Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №4 по курсу

«Операционные системы»

Группа: М8О-209БВ-24

Студент: Котик М. Н.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 11.12.25

Москва, 2024

Постановка задачи

Вариант 26.

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

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

В конечном итоге, в лабораторной работе необходимо получить следующие части:

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа (*программа №1*), которая используют одну из библиотек, используя информацию полученные на этапе компиляции;
- Тестовая программа (*программа №2*), которая загружает библиотеки, используя только их относительные пути и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для *программы №2*). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
2. «1 arg1 arg2 … argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;

«2 arg1 arg2 … argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.
Требуется создать динамические библиотеки, реализующие две функции:

1. **GCF(int A, int B)** - вычисление наибольшего общего делителя
2. *Sort(int array, int size)** - сортировка массива целых чисел

Каждая функция должна иметь две реализации:

- **Реализация 1 для GCF:** Алгоритм Евклида
- **Реализация 2 для GCF:** Наивный алгоритм (перебор)
- **Реализация 1 для Sort:** Пузырьковая сортировка
- **Реализация 2 для Sort:** Быстрая сортировка (Хоара)

Необходимо разработать две программы:

1. **Программа №1:** использует библиотеку с линковкой на этапе компиляции
2. **Программа №2:** динамически загружает библиотеки во время выполнения с возможностью переключения между ними командой "0"

Пользовательский интерфейс должен поддерживать команды:

- "0" - переключить библиотеку (только для программы №2)
- "1 A B" - вычислить НОД чисел A и B
- "2 size n1 n2 ..." - отсортировать массив заданного размера

3.

Общий метод и алгоритм решения

Использованные системные вызовы и функции:

Для работы с динамическими библиотеками:

1. **dlopen(const char filename, int flags)* - загрузка динамической библиотеки в память
2. **dlsym(void *handle, const char symbol)* - получение адреса функции по её имени
3. **dlclose(void handle)* - выгрузка библиотеки из памяти
4. **dlerror(void)** - получение сообщения об ошибке

Для статической линковки:

1. Компоновщик автоматически разрешает зависимости на этапе сборки
2. Библиотека загружается при старте программы через механизмы ОС

Дополнительные системные вызовы (видимые в strace):

1. **openat(AT_FDCWD, "filename", O_RDONLY|O_CLOEXEC)** - открытие файла библиотеки
2. **mmap(NULL, size, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, fd, 0)** - отображение библиотеки в память
3. **munmap(addr, size)** - освобождение отображенной памяти
4. **read(fd, buf, count)** - чтение данных
5. **write(fd, buf, count)** - запись данных
6. **malloc(size)** / **free(ptr)** - управление динамической памятью

Алгоритмическая часть:

Реализация 1 (libfuncs1.so):

- **GCF:** Алгоритм Евклида - итеративное вычисление НОД через остаток от деления
- **Sort:** Пузырьковая сортировка - попарное сравнение и обмен соседних элементов

Реализация 2 (libfuncs2.so):

- **GCF:** Наивный алгоритм - перебор делителей от минимального числа вниз
- **Sort:** Быстрая сортировка (Хоара) - рекурсивное разделение массива относительно опорного элемента

Архитектура программы:

1. **Общий заголовочный файл (libfuncs.h):** определяет интерфейс функций
2. **Две библиотеки:** libfuncs1.so и libfuncs2.so с разными реализациями
3. **Program1:** использует первую библиотеку через статическую линковку
4. **Program2:** динамически загружает библиотеки, позволяет переключаться между ними

Код программы

libfuncs.h

```
#ifndef LIBFUNCS_H
#define LIBFUNCS_H

int GCF(int A, int B);

int *Sort(int *array, int size);

#endif
```

libfuncs1.c

```
#include <stdlib.h>

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

   while (A != 0 && B != 0)
   {
      if (A > B)
      {
         A = A % B;
      }
      else
      {
         B = B % A;
      }
   }
   return A + B;
}

int *Sort(int *array, int size)
{
   if (!array || size <= 0)
      return NULL;
```

```
    int *result = malloc(size * sizeof(int));
    for (int i = 0; i < size; i++)
        result[i] = array[i];

    for (int i = 0; i < size - 1; i++)
    {
        for (int j = 0; j < size - i - 1; j++)
        {
            if (result[j] > result[j + 1])
            {
                int temp = result[j];
                result[j] = result[j + 1];
                result[j + 1] = temp;
            }
        }
    }
    return result;
}
```

