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: Базаргармаев Нима Дондокович

Группа: М8О –301Б-18

Вариант: 2

Преподаватель: Миронов Евгений Сергеевич

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. 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: стек

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\_dll.c - программа, которая подгружает необходимые данные из библиотеки во время исполнения программы.

3) stack.h, stack.c - реализация библиотеки со стеком.

**Статическая библиотека** - это коллекция объектных файлов, которые присоединяются к программе во время линковки программы. Таким образом, статические библиотеки используются только при создании 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 к 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 stack.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.
3. Изменяю переменную среды LD\_LIBRARY\_PATH на местоположение библиотеки со стеком

**Листинг программы.**

**Makefile**

all: main main\_so

main: stack.so main.c

gcc main.c -L. stack.so -o main

main\_so: stack.so main\_so.c

gcc main\_so.c -ldl -o main\_so

stack.so: stack.o

gcc -shared stack.o -o stack.so

stack.o: stack.c

gcc -fPIC -c stack.c

clean:

rm -rf \*.so

rm -rf \*.o

**main.c**

#include "stack.h"

#include <stdio.h>

#include <dlfcn.h>

void print\_menu() {

printf("Menu: \n");

printf("1. Push an element to stack\n");

printf("2. Pop an element from stack\n");

printf("3. Check if stack empty\n");

printf("4. Get the top element from stack\n");

printf("5. Get the size of stack\n");

printf("6. Print stack\n");

printf("7. Exit\n");

printf("8. Print menu\n");

}

int main() {

Stack stack;

stack.size = 0;

for (int i = 0; i < 20; ++i) {

stack.stack[i] = 0;

}

int n = 0;

bool flag = true;

print\_menu();

while (flag) {

printf("Enter the number to select action\n");

scanf("%d", &n);

switch (n) {

case 1: {

int number = 0;

printf("Enter the number to push\n");

scanf("%d", &number);

push(&stack, number);

break;

}

case 2:

pop(&stack);

break;

case 3:

if (stack\_empty(&stack)) {

printf("Stack is empty\n");

} else {

printf("Stack isn't empty\n");

}

break;

case 4:

printf("The top element from stack is: %d\n", top(&stack));

break;

case 5:

printf("The size of stack is: %d\n", size(&stack));

break;

case 6:

print(&stack);

break;

case 7:

flag = false;

break;

case 8:

print\_menu();

break;

}

}

return 0;

}

**main\_so.cpp**

#include "stack.h"

#include <stdio.h>

#include <dlfcn.h>

void print\_menu() {

printf("Menu: \n");

printf("1. Push an element to stack\n");

printf("2. Pop an element from stack\n");

printf("3. Check if stack empty\n");

printf("4. Get the top element from stack\n");

printf("5. Get the size of stack\n");

printf("6. Print stack\n");

printf("7. Exit\n");

printf("8. Print menu\n");

}

int main() {

void\* ext\_library;

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

if (!ext\_library) {

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

return 1;

}

bool (\*stack\_empty)(Stack\* stack) = dlsym(ext\_library, "stack\_empty");

void (\*push)(Stack\* stack, const int value) = dlsym(ext\_library, "push");

void (\*pop)(Stack\* stack) = dlsym(ext\_library, "pop");

int (\*top)(Stack\* stack) = dlsym(ext\_library, "top");

int (\*size)(Stack\* stack) = dlsym(ext\_library, "size");

void (\*print)(Stack\* stack) = dlsym(ext\_library, "print");

Stack stack;

stack.size = 0;

for (int i = 0; i < 20; ++i) {

stack.stack[i] = 0;

}

int n = 0;

bool flag = true;

print\_menu();

while (flag) {

printf("Enter the number to select action\n");

scanf("%d", &n);

switch (n) {

case 1: {

int number = 0;

printf("Enter the number to push\n");

scanf("%d", &number);

(\*push)(&stack, number);

break;

}

case 2:

(\*pop)(&stack);

break;

case 3:

if ((\*stack\_empty)(&stack)) {

printf("Stack is empty\n");

} else {

printf("Stack isn't empty\n");

}

break;

case 4:

printf("The top element from stack is: %d\n", (\*top)(&stack));

break;

case 5:

printf("The size of stack is: %d\n", (\*size)(&stack));

break;

case 6:

(\*print)(&stack);

break;

case 7:

flag = false;

break;

case 8:

print\_menu();

break;

}

}

dlclose(ext\_library);

return 0;

}

**stack.h**

#pragma once

#include <stdbool.h>

typedef struct Stack {

int stack[20];

int size;

} Stack;

bool stack\_empty(Stack\* stack);

void push(Stack\* stack, const int value);

void pop(Stack\* stack);

int top(Stack\* stack);

int size(Stack\* stack);

void print(Stack\* stack);

**stack.c**

#include "stack.h"

#include <stdio.h>

bool stack\_empty(Stack\* stack) {

if (stack->size == 0) {

return true;

}

return false;

}

void push(Stack\* stack, const int value) {

stack->size++;

stack->stack[stack->size - 1] = value;

}

void pop(Stack\* stack) {

if (stack->size == 0) {

printf("Stack is empty\n");

return;

}

stack->stack[stack->size - 1] = 0;

stack->size--;

}

int top(Stack\* stack) {

if (stack->size == 0) {

printf("Stack is empty\n");

return 0;

}

return stack->stack[stack->size - 1];

}

int size(Stack\* stack) {

return stack->size;

}

void print(Stack\* stack) {

for (int i = stack->size - 1; i >= 0; --i) {

printf("%d ", stack->stack[i]);

}

printf("\n");

}

**Тестирование.**

См. следующий пункт.

**Демонстрация работы программы.**

user@DESKTOP-4RFVSK2:~/os/5lab$ ./main

Menu:

1. Push an element to stack

2. Pop an element from stack

3. Check if stack empty

4. Get the top element from stack

5. Get the size of stack

6. Print stack

7. Exit

8. Print menu

Enter the number to select action

1

Enter the number to push

2

Enter the number to select action

1

Enter the number to push

15

Enter the number to select action

1

Enter the number to push

5897

Enter the number to select action

3

Stack isn't empty

Enter the number to select action

4

The top element from stack is: 5897

Enter the number to select action

6

5897 15 2

Enter the number to select action

5

The size of stack is: 3

Enter the number to select action

2

Enter the number to select action

6

15 2

Enter the number to select action

2

Enter the number to select action

6

2

Enter the number to select action

2

Enter the number to select action

6

Enter the number to select action

2

Stack is empty

Enter the number to select action

6

Enter the number to select action

7

user@DESKTOP-4RFVSK2:~/os/5lab$ ./main\_so

Menu:

1. Push an element to stack

2. Pop an element from stack

3. Check if stack empty

4. Get the top element from stack

5. Get the size of stack

6. Print stack

7. Exit

8. Print menu

Enter the number to select action

1

Enter the number to push

654984

Enter the number to select action

6

654984

Enter the number to select action

1

Enter the number to push

6549

Enter the number to select action

6

6549 654984

Enter the number to select action

2

Enter the number to select action

6

654984

Enter the number to select action

4

The top element from stack is: 654984

Enter the number to select action

5

The size of stack is: 1

Enter the number to select action

6

654984

Enter the number to select action

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.