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: Хамзин Т.Н.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 17.12.24

Москва, 2024

Постановка задачи

Вариант 8.

Разработать программу на языке C, вычисляющую интеграл функции sin(x) на заданном отрезке [A, B] с указанным шагом e и сортирующую целочисленный массив. Реализовать две версии алгоритмов для каждой операции: для интегрирования — метод прямоугольников и метод трапеций; для сортировки — пузырьковую сортировку и быструю сортировку Хоара. Оформить каждую пару алгоритмов в виде отдельной динамической библиотеки (.so). Создать две демонстрационные программы: первая должна использовать одну из реализаций путём статической линковки на этапе компиляции, вторая — динамически загружать библиотеки во время выполнения с возможностью их переключения по команде пользователя.

Общий метод и алгоритм решения

Использованные системные вызовы:

- void *dlopen(const char *filename, int flags) - загрузка динамической библиотеки
- void *dlsym(void *handle, const char *symbol) - получение адреса функции из библиотек и
- int dlclose(void *handle) - выгрузка библиотеки
- char *dlerror(void) - получение ошибок

Я разработал две программы, использующие математические функции (вычисление интеграла sin(x) и сортировку массива) через динамические библиотеки: первая программа линкуется с библиотекой статически на этапе компиляции, вторая — загружает реализации динамически во время выполнения, используя dlopen/dlsym для переключения между алгоритмами (метод прямоугольников/трапеций для интеграла и пузырьковая/быстрая сортировка для массива), что демонстрирует различные подходы к интеграции внешнего кода и позволяет сравнивать производительность и точность разных реализаций без перекомпиляции основной программы.

Код программы

Impl1.c

```
#include <math.h>

#include <stdlib.h>

#include "contract.h"
```

```
float SinIntegral(float A, float B, float e) {
      if (e <= 0.0f || B <= A) {
          return 0.0f;
      }

      float integral = 0.0f;
      int n = (int)((B - A) / e);

      for (int i = 0; i < n; i++) {
          float x = A + i * e;
          integral += sinf(x) * e;
      }
      float remaining = B - (A + n * e);
      if (remaining > 0) {
          float x = A + n * e;
          integral += sinf(x) * remaining;
      }

      return integral;
  }
  int* Sort(int* array) {
      if (array == NULL) return NULL;
      int n = array[0];
      if (n <= 1) return array;

      for (int i = 1; i <= n; i++) {
          for (int j = 1; j <= n - i; j++) {
              if (array[j] > array[j + 1]) {
                  int temp = array[j];
                  array[j] = array[j + 1];
                  array[j + 1] = temp;
              }
          }
```

```
    }
    return array;
}
```

Impl2.c

```
#include <math.h>
#include <stdlib.h>
#include "contract.h"

float SinIntegral(float A, float B, float e) {
    if (e <= 0.0f || B <= A) {
        return 0.0f;
    }

    float integral = 0.0f;
    int n = (int)((B - A) / e);
    float prev_value = sinf(A);
    for (int i = 1; i <= n; i++) {
        float x = A + i * e;
        float cur_value = sinf(x);
        integral += (prev_value + cur_value) * e / 2.0f;
        prev_value = cur_value;
    }
    float remaining = B - (A + n * e);
    if (remaining > 0) {
        float x = A + n * e;
        float cur_value = sinf(B);
        integral += (prev_value + cur_value) * remaining / 2.0f;
    }

    return integral;
}

static void quick_sort(int* arr, int left, int right) {
```

```
    if (left >= right) return;

    int pivot = arr[(left + right) / 2];
    int i = left, j = right;

    while (i <= j) {
        while (arr[i] < pivot) i++;
        while (arr[j] > pivot) j--;
        if (i <= j) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++;
            j--;
        }
    }
    quick_sort(arr, left, j);
    quick_sort(arr, i, right);
}

int* Sort(int* array) {
    if (array == NULL) return NULL;
    int n = array[0];
    if (n <= 1) return array;
    quick_sort(&array[1], 0, n - 1);
    return array;
}
```

