

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: Никулин Кристиан Ильич
Группа: М8О-208Б-21
Вариант: 26
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Постановка задачи

Цель работы

Целью является приобретение практических навыков в:

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

Задание

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
1. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

Вариант 26

4	Подсчёт наибольшего общего делителя для двух натуральных чисел	Int GCF(int A, int B)	Алгоритм Евклида	Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.
9	Отсортировать целочисленный массив	Int * Sort(int * array)	Пузырьковая сортировка	Сортировка Хоара

Общие сведения о программе

Для выполнения данной лабораторной работы я использую 4 файла: первые два – re1.cpp и re2.cpp являются исходным кодом для наших динамических библиотек. Файлы prog1.cpp и prog2.cpp являются двумя программами, которые нужно было реализовать по заданию. Prog1.cpp является программой, к которой библиотека подгружается на этапе компиляции, prog2.cpp является программой, к которой библиотека подключается непосредственно во время работы программы.

Также для удобства компиляции всех программ я создал Makefile со следующим набором команд:

all:

g++ -fPIC -c re1.cpp -o d1.o

g++ -shared d1.o -o libd1.so

g++ -fPIC -c re2.cpp -o d2.o

g++ -shared d2.o -o libd2.so

static1: prog1.cpp

g++ prog1.cpp -L. -ld1 -o main1 -Wl,-rpath -Wl,.

static2: prog1.cpp

g++ prog1.cpp -L. -ld2 -o main2 -Wl,-rpath -Wl,.

dynamic: prog2.cpp

g++ prog2.cpp -L. -ld1 -o main3 -Wl,-rpath -Wl,.

clean:

rm -rf *.o *.so main1 main2 main3

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 непрозрачный описатель на загруженный объект.
2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти(указывается одним из аргументов).
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

В самом начале выполнения лабораторной работы я реализовал две библиотеки: `re1.cpp` и `re2.cpp`. В библиотеке `re1.cpp` реализованы алгоритм Евклида и пузырьковая сортировка. В библиотеке `re2.cpp` реализованы те же функции, но другими методами – обычный алгоритм, сортировка Хоара.

Далее в файле `prog1.cpp` я реализовал обычное считывание команды при помощи функции `scanf`. Если вводится 1, то считается алгоритм Евклида. Если вводится 2, то вводится и сортируется массив. Если вводится 3, то программа завершается.

`Prog2.cpp` аналогична `prog1.cpp`, но с небольшими изменениями. Для выбора динамической библиотеки считываем команду. Если это не 0, 1, 2, 3, то прошу ввести правильную команду. Если эта команда 0, то программа меняет библиотеки. Если команда 1, то считается НОД двух чисел. Если команда 2, то сортируется массив. Если команда 3, то программа завершается.

Основные файлы программы

re1.cpp:

```
#include <cmath>
#include <vector>
#include <cstdlib>
#include <iostream>

using namespace std;

extern "C" int GCF(int a, int b);
extern "C" int *Sort(int *array, int first, int last);

int GCF(int a, int b)
{
    cout << "Алгоритм Евклида\n";
    int t;
    while (b != 0)
    {
        t = b;
        b = a % b;
    }
}
```

```

        a = t;
    }
    return a;
}

int *Sort(int *array, int first, int last)
{
    cout << "Сортировка пузырьком\n";
    int n = 5;
    for (int i = 0; i < n - 1; ++i)
    {
        for (int j = 0; j < n - i - 1; ++j)
        {
            if (array[j] > array[j + 1])
            {
                int t;
                t = array[j + 1];
                array[j + 1] = array[j];
                array[j] = t;
            }
        }
    }
    return array;
}

```

re2.cpp:

```

#include <cmath>
#include <vector>
#include <cstdlib>
#include <iostream>

using namespace std;

extern "C" int GCF(int a, int b);
extern "C" int *Sort(int *array, int first, int last);

int GCF(int a, int b)
{
    cout << "Обычный алгоритм\n";
    int t = min(a, b);
    while (t > 1)
    {
        if (a % t == 0 && b % t == 0)
        {
            return t;
        }
        t--;
    }
    return t;
}

```

5

```

// Сортировка Хоара
int *Sort(int *array, int first, int last)
{
    int i = first, j = last;
    double tmp, x = array[(first + last) / 2];

    do
    {
        while (array[i] < x)
        {
            i++;
        }
        while (array[j] > x)
        {
            j--;
        }
        if (i <= j)
        {
            if (i < j)
            {
                tmp = array[i];
                array[i] = array[j];
                array[j] = tmp;
            }
            i++;
            j--;
        }
    } while (i <= j);

    if (i < last)
    {
        Sort(array, i, last);
    }
    if (first < j)
    {
        Sort(array, first, j);
    }
    return array;
}

