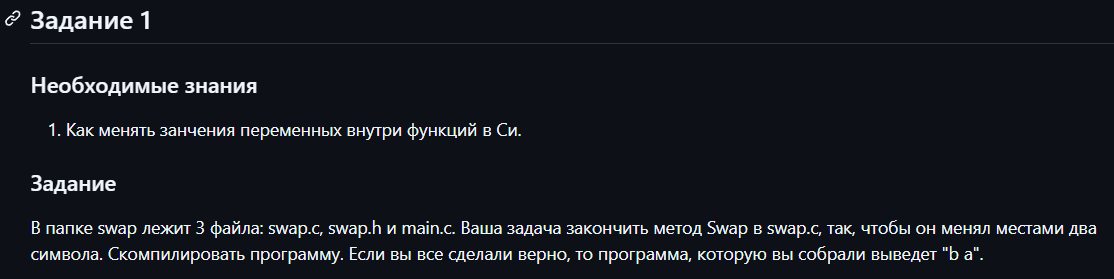
ЛР2



**Изменение значений переменных внутри функций в C**

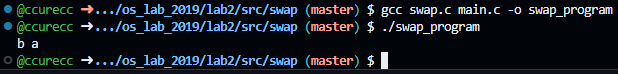
**1. Передача по значению**

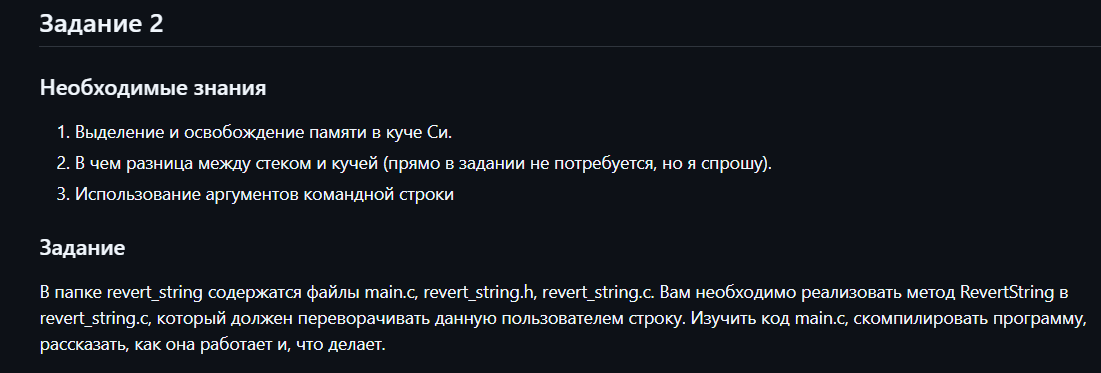
Функция получает **копию** переменной. Изменения не сохраняются после вызова.

**2. Передача по указателю**

Функция получает **адрес** переменной. Изменения сохраняются.

Код поменят, все закоммитил





**1. Выделение и освобождение памяти в куче (Си)**

**Выделение:**

* **malloc(size)** — выделяет size байт неинициализированной памяти.
* **calloc(n, size)** — выделяет память для n элементов по size байт (инициализирует нулями).
* **realloc(ptr, new\_size)** — изменяет размер ранее выделенного блока памяти.

**Освобождение:**

* **free(ptr)** — освобождает память, выделенную malloc/calloc/realloc.

**2. Разница между стеком и кучей**

Стек Куча

Автоматическое управление памятью Ручное управление (malloc/free)

Быстрый доступ Медленнее стека

Ограниченный размер Большой размер (доступна вся RAM)

Хранит локальные переменные Хранит динамические данные

Освобождается при выходе из функции Требуется явное освобождение

**3. Аргументы командной строки**

Передача параметров в программу при запуске.

int main(int argc, char \*argv[]) {

// argc — количество аргументов (включая имя программы)

// argv[] — массив строк (аргументы)

}

Пример:

#include <stdio.h>

int main(int argc, char \*argv[]) {

printf("Программа: %s\n", argv[0]);

for (int i = 1; i < argc; i++) {

printf("Аргумент %d: %s\n", i, argv[i]);

}

}

Запуск:

./program hello 123

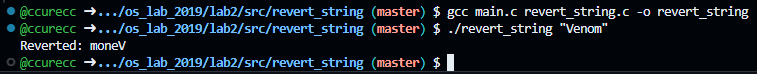
# Выведет:

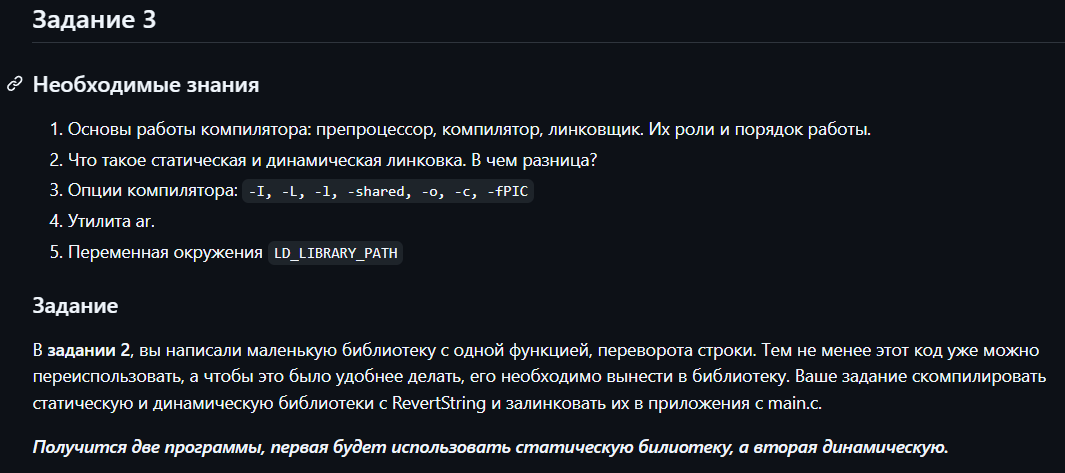
# Программа: ./program

# Аргумент 1: hello

# Аргумент 2: 123

Код поменял, все закоммитил, код мейна изучил, написал подробные комментарии





**1. Этапы работы компилятора**

**1. Препроцессор (gcc -E)**  
Обрабатывает директивы #include, #define, удаляет комментарии.  
**На выходе:** чистый C-код (.i файл).

**2. Компилятор (gcc -S)**  
Преобразует код в ассемблер (.s файл).

**3. Ассемблер (gcc -c)**  
Генерирует объектный файл (.o).

**4. Линковщик (ld)**  
 Связывает объектные файлы и библиотеки в исполняемый файл.

2. Статическая vs динамическая линковка

Статическая Динамическая

Библиотеки (.a) вшиваются в бинарник Библиотеки (.so) подгружаются при запуске

Большой размер файла Меньший размер

Не зависит от системы Требует совместимых .so на целевой системе

Изменения в .so не влияют Обновления .so применяются автоматически

3. Опции компилятора

Опция Назначение Пример

-I Добавить путь к заголовочным файлам gcc -I/usr/local/include main.c

-L Добавить путь к библиотекам gcc -L/usr/local/lib -lm

-l Связать библиотеку gcc -lpthread main.c

-shared Создать динамическую библиотеку gcc -shared -o libfoo.so foo.c

-o Задать имя выходного файла gcc main.c -o program

-c Только компиляция (без линковки) gcc -c file.c

-fPIC Генерация позиционно-независимого кода (для .so) gcc -fPIC -shared -o lib.so file.c

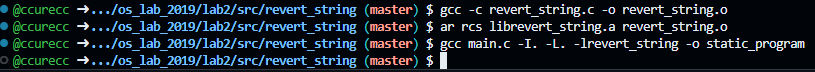
**4. Утилита ar**

Создает и управляет **статическими библиотеками** (.a).

**5. Переменная LD\_LIBRARY\_PATH**

Список путей для поиска **динамических библиотек** (.so) во время выполнения.