Prog1.c

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

```
int main() {
    char input[256];
    char command;
    float a, b, e;
    int *array = NULL;

    printf("Программа 1: Статическая линковка с impl1\n");
    printf("Команды:\n");
    printf("  0 - не используется в этой программе\n");
    printf("  1 A B e - вычислить интеграл sin(x) от A до B с шагом e\n");
    printf("  2 N x1 x2 ... xN - отсортировать массив из N элементов\n");
    printf("  q - выход\n\n");

    while (1) {
        printf("> ");

        if (fgets(input, sizeof(input), stdin) == NULL) {
            break;
        }

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

        if (input[0] == 'q' && (input[1] == '\0' || input[1] == ' ')) {
            break;
        }

        if (sscanf(input, "%c", &command) != 1) {
            printf("Неверный формат команды\n");
            continue;
```

```
}

switch (command) {
    case '0':
        printf("В этой программе переключение реализаций не поддерживается\n");
        break;

    case '1':
        if (sscanf(input, "%*c %f %f %f", &a, &b, &e) != 3) {
            printf("Ошибка: требуется 3 аргумента: A B e\n");
            printf("Пример: 1 0 3.14159 0.01\n");
            continue;
        }

        float result = SinIntegral(a, b, e);
        printf("Результат: %f\n", result);
        break;

    case '2': {
        int count = 0;
        char *token;
        char temp_input[256];
        strcpy(temp_input, input);

        token = strtok(temp_input, " ");
        while (token != NULL) {
            count++;
            token = strtok(NULL, " ");
        }
```

```
int n = count - 1;
if (n <= 0) {
    printf("Ошибка: требуется хотя бы один элемент для сортировки\n");
    printf("Пример: 2 5 9 3 7 1 5\n");
    continue;
}

if (array != NULL) free(array);
array = (int*)malloc((n + 1) * sizeof(int));
if (array == NULL) {
    printf("Ошибка выделения памяти\n");
    continue;
}

array[0] = n;

strcpy(temp_input, input);
token = strtok(temp_input, " ");
token = strtok(NULL, " ");

for (int i = 1; i <= n && token != NULL; i++) {
    array[i] = atoi(token);
    token = strtok(NULL, " ");
}

array = Sort(array);

printf("Отсортированный массив: ");
for (int i = 1; i <= n; i++) {
    printf("%d ", array[i]);
```

```
            }
            printf("\n");
            break;
        }

        default:
            printf("Неизвестная команда. Используйте 0, 1 или 2\n");
            break;
    }
}

if (array != NULL) free(array);
return 0;
}
```

Prog2.c

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

// Объявления типов функций
typedef float (*SinIntegralFunc)(float, float, float);
typedef int* (*SortFunc)(int*);

int main() {
    void *library = NULL;
    SinIntegralFunc SinIntegral = NULL;
    SortFunc Sort = NULL;

    int current_lib = 1;
```

```
char lib_name[2][20] = {"./libimpl1.so", "./libimpl2.so"};
char input[256];
char command;
float a, b, e;
int *array = NULL;

// Загружаем первую библиотеку
printf("Загружаем библиотеку %s\n", lib_name[0]);
library = dlopen(lib_name[0], RTLD_LAZY);
if (!library) {
    fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());
    return 1;
}

// Загружаем функции
SinIntegral = (SinIntegralFunc)dlsym(library, "SinIntegral");
Sort = (SortFunc)dlsym(library, "Sort");

if (!SinIntegral || !Sort) {
    fprintf(stderr, "Ошибка загрузки функций: %s\n", dlerror());
    dlclose(library);
    return 1;
}

printf("Программа 2: Динамическая загрузка библиотек\n");
printf("Команды:\n");
printf("  0 - переключить реализацию (libimpl1.so <-> libimpl2.so)\n");
printf("  1 A B e - вычислить интеграл sin(x) от A до B с шагом e\n");
printf("  2 N x1 x2 ... xN - отсортировать массив из N элементов\n");
printf("  q - выход\n\n");
```

```
while (1) {
    printf("> ");

    // Читаем всю строку
    if (fgets(input, sizeof(input), stdin) == NULL) {
        break;
    }

    // Убираем символ новой строки
    input[strcspn(input, "\n")] = 0;

    // Проверяем на выход
    if (input[0] == 'q' && (input[1] == '\0' || input[1] == ' ')) {
        break;
    }

    // Парсим команду
    if (sscanf(input, "%c", &command) != 1) {
        printf("Неверный формат команды\n");
        continue;
    }

    switch (command) {
        case '0': {
            // Переключение библиотеки
            dlclose(library);

            current_lib = (current_lib == 1) ? 2 : 1;
            printf("Переключаем на библиотеку %s\n", lib_name[current_lib - 1]);
```

```
    library = dlopen(lib_name[current_lib - 1], RTLD_LAZY);
    if (!library) {
        fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror());
        return 1;
    }