libfuncs2.c

```
#include <stdlib.h>

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

    int min = (A < B) ? A : B;
    for (int i = min; i >= 1; i--)
    {
        if (A % i == 0 && B % i == 0)
        {
            return i;
        }
    }
    return 1;
}

void quick_sort(int *arr, int low, int high)
{
    if (low < high)
    {
        int pivot = arr[high];
        int i = low - 1;

        for (int j = low; j < high; j++)
```

```
        {
            if (arr[j] < pivot)
            {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }

        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;

        int pi = i + 1;
        quick_sort(arr, low, pi - 1);
        quick_sort(arr, pi + 1, high);
    }
}

int *Sort(int *array, int size)
{
    if (!array || size <= 0)
        return NULL;

    int *result = malloc(size * sizeof(int));
    for (int i = 0; i < size; i++)
        result[i] = array[i];

    quick_sort(result, 0, size - 1);
    return result;
}
```

program1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "libfuncs.h"

void print_array(int *arr, int size)
{
    printf("[");
    for (int i = 0; i < size; i++)
    {
        printf("%d", arr[i]);
        if (i < size - 1)
        {
```

```
            printf(", ");
        }
    }
    printf("]\n");
}

int main()
{
    printf("Программа 1: Использование библиотеки при линковке\n");
    printf("Команды:\n");
    printf("  0 - справка\n");
    printf("  1 A B - НОД чисел A и B\n");
    printf("  2 size n1 n2 ... - сортировка массива\n");
    printf("  q - выход\n\n");

    char line[256];
    while (1)
    {
        printf("> ");
        if (!fgets(line, sizeof(line), stdin))
            break;

        line[strcspn(line, "\n")] = 0;

        if (line[0] == 'q')
            break;

        if (line[0] == '0')
        {
            printf("Справка:\n");
            printf("  1 A B - вычисление НОД чисел A и B\n");
            printf("  2 size n1 n2 ... - сортировка массива\n");
            printf("      size - размер массива\n");
            printf("      n1 n2 ... - элементы массива\n");
        }
        else if (line[0] == '1')
        {
            int a, b;
            if (sscanf(line + 1, "%d %d", &a, &b) == 2)
            {
                printf("НОД(%d, %d) = %d\n", a, b, GCF(a, b));
            }
            else
            {
                printf("Ошибка формата. Используйте: 1 A B\n");
            }
        }
        else if (line[0] == '2')
        {
            int size;
            char *ptr = line + 2;
```

```
if (sscanf(ptr, "%d", &size) != 1 || size <= 0)
{
   printf("Ошибка: укажите корректный размер массива (> 0)\n");
   continue;
}

while (*ptr && *ptr != ' ')
   ptr++;
while (*ptr == ' ')
   ptr++;

int *arr = malloc(size * sizeof(int));
if (arr == NULL)
{
   printf("Ошибка: не удалось выделить память\n");
   continue;
}

int i;
for (i = 0; i < size; i++)
{
   if (*ptr == '\0')
   {
      break;
   }
   if (sscanf(ptr, "%d", &arr[i]) != 1)
   {
      break;
   }
   while (*ptr && *ptr != ' ')
      ptr++;
   while (*ptr == ' ')
      ptr++;
}

if (i != size)
{
   printf("Ошибка: указано %d элементов, но требуется %d\n", i, size);
   free(arr);
   continue;
}

printf("Исходный массив: ");
print_array(arr, size);

int *sorted = Sort(arr, size);
if (sorted)
{
   printf("Отсортированный массив: ");
   print_array(sorted, size);
```

