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: Петрухин Дмитрий Олегович

Группа: М8О –201Б-18

Вариант: 13

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2019

**Содержание**

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

**Постановка задачи**.

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.

**Задание согласно варианту:**

Структура данных: очередь

Тип данных: целочисленный 32-битный.

**Общие сведения о программе**

Программа состоит из следующих файлов:

1) main.c - программа, которая использует библиотеку на этапе компиляции

2) main\_dyn.c - программа, которая подгружает необходимые данные из библиотеки во время исполнения программы.

3) queue.c и queue.h - реализация библиотеки с очередью.

**Статическая библиотека** - это коллекция объектных файлов, которые присоединяются к программе во время линковки программы. Таким образом статические библиотеки используются только при созданиии программы. Потом в работе самой программы они не принимают участие, в отличие от динамических библиотек.

**Динамическая библиотека** - это созданная специальным образом библиотека, которая присоединяется к результирующей программе в два этапа. Первый этап, это естественно этап компиляции. На этом этапе линковщик встраивает в программу описания требуемых функций и переменных, которые присутствуют в библиотеке. Сами объектные файлы из библиотеки не присоединяются к программе. Присоединение этих объектных файлов(кодов функций) осуществляет системный динамический загрузчик во время запуска программы. Загрузчик проверяет все библиотеки прилинкованные с программе на наличие требуемых объектных файлов, затем загружает их в память и присоединяет их в копии запущенной программы, находящейся в памяти.

Основные системные вызовы:

1) **void \*dlopen(const char \**filename*, int *flag*)** - загружает динамическую библиотеку, имя которой указано в строке *filename*, и возвращает прямой указатель на начало динамической библиотеки. Если *filename* не является полным именем файла (т.е. не начинается с "/"), то файл ищется в следующих местах:

- в разделенном двоеточием списке каталогов, в переменной окружения пользователя **LD\_LIBRARY\_PATH**.

- в списке библиотек, кэшированных в файле */etc/ld.so.cache*.

- в usr/lib и далее в /lib

flag=**RTLD\_LAZY**, подразумевает разрешение неопределенных символов в виде кода, содержащегося в исполняемой динамической библиотеке.

2) **void \*dlsym(void \**handle*, char \**symbol*) -** передаётся указатель на объект, возвращаемый вызовом **[dlopen](http://ru.manpages.org/dlopen/3)**(3) и имя символа (с null в конце). В результате функция возвращает адрес, по которому символ расположен в памяти. Если символ не найден в указанном объекте или во всех общих объектах, которые были автоматически загружены **[dlopen](http://ru.manpages.org/dlopen/3)**(3) на момент загрузки объекта, то **dlsym**() возвращает NULL (поиск, выполняемый **dlsym**(), охватывает всё дерево зависимостей этих общих объектов).

3) **int dlclose(void \*handle*)*** *-* уменьшает на единицу счетчик ссылок на указатель динамической библиотеки *handle*. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается. При успешном завершении возвращает 0.

**Необходимые ключи gcc для создания и использования динамической библиотеки:**

1) -shared — ключ, необходимый для создания shared object файла, т.е самой динамической библиотеки

2) -L. — ключ указывает ранее прописанный путь для динамической библиотеки

LD\_LIBRARY\_PATH=/root:${LD\_LIBRARY\_PATH}

export LD\_LIBRARY\_PATH

указывается путь, где лежат все пользовательские библиотеки(туда где лежит библиотека в данный момент)

3) -fPIC — генерируемый компилятором код становится независимым от адресов, связано это с тем, что все объектные файлы создаваемые обычным образом не имеют представления о том в какие адреса памяти будет загружена использующая их программа. Несколько различных программ могут использовать одну библиотеку, и каждая из них располагается в различном адресном пространстве. Поэтому требуется, чтобы переходы в функциях библиотеки (операции **goto** на ассемблере) использовали не абсолютную адресацию, а относительную.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Создаю файл, где подключаю библиотеку queue.h и использую в ней основные функции библиотеки.
2. Создаю файл, в котором в программе с помощью системных вызовов подгружаю необходимые мне данные, чтобы использовать основные функции библиотеки.

