Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу
«Операционные системы»

Тема работы
“ Работа с динамическими библиотеками”

Студент: Меджидли Махмуд Ибрагим оглы
Группа: М8О-208Б-20
Вариант: 16
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/loshadkaigogo/OS

Постановка задачи

Задача: создать динамические библиотеки, вычисляющие:

А) Количество простых чисел между двумя неотрицательными числами (PrimeCount).

Б) НОД 2 чисел (GCF).

Первая библиотека:

I) Находит (А) наивным алгоритмом: проверяет делимость текущего числа на все предыдущие числа.

II) Находит (Б) алгоритмом Евклида.

Вторая библиотека:

I) Для нахождения (А) применяется решето Эратосфена.

II) Находит (Б) алгоритмом наивным алгоритмом: пытается разделить A и B на все числа, меньшие A.

Присоединять библиотеки 2 способами:

1) На этапе компиляции.

А) При вводе команды 1 A B вычислить значение функции PrimeCount(A, B) и вывести его на консоль.

Б) При вводе команды 2 A B вычислить значение функции GCF(A, B) и вывести его на консоль.

2) Во время исполнения программы.

А) При вводе цифры 0 сменить реализацию контрактов функций.

Б) При вводе команды 1 A B вычислить значение функции PrimeCount(A, B) и вывести его на консоль.

В) При вводе команды 2 A B вычислить значение функции GCF(A, B) и вывести его на консоль.

Общие сведения о программе

В программе используются следующие библиотеки:

- <stdio.h> - для вывода информации на консоль
- <stdlib.h> - для использования основных функций C.
- <dlfcn.h> - для использования функций для работы с динамическими библиотеками.
- <stdbool.h > - для использования макросов true и false.

В задании используются такие команды и строки, как:

- **extern void swap(int*, int*)** – определяет функцию swap как глобальную, доступную во всех модулях. Для этого используется спецификатор extern (также и с функциями PrimeCount и GCF).
- **dlopen(const char* filename, int flags)** – загружает динамическую библиотеку, путь к которой указан в строке filename, с одним обязательным флагом (в моём случае RTLD_LAZY – то есть неопределённые символы разрешаются в виде кода, содержащегося в исполняемой динамической библиотеке). Возвращает указатель на библиотеку, а в случае ошибки – NULL.
- **dlerror()** – получение понятного пользователю текста. Возвращает NULL, если с момента инициализации или последнего вызова не произошло ошибок при выполнении функций dl.
- **void (*swap)(int*, int*);**

  swap = dlsym(handle, «swap») – указателю swap на функцию, имеющую тип void и принимающую 2 целочисленных указателя присваивается адрес, по которому загружается одноимённая функция

swap из библиотеки с дескриптором handle. Тоже самое с функциями PrimeCount и GCF.

- **fprintf(stderror, «%s\n», dlerror()) –** вывод в стандартный поток ошибок результат команды dlerror().
- **dlclose(handle) –** уменьшает на единицу счётчик ссылок на указатель динамической библиотеки handle. Если нет других загруженных библиотек, использующих её символы или её счётчик равен 0, то динамическая библиотека выгружается из памяти.

Общий метод и алгоритм решения

Реализуются 2 библиотеки с заданными алгоритмами вычисления функций. В первой программе с помощью ключевого слова extern функции swap, PrimeCount и GCF импортируются из присоединённой на этапе компиляции библиотеки.

Во второй программе для загрузки библиотеки, присоединённой на этапе выполнения используется команда dlopen. Когда библиотека выгружается или сменяется контекст (а тогда по-любому она выгружается), то используется команда dlclose, так же как и в конце программы. Для загрузки определённых функций из программы (в моём случае всех) используется команда dlsym.

Исходный код

1) Файл functions1.c

```c
#include <stdlib.h>
#include <stdbool.h>

extern void swap(int* a, int* b);
extern int PrimeCount(int A, int B);
extern int GCF (int A, int B);

void swap (int *a, int *b)
{
        *a += *b;
        *b = *a - *b;
        *a -= *b;
}

int PrimeCount (int A, int B)
{
```

```
        if ((A < 0) || (B < 0))
        {
                return -1;
        }
        if (A > B)
        {
                swap(&A, &B);
        }
        int amount = 0;
        bool prime = true;
        for (int i = A; i <= B; i++)
        {
                if (i > 1)
                {
                        for (int j = 2; j * j <= i; j++)
                        {
                                if (!(i % j))
                                {
                                        prime = false;
                                        break;
                                }
                        }
                        if (prime)
                        {
                                amount++;
                        }
                        prime = true;
                }
        }
        return amount;
}

int GCF (int A, int B)
{
        A = abs(A);
        B = abs(B);
        if (A < B)
        {
                swap(&A, &B);
        }
        while (B)
        {
                A %= B;
                swap(&A, &B);
        }
        return A;
}
```

