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»

Выполнил: М. А. Понизяйкин
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель работы:
Приобретение практических навыков в:

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

Задание:
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;
3. «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.

Вариант: 25

№	Описание	Сигнатура	Реализация 1	Реализация 2
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.
8	Перевод числа x из десятичной системы счисления в другую	Char* translation(long x)	Другая система счисления двоичная	Другая система счисления троичная

Метод решения

Алгоритм решения задачи:

1. Пользователь вводит команду в консоль одной из тестовых программ:
   - `0` - переключение реализаций (только для программы с динамической загрузкой);
   - `1 A B` - вызов функции `gcf(A, B)` для вычисления НОД;
   - `2 X` - вызов функции `translation(X)` для перевода числа в другую систему счисления;
2. Для программы `main_link` (и её альтернативной версии `main_link_alt`):
   - Связывание с динамическими библиотеками происходит на этапе линковки;
   - При запуске система автоматически загружает нужные `.so`-файлы (например, `libgcf_euclid.so` и `libtranslation_binary.so`);
   - Вызовы функций `gcf` и `translation` разрешаются до начала выполнения основной логики;
3. Для программы `main_runtime`:
   - На старте создаются объекты класса `DynamicLibrary`, которые вызывают `dlopen` для загрузки библиотек `./libgcf_euclid.so` и `./libtranslation_binary.so`;
   - Через метод `get_function` с использованием `dlsym` получают указатели на функции `gcf` и `translation`;
4. При вводе команды `0` в `main_runtime`:
   - Текущие библиотеки выгружаются через деструкторы `DynamicLibrary` (вызывается `dlclose`);
   - Загружаются альтернативные реализации: `./libgcf_naive.so` и `./libtranslation_ternar`
   - Обновляются указатели на функции, и программа выводит сообщение о переключении.
5. При вводе команды `1 A B`:
   - Вызывается функция `gcf(A, B)`;

- В зависимости от загруженной библиотеки используется либо алгоритм Евклида, либо наивный перебор;
- Результат выводится в консоль.

6. При вводе команды `2 X`:
   - Вызывается функция `translation(X)`;
   - В зависимости от реализации число переводится либо в двоичную, либо в троичную систему счисления.
   - Результат (строка, выделенная через `malloc`) выводится в консоль, после чего освобождается через `free`.
7. Все функции экспортируются из библиотек с использованием `extern "C"`, чтобы избежать `name mangling` и обеспечить корректную работу `dlsym`.
8. Память, выделенная в библиотеках (`malloc`), освобождается в вызывающем коде (`free`), что гарантирует совместимость аллокаторов между модулями.
9. Обработка ошибок:
   - При невозможности загрузить библиотеку (ошибка `dlopen`) выбрасывается исключение `std::runtime_error`;
   - При отсутствии символа (ошибка `dlsym`) также выбрасывается исключение;
   - Программа завершается с информативным сообщением об ошибке.
10. После завершения работы все динамические библиотеки выгружаются автоматически (благодаря `RAII`), ресурсы освобождаются, и программа корректно завершает выполнение.

Архитектура программы:

```
lab4-var25/
├── build/
├── apps/
│   ├── main_link.cpp
│   ├── main_link_alt.cpp
│   └── main_runtime.cpp
├── include/
│   ├── gcf.h
│   ├── translation.h
│   └── platform/
│       └── dynlib.h
└── src/
    ├── gcf_euclid.cpp
    ├── gcf_naive.cpp
    ├── translation_binary.cpp
    ├── translation_ternary.cpp
    └── platform/
        └── dynlib.cpp
```

Ссылки:

- https://pubs.opengroup.org/onlinepubs/9699919799/functions/dlopen.html?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://pubs.opengroup.org/onlinepubs/9699919799/functions/dlsym.html?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://pubs.opengroup.org/onlinepubs/9699919799/functions/dlclose.html?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://man7.org/linux/man-pages/man3/dlopen.3.html?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://www.ibm.com/developerworks/library/l-dynamic-libraries/?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://msdn.microsoft.com/en-us/library/ms235636.aspx?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://msdn.microsoft.com/en-us/library/windows/desktop/ms684175(v=vs.85).aspx?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://msdn.microsoft.com/en-us/library/windows/desktop/ms683212(v=vs.85).aspx?spm=a2ty_o01.29997173.0.0.713351719DZq6l
- https://gcc.gnu.org/wiki/Visibility?spm=a2ty_o01.29997173.0.0.713351719DZq6l

Описание программы

`gcf.h, translation.h` — объявляют функции с `extern "C"` для экспорта без `name mangling`.