**Листинг программы.**

**Файл main.c**

#include "queue.h"

#include <stdlib.h>

void menu() {

printf("\nMenu -- 1\n");

printf("Push -- 2\n");

printf("Pop -- 3\n");

printf("Show top element -- 4\n");

printf("Print queue -- 5\n");

printf("Print size queue -- 6\n");

printf("Exit -- 7\n");

}

int main(){

queue q;//////????

q\_init(&q);

int n, top;

char \* digit = malloc(sizeof(char)\*100);

bool t = true;

menu();

while(t){

printf("Enter the number to select an operation from the menu\n");

scanf("%d", &n);

while(n == 0){

printf("Invalid input of a number.\n");

printf("Enter a number from 1 to 7 to select an operation from the menu\n");

scanf("%d", &n);

}

switch(n){

case 1:

menu();

break;

case 2:

printf("Enter a digit:\n");

scanf("%s", digit);

q\_push(&q, atoi(digit));

break;

case 3:

if(q\_size(&q)==0){

printf("Error: queue is empty");

break;

}

q\_pop(&q);

break;

case 4:

printf("The head of element: ");

printf("%d\n", q\_top(&q));

printf("\n");

break;

case 5:

q\_print(&q);

break;

case 6:

printf("Size queue: ");

printf("%d\n", q\_size(&q));

break;

case 7:

t = false;

break;

}

}

q\_destroy(&q);

return 0;

}

**main\_dyn.c**

#include "queue.h"

#include <dlfcn.h>

#include <stdio.h>

#include <stdlib.h>

void menu() {

printf("\nMenu -- 1\n");

printf("Push -- 2\n");

printf("Pop -- 3\n");

printf("Show top element -- 4\n");

printf("Print queue -- 5\n");

printf("Print size queue -- 6\n");

printf("Exit -- 7\n");

printf("Empty or nor -- 8\n");

}

int main(){

int n;

char \* digit = malloc(sizeof(char)\*100);

bool t = true;

void \*ext\_library;

menu();

ext\_library = dlopen("libqueue.so", RTLD\_LAZY);

if (!ext\_library){

//если ошибка, то вывести ее на экран

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

return 1;

};

void (\*q\_init)(queue\* q) = dlsym(ext\_library,"q\_init");

int (\*q\_top)(const queue\* q) = dlsym(ext\_library, "q\_top");

int (\*q\_pop)(queue\* q) = dlsym(ext\_library, "q\_pop");

int (\*q\_size)(const queue\* q) = dlsym(ext\_library, "q\_size");

void (\*q\_push)(queue\* q, const int arrray) = dlsym(ext\_library, "q\_push");

bool (\*q\_empty)(const queue\* q) = dlsym(ext\_library, "q\_empty");

void (\*q\_destroy)(queue\* q) = dlsym(ext\_library, "q\_destroy");

void (\*q\_print)(const queue\* q) = dlsym(ext\_library, "q\_print");

queue q;

(\*q\_init)(&q);

while(t){

printf("Enter the number to select an operation from the menu\n");

scanf("%d", &n);

while(n == 0){

printf("Invalid input of a number.\n");

printf("Enter a number from 1 to 7 to select an operation from the menu\n");

scanf("%d", &n);

}

switch(n){

case 1:

menu();

break;

case 2:

printf("Enter a digit:\n");

scanf("%s", digit);

(\*q\_push)(&q, atoi(digit));

break;

case 3:

if((\*q\_size)(&q)==0){

printf("Error: queue is empty");

break;

}

(\*q\_pop)(&q);

break;

case 4:

printf("The head of element: ");

printf("%d\n", (\*q\_top)(&q));

printf("\n");

break;

case 5:

(\*q\_print)(&q);

break;

case 6:

printf("Size queue: ");

printf("%d\n", (\*q\_size)(&q));

break;

case 7:

t = false;

break;

case 8:

if( (\*q\_empty)(&q) )

printf("Queue is empty\n");

else

printf("Queue is not empty");

break;

}

}

(\*q\_destroy)(&q);

dlclose(ext\_library);

return 0;

}

**queue.c**

#include "queue.h"

void q\_init(queue\* q) {

q->head = NULL;

q->tail = NULL;

q->size = 0;

}

int q\_top(const queue \*q) {

return q->head->value;

}

int q\_pop(queue\* q) {

int temp = q\_top(q);

queue\_item\* ptr\_to\_free = q->head;

q->head = q->head->next;

if (q->head == NULL) {

q->tail = NULL;

}

free(ptr\_to\_free);

q->size--;

return temp;

}

void q\_push(queue\* q, const int array) {

queue\_item\* new\_elem = malloc(sizeof(queue\_item));

new\_elem->value = array;

new\_elem->next = 0;

if (q->head == NULL) {

q->head = new\_elem;

q->tail = new\_elem;

new\_elem->prev = 0;

} else {

q->tail->next = new\_elem;

new\_elem->prev = q->tail;

q->tail = new\_elem;

}

q->size++;

}

void q\_print(const queue \*q){

if(q\_empty(q)){

printf("<queue is empty>\n");

return;

}

printf("queue: ");

queue\_item \*temp = q->head;

while(temp != NULL){

if(temp->next == NULL){

printf("%d.", temp->value);

