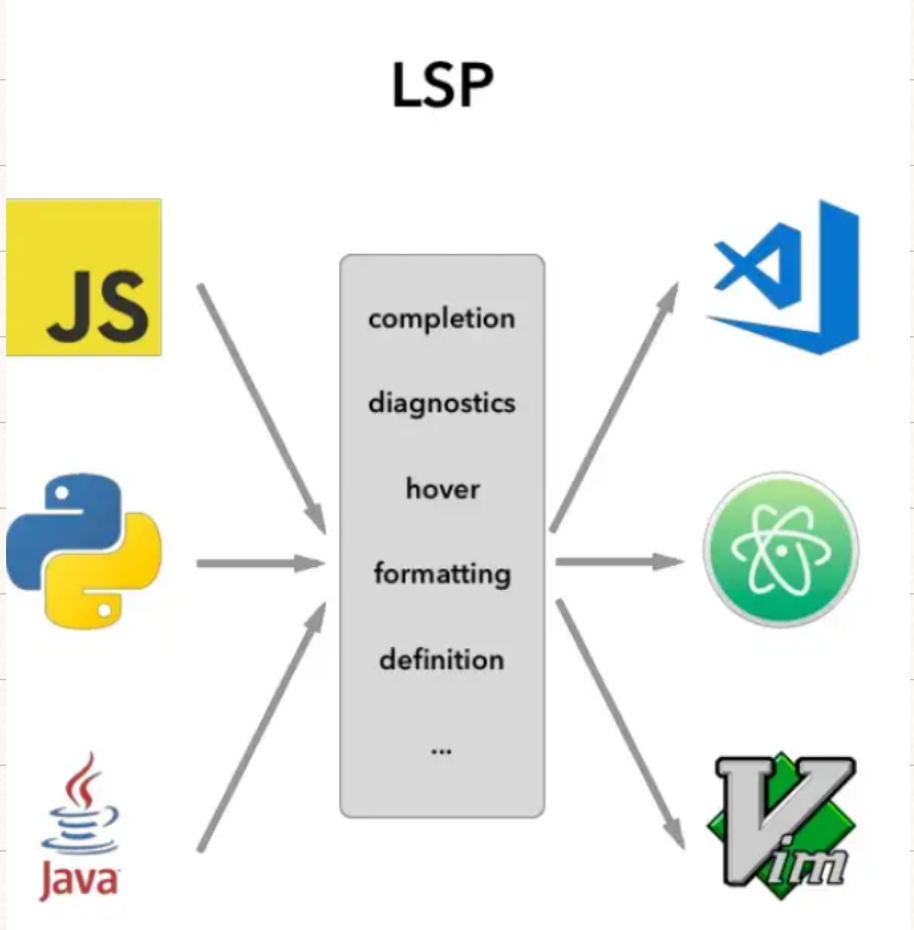
Language Server Protocol

The Language Server protocol is used between a tool (the client) and a language smartness provider (the server) to integrate features like auto complete, goto definition, find all references and alike into the tool.

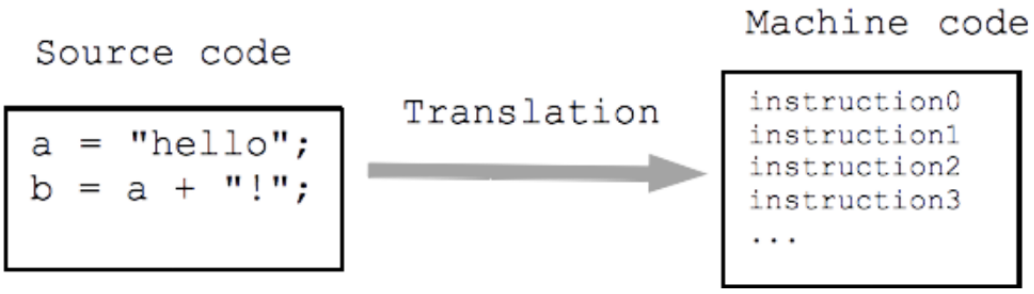
In other words, it is a standard way to communicate between editor and language tools, like linters, style controls and, generally speaking; everything that needs to check the source code.



