|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** №5 по курсу Операционные системы ­  студента группы М8О-207Б-18 Хренова Геннадия Николаевича , № по списку 28  Адреса www, e-mail, jabber, skype [khrenov.gena@yandex.ru](mailto:khrenov.gena@yandex.ru)  Работа выполнена: “7“ декабря 2019 г.  Преподаватель: Миронов Е. С.  Входной контроль знаний с оценкой  Отчёт сдан “ “ 20 г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Использование динамических библиотек
2. **Цель работы**: Создание динамических библиотек и программ, использующих их функции
3. **Задание** (*вариант №* 5): Требуется создать динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами: во время компиляции(на этапе линовки) и во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов
4. **Оборудование** (*лабораторное*):

ЭВМ Pentium , процессор 2.7 ГГц , имя узла сети с ОП 4096 МБ

НМД 2 ГБ. Терминал адрес . Принтер

Другие устройства

*Оборудование ПЭВМ студента, если использовалось:*

Процессор intel core i3 , ОП 3072 , НМД 50 ГБ. Монитор

Другие устройства

1. **Программное обеспечение** (*лабораторное*):

Операционная система семейства Unix , наименование версия

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства Unix , наименование Ubuntu версия 18.04LTS

Интерпретатор команд bash версия

Система программирования версия

Редактор текстов Emacs версия

Утилиты операционной системы

Прикладные системы и программы

Местонахождения и имена файлов программ и данных

1. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

Требуется реализовать очередь с целочисленным 32-битным типом.

Для выполнения работы необходимы следующие команды:

**void \*dlopen(const char \**filename*, int *flag*)** - загружает динамическую библиотеку, имя которой указано в строке *filename*, и возвращает прямой указатель на начало динамической библиотеки.

**RTLD\_LAZY(флаг)** -подразумевает разрешение неопределенных символов в виде кода, содержащегося в исполняемой динамической библиотеке.

**void \*dlsym(void \**handle*, char \**symbol*)**- использует указатель на динамическую библиотеку, возвращаемую dlopen, и оканчивающееся нулем символьное имя, а затем возвращает адрес, указывающий, откуда загружается этот символ. Если символ не найден, то возвращаемым значением dlsym является NULL.

**Dlerror()** - описывает большинство ошибок, происходящих при выполнении любых функций dl (dlopen, dlsym or dlclose), возвращает NULL, если не возникло ошибок с момента инициализации или его последнего вызова.

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

Очередь содержит общие функции

qCreate - создание

qPush - добавление

qPop - удаление

printq - вывод

Для подключения библиотеки на этапе линовки достаточно прописать в начале include и нужную библиотеку. Подгрузка через системные вызовы осуществляется через семество команд dl.

Тесты:

1)qCreate(&q);

qPush(&q, 2);

qPush(&q, 99);

printq(&q);

for (int i = 0; i < 5; i++) {

qPush(&q, i);

}

printq(&q);

printf("size: %dqSize(&q));

qPop(&q);

printf("size: %d\n", qSize(&q))

2) qCreate

qPush 12

qPush 2

printq

qPop

qPop

qPop

end

*Пункты 1-7 отчёта составляются* ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

**q.c**

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include "q.h"

void qCreate(queue\* q)

{

q->first = 0;

q->size = 0;

}

bool qEmpty(const queue\* q)

{

return q->size == 0;

}

int qSize(queue\* q)

{

return q->size;

}

int qTop(queue\* q)

{

if(q->size)

return q->data[q->first];

}

int qPush(queue\* q, int t)

{

if (q->size == POOL\_SIZE){

fprintf(stderr, "this queue is full\n" );

return 1;

}

q-> data[(q->first + q->size++) % POOL\_SIZE] = t;

return 0;

}

int qPop(queue\* q)

{

if (q->size > 0){

q-> size--;

} else{

fprintf(stderr,"there`s no elements in queue! ");

return 2;

