Getting Started with C++

MICHAEL PRICE

Cppcon
The C++ Conference

2023

October 01 - 06

Welcome to CppCon 2023!

Join the #visual_studio channel on CppCon Discord
https://aka.ms/cppcon/discord

- Ask any questions
- Discuss the latest announcements

Take our survey

https://aka.ms/cppcon/start

> ***For what is spherical moves easily, and those bodies which are least stable and have the smallest surface of contact are easily furthered in their motion.*** Generally it is easier to further the motion of a moving body than to move a body at rest. This is likewise the case with all similar things.

attributed to Themistius (Byzantium, 4th century C.E.)

SAMBURSKY, SAMUEL. "SUBLUNAR MECHANICS."
In *The Physical World of Late Antiquity*, 65–66.
Princeton University Press, 1962.

Would you believe me if I told you that you can…

Get a working C++ developer environment…

On the most common operating system…

With the most commonly used tools…

Without using your credit card…

In less than 15 minutes?

Agenda

01

Obtaining tools for your platform

02

Code reuse through libraries

03

Building correct, secure, and safe systems

04

Planning for the future

05

Resources for learning modern C++

Agenda

01
Obtaining tools for your platform

02
Code reuse through libraries

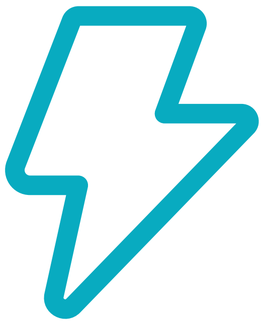
03
Building correct, secure, and safe systems

04
Planning for the future

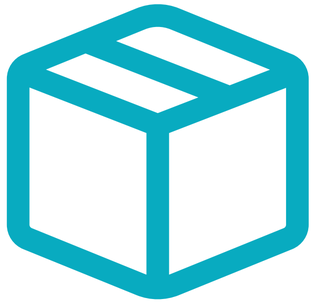
05
Resources for learning modern C++

Tools for your platform

Editors & IDEs



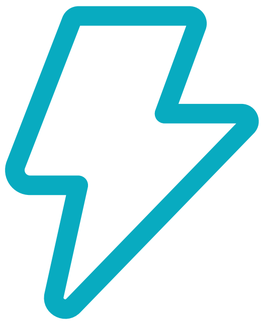
Compiler toolchains



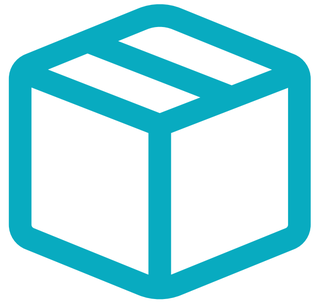
Build & project systems

Tools for your platform

Editors & IDEs



Compiler toolchains



Build & project systems

Editor *or* Integrated Developer Environment (IDE)

A matter of focus and out-of-the-box behavior

- **Focused on text editing**
- Syntax highlighting

- *Often extensible…*
- Build system knowledge
- Invoking compiler tools
- Debugger capabilities

- **Bundled experience**
- Usually tightly integrated with compiler toolchain and other build environment tools
- Sometimes focused on specific target environments
- Usually extensible, but often not needed

Editors & IDEs

Emacs

Sublime Text

Eclipse

Android Studio

Xcode

Neovim

Visual Studio Code

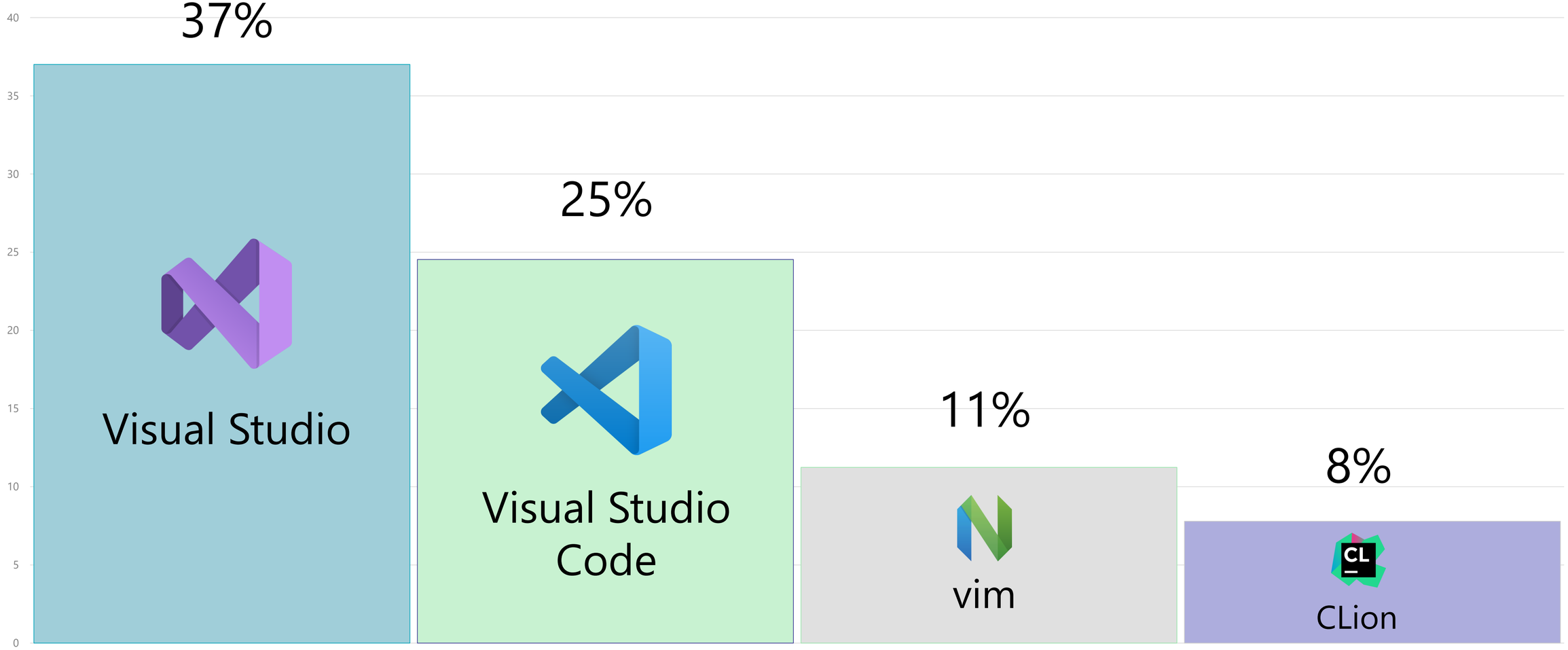
Qt Creator

CLion

Visual Studio

Popular C++ editors and IDEs

"Which development environments (IDEs) or editors do you use for C++ development?"
2023 ISO C++ Developer Survey "Lite"
*data shown is for "primary" IDEs/editors



Popular C++ editors and IDEs

Visual Studio

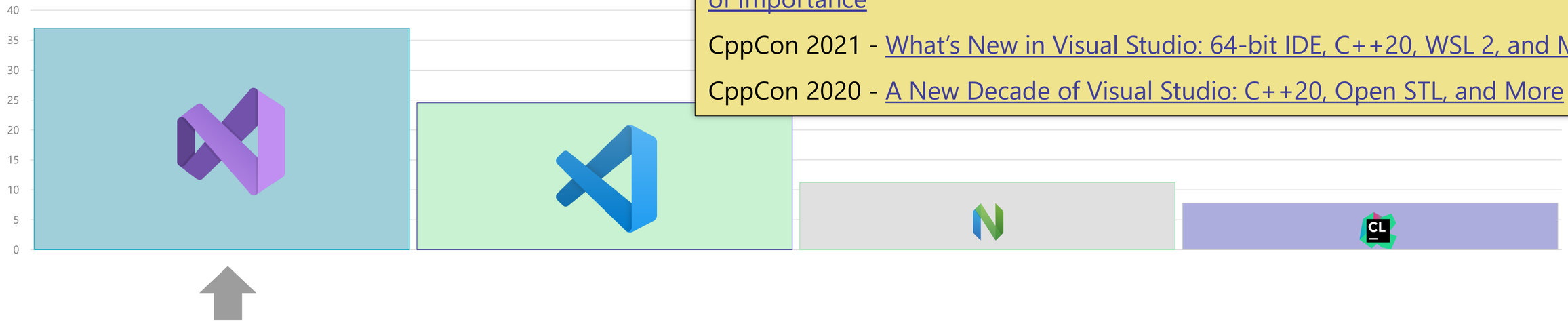
Go Deeper

CppCon 2023 - What's New in Visual Studio: CMake Debugger, Diagnostics Improvements, Video Games, and More

CppCon 2022 - New in Visual Studio 2022 - Conformance, Performance, Features of Importance

CppCon 2021 - What's New in Visual Studio: 64-bit IDE, C++20, WSL 2, and More

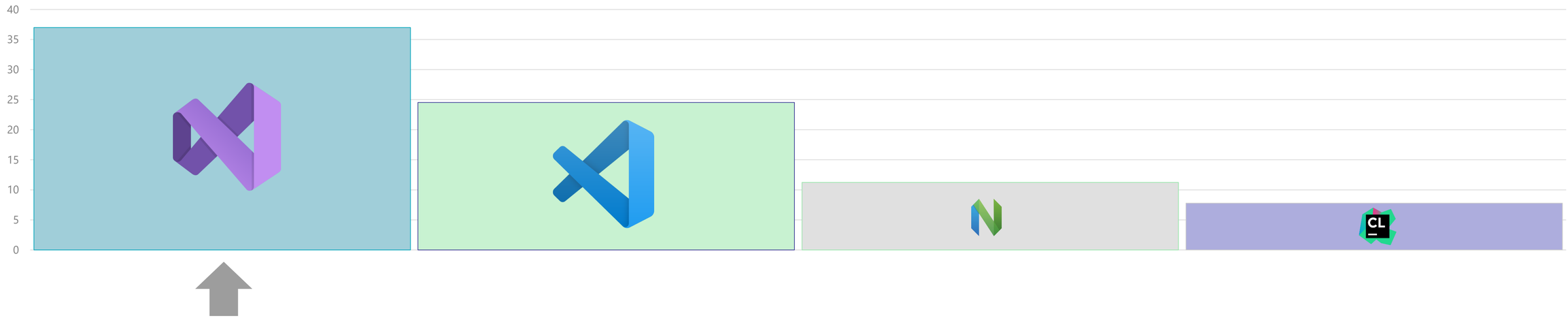
CppCon 2020 - A New Decade of Visual Studio: C++20, Open STL, and More



- https://visualstudio.microsoft.com
- Host: Windows only
- Easily build on Linux with Windows Subsystem for Linux (WSL)
- Comes in 3 editions: Community, Professional, Enterprise
  - Community is free in many cases; up to 5 concurrent users for non-Enterprises

Popular C++ editors and IDEs

Visual Studio



- `winget install Microsoft.VisualStudio.2022.Community --silent --override "--wait --quiet --add ProductLang en-us --add Microsoft.VisualStudio.Workload.NativeDesktop --includeRecommended"`

