Компиляция и линковка

Гусев Илья, Булгаков Илья

Московский физико-технический институт

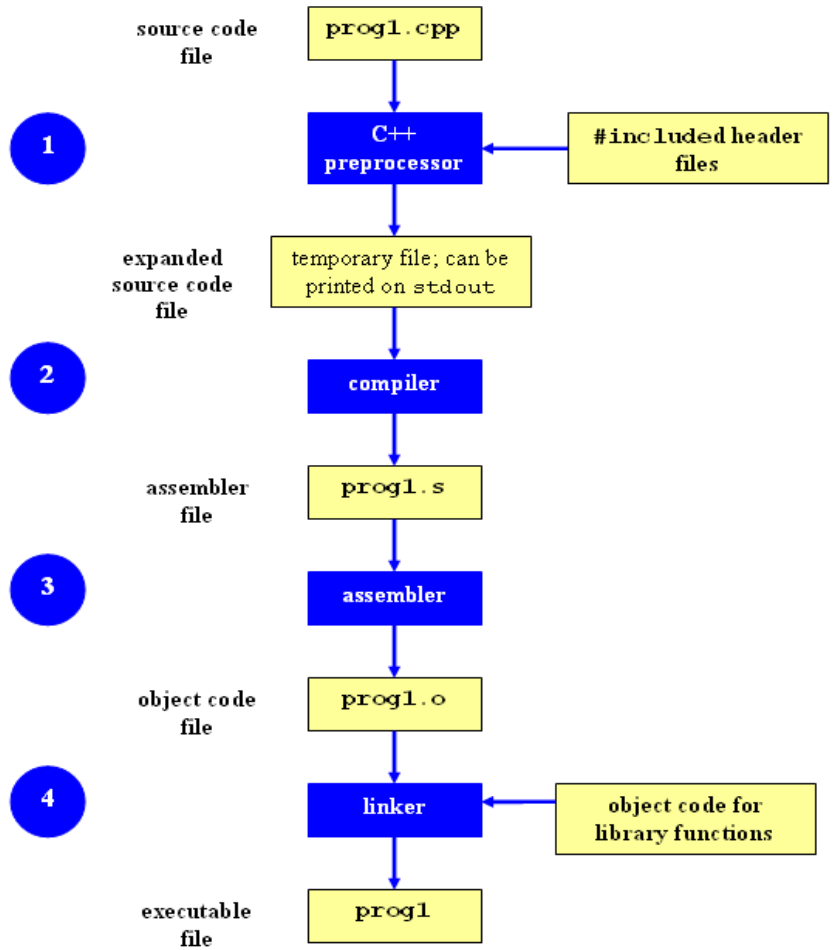
Москва, 2018

Содержание

Общая схема

g++ prog1.cpp

1. Препроцессор копирует содержимое включённых файлов в исходный код, разворачивает макросы и подставляет константы, переопределённые с помощью #define.
2. Развёрнутый исходный код компилируется в платформозависимый ассемблер (.s). Включение вывода: savetemps
3. Ассемблерный код собирается в объектный код платформы (.o). Смотреть: nm, objdump.
4. Объектный код связывается с другими объектными файлами и библиотеками.