```
            free(sorted);
        }
        else
        {
            printf("Ошибка при сортировке\n");
        }

        free(arr);
    }
    else
    {
        if (line[0] != '\0')
        {
            printf("Неизвестная команда. Доступные команды: 0, 1, 2, q\n");
        }
    }
}

return 0;
}
```

program2.c

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

typedef int (*GCF_func)(int, int);
typedef int *(*Sort_func)(int *, int);

void *lib_handle = NULL;
GCF_func gcf_func = NULL;
Sort_func sort_func = NULL;

int load_library(const char *libname)
{
    if (lib_handle)
        dlclose(lib_handle);

    lib_handle = dlopen(libname, RTLD_LAZY);
    if (!lib_handle)
    {
        printf("Ошибка загрузки: %s\n", dlerror());
        return 0;
    }

    gcf_func = (GCF_func)dlsym(lib_handle, "GCF");
    sort_func = (Sort_func)dlsym(lib_handle, "Sort");
```

```
    if (!gcf_func || !sort_func)
    {
        printf("Ошибка загрузки функций: %s\n", dlerror());
        dlclose(lib_handle);
        return 0;
    }

    printf("Загружена библиотека: %s\n", libname);
    return 1;
}

void print_array(int *arr, int size)
{
    printf("[");
    for (int i = 0; i < size; i++)
    {
        printf("%d", arr[i]);
        if (i < size - 1)
        {
            printf(", ");
        }
    }
    printf("]\n");
}

int main()
{
    if (!load_library("./libfuncs1.so"))
    {
        return 1;
    }

    printf("Программа 2: Динамическая загрузка библиотек\n");
    printf("Команды:\n");
    printf("  0 - переключить библиотеку (1/2)\n");
    printf("  1 A B - НОД чисел A и B\n");
    printf("  2 size n1 n2 ... - сортировка массива\n");
    printf("  q - выход\n\n");

    int current_lib = 1;
    char line[256];

    while (1)
    {
        printf("> ");
        if (!fgets(line, sizeof(line), stdin))
            break;

        line[strcspn(line, "\n")] = 0;

        if (line[0] == 'q')
```

```
    break;

if (line[0] == '0')
{
    current_lib = (current_lib == 1) ? 2 : 1;
    char libname[20];
    sprintf(libname, "./libfuncs%d.so", current_lib);

    if (load_library(libname))
    {
        printf("Переключено на библиотеку %d\n", current_lib);
    }
    else
    {
        current_lib = (current_lib == 1) ? 2 : 1;
    }
}
else if (line[0] == '1')
{
    int a, b;
    if (sscanf(line + 1, "%d %d", &a, &b) == 2)
    {
        printf("НОД(%d, %d) = %d\n", a, b, gcf_func(a, b));
    }
    else
    {
        printf("Ошибка формата. Используйте: 1 A B\n");
    }
}
else if (line[0] == '2')
{
    int size;
    char *ptr = line + 2;

    if (sscanf(ptr, "%d", &size) != 1 || size <= 0)
    {
        printf("Ошибка: укажите корректный размер массива (> 0)\n");
        continue;
    }

    while (*ptr && *ptr != ' ')
        ptr++;
    while (*ptr == ' ')
        ptr++;

    int *arr = malloc(size * sizeof(int));
    if (arr == NULL)
    {
        printf("Ошибка: не удалось выделить память\n");
        continue;
    }
```

```
            int i;
            for (i = 0; i < size; i++)
            {
                if (*ptr == '\0')
                {
                    break;
                }
                if (sscanf(ptr, "%d", &arr[i]) != 1)
                {
                    break;
                }
                while (*ptr && *ptr != ' ')
                    ptr++;
                while (*ptr == ' ')
                    ptr++;
            }

            if (i != size)
            {
                printf("Ошибка: указано %d элементов, но требуется %d\n", i, size);
                free(arr);
                continue;
            }

            printf("Исходный массив: ");
            print_array(arr, size);

            int *sorted = sort_func(arr, size);
            if (sorted)
            {
                printf("Отсортированный массив: ");
                print_array(sorted, size);
                free(sorted);
            }
            else
            {
                printf("Ошибка при сортировке\n");
            }

            free(arr);
        }
        else
        {
            if (line[0] != '\0')
            {
                printf("Неизвестная команда. Доступные команды: 0, 1, 2, q\n");
            }
        }
    }
```

```
    if (lib_handle)
        dlclose(lib_handle);
    return 0;
}
```

Протокол работы программы

Тестирование:

Программа 1 (статистическая линковка):

$ ./program1

Программа 1: Использование библиотеки при линковке

Команды:

0 - справка

1 A B - НОД чисел A и B

2 size n1 n2 ... - сортировка массива

q - выход

> 1 15 6

НОД(15, 6) = 3

> 2 3 109 98 4

Исходный массив: [109, 98, 4]

Отсортированный массив: [4, 98, 109]