`gcf_euclid.cpp` — реализует НОД через циклическое деление по модулю.
Основные функции:

- `int gcf(int a, int b)` — вычисляет наибольший общий делитель двух натуральных чисел с использованием классического алгоритма Евклида (через последовательное деление с остатком).

`gcf_naive.cpp` — перебирает все возможные делители от `min(a,b)` до 1.
Основные функции:

- `int gcf(int a, int b)` — вычисляет НОД путём перебора всех возможных делителей от `min(a, b)` до 1. Возвращает первый общий делитель, начиная с наибольшего.

`translation_binary.cpp` — использует побитовые операции `(x & 1, x >>= 1)` для перевода в двоичную систему.
Основные функции:

- `char* translation(long x)` — преобразует целое число `x` в строковое представление в двоичной системе счисления. Обрабатывает отрицательные числа и ноль. Память под строку выделяется через `malloc`.

translation_ternary.cpp —— использует деление на 3 и остаток от деления для троичного представления.
Основные функции:

- char* translation(long x) — преобразует целое число x в строковое представление в троичной системе счисления. Аналогично поддерживает отрицательные значения и ноль. Использует деление на 3 и запись остатков. Память выделяется через malloc.

platform/dynlib.h/cpp — инкапсулирует dlopen, dlsym, dlclose в RAII-класс DynamicLibrary.
Основные функции:

- DynamicLibrary::DynamicLibrary(const std::string& path) —— загружает динамическую библиотеку по указанному пути. Использует dlopen с флагом RTLD_LAZY. В случае ошибки выбрасывает исключение.
- DynamicLibrary:: DynamicLibrary() —— автоматически выгружает библиотеку при уничтожении объекта. Использует dlclose.
- void* DynamicLibrary::get_symbol_impl(const char* name) const —— получает адрес символа (функции или переменной) по его имени из загруженной библиотеки. Использует dlsym. При отсутствии символа выбрасывает исключение.
- template<typename Func> Func get_function(const char* name) const —— шаблонный метод для безопасного получения указателя на функцию заданной сигнатуры. Выполняет приведение через reinterpret_cast.

main_link.cpp и main_link_alt.cpp —— идентичные по коду, различаются только линковкой.
Основные функции:

- main() —— реализует цикл ввода команд пользователя и вызов функций gcf и translation. Команда 0 не поддерживается (выводится соответствующее сообщение). Результаты вычислений выводятся в консоль. Память, полученная от translation, освобождается через free.

main_runtime.cpp —— загружает библиотеки вручную, поддерживает переключение по команде 0.
Основные функции:

- main() —— загружает динамические библиотеки (libgcf_euclid.so и libtranslation_binary.s с помощью обёртки DynamicLibrary.
  Поддерживает команду 0 для переключения на альтернативные реализации (libgcf_naive.so, libtranslation_ternary.so).
  Обрабатывает команды 1 и 2, вызывая соответствующие функции через указатели, полученные через get_function.
  Корректно освобождает память через free и обеспечивает безопасную выгрузку библиотек при завершении.