```

prog1.cpp:

```

#include <stdio.h>
#include <vector>
#include <iostream>

using namespace std;

extern "C" int GCF(int a, int b);
6

```

```

extern "C" int *Sort(int *array, int first, int last);

int main()
{
    int command;
    printf("1 - НОД, 2 - Сортировка, 3 - выход:\n");
    scanf("%d", &command);

    while (1)
    {
        if (command == 1)
        {
            int a, b;
            cin >> a >> b;

            if ((a < 1) || (b < 1))
            {
                printf("Input error\n");
            }
            else
            {
                printf("%d\n", GCF(a, b));
            }
        }
        if (command == 2)
        {
            int n = 5;
            int array[5];
            for (int i = 0; i < n; ++i)
            {
                cin >> array[i];
            }
            int *arr = Sort(array, 0, n - 1);

            for (int i = 0; i < n; ++i)
            {
                printf("%d ", arr[i]);
            }
            printf("\n");
        }
        if (command == 3)
        {
            break;
        }
        scanf("%d", &command);
    }
    return 0;
}

```

prog2.cpp:

```

#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include <vector>
#include <iostream>

using namespace std;

int main()
{
    void *h = NULL;
    int (*GCF)(int a, int b);
    int (*Sort)(int *array, int first, int last);
    int lib;
    printf("0 - изменить библиотеку, 1 - re1 , 2 - re2 , 3 - выход:\n");
    scanf("%d", &lib);

    while ((lib != 1) && (lib != 2))
    {
        printf("Input error, try again:\n");
        scanf("%d", &lib);
    }
    if (lib == 1)
    {
        h = dlopen("./libd1.so", RTLD_LAZY);
    }
    if (lib == 2)
    {
        h = dlopen("./libd2.so", RTLD_LAZY);
    }

    GCF = (int (*)(int, int))dlsym(h, "GCF");
    Sort = (int (*)(int *, int, int))dlsym(h, "Sort");
    unsigned command;
    printf("1 - НОД, 2 - Сортировка:\n");
    scanf("%d", &command);
    while (1)
    {
        if (command == 0)
        {
            if (lib == 1)
            {
                dlclose(h);
                h = dlopen("./libd2.so", RTLD_LAZY);

                GCF = (int (*)(int, int))dlsym(h, "GCF");
                Sort = (int (*)(int *, int, int))dlsym(h, "Sort");
                lib = 2;
                printf("re1 --> re2\n");
                scanf("%d", &command);
            }
        }
    }
}