    SinIntegral = (SinIntegralFunc)dlsym(library, "SinIntegral");
    Sort = (SortFunc)dlsym(library, "Sort");

    if (!SinIntegral || !Sort) {
        fprintf(stderr, "Ошибка загрузки функций: %s\n", dlerror());
        dlclose(library);
        return 1;
    }
    break;
}

case '1': {
    // Интеграл
    if (sscanf(input, "%*c %f %f %f", &a, &b, &e) != 3) {
        printf("Ошибка: требуется 3 аргумента: A B e\n");
        printf("Пример: 1 0 3.14159 0.01\n");
        continue;
    }

    float result = SinIntegral(a, b, e);
    printf("Результат (библиотека %d): %f\n", current_lib, result);
    break;
}
```

```
case '2': {
    int count = 0;
    char *token;
    char temp_input[256];
    strcpy(temp_input, input);

    token = strtok(temp_input, " ");
    while (token != NULL) {
        count++;
        token = strtok(NULL, " ");
    }
    int n = count - 1;
    if (n <= 0) {
        printf("Ошибка: требуется хотя бы один элемент для сортировки\n");
        printf("Пример: 2 5 9 3 7 1 5\n");
        continue;
    }
    if (array != NULL) free(array);
    array = (int*)malloc((n + 1) * sizeof(int));
    if (array == NULL) {
        printf("Ошибка выделения памяти\n");
        continue;
    }
    array[0] = n;
    strcpy(temp_input, input);
    token = strtok(temp_input, " ");
    token = strtok(NULL, " "); // пропускаем команду '2'

    for (int i = 1; i <= n && token != NULL; i++) {
```

```
            array[i] = atoi(token);
            token = strtok(NULL, " ");
        }
        array = Sort(array);
        printf("Отсортированный массив (библиотека %d): ", current_lib);
        for (int i = 1; i <= n; i++) {
            printf("%d ", array[i]);
        }
        printf("\n");
        break;
      }

      default:
        printf("Неизвестная команда. Используйте 0, 1 или 2\n");
        break;
    }
  }

  if (array != NULL) free(array);
  if (library != NULL) dlclose(library);
  return 0;
}
```

Протокол работы программы

```
Тестирование:
$ ./prog2
1 5 10 0.1
Результат (библиотека 1): 1.101053
> 0
Переключаем на библиотеку ./libimpl2.so
> 1 5 10 0.1
Результат (библиотека 2): 1.121798
```

strace -f ./prog2

```
execve("./prog2", ["./prog2"], 0x7ffeca910078 /* 31 vars */) = 0
brk(NULL)                               = 0x36708000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x707d7c2ac000
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=33091, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 33091, PROT_READ, MAP_PRIVATE, 3, 0) = 0x707d7c2a3000
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\0\20t\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
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1926232, ...}, AT_EMPTY_PATH) = 0
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
mmap(NULL, 1974096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x707d7c0c1000
mmap(0x707d7c0e7000, 1400832, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x26000) = 0x707d7c0e7000
mmap(0x707d7c23d000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17c000) =
0x707d7c23d000
mmap(0x707d7c290000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1cf000) = 0x707d7c290000
mmap(0x707d7c296000, 53072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0) = 0x707d7c296000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x707d7c0be000
arch_prctl(ARCH_SET_FS, 0x707d7c0be740) = 0
set_tid_address(0x707d7c0bea10)         = 7211
set_robust_list(0x707d7c0bea20, 24)     = 0
rseq(0x707d7c0bf060, 0x20, 0, 0x53053053) = 0
mprotect(0x707d7c290000, 16384, PROT_READ) = 0
mprotect(0x403000, 4096, PROT_READ)     = 0
mprotect(0x707d7c2df000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
```

```
munmap(0x707d7c2a3000, 33091)           = 0

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

getrandom("\x30\xfe\xab\x58\x75\xfb\x6b\x36", 8, GRND_NONBLOCK) = 8

brk(NULL)                               = 0x36708000

brk(0x36729000)                         = 0x36729000

write(1, "\320\227\320\260\320\263\321\200\321\203\320\266\320\260\320\265\320\274
\320\261\320\270\320\261\320\273\320\270\320\276\321"..., 543агружаем библиотеку
./libimpl1.so

