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: Кудинов Сергей Владимирович
Группа: М80 – 206Б-18
Вариант: 15
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Постановка задачи

Создать динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку двумя способами:

- Подключить библиотеку на этапе линковки
- Подгрузив библиотеку в память с помощью системных вызовов

Вариант задания: 1-4. Библиотека должна обеспечивать работу с вектором вещественных чисел

Общие сведения о программе

Программа представляет собой динамическую библиотеку, состоящую из файлов main_dynamic.c, main_static.c, vector.c, vector.h.

Сборка и подключение библиотеки выполняется системой сборки CMake. Используются команды add_executable, add_library, target_link_library(SHARED).

Используются следующие системные вызовы

1. **dlopen** — загружает динамическую библиотеку и возвращает handle
2. **dlclose —** уменьшает счетчик 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 взаимодействие с динамической библиотекой.

Общий метод и алгоритм решения

- Создать файлы динамической библиотеки.
- Собрать библиотеку с помощью CMake и получить файл с расширением .so
- Написать две программы, одна из которых использует системные вызовы для открытия библиотеки, а другая подключает библиотеку на этапе линковки.
- Собрать эти программы и проверить корректность их работы

Код программы

vector.h:

```
#include "vector.h"
#include <stdio.h>
#include <stdlib.h>

void vectorInit(vector *v)
{
        v->data = NULL;
        v->size = 0;
        v->count = 0;
}

int vectorCount(vector *v)
{
        return v->count;
}
void vectorPrint(vector *v) {
    for (int i = 0; i<v->count; ++i) {
        printf("%lf ",v->data[i]);
    }
    printf("\n");
}
void vectorAdd(vector *v, double e)
{
        if (v->size == 0) {
                v->size = 10;
                v->data = malloc(sizeof(double*) * v->size);
        }


        if (v->size == v->count) {
                v->size *= 2;
                v->data = realloc(v->data, sizeof(void*) * v->size);
        }

        v->data[v->count] = e;
        v->count++;
}

void vectorSet(vector *v, int index, double d)
{
        if (index >= v->count) {
                return;
        }

        v->data[index] = d;
}

double vectorGet(vector *v, int index)
{
        if (index >= v->count) {
                return;
        }

        return v->data[index];
}
```

```
void vectorDelete(vector *v, int index)
{
        if (index >= v->count) {
                return;
        }

        for (int i = index; i<v->count-1; ++i) {
        v[i] = v[i+1];
    }
        v->count--;

}

void vectorFree(vector *v)
{
        free(v->data);
}
```

vector.c:

```
#include "vector.h"
#include <stdio.h>
#include <stdlib.h>

void vectorInit(vector *v)
{
        v->data = NULL;
        v->size = 0;
        v->count = 0;
}

int vectorCount(vector *v)
{
        return v->count;
}
void vectorPrint(vector *v) {
    for (int i = 0; i<v->count; ++i) {
        printf("%lf ",v->data[i]);
    }
    printf("\n");
}
void vectorAdd(vector *v, double e)
{
        if (v->size == 0) {
                v->size = 10;
                v->data = malloc(sizeof(double*) * v->size);
        }


        if (v->size == v->count) {
                v->size *= 2;
                v->data = realloc(v->data, sizeof(void*) * v->size);
        }

        v->data[v->count] = e;
        v->count++;
}
```

```
void vectorSet(vector *v, int index, double d)
{
        if (index >= v->count) {
                return;
        }

        v->data[index] = d;
}

double vectorGet(vector *v, int index)
{
        if (index >= v->count) {
                return;
        }

        return v->data[index];
}

void vectorDelete(vector *v, int index)
{
        if (index >= v->count) {
                return;
        }

        for (int i = index; i<v->count-1; ++i) {
        v[i] = v[i+1];
    }
        v->count--;

}

void vectorFree(vector *v)
{
        free(v->data);
}
```

