ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)"

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа № 5 по курсу

по курсу «Операционные системы»

Группа: М8о-207Б-18

Студент:

Токарев Никита Станиславович

Преподаватель:

Миронов Евгений Сергеевич

Оценка:

Дата:

Москва,2019

Оглавление

1.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.

Вариант 3: Дек,int

2.Структура программы

В данной работе в моя программа состоит из двух файлов:

1. maindnm.c
2. Makefile
3. mainstt.c
4. deque.c
5. deque.h

3.Описание программы

По способу компоновки библиотеки подразделяют на *архивы* (статические библиотеки, static libraries) и совместно используемые (динамические библиотеки, shared libraries). Для создания динамической в Makefile при

создании динамической библиотеки используется параметр -shared. Также при сборке объектного файла для нашей структуры указан параметр -fpic(позиционно независимый код) который используется в динамических библиотеках. Также хотелось бы отметить что при сборке исполняемого файла

с использованием динамической библиотеки, при динамической компоновке в Makefile стоит указать -L. -ldeque(для линковщика при записи в объектный файл). Добавляет текущий каталог в просматриваемый линковщиком так как библиотеки располагаются в линукс либо в каталоге /lib или /usr/lib. -ldeque собственно название библиотеки. А соответственно -Wl,-rpath это параметры для поиска во время выполнения, благодаря которым он ищет библиотеку в текущем каталоге при создании объектного файла. Для этого используется опция компилятора -Wl,option,optargs,... Расшифровываю: передать линковщику (-Wl) опцию option с аргументами optargs. В нашем случае мы передаем линковщику опцию -rpath с аргументом . (точка, текущий каталог). Также вместо -Wl, -rpath можно использовать переменную окружения LD_LIBRARY_PATH. Также хотелось бы отметить, что при реализации maindnm.c использовались системные вызовы:

При вызове dlopen происходит автоматическое разрешение зависимостей между библиотеками. Это значит, что если некая библиотека использует другую библиотеку, функция загрузит и ее. dlopen возвращает дескриптор, используемый для дальнейшей работы с библиотекой. Прототип функции выглядит так: void *dlopen(const char *file, int mode);

По дескриптору с помощью функции `dlsym` находятся адреса символов библиотеки. Функция принимает в качестве параметра дескриптор и строковое имя символа и возвращает искомый адрес: void *dlsym(void *restrict handle, const char *restrict name);

4.Листинг программы

```
//maindnm.c
#include "deque.h"
void help() {
    printf("PushBack: Press a\n");
    printf("PopBack: Press d\n");
    printf("PushFront: Press t\n");
    printf("PopFront: press r\n");
    printf("PRINT: Press p\n");
    printf("HELP: Press h\n");
    printf("SIZE: Press l\n");
    printf("EXIT: Press e\n");
}

int main() {
    void *libHandle;
    libHandle = dlopen("libdeque.so", RTLD_LAZY);
    if (!libHandle) {
        fprintf(stderr, "%s\n", dlerror());
        exit(1);
    }
    dblinkedlist*(*createdblinkedlist)() = dlsym(libHandle, "createdblinkedlist");
    void(*Delete_dblinkedlist)(dblinkedlist **l) = dlsym(libHandle, "Delete_dblinkedlist");
    void(*Printdblinkedlist)(dblinkedlist *list) = dlsym(libHandle, "Printdblinkedlist");
    void(*PushBack)(dblinkedlist *list, int value) = dlsym(libHandle, "PushBack");
    Node*(*PopBack)(dblinkedlist *l) = dlsym(libHandle, "PopBack");
    void(*PushFront)(dblinkedlist *l, int data) = dlsym(libHandle, "PushFront");
    Node*(*PopFront)(dblinkedlist *l) = dlsym(libHandle, "PopFront");
    dblinkedlist *l = (*createdblinkedlist)();
    char c;
    int a;
    help();
    while (true) {
```

```
    scanf("%c", &c);
    switch (c) {
        case 'a':
            printf("Your element\n");
            scanf("%d", &a);
            (*PushBack)(l, a);
        break;
        case 'd':
            (*PopBack)(l);
        break;
        case 't':
            printf("Your element\n");
            scanf("%d", &a);
            (*PushFront)(l,a);
        break;
        case 'e':
            (*Delete_dblinkedlist)(&l);
            dlclose(libHandle);
            return 0;
        break;
        case 'h':
            help();
        break;
        case 'p':
            (*Printdblinkedlist)(l);
        break;
        case 'l':
            printf("size:%lu\n",l->size);
        break;
        case 'r':
            (*PopFront)(l);
        break;
    }
}
```

```
    return 0;
}
//Makefile
CC = gcc
FLAGS = -g -Wall -Werror -Wextra

all: stt dnm

stt: mainstt.o libdeque.so
	$(CC) $(FLAGS) -o stt mainstt.o -L. -ldeque -Wl,-rpath,.

dnm: maindnm.o libdeque.so
	$(CC) $(FLAGS) -rdynamic -o dnm maindnm.o -ldl -Wl,-rpath,.

mainstt.o: mainstt.c
	$(CC) -c $(FLAGS) mainstt.c

maindnm.o: maindnm.c
	$(CC) -c $(FLAGS) maindnm.c

libdeque.so: deque.o
	$(CC) $(FLAGS) -shared -o libdeque.so deque.o

deque.o: deque.c
	$(CC) -c -fPIC $(FLAGS) deque.c

clean:
	rm *.o stt dnm *.so
```