Результаты

Программа успешно создаёт четыре динамические библиотеки (libgcf_euclid.so, libgcf_naive.so, libtranslation_binary.so, libtranslation_ternary.so), реализующие две функции с двумя различными алгоритмами каждая.
Тестовая программа main_link корректно использует библиотеки, подключённые на этапе линковки, и выдаёт результаты вычислений НОД и перевода чисел в заданную систему счисления.
Программа main_runtime загружает библиотеки во время выполнения с помощью системного API (dlopen, dlsym, dlclose), поддерживает переключение реализаций по команде 0 и корректно вызывает функции через полученные указатели.
Все команды обрабатываются в соответствии с заданным форматом ввода:

- 1 A B — вычисление НОД,
- 2 X — перевод числа,
- 0 — переключение реализаций (только в main_runtime).

Память, выделенная в библиотеках, освобождается в вызывающем коде, что предотвращает утечки. Программы корректно обрабатывают ошибки: отсутствие .so-файлов, отсутствие символов в библиотеках, неверный ввод пользователя. В случае ошибки выводится понятное сообщение, и программа завершается безопасно.

Выводы

В ходе выполнения лабораторной работы были приобретены практические навыки создания и использования динамических библиотек. Были реализованы и отлажены две стратегии взаимодействия с динамическими библиотеками:

- implicit linking —— связывание на этапе компиляции (программы main_link и main_link_alt),
- explicit linking —— загрузка библиотек во время выполнения через POSIX API (main_runtime).

Для обеспечения совместимости с dlsym все экспортируемые функции объявлены с использованием extern "C", что предотвращает name mangling и гарантирует корректное разрешение символов. Архитектура программы построена по принципу инкапсуляции: системно-зависимый код выделен в отдельный модуль (platform/dynlib.cpp), что повышает читаемость, безопасность (благодаря RAII) и облегчает возможную адаптацию под другие ОС в будущем. Результаты работы полностью соответствуют требованиям задания и демонстрируют глубокое понимание механизмов динамической загрузки кода, управления памятью и модульного проектирования программ на C++.

Исходная программа

```
#pragma once

extern "C" int gcf(int a, int b);
```

Листинг 1: include/gcf.h

```
#pragma once

extern "C" char* translation(long long x);
```

Листинг 2: include/translation.h

```
#pragma once

#include <memory>
#include <string>

namespace platform {

class DynamicLibrary {
public:
  explicit DynamicLibrary(const std::string &path);
  ~DynamicLibrary();

  DynamicLibrary(const DynamicLibrary &) = delete;
  DynamicLibrary &operator=(const DynamicLibrary &) = delete;

  template <typename Func> Func get_function(const char *name) const {
    return reinterpret_cast<Func>(get_symbol_impl(name));
  }

private:
  void *get_symbol_impl(const char *name) const;
  void *handle_ = nullptr;
};

} // namespace platform
```

Листинг 3: include/platform/dynlib.h

```
#include <cstdlib>
#include <iostream>
#include <string>

#include "gcf.h"
#include "translation.h"

int main() {
  std::string cmd;
  std::cout << "Euclid gcf and binary translation (linking)\n"
            << "Select mode:\n\t0 - switch libraries (not available in static "
               "linkage mode)\n"
            << "\t1 - GCF\n\t2 - translation\n";
  while (std::cin >> cmd) {
    if (cmd == "0") {
```

```
      std::cout << "Switch libraries not available in static linkage.\n";
    } else if (cmd == "1") {
      std::cout << "Write: A B\n";
      int a;
      int b;
      std::cin >> a >> b;
      std::cout << "GCF(" << a << ", " << b << ") = " << gcf(a, b) << "\n";
    } else if (cmd == "2") {
      std::cout << "Write: X\n";
      long long x;
      std::cin >> x;
      char *res = translation(x);
      std::cout << "Translation(" << x << ") = " << res << "\n";
      std::free(res);
    }
  }
  return 0;
}
```