2) Файл functions2.c

```c
#include <stdlib.h>
#include <stdbool.h>

extern void swap(int* a, int* b);
extern int PrimeCount(int A, int B);
extern int GCF(int A, int B);

void swap (int *a, int *b)
{
        *a += *b;
        *b = *a - *b;
        *a -= *b;
}

int PrimeCount (int A, int B)
{
        if ((A < 0) || (B < 0))
        {
                return -1;
        }
        if (A > B)
        {
                swap(&A, &B);
        }
        bool* sieve = (bool*)calloc((B + 1), sizeof(bool));
        int amount = 0;
        sieve[0] = true;
        if (B > 0)
        {
                sieve[1] = true;
        }
        for (int i = 2; i <= B; i++)
        {
                if (sieve[i])
                {
                        continue;
                }
                else
                {
                        if (i >= A)
                        {
                                amount++;
                        }
                        for (int j = i * 2; j <= B; j += i)
                        {
                                sieve[j] = true;
                        }
                }
        }
        free(sieve);
        return amount;
}

int GCF (int A, int B)
{
        A = abs(A);
        B = abs(B);
```

```
        if (A > B)
        {
                swap(&A, &B);
        }
        int g = 1;
        for (int i = 2; i <= A; i++)
        {
                if ((!(A % i)) && (!(B % i)))
                {
                        g = i;
                }
        }
        return g;
}
```

3) Файл task1.c

```
#include <stdio.h>

extern void swap(int*, int*);
extern int PrimeCount(int, int);
extern int GCF(int, int);

int main()
{
        int ind, A, B;
        while ((scanf("%d", &ind)) != EOF)
        {
                if (ind == 1)
                {
                        scanf("%d %d", &A, &B);
                        int result = PrimeCount(A, B);
                        if (result == -1)
                        {
                                printf("A and B must be non-negative numbers\n");
                        }
                        else
                        {
                                printf("%d\n", result);
                        }
                }
                else if (ind == 2)
                {
                        scanf("%d %d", &A, &B);
                        printf("%d\n", GCF(A, B));
                }
        }
        return 0;
}
```

4) Файл task2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include <stdbool.h>

const char* path1 = "libfunctions1.so";
const char* path2 = "libfunctions2.so";
void (*swap)(int*, int*);
```

```
int (*PrimeCount)(int, int);
int (*GCF)(int, int);
void *handle = NULL;
bool first = true;

void Load()
{
        const char *path;
        if(first)
        {
                path = path1;
        }
        else
        {
                path = path2;
        }
        handle = dlopen(path, RTLD_LAZY);
        if(!handle)
        {
                fprintf(stderr, "%s\n", dlerror());
                exit(EXIT_FAILURE);
        }
}

void Unload()
{
        dlclose(handle);
}

void loadContext()
{
        Load();
        swap = dlsym(handle, "swap");
        PrimeCount = dlsym(handle, "PrimeCount");
        GCF = dlsym(handle, "GCF");
        char* error;
        if((error = dlerror()))
        {
                fprintf(stderr, "%s\n", error);
                exit(EXIT_FAILURE);
        }
}

