Содержание

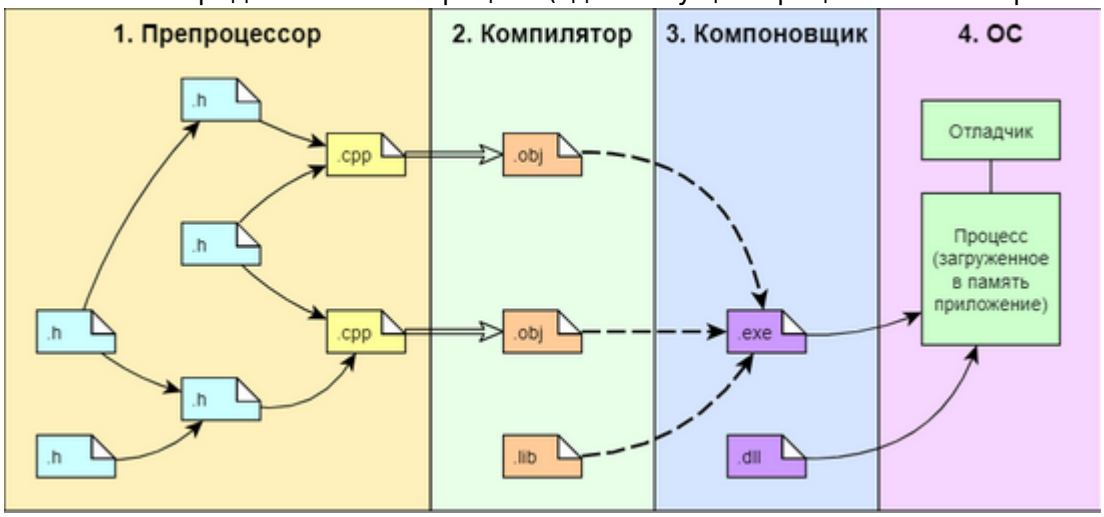
компиляция

(https://habr.com/ru/articles/478124/)

Программа, по сути своей, набор команд, описанных двоичными кодами, понятными процессору. Поскольу таким образом писать нереально, команды процессора можно описать как набор мнимоников языка ассемблера, компилируемых в машинные коды. Но и с ассемблером не все так просто, ведь его синтаксис отличен от процессора к процессору (и зависит от разрядности), потому сущевствуют языки высокого уровня: языки, чей исходный код проходит процесс сборки[1)] в язык ассемблера, компилируемый в двоичные коды.

Сборка программы - это компиляция исходного кода из одного или нескольких файлов и последующее связывание этих файлов в исполняемый файл, библиотеку динамической загрузки или статическую библиотеку.

Схематично представленный процесс (здесь опущен процесс ассемблирования):



Результат работы компилятора - перемещаемый объектный файл, а результатом работы компоновщика - файл, готовый к исполнению.

Компоновщик выполняет двойную роль:

1. Физически объединяет указанные в списке связей файлы в 1 файл.
2. Решает проблему внешних ссылок и обращений к памяти. Внешняя ссылка делается каждый раз, когда в коде файла упоминается код из другого файла. Это случается либо при вызове функции, либо при упоминании глобальной переменной.

Этапы компиляции

Препроцессинг

Препроцессор - это *макро процессор*, который преобразовывает вашу программу для дальнейшего компилирования. На данной стадии происходит происходит работа с препроцессорными директивами. Например, препроцессор добавляет (`#include`) хэдеры в код, убирает комментарии, заменяет макросы (`#define`) их значениями, выбирает нужные

куски кода в соответствии с условиями `#if`, `#ifdef` и `#ifndef`.

Хэдеры, включенные в программу с помощью директивы `#include`, рекурсивно проходят стадию препроцессинга и включаются в выпускаемый файл. Однако, каждый хэдер может быть открыт во время препроцессинга несколько раз, поэтому, обычно, используются специальные препроцессорные директивы, предохраняющие от циклической зависимости.

файлы программы проходят препроцессинг независимо друг от друга.

Компиляция / Ассемблирование

Сначала файлы, прошедшие препроцессинг проходят через лексический анализатор, выявляющий синтаксические ошибки. Если ошибок нет - файлы передаются компилятору.

На данном шаге компилятор выполняет свою главную задачу - компилирует, то есть преобразует полученный на прошлом шаге код без директив в *ассемблерный код*. Это промежуточный шаг между высокоуровневым языком и машинным (бинарным) кодом.

Так как x86 процессоры исполняют команды на бинарном коде, необходимо перевести ассемблерный код в машинный с помощью **ассемблера**. Ассемблер преобразовывает ассемблерный код в машинный код, сохраняя его в **объектном файле**.

Объектный файл - это созданный ассемблером промежуточный файл, хранящий кусок машинного кода. Этот кусок машинного кода, который еще не был связан вместе с другими кусками машинного кода в конечную выполняемую программу, называется *объектным кодом*.

Далее возможно сохранение данного объектного кода в **статические библиотеки** для того, чтобы не компилировать данный код снова.

Компоновка / Линковка

Объектных файлов может быть много и нужно их всех соединить в единый исполняемый файл с помощью компоновщика (линкера). Он связывает все объектные файлы и статические библиотеки в единый исполняемый файл, который мы и сможем запустить в дальнейшем. Для того, чтобы понять как происходит связка, следует рассказать о *таблице символов*. Линкер добавляет только те реализации библиотечных ф-ий, которые используются в программе.

[1)]

чаще всего для упрощения называемого компиляцияей, хотя компиляция - лишь один из этапов сборки