- https://aka.ms/getvisualstudio

Popular C++ editors and IDEs

Visual Studio Code

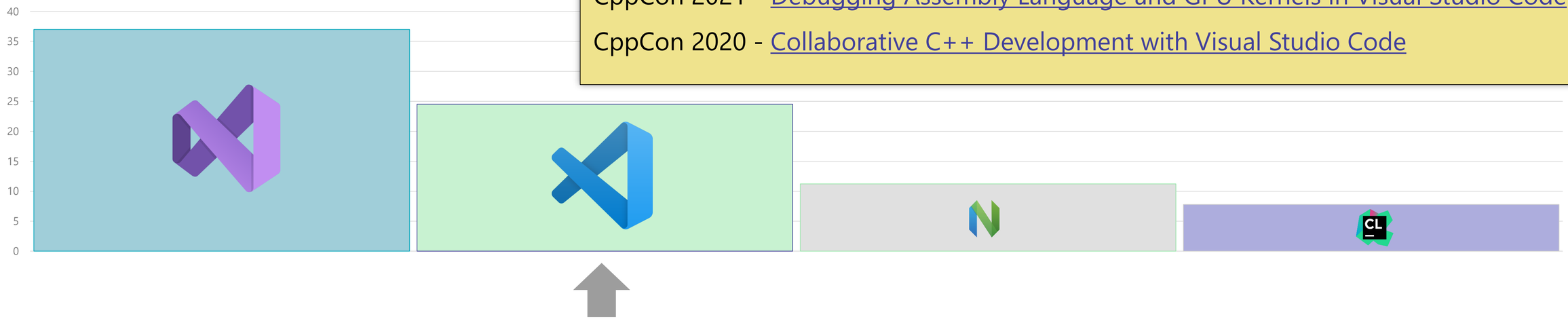
Go Deeper

CppCon 2023 - How Visual Studio Code Can Help You Code More Efficiently in C++

CppCon 2022 - What's New for You in Visual Studio Code: Clang-Tidy, makefile, CMake, GitHub and More

CppCon 2021 - Debugging Assembly Language and GPU Kernels in Visual Studio Code

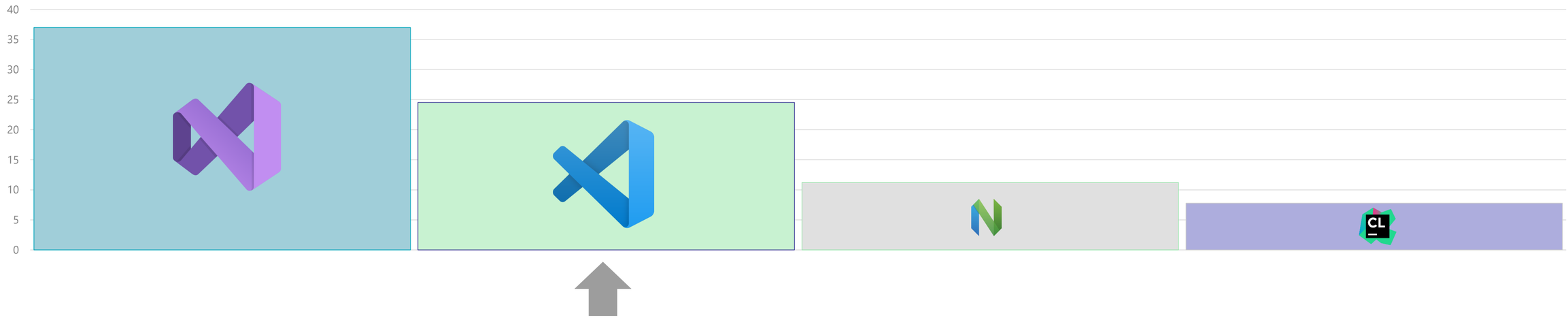
CppCon 2020 - Collaborative C++ Development with Visual Studio Code



- https://code.visualstudio.com
- Hosts: Windows, macOS, Linux, any modern web browser
- Install the C++ Tools extensions to get started easily with C++
- You will need to acquire a compiler toolchain separately
- Free license

Popular C++ editors and IDEs

Visual Studio Code



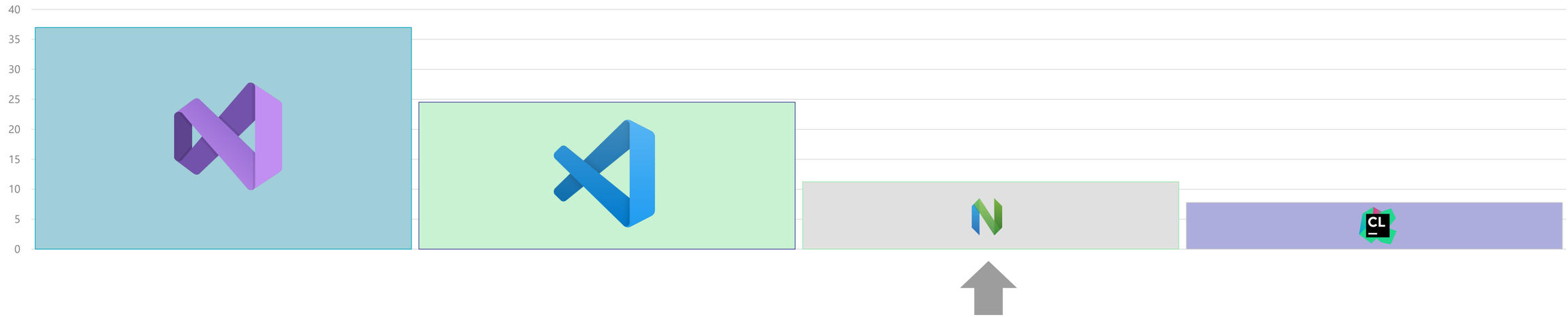
- Windows: `winget install Microsoft.VisualStudioCode -silent ; code --install-extension ms-vscode.cpptools-extension-pack`
- Ubuntu: `sudo snap install code --classic`
- macOS: use the downloaded installer

Popular C++ editor/IDEs

vim and Neovim

Go Deeper

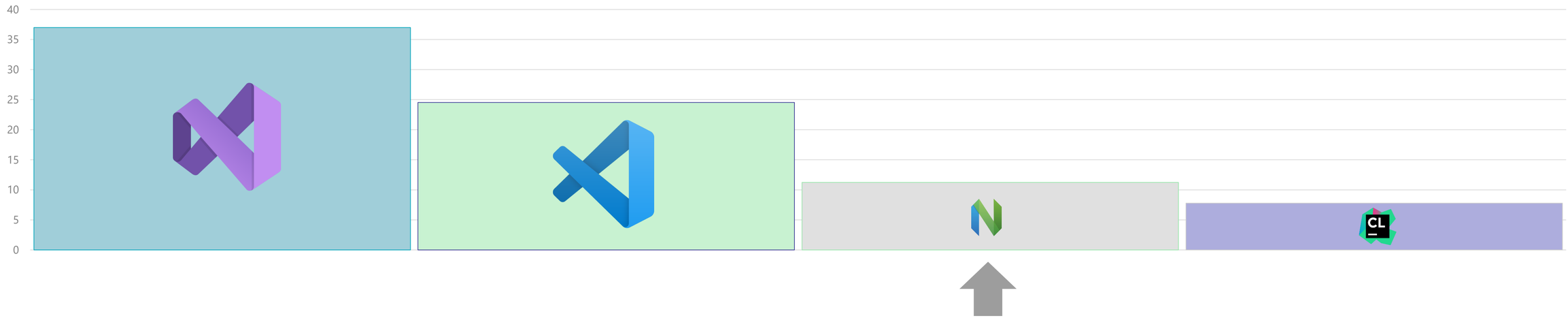
CppCon 2022 - C++ Coding with Neovim



- https://www.vim.org and https://neovim.io
- Hosts: Most Oses
- You will need to acquire a compiler toolchain separately
- Free license

Popular C++ editor/IDEs

vim and Neovim



- Windows: `winget install Neovim.Neovim`
- Ubuntu: `sudo apt-get install neovim`
- macOS: `brew install neovim` or `sudo port install neovim`

Popular C++ editors and IDEs

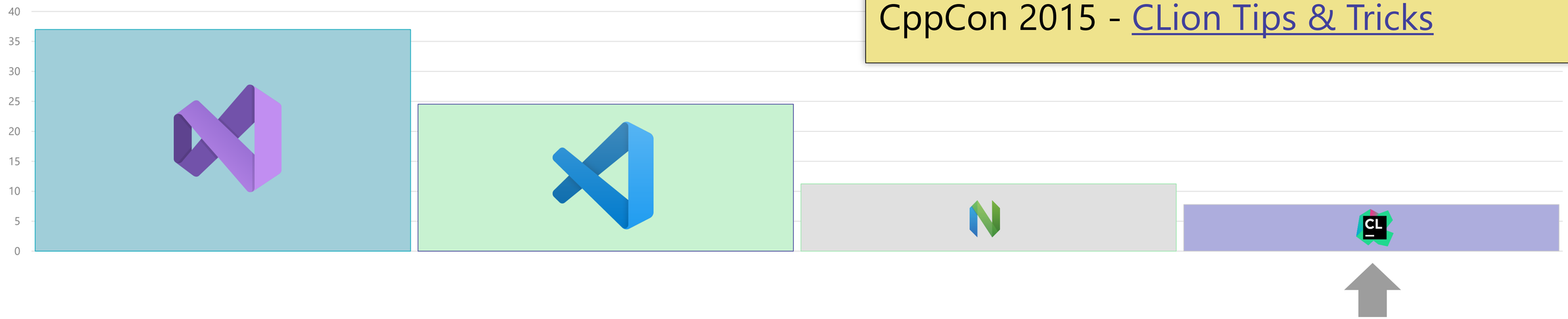
CLion

Go Deeper

CLion | The CLion Blog (jetbrains.com)

CLion - YouTube

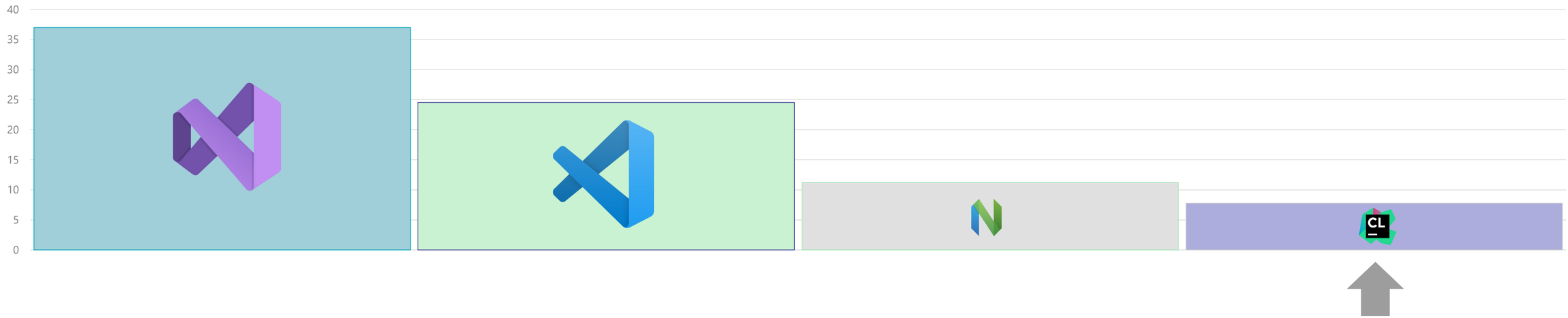
CppCon 2015 - CLion Tips & Tricks



- https://www.jetbrains.com/clion
- Hosts: Windows, macOS, Linux
- Only ships a compiler toolchain (MinGW/GCC) on Windows
- Free 30-day trial