Листинг 4: apps/main_link.cpp

```
#include <cstdlib>
#include <iostream>
#include <string>

#include "gcf.h"
#include "translation.h"

int main() {
  std::string cmd;
  std::cout << "Naive gcf and ternary translation (linking)\n"
            << "Select mode:\n\t0 - switch libraries (not available in static "
               "linkage mode)\n"
            << "\t1 - GCF\n\t2 - translation\n";
  while (std::cin >> cmd) {
    if (cmd == "0") {
      std::cout << "Switch libraries not available in static linkage.\n";
    } else if (cmd == "1") {
      std::cout << "Write: A B\n";
      int a;
      int b;
      std::cin >> a >> b;
      std::cout << "GCF(" << a << ", " << b << ") = " << gcf(a, b) << "\n";
    } else if (cmd == "2") {
      std::cout << "Write: X\n";
      long long x;
      std::cin >> x;
      char *res = translation(x);
      std::cout << "Translation(" << x << ") = " << res << "\n";
      std::free(res);
    }
  }
  return 0;
}
```

Листинг 5: apps/main_link_alt.cpp

```cpp
#include <iostream>
#include <memory>
#include <string>

#include "platform/dynlib.h"

using gcf_func_t = int (*)(int, int);
using translation_func_t = char *(*)(long long);

int main() {
  std::string gcf_path = "./libgcf_euclid.so";
  std::string trans_path = "./libtranslation_binary.so";

  std::unique_ptr<platform::DynamicLibrary> gcf_lib;
  std::unique_ptr<platform::DynamicLibrary> trans_lib;

  try {
    gcf_lib = std::make_unique<platform::DynamicLibrary>(gcf_path);
    trans_lib = std::make_unique<platform::DynamicLibrary>(trans_path);
  } catch (const std::exception &e) {
    std::cerr << "Error: " << e.what() << "\n";
    return 1;
  }

  auto gcf_func = gcf_lib->get_function<gcf_func_t>("gcf");
  auto trans_func = trans_lib->get_function<translation_func_t>("translation");

  std::cout << "Euclid gcf and binary translation (runtime)\n"
            << "Select mode:\n\t0 - switch libraries\n"
            << "\t1 - GCF\n\t2 - translation\n";
  std::string cmd;
  while (std::cin >> cmd) {
    if (cmd == "0") {
      if (gcf_path == "./libgcf_euclid.so") {
        gcf_path = "./libgcf_naive.so";
        trans_path = "./libtranslation_ternary.so";
      } else {
        gcf_path = "./libgcf_euclid.so";
        trans_path = "./libtranslation_binary.so";
      }

      try {
        gcf_lib = std::make_unique<platform::DynamicLibrary>(gcf_path);
        trans_lib = std::make_unique<platform::DynamicLibrary>(trans_path);

        gcf_func = gcf_lib->get_function<gcf_func_t>("gcf");
        trans_func = trans_lib->get_function<translation_func_t>("translation");
        std::cout << "Switched to alternative.\n";
      } catch (const std::exception &e) {
        std::cerr << "Switch failed: " << e.what() << "\n";
      }
    } else if (cmd == "1") {
      std::cout << "Write: A B\n";
      int a;
      int b;
      std::cin >> a >> b;
      std::cout << "GCF(" << a << ", " << b << ") = " << gcf_func(a, b) << "\n";
    } else if (cmd == "2") {
```

```
      std::cout << "Write: X\n";
      long long x;
      std::cin >> x;
      char *res = trans_func(x);
      std::cout << "Translation(" << x << ") = " << res << "\n";
      std::free(res);
    }
  }

  return 0;
}
```

Листинг 6: apps/main_runtime.cpp

```
#include "gcf.h"

int gcf(int a, int b) {
  while (b != 0) {
    int temp = b;
    b = a % b;
    a = temp;
  }
  return a;
}
```

Листинг 7: src/gcf_euclid.cpp

```
#include <algorithm>

#include "gcf.h"

int gcf(int a, int b) {
  int min_value = std::min(a, b);
  for (int i = min_value; i >= 1; --i) {
    if (a % i == 0 && b % i == 0)
      return i;
  }
  return 1;
}
```

Листинг 8: src/gcf_naive.cpp

```
#include <cstdlib>
#include <cstring>

#include "translation.h"

const int8_t BUFFER = 65;

char *translation(long long x) {
  if (x == 0) {
    char *res = static_cast<char *>(malloc(2));
    std::strcpy(res, "0");
    return res;
  }