}

for (int i = 0; i < q->size; i++){

q->data[i] = q->data[i+1];

}

return 0;

}

void printq(queue\* q)

{

for (int i = 0; i < q->size; i++){

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

}

printf("\n");

}

**q.h**

#ifndef \_QUEUE\_H\_

#define \_QUEUE\_H\_

#include <stdbool.h>

#include <stdint.h>

#define POOL\_SIZE 100

#define QueueElemType int32\_t

typedef struct

{

int first;

size\_t size;

int32\_t data[POOL\_SIZE];

}queue;

void qCreate(queue\* q);

bool qEmpty(const queue\* q);

int qSize(queue\* q);

int qTop(queue\* q);

int qPush(queue\* q, int t);

int qPop(queue\* q);

void printq(queue\* q);

#endif

**main.c(1)**

#include <stdio.h>

#include "q.h"

int main() {

queue q;

qCreate(&q);

qPush(&q, 2);

qPush(&q, 99);

printf("size: %d\n", qSize(&q));

printq(&q);

for (int i = 0; i < 5; i++) {

qPush(&q, i);

}

printq(&q);

printf("size: %d\n", qSize(&q));

qPop(&q);

printf("size: %d\n", qSize(&q));

}

**main.c(2)**

#include <stdio.h>

#include <string.h>

#include <dlfcn.h>

#include "q.h"

#define MAX\_SIZE 100

#define LIB\_IN\_USE "../libq.so"

void dl\_error() {

if (dlerror() != NULL) {

fprintf(stderr, "dlsym error\n");

return;

}

}

void exec\_method(char \*method, queue\* q, QueueElemType argument) {

void\* dl\_handle;

fprintf(stdout, "trying to do \"%s\"\n", method);

dl\_handle = dlopen(LIB\_IN\_USE, RTLD\_LAZY);

if (!dl\_handle) {

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

return;

}

if (strcmp(method, "qCreate") == 0) {

fprintf(stdout, "creating...\n");

void (\*func)(queue\*);

func = dlsym(dl\_handle, method);

dl\_error();

(\*func)(q);

} else if (strcmp(method, "qPush") == 0) {

fprintf(stdout, "pushing...\n");

int (\*func)(queue\*, QueueElemType);

func = dlsym(dl\_handle, method);

dl\_error();

if ((\*func)(q, argument)) {

fprintf(stderr, "queue is full\n");

}

} else if (strcmp(method, "qPop") == 0) {

fprintf(stdout, "popping...\n");

int (\*func)(queue\*);

func = dlsym(dl\_handle, method);

dl\_error();

if ((\*func)(q)) {

fprintf(stderr, "queue is empty\n");

}

} else if (strcmp(method, "printq") == 0) {

void (\*func)(queue\*, FILE\*);

func = dlsym(dl\_handle, method);

dl\_error();

(\*func)(q, stdout);

} else {

fprintf(stderr, "invalid input\n");

}

dlclose(dl\_handle);

return;

}

int main() {

char buf[MAX\_SIZE];

char func\_name[MAX\_SIZE];

int argument;

queue q;

for (;;) {

fprintf(stdout, "enter function: ");

fgets(buf, MAX\_SIZE, stdin);

if (!strncmp(buf, "end", 3)) {

break;

}

sscanf(buf, "%s %d", func\_name, &argument);

exec\_method(func\_name, &q, argument);

}

}

**9.Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |

1. Замечание автора по существу работы
2. Выводы: Библиотеки желательно использовать всегда, когда программу можно условно разбить на модули. Это структурирует код и делает его более понятным. Библиотеки бывают статические и динамические. Статические подключаются на этапе линовки, а динамические — во время выполнения программы. Из плюсов статической — её намного проще использовать. Динамическая лучше статической в плане памяти — динамическую программы могут использовать совместно, а статическую каждой программе приходится копировать к себе.