}else{

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

}

temp = temp->next;

}

printf("\n");

}

bool q\_empty(const queue\* q) {

return q->head == NULL;

}

int q\_size(const queue\* q) {

return q->size;

}

void q\_destroy(queue\* q) {

queue\_item\* start = q->head;

while (start != NULL) {

queue\_item\* next = start->next;

free(start);

start = next;

}

q->head = NULL;

q->tail = NULL;

q->size = 0;

}

**queue.h**

#ifndef OS\_LAB\_3\_QUEUE\_H

#define OS\_LAB\_3\_QUEUE\_H

#include <stdbool.h>

#include <stdlib.h>

#include <stdio.h>

typedef struct QueueItem queue\_item;

struct QueueItem {

struct QueueItem\* next;

struct QueueItem\* prev;

int value;

};

typedef struct Queue queue;

struct Queue {

queue\_item\* head;

queue\_item\* tail;

int size;

};

void q\_init(queue\* q);

int q\_top(const queue\* q);

int q\_pop(queue\* q);

int q\_size(const queue\* q);

void q\_push(queue\* q, const int arrray);

bool q\_empty(const queue\* q);

void q\_destroy(queue\* q);

void q\_print(const queue\* q);

#endif //OS\_LAB\_3\_QUEUE\_H

**Makefile**

all: main main\_dyn

main: libqueue.so main.c

gcc main.c -L. -lqueue -o main

echo main is builded

main\_dyn: libqueue.so main\_dyn.c

gcc -Wall -Werror -Wextra main\_dyn.c -o main\_dyn

echo main\_dyn is builded

libqueue.so: queue.o

gcc -shared queue.o -o libqueue.so

echo libqueue.so is builded

queue.o: queue.c

gcc -Wall -Werror -Wextra -fPIC -c queue.c

clean:

rm -rf \*.so

rm -rf \*.o

**Тестирование.**

**test1:** Запуск программы, которая использует библиотеку уже на этапе компиляции.

2 11

2 fishes

4

2 9

5

3

6

3

3

3

2 11

0

7

**test2:** Запуск программы, которая выгружает данные из библиотеки во время исполнения программы

3

2 54

2 -11

4

2 99

5

8

7

**Демонстрация работы программы.**

➜ os\_lab\_05 make

gcc -Wall -Werror -Wextra -fPIC -c queue.c

gcc -shared queue.o -o libqueue.so

echo libqueue.so is builded

libqueue.so is builded

gcc main.c -L. -lqueue -o main

echo main is builded

main is builded

gcc -Wall -Werror -Wextra main\_dyn.c -o main\_dyn

echo main\_dyn is builded

main\_dyn is builded

➜ os\_lab\_05 ./main

Menu -- 1

Push -- 2

Pop -- 3

Show top element -- 4

Print queue -- 5

Print size queue -- 6

Exit -- 7

Enter the number to select an operation from the menu

2

Enter a digit:

11

Enter the number to select an operation from the menu

2

Enter a digit:

fshgs

Enter the number to select an operation from the menu

4

The head of element: 11

Enter the number to select an operation from the menu

2

Enter a digit:

9

Enter the number to select an operation from the menu

5

queue: 11, 0, 9.

Enter the number to select an operation from the menu

3

Enter the number to select an operation from the menu

6

Size queue: 2

Enter the number to select an operation from the menu

3

Enter the number to select an operation from the menu

3

Enter the number to select an operation from the menu

3

Error: queue is emptyEnter the number to select an operation from the menu

2

Enter a digit:

11

Enter the number to select an operation from the menu

0

Invalid input of a number.

Enter a number from 1 to 7 to select an operation from the menu

7

➜ os\_lab\_05 ./main\_dyn

Menu -- 1

Push -- 2

Pop -- 3

Show top element -- 4

Print queue -- 5

Print size queue -- 6

Exit -- 7

Empty or nor -- 8

Enter the number to select an operation from the menu

3

Error: queue is emptyEnter the number to select an operation from the menu

2

Enter a digit:

54

Enter the number to select an operation from the menu

2

Enter a digit:

-11

Enter the number to select an operation from the menu

4

The head of element: 54

Enter the number to select an operation from the menu

2

Enter a digit:

99

Enter the number to select an operation from the menu

6

Size queue: 3

Enter the number to select an operation from the menu

8

Queue is not emptyEnter the number to select an operation from the menu

7

**Вывод.**

Динамические библиотеки очень полезный инструмент. Файлы, использующие динамические библиотеки на практике в больших проектах занимают гораздо меньший размер, чем файлы, использующие статические библиотеки. Также нет необходимости перекомпилировать библиотеку в случае изменений основного файла программы и использовать одну библиотеку для нескольких различных проектов.