5.Результат работы

```
nikita@nikita-HP:~/oc/lr5v3$ make
gcc -c -g -Wall -Werror -Wextra mainstt.c
gcc -c -fPIC -g -Wall -Werror -Wextra deque.c
```

```
gcc -g -Wall -Werror -Wextra -shared -o libdeque.so deque.o
gcc -g -Wall -Werror -Wextra -o stt mainstt.o -L. -ldeque -Wl,-rpath,.
gcc -c -g -Wall -Werror -Wextra maindnm.c
gcc -g -Wall -Werror -Wextra -rdynamic -o dnm maindnm.o -ldl -Wl,-rpath,.
nikita@nikita-HP:~/oc/lr5v3$ ./stt
PushBack: Press a
PopBack: Press d
PushFront: Press t
PopFront: press r
PRINT: Press p
HELP: Press h
SIZE: Press l
EXIT: Press e
a
Your element
3
p
null <-> 3  <-> null
a
Your element
6
p
null <-> 3  <-> 6  <-> null
t
Your element
9
p
null <-> 9  <-> 3  <-> 6  <-> null
r
d
d
p
List is empty
t
```

Your element

4

p

null <-> 4 <-> null

t

Your element

4

e

nikita@nikita-HP:~/oc/lr5v3$ ./dnm

PushBack: Press a

PopBack: Press d

PushFront: Press t

PopFront: press r

PRINT: Press p

HELP: Press h

SIZE: Press l

EXIT: Press e

a

Your element

3

a

Your element

4

a

Your element

5

p

null <-> 3 <-> 4 <-> 5 <-> null

t

Your element

2

t

Your element

1

```
p
null <-> 1  <-> 2  <-> 3  <-> 4  <-> 5  <-> null
r
p
null <-> 2  <-> 3  <-> 4  <-> 5  <-> null
r
p
null <-> 3  <-> 4  <-> 5  <-> null
e
```

6.Вывод

В чем же отличие подключения динамической библиотеки на стадии линковки и во время исполнения? На стадии линковки адреса данных функций подгружаются в объектный файл и при запуске программы мы вызываем данные функции, если же во время исполнения то мы вручную с помощью системных вызовов dlopen и тд. Получаем и определяем адреса во время исполнения программы. Основными плюсами использования динамических библиотек является экономия памяти и возможность использования данной библиотеки несколькими программами и соответственно при изменении в библиотеке чего-либо в этих программах эти изменения будут отображены. Экономия памяти происходит за счет того, что в исполняемый файл подгружаются адреса на наши функции в динамической библиотеке.