Platform Internals

Armine Hayrapetyan

Outline

- Preprocessing
- Compiling
- Assembly
- Linking

Compilation Process For C Programs

- Preprocessing
- Compiling
- Assembly
- Linking

Preprocessing

Preprocessing is the first step. The preprocessor obeys commands that begin with # (known as directives) by:

- removing comments
- expanding macros
- expanding included files

If you included a header file such as **#include <stdio.h>**, it will look for the **stdio.h** file and copy the header file into the source code file.

The preprocessor also generates macro code and replaces symbolic constants defined using **#define** with their values.

Compiling

Compiling is the second step. It takes the output of the preprocessor and generates assembly language, an intermediate human readable language, specific to the target processor.

Assembly

Assembly is the third step of compilation. The assembler will convert the assembly code into pure binary code or machine code (zeros and ones). This code is also known as object code.

Linking

Linking is the final step of compilation. The linker merges all the object code from multiple modules into a single one. If we are using a function from libraries, linker will link our code with that library function code.

In static linking, the linker makes a copy of all used library functions to the executable file. In dynamic linking, the code is not copied, it is done by just placing the name of the library in the binary file.

References

- https://medium.com/datadriveninvestor/compilation-process-db17c3b58e62