# Linker

A linker is a software utility that combines multiple object files and libraries into a single executable or shared library. It is an essential component of the software development process, particularly in compiled programming languages like C, C++, and Fortran.

When you write a program in a compiled language, it is typically divided into multiple source files. Each source file is individually compiled into an object file, which contains the machine code specific to that file. The linker's primary role is to resolve references between different object files and libraries, ensuring that all the necessary symbols (functions, variables, etc.) are properly linked together.

Here's a high-level overview of how the linker works:

1. Compilation: Each source file (.c, .cpp, .f) is separately compiled by the compiler, generating an object file (.o, .obj) for each source file. The object files contain machine code and additional information about symbols used and symbols provided.

2. Symbol resolution: The linker analyzes all the object files and libraries provided, looking for undefined symbols. An undefined symbol is a reference to a symbol (function or variable) that is used but not defined in the current object file.

3. Symbol table: The linker creates a symbol table that keeps track of all the symbols defined and referenced in the object files. It resolves the undefined symbols by searching for their definitions in other object files or libraries.

4. Linking: The linker performs the actual linking process, which involves combining the object files and resolving symbol references. It determines the final memory addresses for symbols and updates the necessary references accordingly.

5. Relocation: The linker performs relocation, adjusting the addresses of symbols based on the final memory layout. This step ensures that the linked executable or shared library can be loaded and executed correctly.

6. Output generation: Finally, the linker produces the linked output, which can be an executable file or a shared library, depending on the desired outcome. This output file is ready to be executed or used by other programs.

Linkers often provide additional functionality, such as support for dynamic linking, which allows the program to use shared libraries at runtime. They can also handle complex scenarios involving multiple object files, libraries, and external dependencies.

Different programming languages and platforms have their own linkers. For example, on Unix-like systems, the GNU linker (ld) is commonly used, while on Windows, the Microsoft linker (link.exe) is prevalent.

Understanding the role of the linker is crucial when working with compiled languages, as it ensures that all the necessary code and libraries are properly connected, resulting in a functional and executable program.