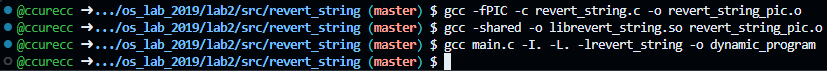
Создание статической библиотеки



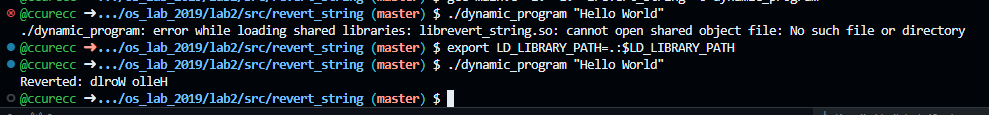
Запуск файла со статической библиотекой



Создание динамической библиотеки



Запуск файла с динамической библиотекой



1. **Статическая библиотека**:

ar rcs создает архив (.a) из объектных файлов

При линковке код библиотеки включается в исполняемый файл

Плюсы: не требует наличия библиотеки при запуске

Минусы: увеличивает размер программы

1. **Динамическая библиотека**:

-fPIC генерирует позиционно-независимый код

-shared создает разделяемую библиотеку (.so)

При запуске требуется наличие библиотеки

Плюсы: экономия памяти, возможность обновления без перекомпиляции

1. **Ключевые опции gcc**:

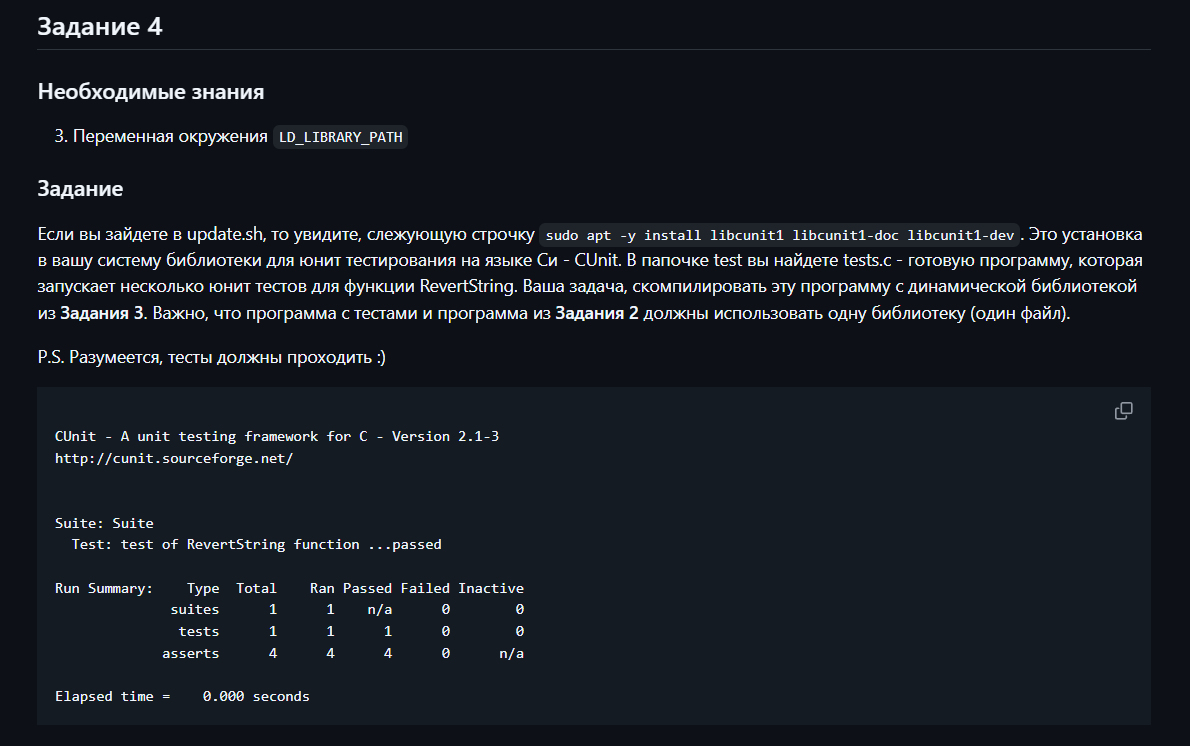
-I. - искать заголовочные файлы в текущей директории

-L. - искать библиотеки в текущей директории

-lrevert\_string - линковаться с librevert\_string.a/so

-o - указать имя выходного файла

1. **LD\_LIBRARY\_PATH**:

Указывает дополнительные пути для поиска динамических библиотек

Тесты не компилировались в папке tests, поэтому я скопировал их в папку из второго и третьего задания  