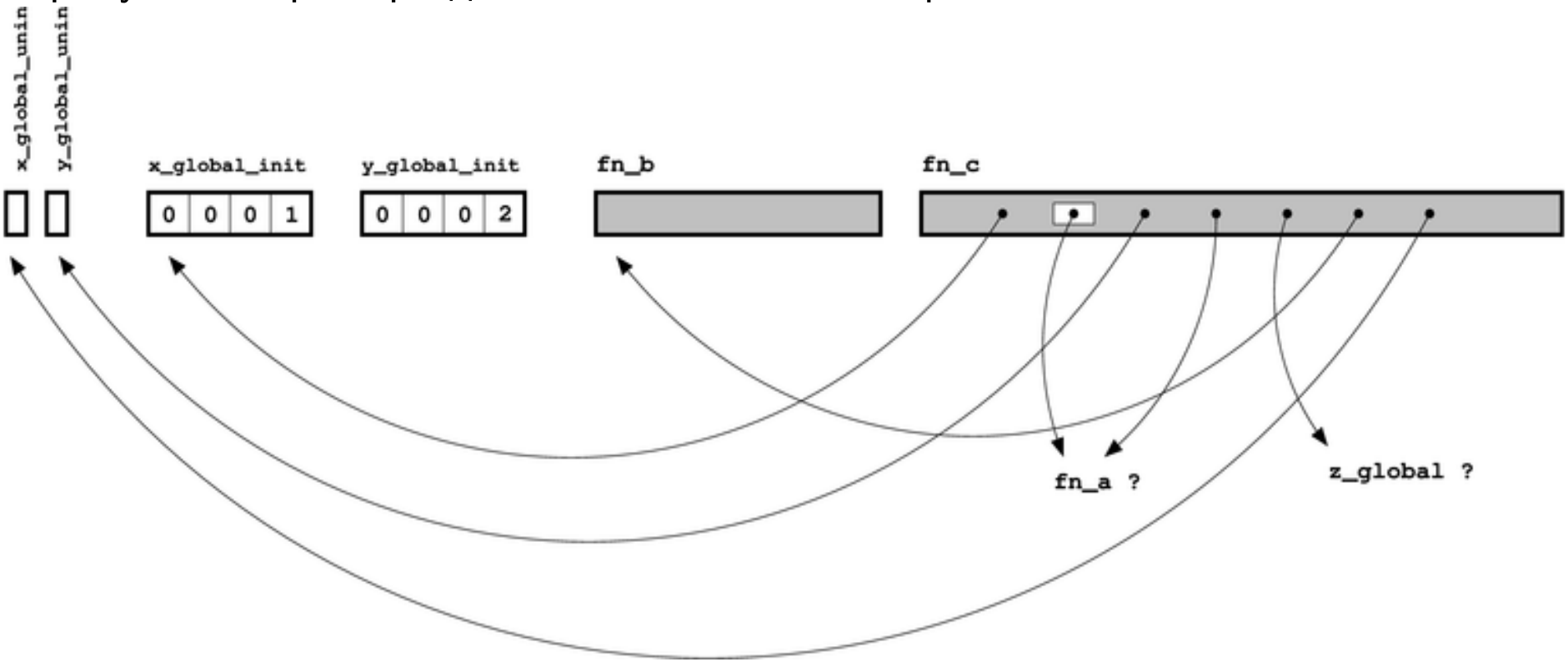


Компиляция

1. Токенизация
2. Построение промежуточного дерева по грамматике
3. Построение abstract-syntax tree (AST)
4. Построение платформо-зависимого дерева
5. Создание ассемблерного кода.

