_type \*d);

void (\*deque\_push\_front)(deque\_type \*d, deque\_val\_type v);

void (\*deque\_push\_back)(deque\_type \*d, deque\_val\_type v);

deque\_val\_type (\*deque\_pop\_front)(deque\_type \*d);

deque\_val\_type (\*deque\_pop\_back)(deque\_type \*d);

deque\_val\_type (\*deque\_peek\_front)(deque\_type \*d);

deque\_val\_type (\*deque\_peek\_back)(deque\_type \*d);

char \*err;

void \*libHandle;

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

if (!libHandle) {

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

return 1;

}

deque\_alloc = dlsym(libHandle, "deque\_alloc");

deque\_free = dlsym(libHandle, "deque\_free");

deque\_is\_empty = dlsym(libHandle, "deque\_is\_empty");

deque\_push\_front = dlsym(libHandle, "deque\_push\_front");

deque\_push\_back = dlsym(libHandle, "deque\_push\_back");

deque\_pop\_front = dlsym(libHandle, "deque\_pop\_front");

deque\_peek\_front = dlsym(libHandle, "deque\_peek\_front");

deque\_pop\_back = dlsym(libHandle, "deque\_pop\_back");

deque\_peek\_back = dlsym(libHandle, "deque\_peek\_back");

if(err = dlerror()) {

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

return 1;

}

deque\_type \*d = (\*deque\_alloc)();

assert(d != NULL);

int k = 1;

char str[32];

char str2[32];

while (k) {

printf("0. Exit\n");

printf("1. Push back\n");

printf("2. Push front\n");

printf("3. Check back\n");

printf("4. Check front\n");

printf("5. Pop back\n");

printf("6. Pop front\n");

printf("7. Check empty\n");

printf("8. Pop all and show\n");

scanf("%d", &k);

if (k == 0) {

deque\_free(d);

return 0;

}

if (k == 1) {

fgets(str, sizeof(str), stdin);

(\*deque\_push\_back)(d, str);

}

if (k == 2) {

printf("Enter a string to push into deque\n");

fgets(str2, sizeof(str2), stdin);

(\*deque\_push\_front)(d, str2);

}

if (k == 3) {

if ((\*deque\_is\_empty)(d) != true) {

printf("%s\n", (\*deque\_peek\_back)(d));

}

}

if (k == 4) {

if ((\*deque\_is\_empty)(d) != true) {

printf("%s\n", (\*deque\_peek\_front)(d));

}

}

if (k == 5) {

if ((\*deque\_is\_empty)(d) != true) {

(\*deque\_pop\_back)(d);

}

}

if (k == 6) {

if ((\*deque\_is\_empty)(d) != true) {

(\*deque\_pop\_front)(d);

}

}

if (k == 7) {

if ((\*deque\_is\_empty)(d) != true) {

printf("Not Empty\n");

}

else if ((\*deque\_is\_empty)(d) == true) {

printf("Empty\n");

}

}

if (k == 8) {

while ((\*deque\_is\_empty)(d) != true) {

printf("%s\n", (\*deque\_pop\_front)(d));

}

}

}

return 0;

}

Stat.c

#include <stdio.h>

#include <assert.h>

#include "deque.h"

int main() {

deque\_type \*d = deque\_alloc();

assert(d != NULL);

int k = 1;

char str[32];

char str2[32];

while (k) {

printf("0. Exit\n");

printf("1. Push back\n");

printf("2. Push front\n");

printf("3. Check back\n");

printf("4. Check front\n");

printf("5. Pop back\n");

printf("6. Pop front\n");

printf("7. Check empty\n");

printf("8. Pop all and show\n");

scanf("%d", &k);

getchar();

if (k == 0) {

deque\_free(d);

return 0;

}

if (k == 1) {

printf("Enter a string to push into deque\n");

fgets(str, sizeof(str), stdin);

deque\_push\_back(d, str);

}

if (k == 2) {

printf("Enter a string to push into deque\n");

fgets(str2, sizeof(str2), stdin);

deque\_push\_front(d, str2);

}

if (k == 3) {

if (deque\_is\_empty(d) != true) {

printf("%s\n", deque\_peek\_back(d));

}

}

if (k == 4) {

if (deque\_is\_empty(d) != true) {

printf("%s\n", deque\_peek\_front(d));

}

}

if (k == 5) {

if (deque\_is\_empty(d) != true) {

deque\_pop\_back(d);

}

}

if (k == 6) {

if (deque\_is\_empty(d) != true) {

deque\_pop\_front(d);

}

}

if (k == 7) {

if (deque\_is\_empty(d) != true) {

printf("Not Empty\n");

}

else if (deque\_is\_empty(d) == true) {

printf("Empty\n");

}

}

if (k == 8) {

while (deque\_is\_empty(d) != true) {

printf("%s\n", deque\_pop\_front(d));

}

}

}

deque\_push\_back(d, "helle");

deque\_push\_back(d, "world");

while (deque\_is\_empty(d) != true) {

printf("%s\n", deque\_pop\_front(d));

}

deque\_free(d);

return 0;

}

Демонстрация работы программы.

0. Exit

1. Push back

2. Push front

3. Check back

4. Check front

5. Pop back

6. Pop front

7. Check empty

8. Pop all and show

1 5

(5)

2 7

(7 5)

3

(5)

4

(7)

5

(7)

7

Not empty.

8

(7)

Выводы:

В процессе выполнения работы были изучены принципы построения динамических библиотек и создания программ, которые используют функции динамических библиотек. Для

демонстрации работы была также создана статическая библиотека без использования функций динамических библиотек. Четыре функции dlopen (), dlsym (), dlclose (),

dlerror () реализуют интерфейс для загрузчика динамической линковки. Были некоторые проблемы при считывании строк с клавиатуры и неправильно их интерпретирование,

но это не связано с основным функционалом программы и частично устраняется getchar коммандами.