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: [ Нелюбин В.С ]

Группа: М8О–206Б–19

Вариант: 18

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

* Создание динамических библиотек
* Создание программ, которые используют функции динамических библиотек

## Задание

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.

Задачи:

Подсчёт количества простых чисел на отрезке [A, B]; Рассчёт значения числа

е

**Общие сведения о программе**

Программа компилируется из файла main.c. Также используется заголовочные файлы: stdio.h, stdbool.h, stdlib.h, string.h, ctype.h, vecmd5.h . В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамический общий объект (общую библиотеку) из файла, имя которого указано в строке filename (завершается null) и возвращает указатель на загруженный объект.
2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти. Позволяет использовать содержимое динамического объекта.
3. **dlclose** – уменьшает счётчик ссылок на динамически загружаемый общий объект, на который ссылается handle. Если счётчик ссылок достигает нуля, то объект выгружается. Все общие объекты, которые были автоматически загружены при вызове dlopen() для объекта, на который ссылается handle, рекурсивно закрываются таким же способом.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы с димамическими объектами (dlsym, dlopen, dlclose).
2. Написать две простые библиотеки с разными реализациями заданных функций.
3. Написать две программы. Одна подключает динамический объект на этапе сборки, а другая может свободно переключаться между двумя библиотеками.

**Основные файлы программы**

**prg1.c:**

#include <stdio.h>

#include <stdlib.h>

int primeCount(int A,int B);

float E(int x);

int main(){

int num=255,ag1,ag2,i;

char\*inpt=malloc(sizeof(char)\*num);

while(NULL!=fgets (inpt, num, stdin)){

if(inpt[0]=='\n'){

break;

}

if(inpt[0]=='0'){

printf("change library here\n");

continue;

}

if(inpt[0]=='1'){

ag1=0;

ag2=0;

for(i=2;((inpt[i]>='0')&&(inpt[i]<='9'));i++){

ag1=ag1\*10+inpt[i]-'0';

}

i++;

for(i;((inpt[i]>='0')&&(inpt[i]<='9'));i++){

ag2=ag2\*10+inpt[i]-'0';

}

printf("there are %d prime numbers between %d and %d\n",primeCount(ag1,ag2),ag1,ag2);

continue;

}

if(inpt[0]=='2'){

ag1=0;

ag2=0;

for(i=2;((inpt[i]>='0')&&(inpt[i]<='9'));i++){

ag1=ag1\*10+inpt[i]-'0';

}

printf("e is %f with %d iterations\n",E(ag1),ag1);

continue;

}

}

return 0;

}

**test2.c**

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

int main( int argc, char \*argv[] ){

int (\*primeCount)(int, int);

float (\*E)(int);

if( argc != 2 ) {

printf("bad command line arguments :(\n");

exit(-1);

}

char mode=argv[1][0];

if((mode!='1')&&(mode!='2')){

printf("bad comment. arguments should be either 1 or 2\n");

exit(-1);

}

char\*lib1="libvar1.so";

char\*lib2="libvar2.so";

void\*copen;

if(mode=='1'){

copen=dlopen(lib1, RTLD\_LAZY);

mode='2';

}

else{

copen=dlopen(lib2,RTLD\_LAZY);

mode='1';

}

if(!copen){

printf("cannot open dynamic library\n");

exit(-1);

}

\*(void \*\*) (&E) = dlsym(copen, "E");

\*(void \*\*) (&primeCount) = dlsym(copen, "primeCount");

int num=255,ag1,ag2,i;

char\*inpt=malloc(sizeof(char)\*num);

while(NULL!=fgets (inpt, num, stdin)){

if(inpt[0]=='\n'){

break;

}

if(inpt[0]=='0'){

printf("change library here\n");

if(mode=='1'){

dlclose(copen);

copen=dlopen(lib1, RTLD\_LAZY);

mode='2';

}

else{

dlclose(copen);

copen=dlopen(lib2,RTLD\_LAZY);

mode='1';

}

if(copen==0){

printf("cannot open dynamic library\n");

exit(-1);

}

\*(void \*\*) (&E) = dlsym(copen, "E");

\*(void \*\*) (&primeCount) = dlsym(copen, "primeCount");

continue;

}

if(inpt[0]=='1'){

ag1=0;

ag2=0;

for(i=2;((inpt[i]>='0')&&(inpt[i]<='9'));i++){

ag1=ag1\*10+inpt[i]-'0';

}

i++;

for(i;((inpt[i]>='0')&&(inpt[i]<='9'));i++){

ag2=ag2\*10+inpt[i]-'0';

}

printf("there are %d prime numbers between %d and %d\n",primeCount(ag1,ag2),ag1,ag2);

continue;

}

if(inpt[0]=='2'){

ag1=0;

ag2=0;

for(i=2;((inpt[i]>='0')&&(inpt[i]<='9'));i++){

ag1=ag1\*10+inpt[i]-'0';

}

printf("e is %f with %d iterations\n",E(ag1),ag1);

continue;

}

}

return 0;

}

**var1.c**

int primeCount(int A,int B){

int rez=0,i,j;

for(i=A;i<=B;i++){

for(j=2;j<i;j++){

if((i%j)==0){

j=0;

break;

}

}

if(j){

rez++;

}

}

return rez;

}

float E(int x){

double b,ex=1;

b=1/(double)x +1;

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

ex=ex\*b;

}

return (float)ex;

}

**var2.c**

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

void fil(int\*ar,int pos){

int ee=pos/(8\*sizeof(int));

int id=pos-ee\*8\*sizeof(int);

ar[ee]=ar[ee]|(1<<id);

}

int chek(int\*ar,int pos){

int ee=pos/(8\*sizeof(int));

int id=pos-ee\*8\*sizeof(int);

return !((ar[ee])&(1<<id));

}

int primeCount(int A,int B){

int rez=0,i,j;

for(i=0;sizeof(int)\*8\*i<B;i++);

int\*tbl=malloc(sizeof(int)\*i);

for(j=0;j<i;j++){

tbl[j]=0;

};

for(j=2;j<=B;j++){

if(chek(tbl,j)){

if(j>1){

for(i=j\*2;i<=B;i+=j){

fil(tbl,i);

}

}

if(j>=A){

rez++;

}

}

}

free(tbl);

return rez;

}

float E(int x){

double l=1,rez=1;

for(int n=1;n<x;n++){

l/=n;

rez+=l;

}

return (float)rez;

}

**Пример работы**

aaaaa@TRASHBox:~/Desktop/OC/laba5$ export LD\_LIBRARY\_PATH=.:$LD\_LIBRARY\_PATH

aaaaa@TRASHBox:~/Desktop/OC/laba5$ ./var1.out

1 5 10

there are 2 prime numbers between 5 and 10

2 2

e is 2.250000 with 2 iterations

aaaaa@TRASHBox:~/Desktop/OC/laba5$ ./var2.out

1 90 99

there are 1 prime numbers between 90 and 99

2 2

e is 2.000000 with 2 iterations

aaaaa@TRASHBox:~/Desktop/OC/laba5$ ./switchy.out 1

1 228 420

there are 32 prime numbers between 228 and 420

2 5

e is 2.488320 with 5 iterations

0

change library here

1 360 1337

there are 145 prime numbers between 360 and 1337

2 5

e is 2.708333 with 5 iterations

aaaaa@TRASHBox:~/Desktop/OC/laba5$

**Вывод**

Крайне полезный приёмы. Возможность менять реализации функций по время работы может пригодиться в личных проектах. Так же может упростить работу с SDL и другими библиотеками, использование которых вызывало много проблем в прошлом.