Popular C++ editors and IDEs

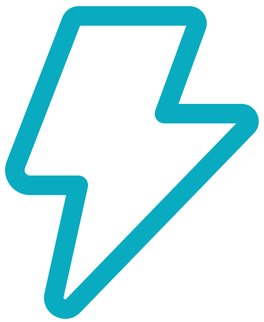
CLion



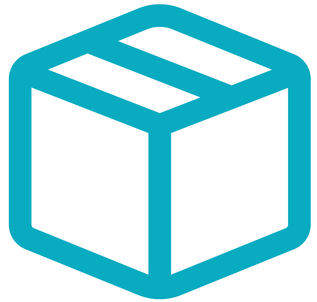
- All Platforms: Use the JetBrains Toolbox App
- Windows: Use the downloaded installer
- Ubuntu: `sudo snap install clion --classic`
- macOS: Use the downloaded installer

Tools for your platform

Editors & IDEs



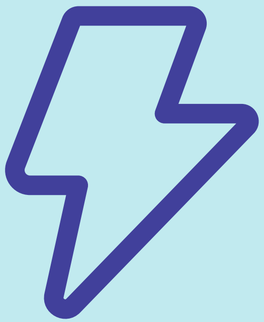
Compiler toolchains



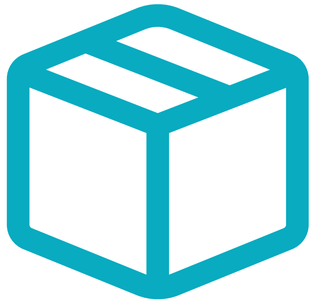
Build & project systems

Tools for your platform

Editors & IDEs



Compiler toolchains



Build & project systems

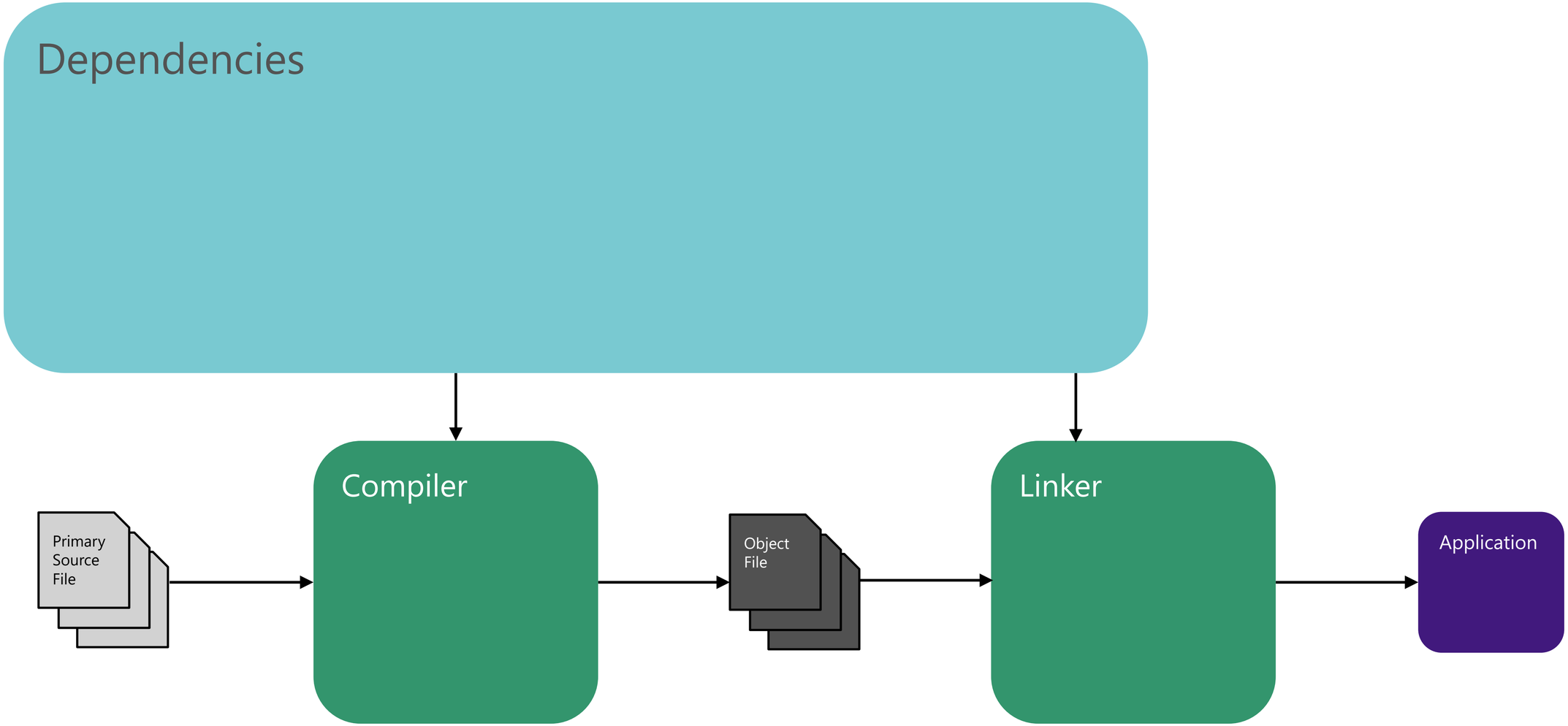
From source to executable

A brief description

Go Deeper

CppCon 2021 - Back to Basics: Compiling and Linking

CppCon 2020 - Back to Basics: The Structure of a Program



The Big 3 – Compiler Toolchains

The Big 3 – Compiler Toolchains

Microsoft Visual C++

Microsoft STL

Standard library implementation

cl.exe

Creates object files (.obj)

lib.exe

Creates static libraries (.lib)

link.exe

Creates executables (.exe) and dynamic libraries (.dll)

Visual Studio Debugger

Attaches to processes and lets developer “step” through code

The Big 3 – Compiler Toolchains

Microsoft Visual C++

- Available from following Visual Studio workloads
- Desktop development with C++

  `Microsoft.VisualStudio.Workload.NativeDesktop`

- Mobile development with C++

  `Microsoft.VisualStudio.Workload.NativeMobile`

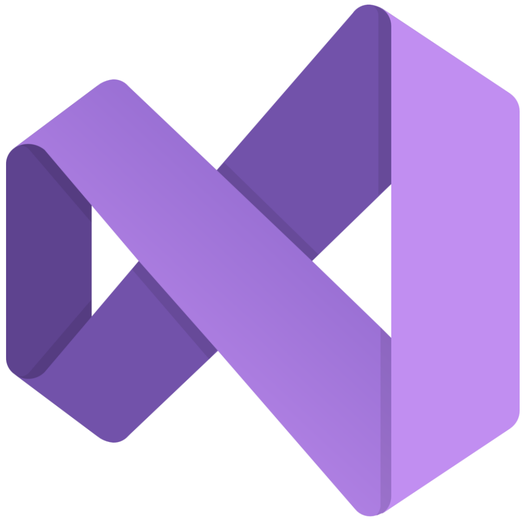
- Game development with C++

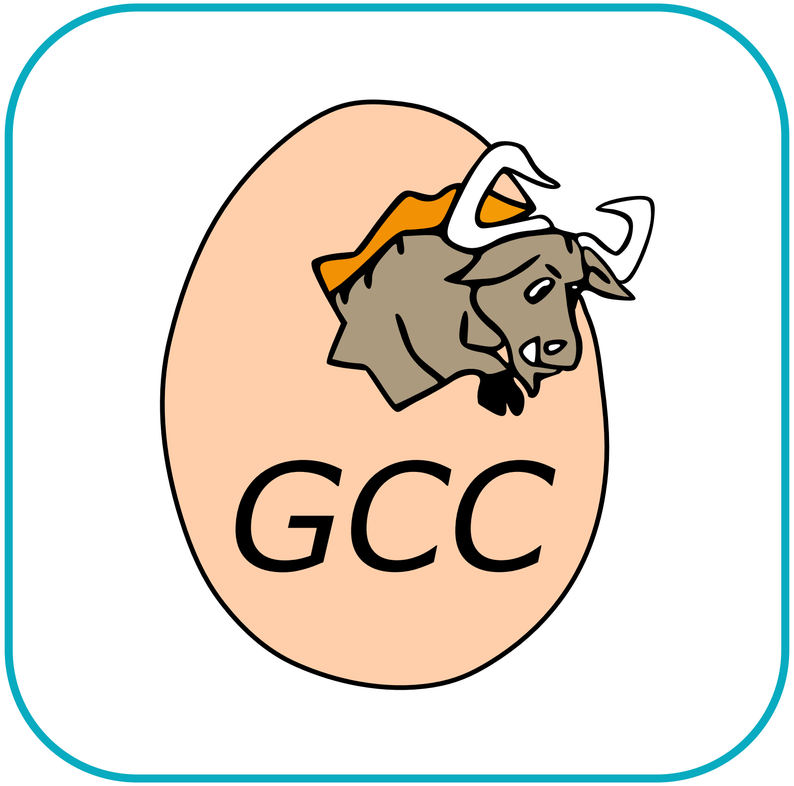
  `Microsoft.VisualStudio.Workload.NativeGame`

- Linux and embedded development with C++

  `Microsoft.VisualStudio.Workload.NativeCrossPlat`

The Big 3 – Compiler Toolchains





The Big 3 – Compiler Toolchains

GNU Compiler Collection (GCC)

The Big 3 – Compiler Toolchains

GNU Compiler Collection (GCC)

gcc / g++

Creates object files (.o)

ar

Creates static libraries (.a)

ld

Creates executables and shared libraries (.so)

gdb

Attaches to processes and lets developer “step” through code

libstdc++

Standard library implementation

The Big 3 – Compiler Toolchains

GNU Compiler Collection (GCC)

https://gcc.gnu.org/install/binaries.html

Building from source

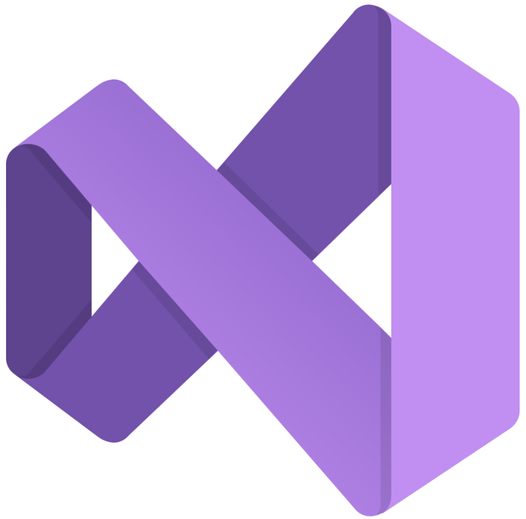
- Repository
  - `git clone git://gcc.gnu.org/git/gcc.git`
- https://gcc.gnu.org/install/configure.html
- https://gcc.gnu.org/install/build.html