LLVM

> *LLVM is a collection of the modular and reusable compiler and toolchain technologies. "LLVM" is not an acronym; it is the full name of the project.*

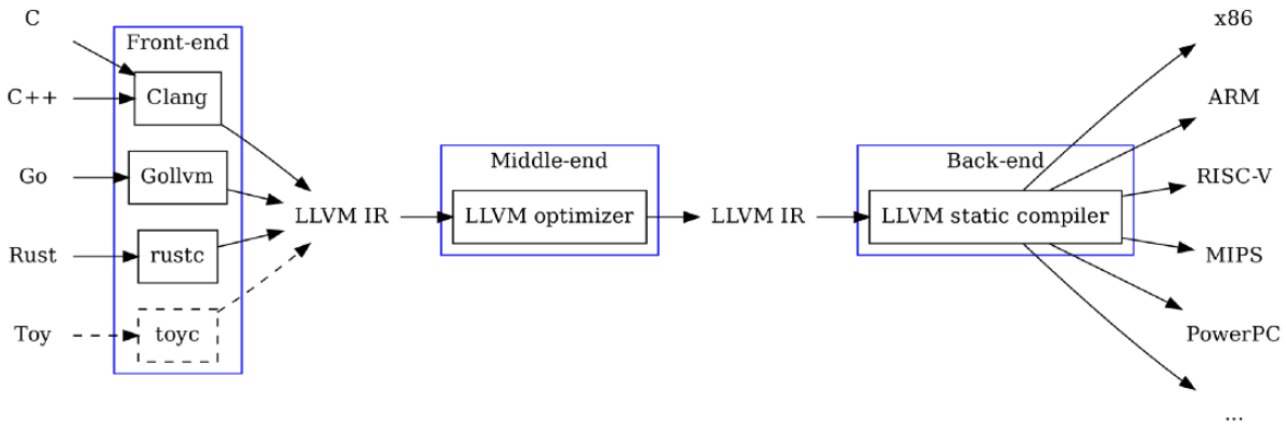
Let's take the below diagram,



Human readable to Machine readable

We write source code, in the language of our choice. This code is human-readable, but the machine on which it runs does not understand it. That's why we compile, which converts the source code and translates it into something which the machine understands.

LLVM is famous for **Intermediate Representation(LLVM IR)** and **IR Optimizer.** During a program compilation, the source language is converted into this **IR,** and the output is finally run on a target architecture (x86, ARM, etc.)



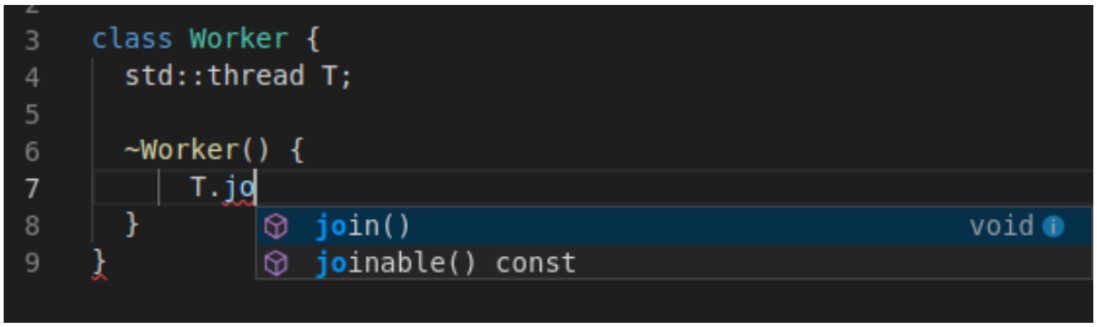
Compilers Overview

- **Front-end**: compiles source language to IR.
- **Middle-end**: optimizes IR.
- **Back-end**: compiles IR to machine code.

clangd

clangd understands your C++ code and adds smart features to your editor: code completion, compile errors, go-to-definition and more.

`clangd` is a *language server* that can work with many editors via a plugin. Here's Visual Studio Code with the clangd plugin, demonstrating code completion:



Project setup

To understand your source code, clangd needs to know your build flags. (This is just a fact of life in C++, source files are not self-contained).

By default, clangd will assume your code is built as `clang some_file.cc`, and you'll probably get spurious errors about missing `#include`d files, etc. There are a couple of ways to fix this.

`compile_commands.json`

This file provides compile commands for every source file in a project. It is usually generated by tools.

clangd will look in the parent directories of the files you edit looking for it, and also in subdirectories named `build/`. For example, if editing `$SRC/gui/window.cpp`, we search in `$SRC/gui/`, `$SRC/gui/build/`, `$SRC/`, `$SRC/build/`, ...