) = 54

openat(AT_FDCWD, "./libimpl1.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|0755, st_size=15312, ...}, AT_EMPTY_PATH) = 0

getcwd("/workspaces/lab1/lab4/src", 128) = 26

mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x707d7c2a7000

mmap(0x707d7c2a8000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x707d7c2a8000

mmap(0x707d7c2a9000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x707d7c2a9000

mmap(0x707d7c2aa000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x707d7c2aa000

close(3)                                = 0

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=33091, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 33091, PROT_READ, MAP_PRIVATE, 3, 0) = 0x707d7c0b5000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=911904, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 913680, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x707d7bfd5000

mmap(0x707d7bfe5000, 475136, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x10000) = 0x707d7bfe5000

mmap(0x707d7c059000, 368640, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x84000) =
0x707d7c059000

mmap(0x707d7c0b3000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xdd000) = 0x707d7c0b3000

close(3)                                = 0
```

```
mprotect(0x707d7c0b3000, 4096, PROT_READ) = 0

mprotect(0x707d7c2aa000, 4096, PROT_READ) = 0

munmap(0x707d7c0b5000, 33091)           = 0

write(1, "\320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260 2:
\320\224\320\270\320\275\320\260\320\274"..., 83Программа 2: Динамическая загрузка библиотек

) = 83

write(1, "\320\232\320\276\320\274\320\260\320\275\320\264\321\213:\n", 16Команды:

) = 16

write(1, "  0 -
\320\277\320\265\321\200\320\265\320\272\320\273\321\216\321\207\320\270\321\202\321\214
\321\200\320"..., 80  0 - переключить реализацию (libimpl1.so <-> libimpl2.so)

) = 80

write(1, "  1 A B e -
\320\262\321\213\321\207\320\270\321\201\320\273\320\270\321\202\321\214 \320"..., 85  1 A B
e - вычислить интеграл sin(x) от A до B с шагом e

) = 85

write(1, "  2 N x1 x2 ... xN - \320\276\321\202\321\201\320\276\321\200\321"..., 87  2 N x1
x2 ... xN - отсортировать массив из N элементов

) = 87

write(1, "  q - \320\262\321\213\321\205\320\276\320\264\n", 17  q - выход

) = 17

write(1, "\n", 1

)                       = 1

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

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

read(0, 1 2 10 0.01

"1 2 10 0.01\n", 1024)          = 12

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202
(\320\261\320\270\320\261\320\273\320\270\320\276"..., 54Результат (библиотека 1): 0.430189

) = 54

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

read(0, 0

"0\n", 1024)                    = 2

munmap(0x707d7c2a7000, 16408)           = 0

munmap(0x707d7bfd5000, 913680)          = 0
```

```
write(1,
"\320\237\320\265\321\200\320\265\320\272\320\273\321\216\321\207\320\260\320\265\320\274
\320\275\320\260 \320\261\320\270"..., 63Переключаем на библиотеку ./libimpl2.so

) = 63

openat(AT_FDCWD, "./libimpl2.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|0755, st_size=15416, ...}, AT_EMPTY_PATH) = 0

getcwd("/workspaces/lab1/lab4/src", 128) = 26

mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x707d7c2a7000

mmap(0x707d7c2a8000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x707d7c2a8000

mmap(0x707d7c2a9000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x707d7c2a9000

mmap(0x707d7c2aa000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x707d7c2aa000

close(3)                                = 0

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=33091, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 33091, PROT_READ, MAP_PRIVATE, 3, 0) = 0x707d7c0b5000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=911904, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 913680, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x707d7bfd5000

mmap(0x707d7bfe5000, 475136, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x10000) = 0x707d7bfe5000

mmap(0x707d7c059000, 368640, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x84000) =
0x707d7c059000

mmap(0x707d7c0b3000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xdd000) = 0x707d7c0b3000

close(3)                                = 0

mprotect(0x707d7c0b3000, 4096, PROT_READ) = 0

mprotect(0x707d7c2aa000, 4096, PROT_READ) = 0

munmap(0x707d7c0b5000, 33091)           = 0

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

read(0, 1 2 10 0.01
```

```
"1 2 10 0.01\n", 1024)          = 12

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202
(\320\261\320\270\320\261\320\273\320\270\320\276"..., 54Результат (библиотека 2): 0.422921

) = 54

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

read(0, ^C0x36709dc0, 1024)              = ? ERESTARTSYS (To be restarted if SA_RESTART is
set)

strace: Process 7211 detached
```

Вывод

Данная лабораторная работа демонстрирует механизм динамической загрузки библиотек (dynamic loading) во время выполнения программы. Основная программа интерактивно загружает различные реализации математических функций из внешних разделяемых объектов (libimpl1.so и libimpl2.so), используя функции dlopen, dlsym и dlclose. Реализация позволяет переключаться между альтернативными алгоритмами вычисления интеграла и сортировки без перекомпиляции основного кода. Работа подтверждает эффективность подхода плагинной архитектуры для создания модульных и расширяемых приложений, где функциональность может быть изменена простой заменой динамических библиотек.