The Big 3 – Compiler Toolchains

GNU Compiler Collection (GCC)

Ubuntu
• `sudo apt-get install build-essential`
Windows
• MSYS - `pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain`
macOS
• `sudo brew install gcc` • `sudo port install gcc`

The Big 3 – Compiler Toolchains



The Big 3 – Compiler Toolchains

Clang/LLVM

The Big 3 – Compiler Toolchains

Clang/LLVM

clang /
clang++

Creates object files (.o)

ar

Creates static libraries (.a)

lld

Creates executables and shared libraries (.so)

llvm

Attaches to processes and lets developer “step” through code

libc++

Standard library implementation

The Big 3 – Compiler Toolchains

Clang/LLVM

https://github.com/llvm/llvm-project/releases

Building from source

- Repository
  - `git clone https://github.com/llvm/llvm-project.git`
- https://llvm.org/docs/GettingStarted.html#getting-the-source-code-and-building-llvm

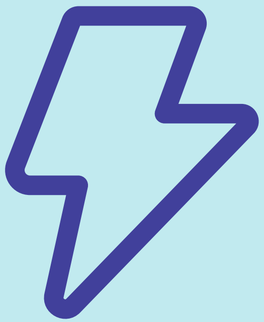
The Big 3 – Compiler Toolchains

Clang/LLVM

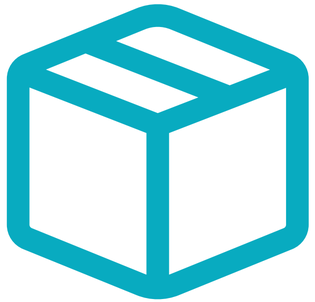
Ubuntu
• `sudo apt-get install clang`
Windows
• MSYS - `pacman -S mingw-w64-x86_64-clang`
macOS
• Install Xcode   • Apple Clang is not quite Clang/LLVM • `sudo port install clang-16`

Tools for your platform

Editors & IDEs



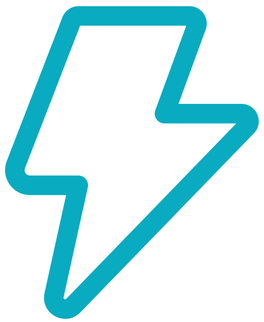
Compiler toolchains



Build & project systems

Tools for your platform

Editors & IDEs

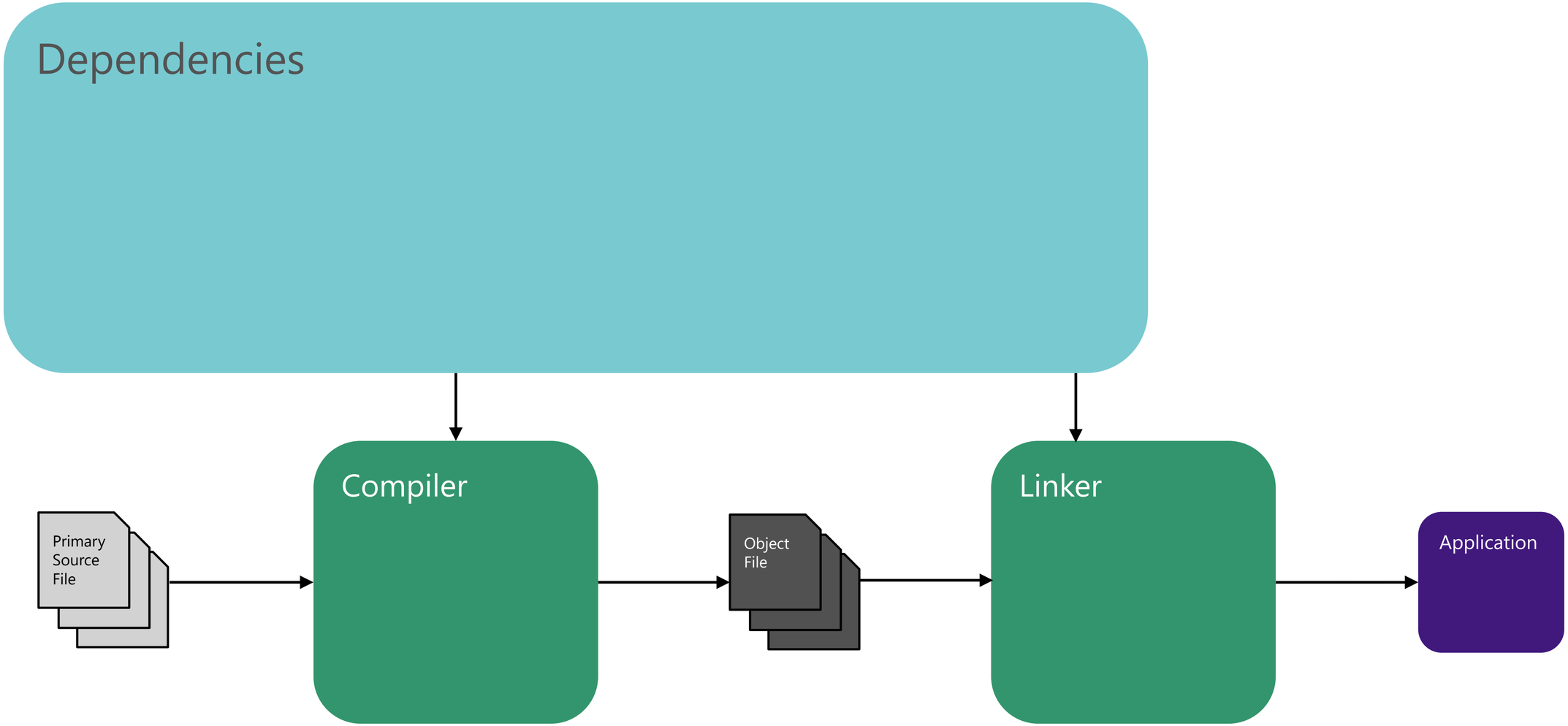


Compiler toolchains

Build & project systems

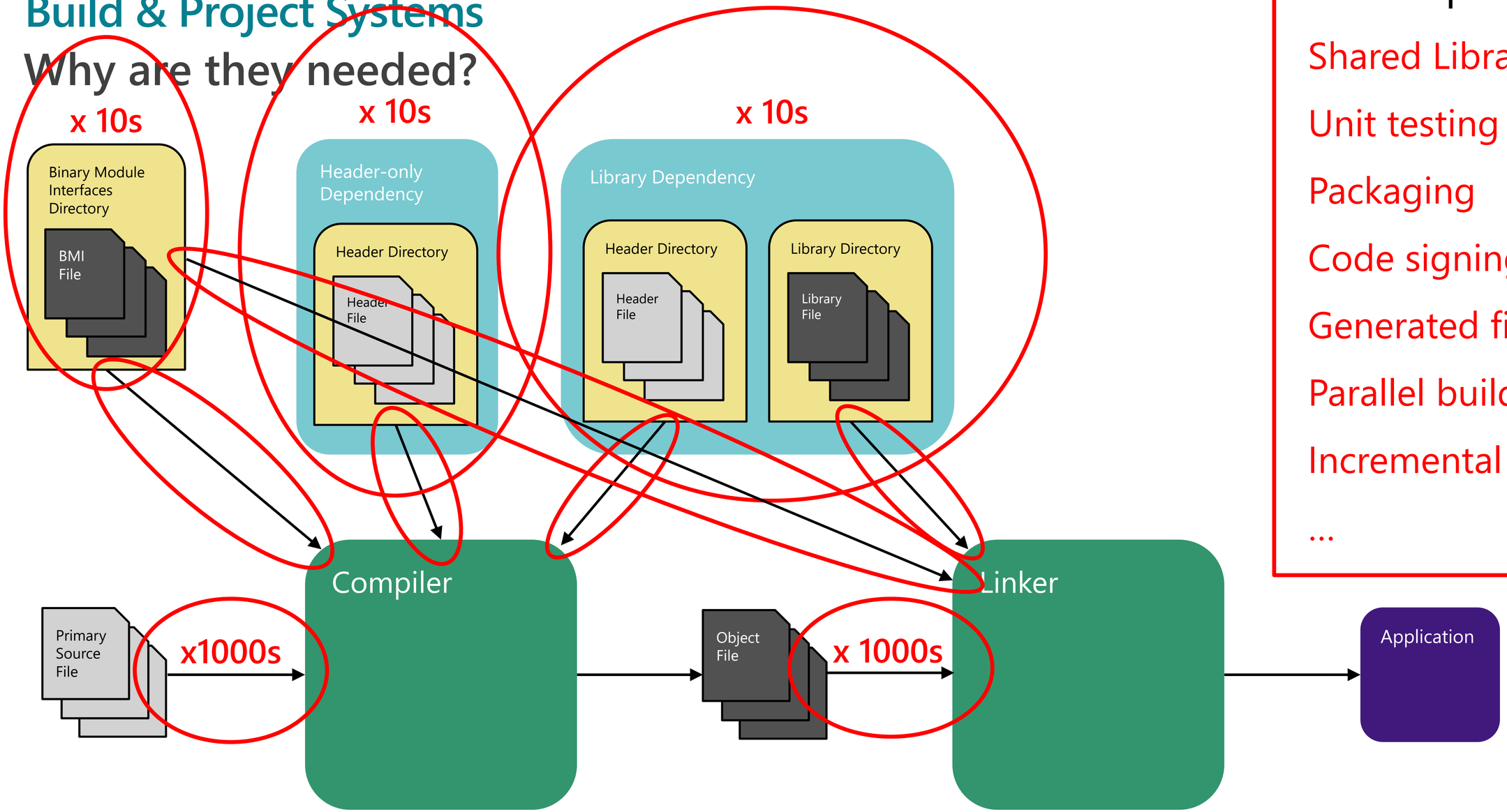
Build & Project Systems

Why are they needed?



Build & Project Systems

Why are they needed?