main_static.c:

```
#include "vector.h"
#include <stdlib.h>
#include <stdio.h>
int main(void)
{
    double key = 0;
    int ind;
    printf("This is compile-time linking\n\n");


    int act = 0;
    vector *v = (vector*)malloc(sizeof(vector));
    printf("Choose an operation:\n");
    printf("1) Initialize the vector\n");
    printf("2) Push back\n");
    printf("3) Insert in random place\n");
    printf("4) Print vector\n");
```

```
    printf("5) delete last element vector\n");

    printf("0) Exit\n");

    while (scanf("%d", &act) && act) {
        switch(act) {
        case 1:
            printf("Enter key to initialize the queue: ");
            vectorInit(v);
            break;

        case 2:
            printf("Enter key: ");
            scanf("%lf", &key);
            vectorAdd(v, key);
            break;

        case 3:
            printf("Enter index: ");
            scanf("%d", &ind);
            printf("Enter key: ");
            scanf("%lf", &key);
            vectorSet(v,ind,key);
            break;

        case 4:
            vectorPrint(v);
            break;
        case 5:
            printf("Enter index: ");
            scanf("%d", &ind);
            vectorDelete(v,ind);
        default:
            printf("Incorrect command\n");
            break;
        }
    printf("Choose an operation:\n");
    printf("1) Initialize the vector\n");
    printf("2) Push back\n");
    printf("3) Insert in random place\n");
    printf("4) Print vector\n");
    printf("5) delete last element vector\n");

    printf("0) Exit\n");
    }
    (*vectorFree)(&v);
    free(v);


    return 0;
}
```

main_dynamic.c:

```
#include <dlfcn.h>

#include "vector.h"
```

```
int main(void)
{
   void (*vectorInit)(vector* v);
    int (*vectorCount)(vector* v);
    void (*vectorAdd)(vector* v, double d);
    void (*vectorSet)(vector* v, int index, double d);
    double (*vectorGet)(vector* v, int index);
    void (*vectorDelete)(vector* v, int index);
    void (*vectorFree)(vector* v);
    void (*vectorPrint)(vector* v);

    void* libHandle;
    libHandle = dlopen("./libvector.so", RTLD_LAZY);
    if (!libHandle) {
        perror("dlopen error");
        exit(-1);
    }

    vectorInit = (void (*)(void))dlsym(libHandle, "vectorInit");
    vectorCount = (int (*)(void))dlsym(libHandle, "vectorCount");
    vectorAdd = (void (*)(void))dlsym(libHandle, "vectorAdd");
    vectorSet = (void (*)(void))dlsym(libHandle, "vectorSet");
    vectorGet = (double (*)(void))dlsym(libHandle, "vectorGet");
    vectorDelete = (void (*)(void))dlsym(libHandle, "vectorDelete");
    vectorFree = (void (*)(void))dlsym(libHandle, "vectorFree");
    vectorPrint = (void (*)(void))dlsym(libHandle, "vectorPrint");

    int act = 0;
    double key = 0;
    int ind;
    vector *v = (vector*)malloc(sizeof(vector));
    printf("Choose an operation:\n");
    printf("1) Initialize the vector\n");
    printf("2) Push back\n");
    printf("3) Insert in random place\n");
    printf("4) Print vector\n");
    printf("0) Exit\n");

    while (scanf("%d", &act) && act) {
        switch(act) {
        case 1:
            printf("Enter key to initialize the queue: ");
            ( *vectorInit)(v);
            break;

        case 2:
            printf("Enter key: ");
            scanf("%lf", &key);
            (*vectorAdd)(v, key);
            break;

        case 3:
            printf("Enter index: ");
            scanf("%d", &ind);
            printf("Enter key: ");
            scanf("%lf", &key);
            (*vectorSet)(v,ind,key);
            break;
```

```
            case 4:
                (*vectorPrint)(v);
                break;
            case 5:
                printf("Enter index: ");
                scanf("%d", &ind);
                (*vectorDelete)(v,ind);
            default:
                printf("Incorrect command\n");
                break;
            }
        printf("Choose an operation:\n");
        printf("1) Initialize the vector\n");
        printf("2) Push back\n");
        printf("3) Insert in random place\n");
        printf("4) Print vector\n");
        printf("0) Exit\n");
        }
        (*vectorFree)(&v);
        free(v);
        if (dlclose(libHandle) != 0) {
            perror("dlclose error");
            exit(-1);
        }

        return 0;
    }
```

Вывод

В результате данной работы я научился создавать статические и динамические библиотеки, ознакомился с тем, как они размещаются в памяти и каким образом программы могут подключать их(с помощью системных вызовов или на этапе линковки).