void Change()
{
        Unload();
        first ^= true;
```

```
        loadContext();
}

int main()
{
        loadContext();
        int ind = 0;
        while (scanf("%d", &ind) != EOF)
        {
                if(!ind)
                {
                        Change();
                        printf("Contract was changed. ");
                        if(first)
                        {
                                printf("Now context is first\n");
                        }
                        else
                        {
                                printf("Now context is second\n");
                        }
                        continue;
                }
                if(ind == 1)
                {
                        int A, B, result;
                        scanf("%d %d", &A, &B);
                        result = PrimeCount(A, B);
                        if (result == -1)
                        {
                                printf("A and B must be non-negtive whole numbers\n");
                        }
                        else
                        {
                                printf("%d\n", result);
                        }
                }
                else if (ind == 2)
                {
                        int A, B;
                        scanf("%d %d", &A, &B);
                        printf("%d\n", GCF(A, B));
                }
        }
        Unload();
}
```

Демонстрация работы программы

Программа запускается так:

gcc -fPIC -c functions1.c -o functions1.o

gcc -fPIC -c functions2.c -o functions2.o

gcc -shared -o libfunctions1.so functions1.o

gcc -shared -o libfunctions2.so functions2.o

gcc task1.c -L. -lfunctions1 -o main1.1 -Wl,-rpath -Wl,.

(подключение библиотеки на этапе компиляции; компилятор будет искать библиотеку libfunctions1.so по адресу . – то есть в текущей директории)

gcc task1.c -L. -lfunctions2 -o main1.2 -Wl,-rpath -Wl,.

gcc task2.c -ldl -o main2 -Wl,-rpath -Wl,.

(подключение библиотеки на этапе выполнения; компилятор будет искать библиотеки, упомянутые в программе по адресу . – то есть в текущей директории)

Тест 1 – библиотека с 1 реализацией функций подсоединяется к прогрмме task1.c на этапе компиляции

Количество простых чисел между 3 и 7, как и между 7 и 3 равняется трём (это 3, 5 и 7).

НОД чисел 43 и 71 – 1 (числа взаимно-простые).

НОД чисел 44 и 1089 – 11.

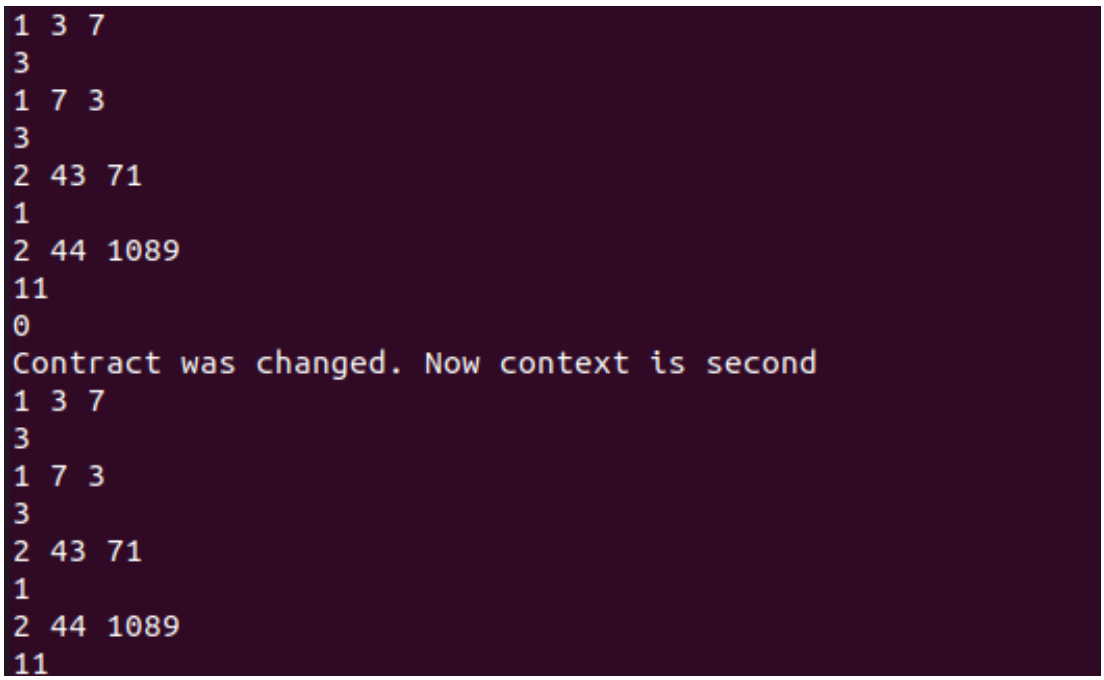
./main1.1

Тест 2 – библиотека со 2 реализацией функций подсоединяется к программе task1.c на этапе компиляции

Результат аналогичен

Тест 3 – библиотеки с 1 и 2 реализациями функций подсоединяются к программе task2.c на этапе выполнения. Во время выполнения программы также сменится контекст.

Результаты такие же, как и с присоединением библиотек на этапе компиляции



Выводы

Несмотря на многочисленное применение команды gcc в операционной системе Ubuntu, я не имел опыта в присоединении библиотек, теперь этот минус устранён.
Также узнал, что при присоединении библиотеки на этапе компиляции нет необходимости помещать библиотеку в один из специальных каталогов, модифицировать переменные окружения и выполнять "ldconfig".