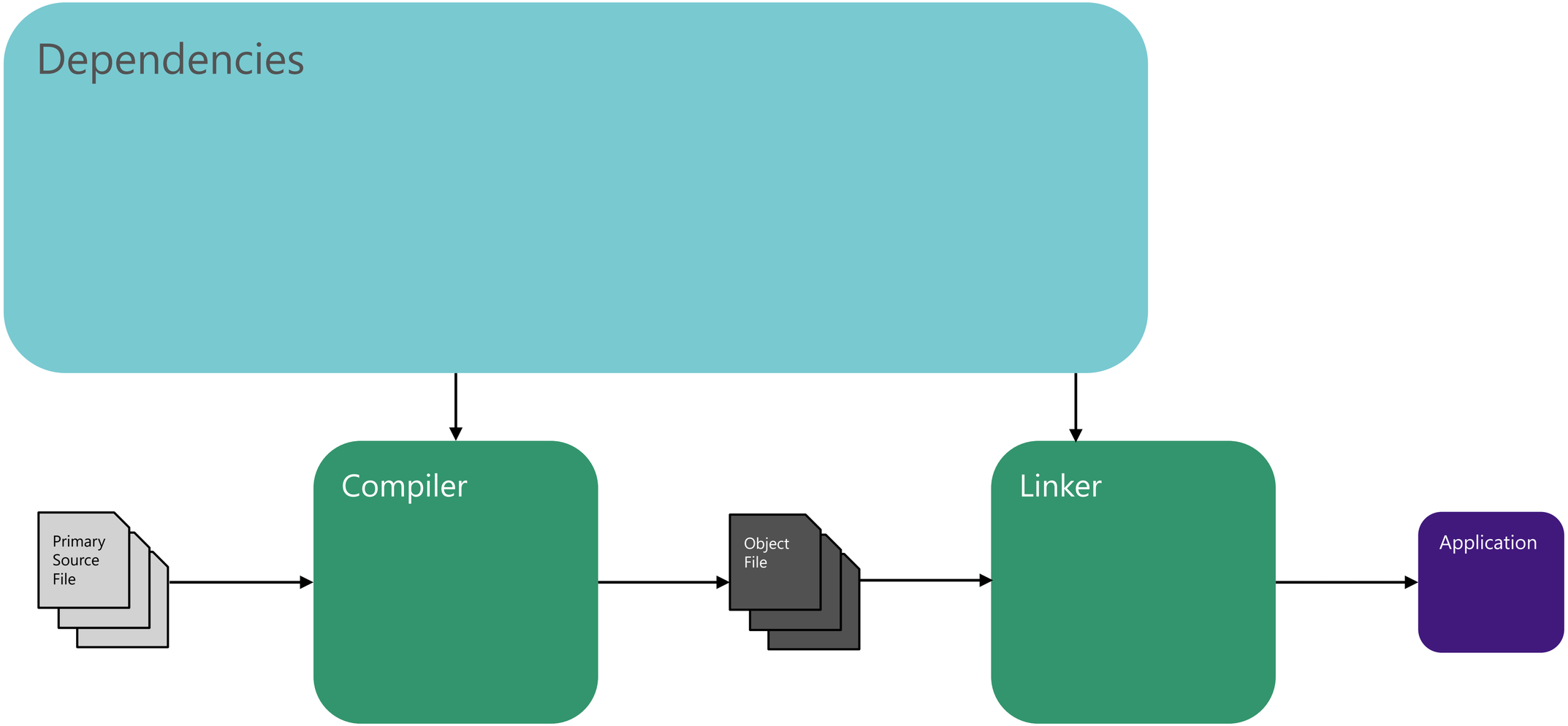


Not depicted

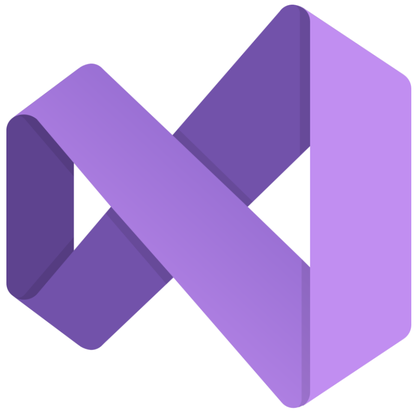
- Shared Libraries
- Unit testing
- Packaging
- Code signing
- Generated files
- Parallel builds
- Incremental builds
- …