```

```
  bool neg = x < 0;
  if (neg)
    x = -x;

  char buffer[BUFFER];
  int i = 0;
  while (x > 0) {
    buffer[i++] = '0' + (x & 1);
    x >>= 1;
  }
  if (neg)
    buffer[i++] = '-';

  char *result = static_cast<char *>(malloc(i + 1));
  for (int j = 0; j < i; ++j) {
    result[j] = buffer[i - 1 - j];
  }
  result[i] = '\0';
  return result;
}
```

Листинг 9: src/translation_binary.cpp

```
#include "translation.h"

#include <cstdlib>
#include <cstring>

const int8_t BUFFER = 65;

char *translation(long long x) {
  if (x == 0) {
    char *res = static_cast<char *>(malloc(2));
    std::strcpy(res, "0");
    return res;
  }

  bool neg = x < 0;
  if (neg)
    x = -x;

  char buffer[BUFFER];
  int i = 0;
  while (x > 0) {
    buffer[i++] = '0' + (x % 3);
    x /= 3;
  }
  if (neg)
    buffer[i++] = '-';

  char *result = static_cast<char *>(malloc(i + 1));
  for (int j = 0; j < i; ++j) {
    result[j] = buffer[i - 1 - j];
  }
  result[i] = '\0';
  return result;
```

```
}
```

Листинг 10: src/translation_ternary.cpp

```
#include "platform/dynlib.h"

#include <dlfcn.h>
#include <stdexcept>
#include <string>

namespace platform {

DynamicLibrary::DynamicLibrary(const std::string &path)
    : handle_(dlopen(path.c_str(), RTLD_LAZY)) {
  if (!handle_) {
    throw std::runtime_error("Failed to load library: " + path);
  }
}

DynamicLibrary::~DynamicLibrary() {
  if (handle_) {
    dlclose(handle_);
  }
}

void *DynamicLibrary::get_symbol_impl(const char *name) const {
  void *symbol = dlsym(handle_, name);
  if (!symbol) {
    throw std::runtime_error("Symbol not found: " + std::string(name));
  }
  return symbol;
}

} // namespace platform
```