VS Code

`Ctrl + Shift + 左方括号`	折叠代码块
`Ctrl + Shift + 右方括号`	展开代码块

Alt + click	Insert cursor

先按 ctrl + k，再按 ctrl + i 可以查看鼠标悬浮信息

按 ctrl + tab 切换窗口，按 ctrl + 1 或 ctrl + 2 或类似，来切换文件组

ctrl + w 关闭当前 tab

F11	Toggle full screen
Ctrl + =/-	Zoom in /out
Ctrl + B	Toggle sidebar visibility
Ctrl + K Z	Zen Mode
Ctrl+`	显示集成终端 Show integrated terminal

文件状态栏中的 outline 很有用

(shift + F10)

→ 右键打开还有很多 peek 选项，可用键盘的 menu 键代替 right click

`F12` 跳转到定义，这个没啥好说的，跳转到函数或符号的定义，这是高频操作。

`Alt + F12` 以预览方式在当前页面显示定义，都是查看定义，相对 `F12` 的优点是不会跳出当前文件到定义文件，而是在当前文件打开一个小窗口预览

`Shift + F12` 查看光标所在函数或变量的引用，就像 `Alt +F12` 一样以预览方式在当前文件打开引用的文件列表。

Ctrl + Shift + P	Show command palette → 如可以搜索 restart language server
Ctrl + P	Go to file
Ctrl + F	Find → ctrl + shift + f 全局查找
Ctrl + H	Replace

Ctrl + S	Save
Ctrl + Shift + S	Save As
Ctrl + /	Toggle line comment
Ctrl + T	Show all Symbols

ctrl + shift + L refactoring some text

F2 rename symbol

Useful settings:

format on save

sticky scroll

remote plugin

C / C++ Debugging Tools

Valgrind is a flexible program for debugging and profiling Linux executables. It consists of a core, which provides a synthetic CPU in software, and a series of debugging and profiling tools. The architecture is modular, so that new tools can be created easily and without disturbing the existing structure.

Valgrind analyzes your application by running it on a synthetic CPU and instrumenting the existing application code as it is executed. It then prints "commentary" clearly identifying each process involved in application execution to a user-specified file descriptor, file, or network socket. The level of instrumentation varies depending on the Valgrind tool in use, and its settings, but it is important to note that executing the instrumented code can take 4-50 times longer than normal execution.

Valgrind can be used on your application as-is, without recompiling. However, because Valgrind uses debugging information to pinpoint issues in your code, if your application and support libraries were not compiled with debugging information enabled, recompiling to include this information is highly recommended.

Massif is a Valgrind tool that measures how much heap memory an application uses. It also has a visualizer.

Sanitizers

有一系列工具，如 address 和 thread sanitizers

Address Sanitizer (ASan)

A memory error detector for C / C++.

Very fast with an average slowdown of ~2x.

Integrated in LLVM and GCC.

编译选项

-fsanitize=address 使能Address Sanitizer工具

-fsanitize=leak 只使能Leak Sanitizer，检测内存泄漏问题

-fno-omit-frame-pointer 检测到内存错误时打印函数调用栈

-O1 代码优化选项，可以打印更清晰的函数调用栈

编译的时候，该 flag 需要加到所有 codes，包括 library。

CXXFLAGS 和 LDFLAG (linker) 都要加该 flag。

If a bug is detected, the program will print an error message and exit immediately.

conda

Verify that conda is installed and running on your system by typing:

```
conda --version
```

Common command	effect
conda install `package_name(==version)`	Install a certain (version) in the current activation environment `version`)Bag
conda search `package_name`	Find a package in Anaconda REPO
conda list (--name ENVNAME)	View Current / Specify Name Environment Installation Package
conda remove/uninstall `package_name`	Uninstall a package
Conda remove -n [or --name] `env_name` --all	Delete the Env_name environment and all the packages underout
conda update pkg_name	Update PKG
conda env export > environment.yaml	Export the package information of the current environment
conda env create -f[--file] environment.yaml	Create a new virtual environment with configuration files
conda create --clone ENVNAME --name NEWENV	Replication and rename an environment
conda clean -a	Remove Index Cache, Lock Files, Unused Cache Packages, And Tarballs.

使用 Miniconda，默认安装的包比较少，省空间。
使用 conda 创造虚拟环境，不同虚拟环境之间不会有包冲突。
初始环境叫 base。