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: Косогоров Владислав Валерьевич
Группа: М80 – 206Б-18
Вариант: 5-2
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: 5-2. Библиотека должна обеспечивать работу со очередью вещественных 64-битных чисел.

Общие сведения о программе

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

queue.h

```
#ifndef D_QUEUE_H

#define D_QUEUE_H
```

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

typedef struct node {
    double data;
    struct node* next;
} Node;

Node* Initialize(double val);
void PushFront(Node* head, double data);
void PopBack(Node** head);
void Destroy(Node** head);
void QueuePrint(Node* head);
#endif // D_QUEUE_H
```

queue.c

```
#include <assert.h>
#include "queue.h"

Node* Initialize(double val) {
    Node* res = (Node*)malloc(sizeof(Node));
    res->data = val;
    res->next = NULL;
    return res;
}

void PushFront(Node* head, double data) {
    assert(head != NULL);
```

```
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->next = head->next;
    head->next = newNode;
    newNode->data = head->data;
    head->data = data;
}

void PopBack(Node** head) {
    assert(head != NULL);
    if ((*head)->next == NULL) {
        Destroy(head);
        return;
    }
    Node* current = *head;
    while (current->next->next != NULL) {
        current = current->next;
    }
    free(current->next);
    current->next = NULL;
}

void QueuePrint(Node* head) {
    assert (head != NULL);
    Node* current = head;
    while (current) {
        printf("%lf ", current->data);
        current = current->next;
    }
    putchar('\n');
```

```
}


void Destroy(Node** head)  {
    Node* current = *head;
    Node* next;
    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }
    *head = NULL;
}
```

staticmain.c

```
#include "queue.h"

int main(void)
{
    int act = 0;
    double key = 0;
    Node* queue = NULL;
    printf("This is compile-time linking\n\n");
    printf("Choose an operation:\n");
    printf("1) Initialize the queue\n");
    printf("2) Push front\n");
    printf("3) Pop back\n");
    printf("4) Print queue\n");
    printf("0) Exit\n");
```

```
while (scanf("%d", &act) && act != 0) {
    switch(act) {
    case 1:
        printf("Enter key to initialize the queue: ");
        scanf("%lf", &key);
        queue = Initialize(key);
        break;
    case 2:
        printf("Enter key: ");
        scanf("%lf", &key);
        PushFront(queue, key);
        break;
    case 3:
        PopBack(&queue);
        break;
    case 4:
        QueuePrint(queue);
        break;
    default:
        printf("Incorrect command\n");
        break;
    }

    printf("Choose an operation:\n");
    printf("1) Initialize the queue\n");
    printf("2) Push front\n");
    printf("3) Pop back\n");
    printf("4) Print queue\n");
```

```
        printf("0) Exit\n");
    }

    Destroy(&queue);
    return 0;
}
```

dynamicmain.c

```
#include <dlfcn.h>

#include "queue.h"

int main(void)
{
    Node* (*Initialize)(double val);
    void (*PushFront)(Node* head, double data);
    void (*PopBack)(Node** head);
    void (*Destroy)(Node** head);
    void (*QueuePrint)(Node* head);

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

    Initialize = (Node (*)(void))dlsym(libHandle, "Initialize");
    PushFront = (void (*)(void))dlsym(libHandle, "PushFront");
```

```
PopBack = (void (*)(void))dlsym(libHandle, "PopBack");
Destroy = (void (*)(void))dlsym(libHandle, "Destroy");
QueuePrint = (void (*)(void))dlsym(libHandle, "QueuePrint");

int act = 0;
double key = 0;
Node* queue = NULL;

printf("Choose an operation:\n");
printf("1) Initialize the queue\n");
printf("2) Push front\n");
printf("3) Pop back\n");
printf("4) Print queue\n");
printf("0) Exit\n");

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

    case 2:
        printf("Enter key: ");
        scanf("%lf", &key);
        (*PushFront)(queue, key);
        break;
```

```
        case 3:
            (*PopBack)(&queue);
            break;

        case 4:
            (*QueuePrint)(queue);
            break;

        default:
            printf("Incorrect command\n");
            break;
        }
        printf("Choose an operation:\n");
        printf("1) Initialize the queue\n");
        printf("2) Push front\n");
        printf("3) Pop back\n");
        printf("4) Print queue\n");
        printf("0) Exit\n");
    }
    (*Destroy)(&queue);

    if (dlclose(libHandle) != 0) {
        perror("dlclose error");
        exit(-1);
    }
    return 0;
}
```

Демонстрация работы программы

…/prog_3_sem/os/lab5 ż ./run-dyn

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

1

Enter key to initialize the queue: 1

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

2

Enter key: 15

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

4

15.000000 1.000000

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

3

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

4

15.000000

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

2

Enter key: 123

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

4

123.000000 15.000000

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

2

Enter key: 12345

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

4

12345.000000 123.000000 15.000000

Choose an operation:

1) Initialize the queue

2) Push front

3) Pop back

4) Print queue

0) Exit

Вывод

В результате данной работы я научился создавать статические и динамические библиотеки, ознакомился с тем, как они размещаются в памяти и принципы их применения на примере работы с линейными структурами данных.