> q

Программа 2 (динамическая загрузка):

$ ./program2

Загружена библиотека: ./libfuncs1.so

Программа 2: Динамическая загрузка библиотек

Команды:

0 - переключить библиотеку (1/2)

1 A B - НОД чисел A и B

2 size n1 n2 ... - сортировка массива

q - выход

> 1 15 6

НОД(15, 6) = 3

> 2 3 109 98 4

Исходный массив: [109, 98, 4]

Отсортированный массив: [4, 98, 109]

> 0

Загружена библиотека: ./libfuncs2.so

Переключено на библиотеку 2

> 1 15 6

НОД(15, 6) = 3

> 2 5 3 9 1 4

Исходный массив: [3, 9, 1, 4, 5]

Отсортированный массив: [1, 3, 4, 5, 9]

> q

Вывод strace

Program1 (статистическая линковка)

```
execve("./program1", ["./program1"], 0x7ffdb8491a60 /* 28 vars */) = 0
brk(NULL) = 0x55cc646aa000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff232fc350) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa0f0241000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libfuncs1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libfuncs1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
```

openat(AT_FDCWD, "./tls/haswell/x86_64/libfuncs1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/haswell/libfuncs1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/x86_64/libfuncs1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/libfuncs1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./haswell/x86_64/libfuncs1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./haswell/libfuncs1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./x86_64/libfuncs1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./libfuncs1.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0777, st_size=15528, ...}, AT_EMPTY_PATH) = 0

getcwd("/mnt/c/Users/mkoti/os_laba1/lab4", 128) = 33

mmap(NULL, 16432, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fa0f023c000

mmap(0x7fa0f023d000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7fa0f023d000

mmap(0x7fa0f023e000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7fa0f023e000

mmap(0x7fa0f023f000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7fa0f023f000

close(3) = 0

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=20892, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 20892, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fa0f0236000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fa0f000d000

mprotect(0x7fa0f0035000, 2023424, PROT_NONE) = 0

mmap(0x7fa0f0035000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fa0f0035000

mmap(0x7fa0f01ca000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fa0f01ca000

mmap(0x7fa0f0223000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7fa0f0223000

mmap(0x7fa0f0229000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fa0f0229000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa0f000a000

arch_prctl(ARCH_SET_FS, 0x7fa0f000a740) = 0

set_tid_address(0x7fa0f000aa10) = 1455

set_robust_list(0x7fa0f000aa20, 24) = 0

rseq(0x7fa0f000b0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fa0f0223000, 16384, PROT_READ) = 0

mprotect(0x7fa0f023f000, 4096, PROT_READ) = 0

mprotect(0x55cc60fb5000, 4096, PROT_READ) = 0

mprotect(0x7fa0f027b000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7fa0f0236000, 20892) = 0

newfstatat(1, "", {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

getrandom("\xe7\xc3\xb8\xf2\xfa\xb3\xe7\xf3", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x55cc646aa000

brk(0x55cc646cb000) = 0x55cc646cb000

write(1, "Программа 1: Использование библиотеки при линковке\n", 94) = 94

write(1, "Команды:\n", 16) = 16

write(1, " 0 - справка\n", 21) = 21

write(1, " 1 A B - НОД чисел A и B\n", 35) = 35

write(1, " 2 size n1 n2 ... - сортировка массива\n", 57) = 57

write(1, " q - выход\n", 17) = 17

write(1, "\n", 1) = 1

newfstatat(0, "", {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

write(1, "> ", 2) = 2

read(0, "1 15 6\n", 1024) = 7

write(1, "НОД(15, 6) = 3\n", 18) = 18

write(1, "> ", 2) = 2

read(0, "2 3 109 98 4\n", 1024) = 13

write(1, "Исходный массив: [109, 98, 4]\n", 44) = 44

write(1, "Отсортированный массив: [4, 98, 109]\n", 58) = 58

write(1, "> ", 2) = 2

read(0, "q\n", 1024) = 2

exit_group(0) = ?