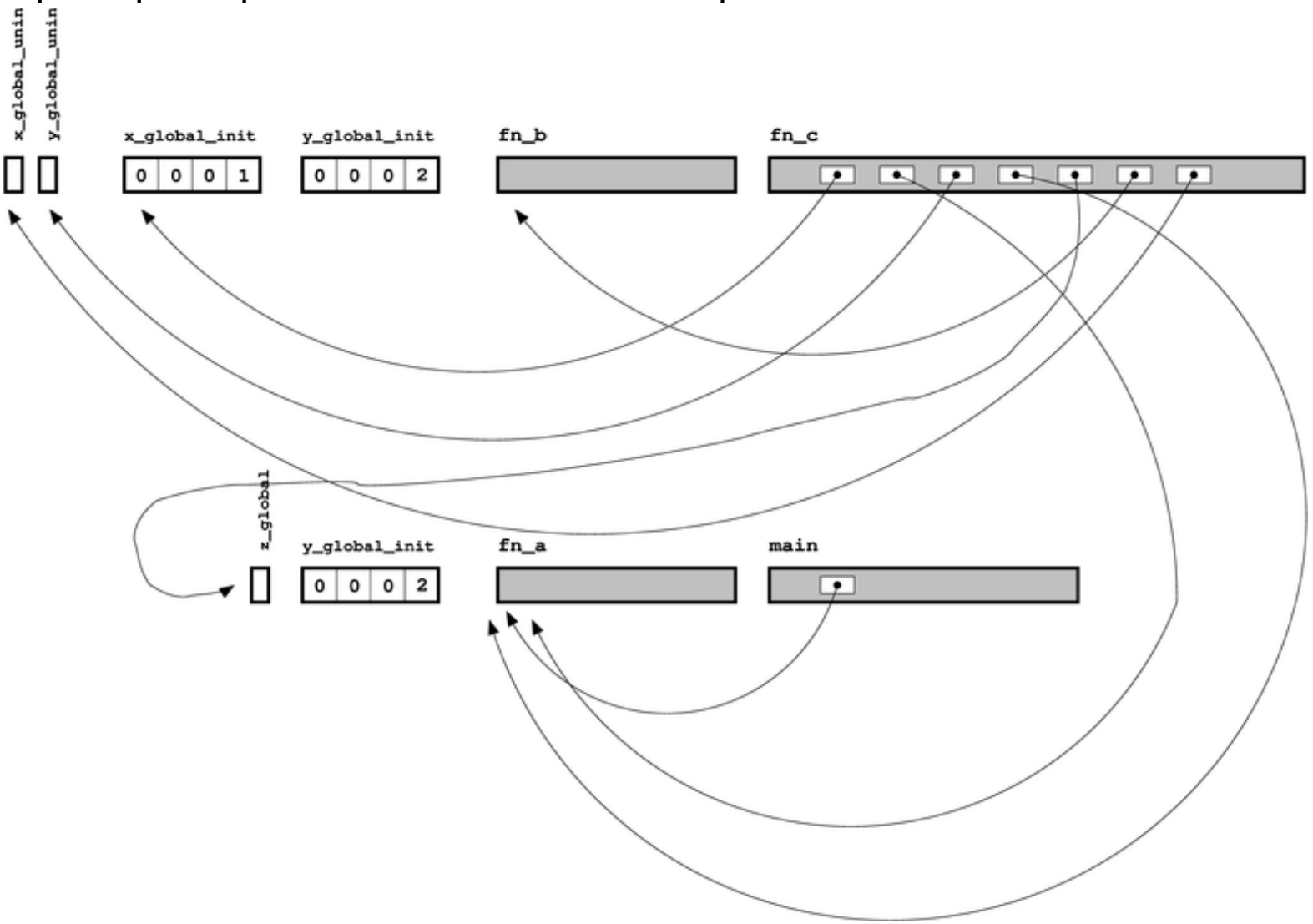
Линковка

На выходе с компилятора приходят объектные файлы. Они ничего не знают о существовании друг друга и их надо объединить в один, заполнив их "пропуски". Пример одиночного объектного файла:



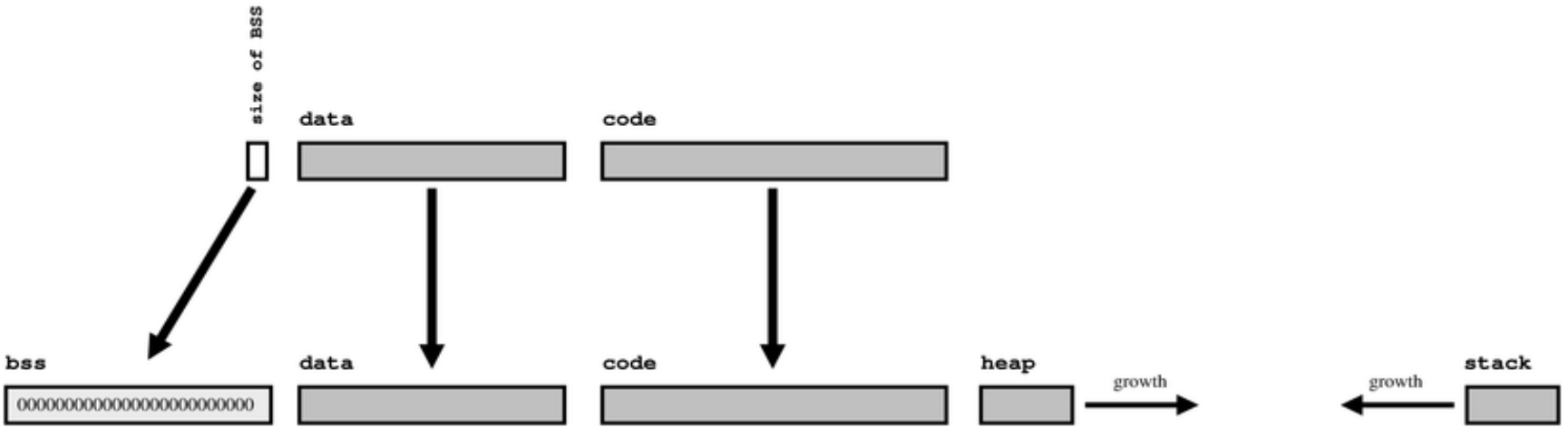
Линковка - собранный файл

Пример собранного исполняемого файла:



Загрузка исполняемого файла

1. Глобальные переменные кладутся в память процесса as is.
2. Локальные переменные кладутся на стек, который растёт при вызовах функций и уменьшается при их завершении.
3. Динамические переменные расходуют память кучи, системные malloc и free управляют этой частью процесса.
4. bss - фрагмент неинициализированных пременных, заполнен нулями
5. Кроме того, на затронут mapping файлов в память и пространство ядра (kernel space).



Статические библиотеки

1. Самая простая версия библиотек
2. Linux: lib*.a, Windows: *.lib
3. Библиотека может состоять из нескольких .o файлов.
4. Библиотека участвует в линковке объектными файлами, то есть тянется не вся библиотека, а только объектники, которые заполняют какой-либо 'пропуск'. При этом, тянется всё, что есть в этом объектном файле, в том числе новые 'пропуски'.
5. Линковка библиотек только после линковки частей программы, и только если есть 'пропуски'.
6. Библиотеки линкуются строго по порядку. Если вдруг библиотека-2 (идущая после библиотеки-1) требует что-то, что есть в незагруженном объектнике библиотеки-1, линкер это что-то не найдёт!

Проблемы статических библиотек

1. Каждый исполняемый файл содержит копию библиотеки → исполняемые файлы занимают кучу места.
2. Если мы хотим поменять код библиотеки нужно перелинковывать ВСЕ исполняемые файлы.

Shared библиотеки

1. *.so, *.dll, *.dylib.
2. Линкер просто оставляет некоторые 'пропуски' открытыми, записывая, что они должны заполняться из такой-то shared библиотеки.
3. Код библиотеки не включается в исполняемый файл.
4. Во время запуска программы (до исполнения main) операционная система натравливает маленькую версию линкера (ld.so) на оставшиеся 'пропуски'.
5. Если мы захотим поменять код, скажем, printf - нам не нужно перелинковывать все исполняемые файлы.
6. Загрузка shared библиотек происходит целиком, а не по объектным файлам!
7. *ldd* <имя исполняемого файла> чтобы посмотреть, какие библиотеки он захватывает.

Динамически загружаемые библиотеки (настоящие DLL)

1. Загрузка бибилиотеки в любом месте программы, не только до main.
2. dlopen и dlsym (или LoadLibrary и GetProcAddress).
3. dlopen - берёт библиотеку и загружает её в память процесса.
4. dlsym - берёт имя символа и возвращает указатель на его местонахождение в в загруженной области памяти.

Перегрузка функций

1. В Си нельзя перегружать функции, в C++ можно.
2. Пример: int max(int x, int y); float max(float x, float y);
3. А что делать линкеру, он же по имени символы ищет?

Mangling

1. Решение:
   1. int max(int x, int y) -> имя - _Z3maxii
   2. float max(float x, float y) -> имя - _Z3maxff
2. Всё сложнее для классов.
3. И ещё сложнее для шаблонов.
4. extern C у объявления и определения - убирает mangling для совместимости кода на Си и C++.
5. extern C игнорируется методами класса.

Полезные ссылки I

C++11 n3690
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3690.pdf

Статья про линковку
https://www.lurklurk.org/linkers/linkers.htmlcfile

The C++ compilation process
http://faculty.cs.niu.edu/ mcmahon/CS241/Notes/compile.html

Хабр: Организация памяти
https://habrahabr.ru/company/smart$_s$*oft*/*blog*/185226/