```



```

        continue;
    }
    else
    {
        dlclose(h);
        h = dlopen("./libd1.so", RTLD_LAZY);

        GCF = (int (*)(int, int))dlsym(h, "GCF");
        Sort = (int (*)(int *, int, int))dlsym(h, "Sort");
        lib = 1;
        printf("re2 --> re1\n");
        scanf("%d", &command);
        continue;
    }
}
if (command == 1)
{
    int a, b;
    scanf("%d%d", &a, &b);
    if ((a < 1) || (b < 1))
    {
        printf("Input error\n");
    }
    else
    {
        printf("%d\n", GCF(a, b));
    }
}
if (command == 2)
{
    int n = 5;
    int array[n];
    for (int i = 0; i < n; ++i)
    {
        cin >> array[i];
    }
    int *arr = Sort(array, 0, n - 1);

    for (int i = 0; i < n; ++i)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
if (command == 3)
{
    break;
}
scanf("%d", &command);
}

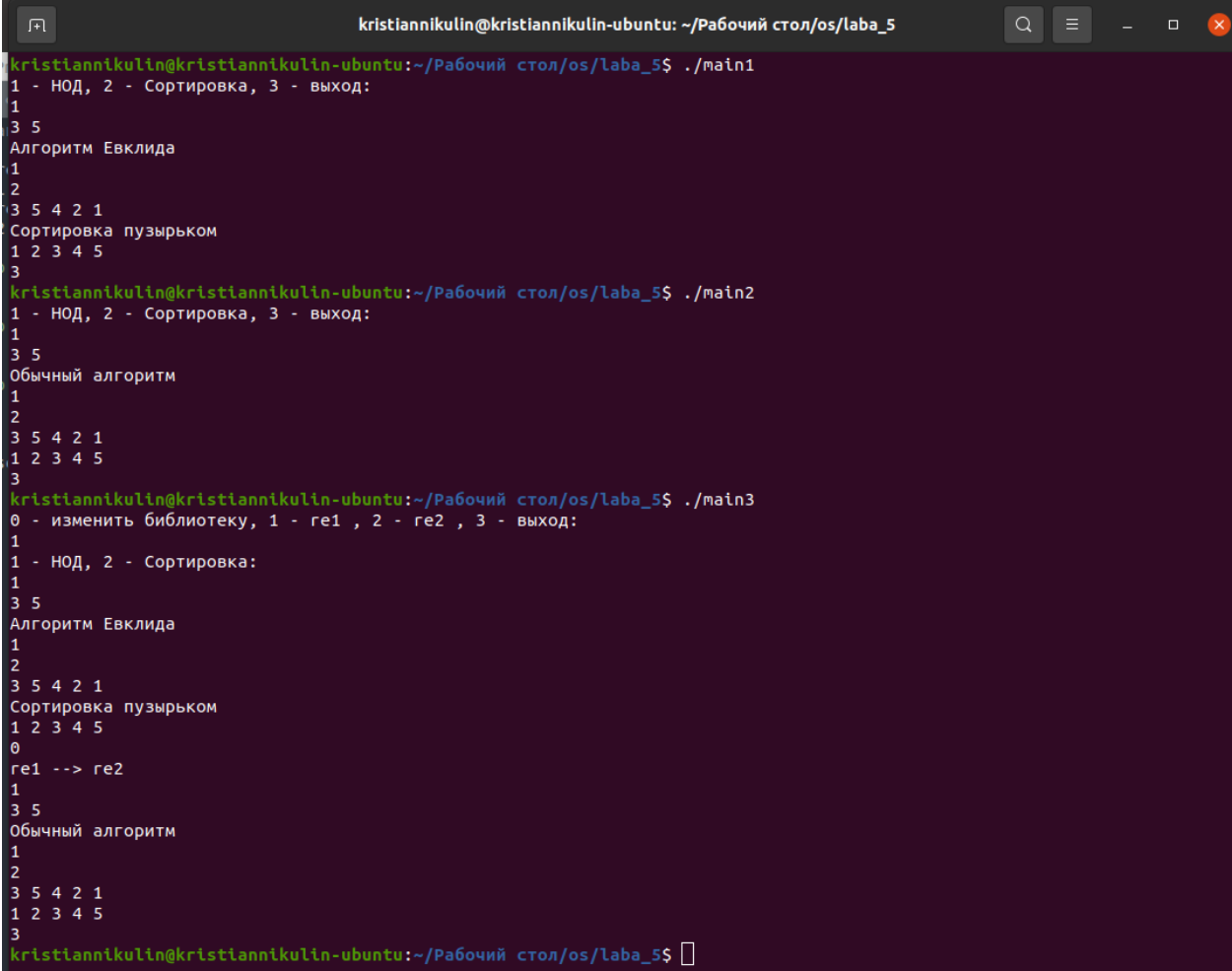
```

```

    dlclose(h);
    return 0;
}

```

Пример работы



```

kristiannikulin@kristiannikulin-ubuntu: ~/Рабочий стол/os/laba_5
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_5$ ./main1
1 - НОД, 2 - Сортировка, 3 - выход:
1
3 5
Алгоритм Евклида
1
2
3 5 4 2 1
Сортировка пузырьком
1 2 3 4 5
3
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_5$ ./main2
1 - НОД, 2 - Сортировка, 3 - выход:
1
3 5
Обычный алгоритм
1
2
3 5 4 2 1
1 2 3 4 5
3
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_5$ ./main3
0 - изменить библиотеку, 1 - ge1 , 2 - ge2 , 3 - выход:
1
1 - НОД, 2 - Сортировка:
1
3 5
Алгоритм Евклида
1
2
3 5 4 2 1
Сортировка пузырьком
1 2 3 4 5
0
ge1 --> ge2
1
3 5
Обычный алгоритм
1
2
3 5 4 2 1
1 2 3 4 5
3
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_5$ 

```

Выводы

Данная лабораторная работа научила меня пользоваться dl-функциями, благодаря реализации исполняемых файлов по заданию, я закрепил навык работы с динамическими библиотеками и узнал их отличие от статических библиотек.