Build & Project Systems

“Hide” some of the complexity



Build & Project Systems



MSBuild

GNU Make

忍者

Ninja

Build & Project Systems

Ships with Visual Studio

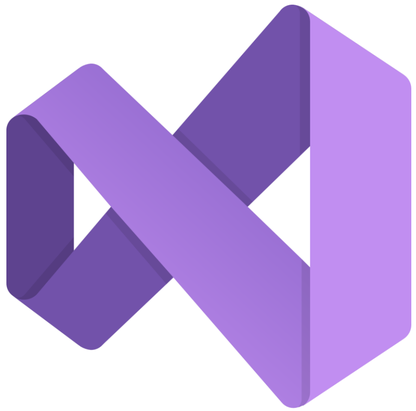
Runs on many platforms

Open source, built with .NET
https://github.com/dotnet/msbuild

Natively supports

- C++
- C#
- Typescript
- etc…

Build & Project Systems



MSBuild

GNU Make

忍者

Ninja

Build & Project Systems

Build & Project Systems

GNU Make

A part of the GNU ecosystem

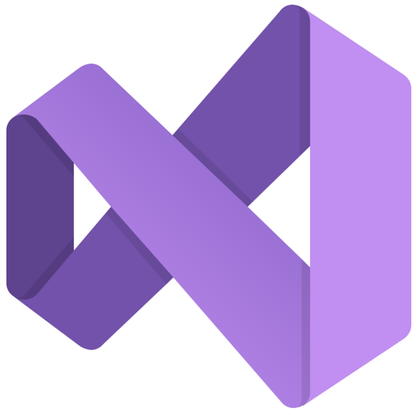
Runs on many platforms

Open source
https://www.gnu.org/software/make

Language agnostic

!!! Watch out for tabs vs. spaces

Build & Project Systems



MSBuild

GNU Make

忍者

Ninja

Build & Project Systems

Runs on many platforms

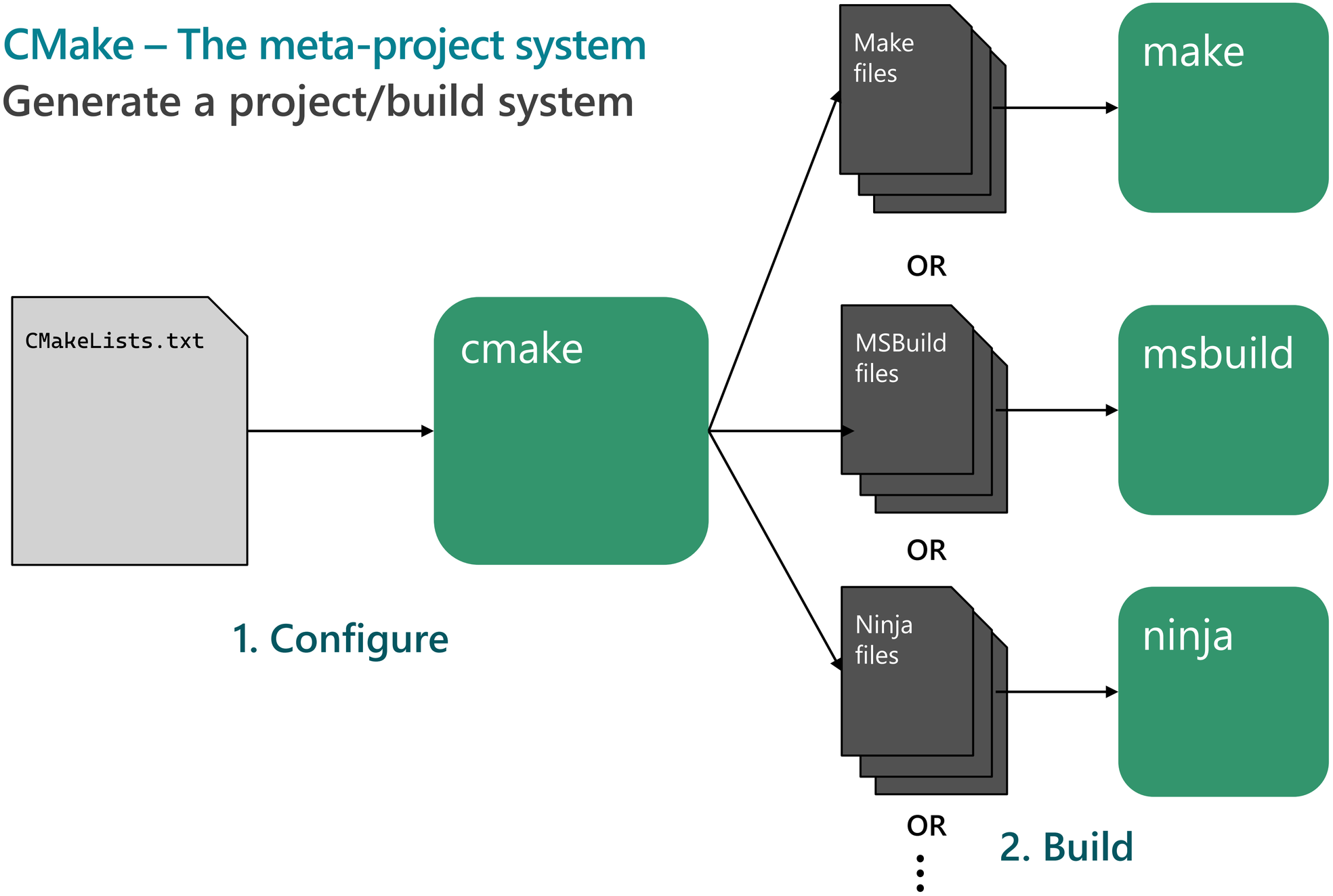
Open source
https://ninja-build.org

Language agnostic

Focused on minimal rebuilds
Intended to be used with a generator

CMake – The meta-project system

Generate a project/build system

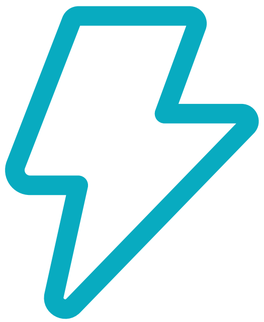


Tools for your platform

Go Deeper

CppCon 2018 - What to Expect from a Next-Generation C++ Build System

Editors & IDEs

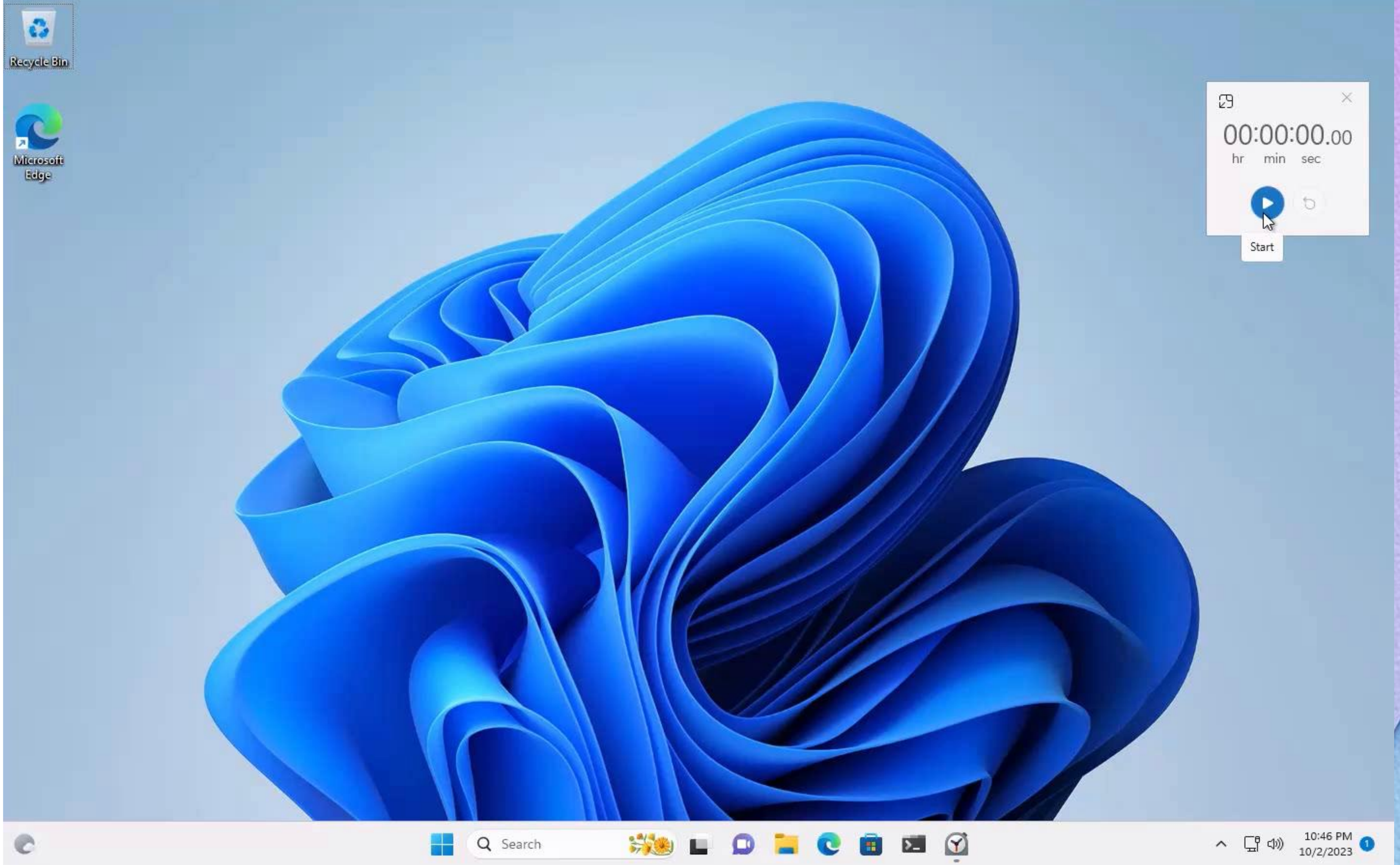


Compiler toolchains

Build & project systems

Demonstration

Zero-to-debugging in 10 minutes


Recycle Bin
Microsoft Edge
00:00:00.00
hr min sec
Start
Search
10:46 PM
10/2/2023

Agenda

01
Obtaining tools for your platform

02
Code reuse through libraries

03
Building correct, secure, and safe systems

04
Planning for the future

05
Resources for learning modern C++

Agenda

01

Obtaining tools for your platform

02

Code reuse through libraries

03

Building correct, secure, and safe systems

04

Planning for the future

05

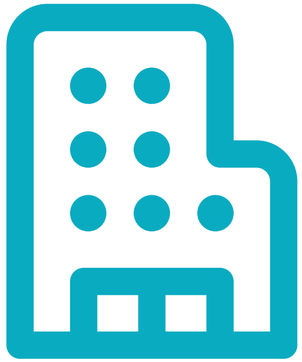
Resources for learning modern C++

You can’t just take two libraries and expect them to work together. Many do, but in general quite a few concerns must be addressed for successful joint use.

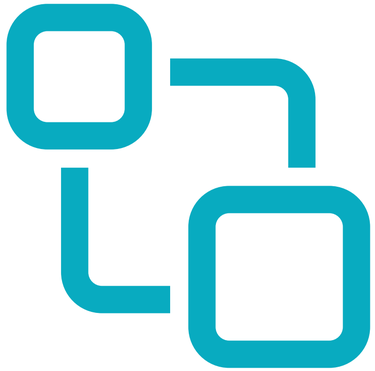
Bjarne Stroustrup

B. Stroustrup: The Design and Evolution of C++.
Addison Wesley, ISBN 0-201-54330-3. March 1994.

Code reuse with libraries



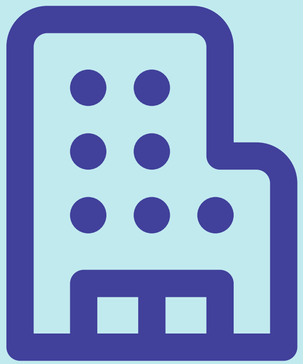
About Libraries



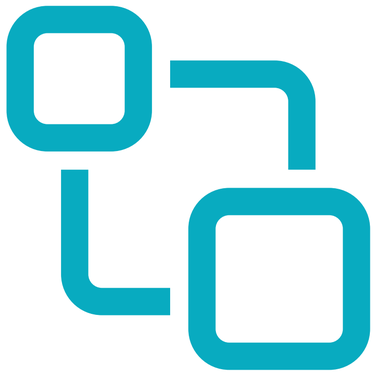
Managing Libraries

Finding Libraries

Code reuse with libraries



About Libraries



Managing Libraries

Finding Libraries

A quick word on libraries

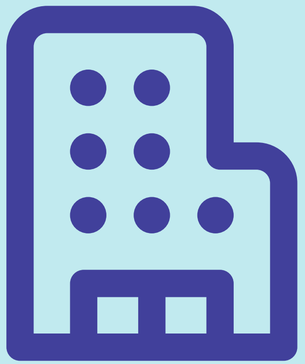
Why should you use libraries?

- Domain expertise
- Leverage prior effort
- Isolation of concerns

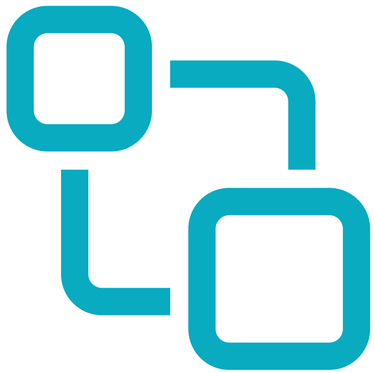
Things to think about

- Licensing
  https://tldrlegal.com
- Testing
- Integration

Code reuse with libraries



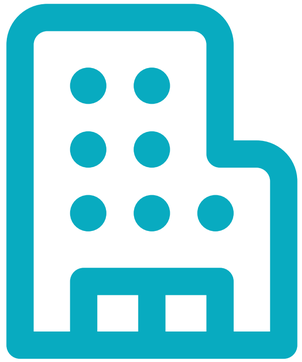
About Libraries



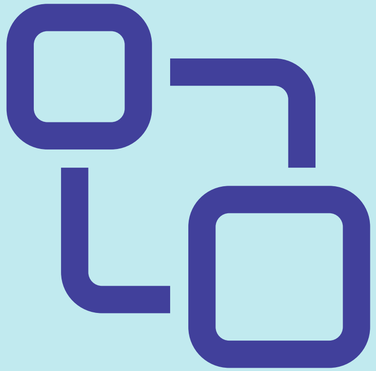
Managing Libraries

Finding Libraries

Code reuse with libraries



About Libraries



Managing Libraries

Finding Libraries

Installing a package manager

vcpkg

Go Deeper

vcpkg on Microsoft Learn - https://learn.microsoft.com/vcpkg

Windows: `iex (iwr -useb "https://aka.ms/vcpkg-init.ps1")`

Linux/macOS: `source <(wget -O - https://aka.ms/vcpkg-init.sh)`

Now bundled with Visual Studio C++ workloads

https://github.com/Microsoft/vcpkg

Installing a package manager

Conan

Go Deeper

Conan Docs - https://docs.conan.io

CppCon 2022 - What's New in Conan 2.0

Install Python 3 and pip

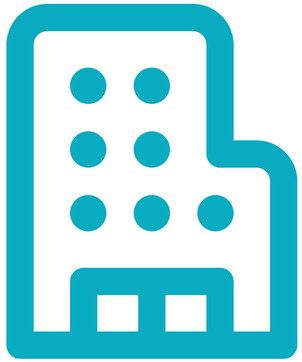
```
pip install conan
```

https://github.com/conan-io/conan

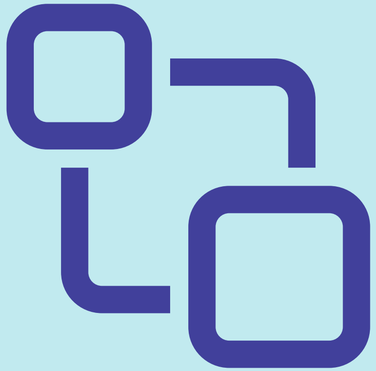
Code reuse with libraries

Go Deeper

CppCon 2022 - C++ Dependencies Don’t Have To Be Painful



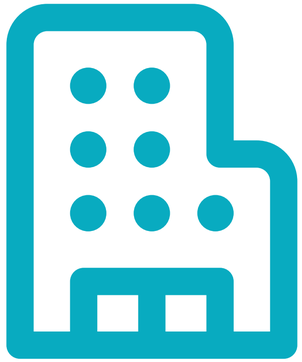
About Libraries



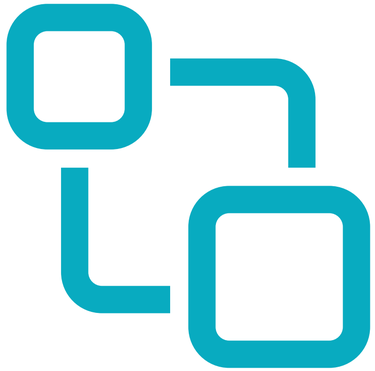
Managing Libraries

Finding Libraries

Code reuse with libraries



About Libraries



Managing Libraries

Finding Libraries

Finding a library

`vcpkg search <query-string>`

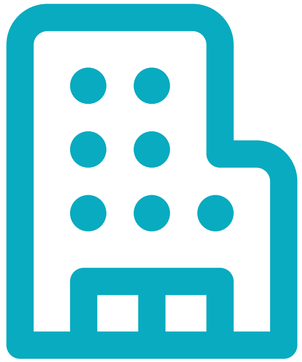
https://vcpkg.io/en/packages

`conan search <reference>`

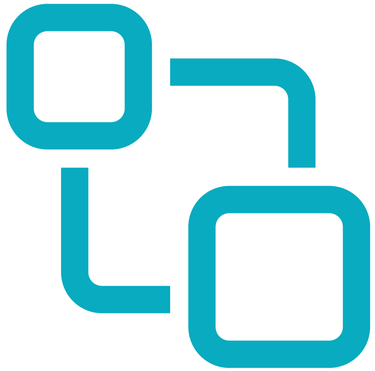
https://conan.io/center

See the cppreference list of libraries: https://en.cppreference.com/w/cpp/links/libs

Code reuse with libraries



About Libraries



Managing Libraries

Finding Libraries

Agenda

01

Obtaining tools for your platform

02

Code reuse through libraries

03

Building correct, secure, and safe systems

04

Planning for the future

05

Resources for learning modern C++

Agenda

01
Obtaining tools for your platform

02
Code reuse through libraries

03
Building correct, secure, and safe systems

04
Planning for the future

05
Resources for learning modern C++

Build reliable and secure C++ programs
https://aka.ms/cpp/security

> **To ensure that software is sufficiently safe and secure, software must be designed, built, delivered, and maintained well. Frequent and thorough verification by developers as early as possible in the software development lifecycle (SDLC) is one critical element of software security assurance.**

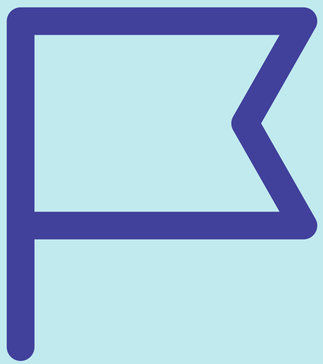
NISTIR 8397 – Guidelines on Minimum Standards for Developer Verification of Software.
https://doi.org/10.6028/NIST.IR.8397