Листинг 11: src/platform/dynlib.cpp

Strace

```
execve("./main_runtime", ["./main_runtime"], 0x7ffc4db5f420 /* 27 vars */) = 0
brk(NULL)                               = 0x578834a10000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70b06d05
readlinkat(AT_FDCWD, "/proc/self/exe", "/home/yamaksush/MAI_OS_Labs/lab4"..., 4096) =
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/glibc-hwcaps/x86-64-v3
newfstatat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/glibc-hwcaps/x86-6
openat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/glibc-hwcaps/x86-64-v2
newfstatat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/glibc-hwcaps/x86-6
openat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/libstdc++.so.6", O_RDO
newfstatat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/", {st_mode=S_IFDI
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=20919, ...}) = 0
mmap(NULL, 20919, PROT_READ, MAP_PRIVATE, 3, 0) = 0x70b06d04b000
```

```
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.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) = 8
fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06cc00000
mmap(0x70b06cc9d000, 1343488, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x70b06cde5000, 552960, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e5
mmap(0x70b06ce6c000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x70b06ce7a000, 12608, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
close(3)                                = 0
openat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/libgcc_s.so.1", O_RDON
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 8
fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06d01d000
mmap(0x70b06d021000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x70b06d045000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2800
mmap(0x70b06d049000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                = 0
openat(AT_FDCWD, "/home/yamaksush/MAI_OS_Labs/lab4-var25/build/libc.so.6", O_RDONLY|O
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\220\243\2\0\0\0\0\0"..., 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, 6
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 6
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06c800000
mmap(0x70b06c828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRIT
mmap(0x70b06c9b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0
mmap(0x70b06c9ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x70b06ca05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS
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) = 8
fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06cf34000
mmap(0x70b06cf44000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE
mmap(0x70b06cfc3000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8f0
mmap(0x70b06d01b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70b06cf3
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x70b06cf
arch_prctl(ARCH_SET_FS, 0x70b06cf2f740) = 0
set_tid_address(0x70b06cf2fa10)         = 44615
set_robust_list(0x70b06cf2fa20, 24)     = 0
rseq(0x70b06cf30060, 0x20, 0, 0x53053053) = 0
mprotect(0x70b06c9ff000, 16384, PROT_READ) = 0
mprotect(0x70b06d01b000, 4096, PROT_READ) = 0
mprotect(0x70b06d049000, 4096, PROT_READ) = 0
```

```
mprotect(0x70b06ce6c000, 45056, PROT_READ) = 0
mprotect(0x57881da92000, 4096, PROT_READ) = 0
mprotect(0x70b06d089000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x70b06d04b000, 20919)           = 0
futex(0x70b06ce7a7bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
getrandom("\xc4\xe1\x1f\xa1\x2d\x9f\x7a\xe8", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x578834a10000
brk(0x578834a31000)                     = 0x578834a31000
openat(AT_FDCWD, "./libgcf_euclid.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) = 83
fstat(3, {st_mode=S_IFREG|0755, st_size=15112, ...}) = 0
getcwd("/home/yamaksush/MAI_OS_Labs/lab4-var25/build", 128) = 45
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06d04c000
mmap(0x70b06d04d000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3
mmap(0x70b06d04e000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
mmap(0x70b06d04f000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                = 0
mprotect(0x70b06d04f000, 4096, PROT_READ) = 0
openat(AT_FDCWD, "./libtranslation_binary.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) = 83
fstat(3, {st_mode=S_IFREG|0755, st_size=15664, ...}) = 0
getcwd("/home/yamaksush/MAI_OS_Labs/lab4-var25/build", 128) = 45
mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06cf2a000
mmap(0x70b06cf2b000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3
mmap(0x70b06cf2c000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
mmap(0x70b06cf2d000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                = 0
mprotect(0x70b06cf2d000, 4096, PROT_READ) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "Euclid gcf and binary translatio"..., 44) = 44
write(1, "Select mode:\n\t0 - switch librari"..., 35) = 35
write(1, "\t1 - GCF\n\t2 - translation\n", 26) = 26
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
read(0, "0\n", 1024)                    = 2
openat(AT_FDCWD, "./libgcf_naive.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) = 83
fstat(3, {st_mode=S_IFREG|0755, st_size=15312, ...}) = 0
getcwd("/home/yamaksush/MAI_OS_Labs/lab4-var25/build", 128) = 45
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06cf25000
mmap(0x70b06cf26000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3
mmap(0x70b06cf27000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
mmap(0x70b06cf28000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                = 0
mprotect(0x70b06cf28000, 4096, PROT_READ) = 0
munmap(0x70b06d04c000, 16400)           = 0
openat(AT_FDCWD, "./libtranslation_ternary.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) = 83
```

```
fstat(3, {st_mode=S_IFREG|0755, st_size=15664, ...}) = 0
getcwd("/home/yamaksush/MAI_OS_Labs/lab4-var25/build", 128) = 45
mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x70b06d04c000
mmap(0x70b06d04d000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 
mmap(0x70b06d04e000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
mmap(0x70b06d04f000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
close(3)                                = 0
mprotect(0x70b06d04f000, 4096, PROT_READ) = 0
munmap(0x70b06cf2a000, 16416)           = 0
write(1, "Switched to alternative.\n", 25) = 25
read(0, "1\n", 1024)                    = 2
write(1, "Write: A B\n", 11)            = 11
read(0, "10 6\n", 1024)                 = 5
write(1, "GCF(10, 6) = 2\n", 15)        = 15
read(0, "2\n", 1024)                    = 2
write(1, "Write: X\n", 9)               = 9
read(0, "48\n", 1024)                   = 3
write(1, "Translation(48) = 1210\n", 23) = 23
read(0, 0x578834a24600, 1024)           = ? ERESTARTSYS (To be restarted if SA_RESTAR
--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---
+++ killed by SIGINT +++
```