**ЗАДАНИЕ**

2. Написать статическую библиотеку с функцией hello\_from\_static\_lib() и

использовать ее в hello.c:

hello.c

*#include <stdio.h>*

*void hello\_from\_static\_lib();*

*int main() {*

*hello\_from\_static\_lib();*

*return 0;*

*}*

hello\_static.c

*#include <stdio.h>*

*void hello\_from\_static\_lib(void) {*

*printf("Hello, World! (static lib)\n");*

*}*

• Статическая библиотека — это набор объектных файлов (.o), объединённых в один файл

Когда программа компилируется со статической библиотекой, код из библиотеки копируется в исполняемый файл. В результате готовая программа становится "самодостаточной" и не требует дополнительных файлов при запуске.

На Unix/Linux/macOS стандартные статические библиотеки хранятся в системных директориях, например:

/usr/lib/

/usr/local/lib/

Статические библиотеки подключаются на этапе компиляции и линковки.

После компиляции библиотека больше не нужна – её код уже включён в исполняемый файл.

статические библиотеки увеличивают размер исполняемого файла, но упрощают распространение.

*gcc hello\_static.c -c -o hello\_static.o -Wall*

• gcc - запускает компилятор GCC.

• hello\_static.c - исходный файл, который компилируем.

• -c - (compile only) компилирует в объектный файл (.o), но не выполняет линковку (не создаёт исполняемый файл (это нужно для сборки библиотеки)). Без -c GCC сразу выполнял бы компиляцию + линковку, создавая исполняемый файл.

• -o hello\_static.o (Output file) - указывает имя выходного файла (в данном случае hello\_static.o).

Без этого флага GCC назвал бы его hello\_static.o по умолчанию, но с -c он мог бы создать a.out, если не указать -o.

*ar r libhello\_static.a hello\_static.o*

• ar (archive) — это утилита для создания, обновления и извлечения файлов архивов, используемых для статических библиотек.

Фактически, ar создаёт один файл (.a), содержащий несколько объектных файлов (.o). Это эквивалент .zip или .tar, но предназначен специально для объектных файлов, чтобы компилятор мог их использовать при сборке программ.

• r (replace) – добавляет hello\_static.o в библиотеку. Если файл уже есть, он заменяется новой версией.

• libhello\_static.a - имя создаваемой библиотеки.

• hello\_static.o - объектный файл, который добавляется в библиотеку.

• Если libhello\_static.a ещё не существует, ar создаст новый архив.

Если libhello\_static.a уже есть, то ar обновит объектный файл hello\_static.o внутри архива.

*ar: creating archive libhello\_static.a*

• Это значит, что библиотеки не существовало, и ar создал её с нуля.

*gcc hello.c -o hello -lhello\_static -L. -Wall*

• Эта команда компилирует hello.c в исполняемый файл hello, используя статическую библиотеку libhello\_static.a.

• -lhello\_static – подключение библиотеки

Флаг -l (-l<имя\_библиотеки>) сообщает компилятору, что нужно использовать статическую или динамическую библиотеку.

Флаг -lhello\_static означает libhello\_static.a.

GCC автоматически добавляет lib в начало и .a в конец.То есть, он ищет libhello\_static.a.

• -L. означает "искать в текущей директории

*./hello*

*Hello, World! (static lib)*

a. посмотреть исполняемый файл на предмет того будет ли функция

hello\_from\_static\_lib() unresolved. Почему?

*nm ./hello*

*0000000100000000 T \_\_mh\_execute\_header*

*0000000100003f64 T \_hello\_from\_static\_lib*

*0000000100003f38 T \_main*

*U \_printf*

• T (Text section) → Функция определена внутри исполняемого файла.

• U (Undefined) → Функция не определена, но нужна (unresolved symbol).

• Нет, не будет, потому что у неё статус T, а не U.

Это значит, что функция hello\_from\_static\_lib() уже встроена в hello и полностью определена.

• Если бы она была unresolved, то:

Она была бы помечена U (undefined).

Программа не запустилась бы, так как линкер не смог бы найти её реализацию.

• Почему она не unresolved?

Мы подключили библиотеку libhello\_static.a на этапе компиляции.

hello\_from\_static\_lib() реально используется в hello.c, поэтому компилятор добавил её код в hello.

b. где находится код этой функции?

1. В исходном коде (hello\_static.c)

Код функции hello\_from\_static\_lib() изначально написан в файле hello\_static.c:

2. В объектном файле (hello\_static.o)

Мы компилируем hello\_static.c в объектный файл с помощью команды: gcc hello\_static.c -c -o hello\_static.o -Wall

Компилятор превращает hello\_static.c в объектный файл (.o), который содержит скомпилированный машинный код функции.

3. В статической библиотеке (libhello\_static.a)

Далее, мы создаём статическую библиотеку:

ar r libhello\_static.a hello\_static.o

ar (архиватор) собирает hello\_static.o в архив libhello\_static.a.

Теперь libhello\_static.a содержит код hello\_from\_static\_lib().

4. В исполняемом файле (hello)

Теперь компилируем hello.c, подключая статическую библиотеку:

gcc hello.c -o hello -lhello\_static -L. -Wall

Файл hello.c ссылается на hello\_from\_static\_lib():

void hello\_from\_static\_lib(); // Объявление, но нет реализации

Линкер находит её в libhello\_static.a и добавляет в hello.