Correctness & Safety

Finding
Bugs

Unit
Testing

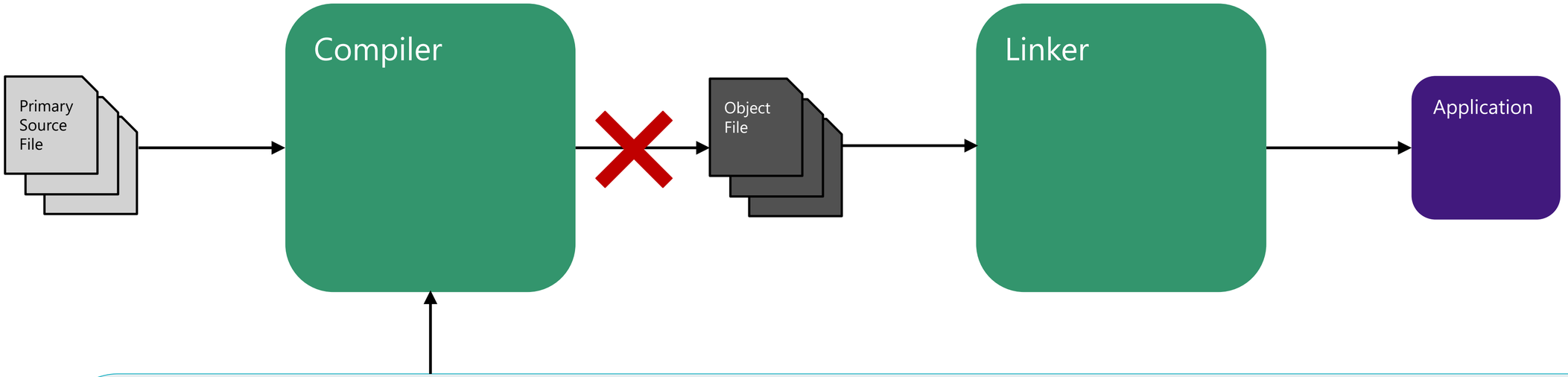
Correctness & Safety



Finding
Bugs

Unit
Testing

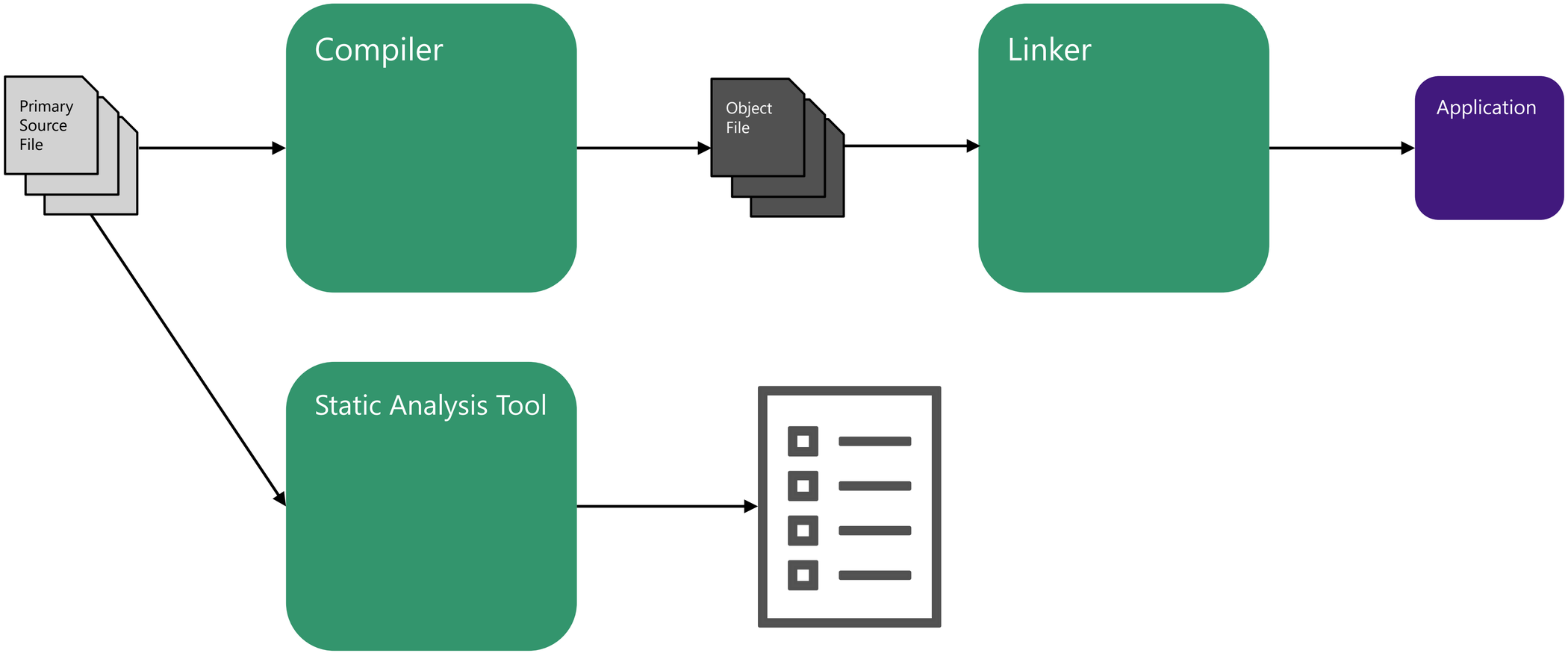
Compiler Warnings



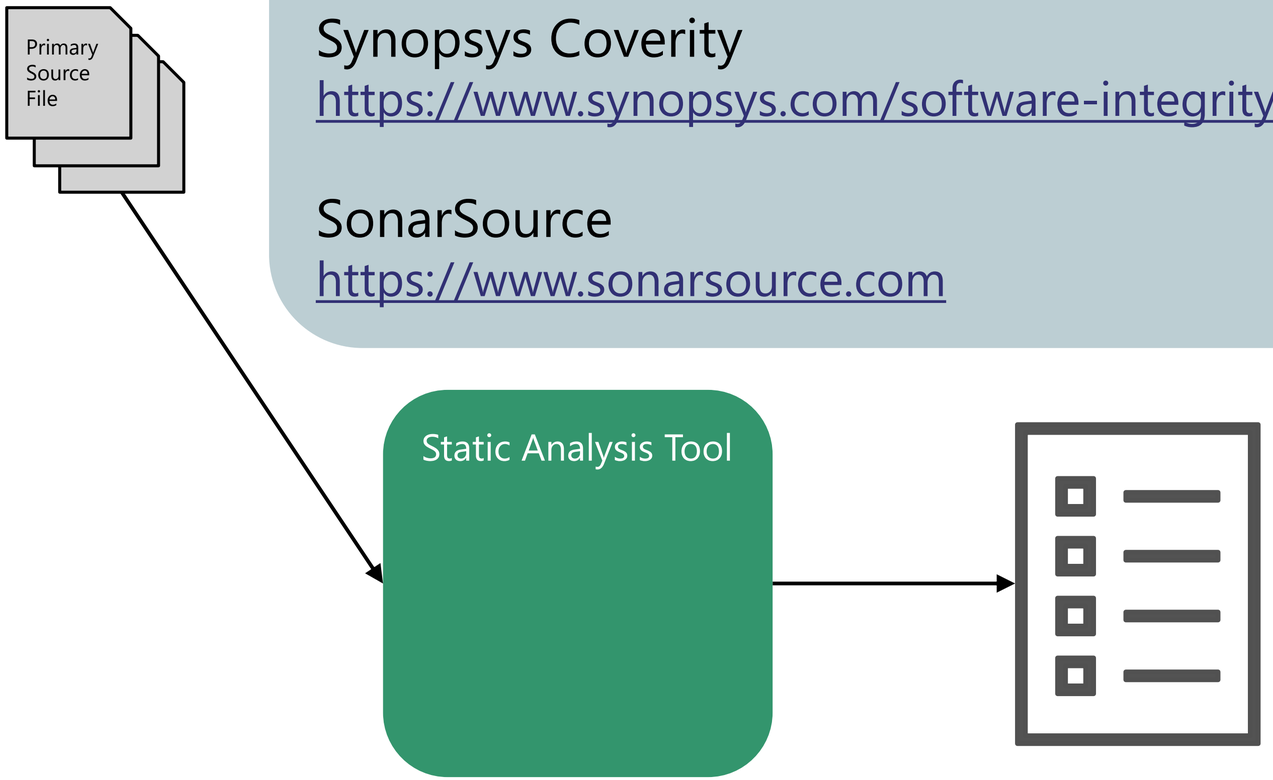
Compiler Switches

- Visual C++
  `/WX /W4 /permissive-`
  See also `/analyze /analyze:plugin EspxEngine.dll`
- GCC / Clang
  `-Werror -Wall -Wextra -Wshadow -Wnon-virtual-dtor -pedantic`

Static Analysis



Static Analysis



Microsoft C/C++ Code Analysis
https://learn.microsoft.com/en-us/cpp/code-quality

Synopsys Coverity
https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html

SonarSource
https://www.sonarsource.com

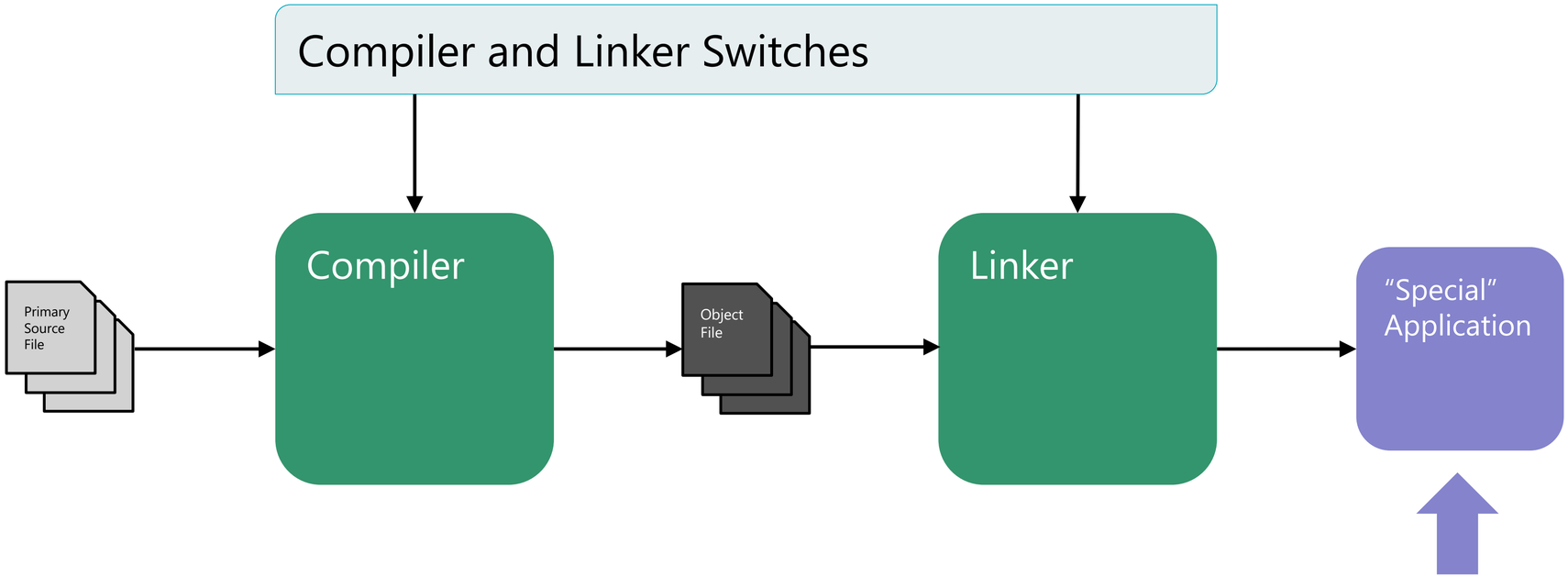
Clang Static Analyzer
https://clang-analyzer.llvm.org

clang-tidy
https://clang.llvm.org/extra/clang-tidy

Cppcheck
https://github.com/danmar/cppcheck

PVS-Studio
https://pvs-studio.com

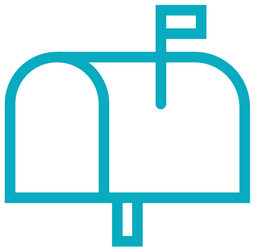
Dynamic analysis



You run this “special” version of your application and the dynamic analysis tools will report any issues found during execution