+++ exited with 0 +++

Program2 (динамическая загрузка)

execve("./program2", ["./program2"], 0x7fff1ad3cba0 /* 28 vars */) = 0

brk(NULL) = 0x560d81fe7000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd7916fe20) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ffbef170000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=20892, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 20892, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ffbef16a000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ffbeef41000

mprotect(0x7ffbeef69000, 2023424, PROT_NONE) = 0

mmap(0x7ffbeef69000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7ffbeef69000

mmap(0x7ffbef0fe000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7ffbef0fe000

mmap(0x7ffbef157000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7ffbef157000

mmap(0x7ffbef15d000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ffbef15d000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ffbeef3e000

arch_prctl(ARCH_SET_FS, 0x7ffbeef3e740) = 0

set_tid_address(0x7ffbeef3ea10) = 1459

set_robust_list(0x7ffbeef3ea20, 24) = 0

rseq(0x7ffbeef3f0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7ffbef157000, 16384, PROT_READ) = 0

mprotect(0x560d61e48000, 4096, PROT_READ) = 0

mprotect(0x7ffbef1aa000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7ffbef16a000, 20892) = 0

getrandom("\x13\x5c\xd8\xb4\x6b\x96\xdb\x2f", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x560d81fe7000

brk(0x560d82008000) = 0x560d82008000

openat(AT_FDCWD, "./libfuncs1.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0777, st_size=15528, ...}, AT_EMPTY_PATH) = 0

getcwd("/mnt/c/Users/mkoti/os_laba1/lab4", 128) = 33

mmap(NULL, 16432, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ffbef16b000

```
mmap(0x7ffbef16c000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7ffbef16c000

mmap(0x7ffbef16d000, 4096, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7ffbef16d000

mmap(0x7ffbef16e000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7ffbef16e000

close(3) = 0

mprotect(0x7ffbef16e000, 4096, PROT_READ) = 0

newfstatat(1, "", {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0

write(1, "Загружена библиотека: ./libfuncs1.so\n", 56) = 56

write(1, "Программа 2: Динамическая загрузка библиотек\n", 83) = 83

write(1, "Команды:\n", 16) = 16

write(1, "  0 - переключить библиотеку (1/2)\n", 56) = 56

write(1, "  1 A B - НОД чисел A и B\n", 35) = 35

write(1, "  2 size n1 n2 ... - сортировка массива\n", 57) = 57

write(1, "  q - выход\n", 17) = 17

write(1, "\n", 1) = 1

newfstatat(0, "", {st_mode=S_IFCHR|0600, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0

write(1, "> ", 2) = 2

read(0, "1 15 6\n", 1024) = 7

write(1, "НОД(15, 6) = 3\n", 18) = 18

write(1, "> ", 2) = 2

read(0, "2 3 109 98 4\n", 1024) = 13

write(1, "Исходный массив: [109, 98, 4]\n", 44) = 44

write(1, "Отсортированный массив: [4, 98, 109]\n", 58) = 58

write(1, "> ", 2) = 2

read(0, "q\n", 1024) = 2

munmap(0x7ffbef16b000, 16432) = 0

exit_group(0) = ?

+++ exited with 0 +++
```

Системные вызовы в strace:

Для обеих программ:

- execve() – запуск исполняемого файла
- brk() – управление размером кучи
- mmap() – отображение файлов/памяти в адресное пространство
- write() / read() – ввод-вывод данных
- openat() – открытие файлов (библиотек)

Только для Program1 (статистическая линковка):

- Множественные openat() вызовы – поиск библиотек в стандартных путях системой
- Нет явного munmap() для библиотеки – выгрузка происходит автоматически

Только для Program2 (динамическая загрузка):

- Прямой openat() с указанием пути – явная загрузка библиотеки программой
- munmap() – явная выгрузка библиотеки из памяти перед завершением

Вывод

В ходе выполнения лабораторной работы были успешно реализованы две динамические библиотеки с различными алгоритмами вычисления НОД и сортировки массивов. Разработаны две программы, демонстрирующие разные подходы к использованию библиотек: статическая линковка на этапе компиляции и динамическая загрузка во время выполнения. Анализ системных вызовов с помощью strace показал, что при статической линковке библиотека загружается автоматически при старте программы через механизмы ОС, в то время как динамическая загрузка позволяет явно управлять жизненным циклом библиотек. Основная сложность заключалась в корректной обработке пользовательского ввода и обеспечении переключения между библиотеками без утечек памяти. Оба подхода имеют свои преимущества: статическая линковка проще в реализации, а динамическая загрузка обеспечивает большую гибкость и возможность "горячей" замены функционала