**ОБЩАЯ ТЕОРИЯ**

**Этапы компиляции в C**

Компиляция программы на C проходит через четыре основных этапа:

1) Препроцессинг (Preprocessing)

2) Компиляция (Compilation)

3) Ассемблирование (Assembly)

4)Редактирование связей (Linking)

1. **Препроцессинг** (cpp)

Препроцессор — это первый этап компиляции, который обрабатывает директивы препроцессора (например, #include, #define, #ifdef).

Что происходит:

- Подключение заголовочных файлов (#include).

- Замена макросов (#define).

- Обработка условных директив (#ifdef, #ifndef, #endif).

- Удаление комментариев.

2. **Компиляция** (cc1)

На этом этапе компилятор переводит C-код в промежуточный код на ассемблере.

Что происходит:

- Проверка синтаксиса и семантики.

- Оптимизация кода.

- Генерация ассемблерного кода.

3. **Ассемблирование** (as)

Ассемблер берет ассемблерный код и переводит его в объектный файл (машинные инструкции, .o).

Объектный файл содержит:

- Машинные инструкции процессора.

- Символьные таблицы (функции, переменные).

- Незавершённые ссылки на внешние функции.

4. **Редактирование связей** (Linking, ld)

Линковщик (ld) объединяет все объектные файлы (.o) и подключает стандартные библиотеки.

Что происходит:

- Разрешение внешних ссылок (printf → libc).

- Оптимизация расположения кода.

- Создание исполняемого файла (a.out или hello).

**Как работают #include?**

#include <stdio.h>

При компиляции #include <stdio.h> заменяется на содержимое файла stdio.h.

Это означает, что код библиотеки прямо вставляется в исходник.

**ЗАДАНИЕ**

1. Написать программу hello.c, которая выводит фразу “Hello world”:

hello.c

*#include <stdio.h>*

*void PrintHello(void) {*

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

*}*

*int main() {*

*PrintHello();*

*return 0;*

*}*

a. получить исполняемый файл;

*gcc hello.c -o hello -Wall*

• Флаг -Wall включает предупреждения компилятора.

• Объектный файл (.o) – это промежуточный файл, который создаётся компилятором после компиляции кода, но перед созданием исполняемого файла. Он содержит машинный код, но ещё не является полностью готовой программой.

Компиляция → объектный файл (.o)

Компилятор преобразует C-код в машинный код и создаёт объектный файл hello.o. Этот файл ещё нельзя запустить – в нём только скомпилированные инструкции, но без подключения стандартных библиотек.

• Исполняемый файл – это окончательный файл, который можно запустить. Он создаётся после компоновки (линковки) объектных файлов и библиотек.

Линковка → исполняемый файл

Линкер соединяет объектный файл с библиотеками (libc, где находится printf()) и создаёт исполняемый файл hello, который уже можно запускать.

b. посмотреть unresolved symbols (puts, printf) с помощью nm;

*nm ./hello*

*0000000100003f44 T \_PrintHello*

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

*0000000100003f60 T \_main*

*U \_printf*

• Unresolved symbols (неразрешённые символы) – это функции или переменные, которые используются в коде, но не определены в объектном файле. Они должны быть найдены во время линковки или во время выполнения программы.

• 0000000100003f44 – адрес в памяти, где находится \_PrintHello.

• T – символ "T" (Text section) означает, что функция определена в этом файле и лежит в разделе кода (текста).

• \_PrintHello – имя функции.

• 0000000100000000 – адрес начала исполняемого файла.

• T – значит, этот символ определён в разделе кода.

• \_\_mh\_execute\_header – это специальный символ, который указывает точку входа в исполняемый файл на macOS.

• 0000000100003f60 – адрес функции main().

• T – функция определена в этом файле и находится в разделе кода.

• \_main – это стандартное имя функции main() в исполняемом файле (на macOS добавляется \_).

• U – Unresolved (неразрешённый символ). Это означает, что printf не определён в этом файле.

• \_printf – имя функции printf(), которая лежит в библиотеке libc.

c. посмотреть зависимости (ldd);

*otool -L hello*

*hello:*

*/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1336.61.1)*

• При компиляции программа может использовать внешние библиотеки (например, стандартную libc, где лежат printf(), puts() и другие функции). Вместо того чтобы копировать код библиотек в исполняемый файл, программа просто ссылается на них.

• При запуске операционная система загружает нужные библиотеки в память и "подключает" их к программе. Это называется динамическое связывание.

• otool -L hello - показывает, какие библиотеки нужны программе.

• hello:

Это имя исполняемого файла, который анализируется.

• /usr/lib/libSystem.B.dylib – библиотека, которая подключена к программе.

Это основная системная библиотека macOS, аналог libc в Linux.

Включает в себя функции ввода-вывода (printf, puts, malloc и т. д.), работу с процессами, памятью, файловой системой и т. д.

• compatibility version 1.0.0 – минимальная версия библиотеки, с которой программа совместима.

• current version 1336.61.1 – фактическая версия библиотеки в системе.

• Программа hello зависит от /usr/lib/libSystem.B.dylib, потому что в коде есть printf(), а эта функция находится в стандартной библиотеке.

macOS автоматически подгрузит библиотеку при запуске программы.

d. запустить

*./hello*