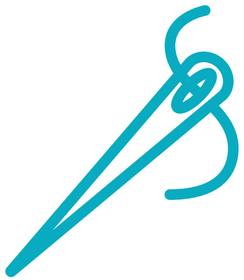
Dynamic analysis

Sanitizers



AddressSanitizer (ASan)

LeakSanitizer (LSan)



ThreadSanitizer (TSan)

UndefinedBehaviorSanitizer (UBSan)

https://github.com/google/sanitizers/

Available on GCC and Clang

Visual C++ currently only supports ASan, but also supports “continue on error” mode

Which approach is best?

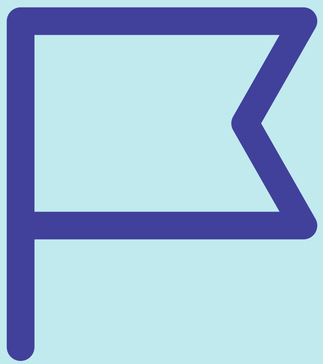
Static Analysis	vs.	Dynamic Analysis
Finds likely problems		Finds definite problems
Single run finds lots of problems		Rinse and repeat
Computationally expensive		Runs for as long as you want**
Does not affect product		Do **not** ship instrumented product

Which approach is best?

Use BOTH!

Static Analysis	AND	Dynamic Analysis
Finds potential problems		Finds definite problems
Single run finds lots of problems		Rinse and repeat
Computationally expensive		Runs for as long as you want**
Does not affect product		Do **not** ship instrumented product

Correctness & Safety



Finding
Bugs

Unit
Testing

Correctness & Safety

Finding
Bugs

Unit
Testing

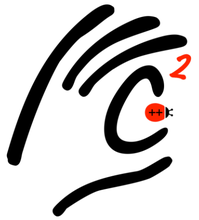
Unit testing

Tools & Frameworks

GoogleTest

https://google.github.io/googletest/

Package Managers: **`gtest`**



Catch2

https://github.com/catchorg/Catch2

Package Managers: **`catch2`**

Microsoft Unit Testing Framework for C++

https://learn.microsoft.com/visualstudio/test/how-to-use-microsoft-test-framework-for-cpp

Only available with Visual Studio

Boost.Test

https://boost.org/libs/test

vcpkg: **`boost-test`**
Conan: **`boost`**

Correctness & Safety

Go Deeper

CppCon 2023 - Back to Basics: Testing

Finding
Bugs

Unit
Testing

Agenda

01
Obtaining tools for your platform

02
Code reuse through libraries

03
Building correct, secure, and safe systems

04
Planning for the future

05
Resources for learning modern C++

Agenda

01
Obtaining tools for your platform

02
Code reuse through libraries

03
Building correct, secure, and safe systems

04
Planning for the future

05
Resources for learning modern C++

> Within Google, we sometimes say, “Software engineering is programming integrated over time.”

Winters, et al. *Software Engineering at Google: Lessons Learned from Programming Over Time*. United States, O'Reilly Media, 2020.

Source control & continuous integration

Git

https://git-scm.com

Continuous integration tools

Hosted source control providers

GitHub

https://github.com

Azure DevOps

https://azure.microsoft.com/products/devops

BitBucket

https://bitbucket.org

GitLab

https://about.gitlab.com

Jenkins

https://www.jenkins.io

Go Deeper

CppCon 2022 - GitHub Features Every C++ Developer Should Know

Agenda

01
Obtaining tools for your platform

02
Code reuse through libraries

03
Building correct, secure, and safe systems

04
Planning for the future

05
Resources for learning modern C++

Agenda

01

Obtaining tools for your platform

02

Code reuse through libraries

03

Building correct, secure, and safe systems

04

Planning for the future

05

Resources for learning modern C++

C++ Conferences

A sampling

Meeting C++ 2023

Cpp North

accu conference

C++ now

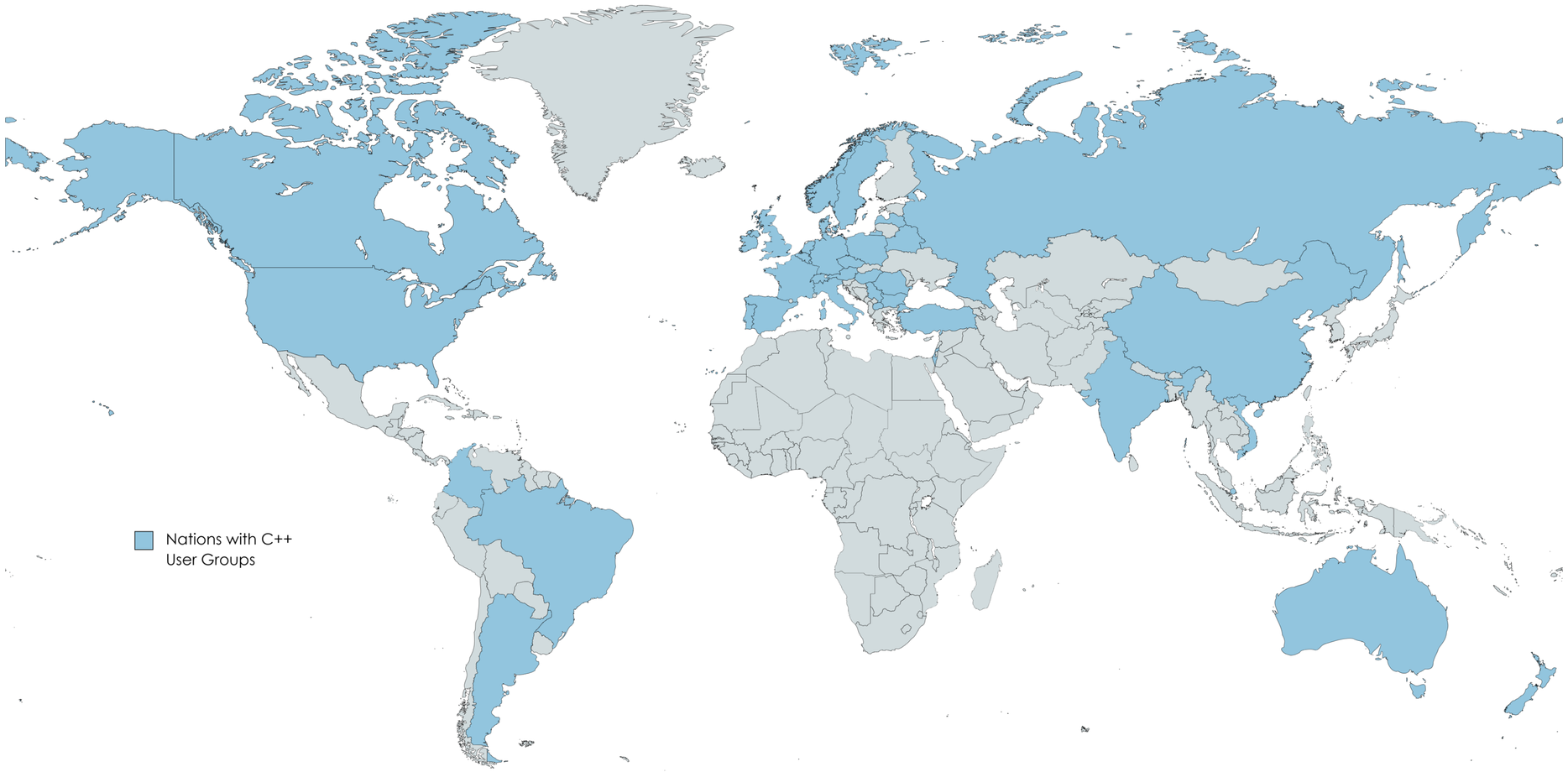
code::dive

- CppCon
- Meeting C++
- CppNorth
- ACCU
- C++ Now
- C++ on Sea
- Code::Dive
- CppIndia
- Core C++
- Pure Virtual C++

Core C++ 2023

Pure Virtual C++

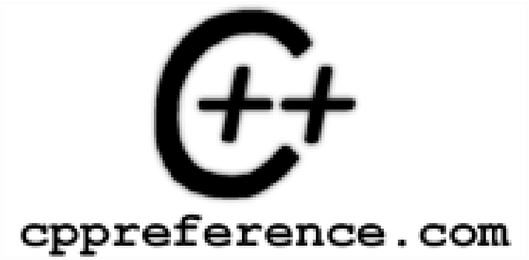
C++ User Groups



https://isocpp.org/wiki/faq/user-groups-worldwide

Learning Resources

A non-comprehensive list



https://cppinsights.io

https://compiler-explorer.com

C++ documentation on Microsoft Learn
https://learn.microsoft.com/cpp

Microsoft C++ Team Blog
https://aka.ms/cpp

Agenda

01

Obtaining tools for your platform

02

Code reuse through libraries

03

Building correct, secure, and safe systems

04

Planning for the future

05

Resources for learning modern C++

The mission of the Microsoft C++ Team is to empower every C++ developer and their team to achieve more.

Our sessions

Monday 2nd

- Lifetime Safety in C++ – Gabor Horvath
- Informal Birds of a Feather for Cpp2/cppfront – Herb Sutter

Tuesday 3rd

- What's New in Visual Studio – David Li & Mryam Girmay

Thursday 5th

- Cooperative C++ Evolution: Towards a Typescript for C++ – Herb Sutter (Keynote)
- How Visual Studio Code Can Help You Develop More Efficiently in C++ – Alexandra Kemper & Sinem Akinci
- Regular, Revisited – Victor Ciura

Friday 6th

- Getting Started with C++ – Michael Price

Enjoy the rest of the conference!

Join the #visual_studio channel on CppCon Discord
https://aka.ms/cppcon/discord

- Ask any questions
- Discuss the latest announcements

Take our survey

https://aka.ms/cppcon/start