



مقام معظم رهبری:

علم برای یک ملت مهم‌ترین ابزار آبرو و پیشرفت و اقتدار است.

۱۳۹۴/۰۸/۲۰





Qt Training in C++

Lecturer: Ali Panahi

What is Qt ?

Qt is cross-platform software for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms such as Linux, Windows, macOS, Android or embedded systems with little or no change in the underlying codebase while still being a native application with native capabilities and speed.

Qt is currently being developed by The Qt Company, a publicly listed company, and the Qt Project under open-source governance, involving individual developers and organizations working to advance Qt. Qt is available under both commercial licenses and open-source GPL 2.0, GPL 3.0, and LGPL 3.0 licenses.

Why use Qt ?

- Design and develop great user experiences
- Qt saves you money
- Get your products to market faster
- Performance, delivered
- One framework, fewer dependencies
- Develop for any platform
- We speak many languages
- Flexible. Reliable. Qt.
- Open source and future-proof

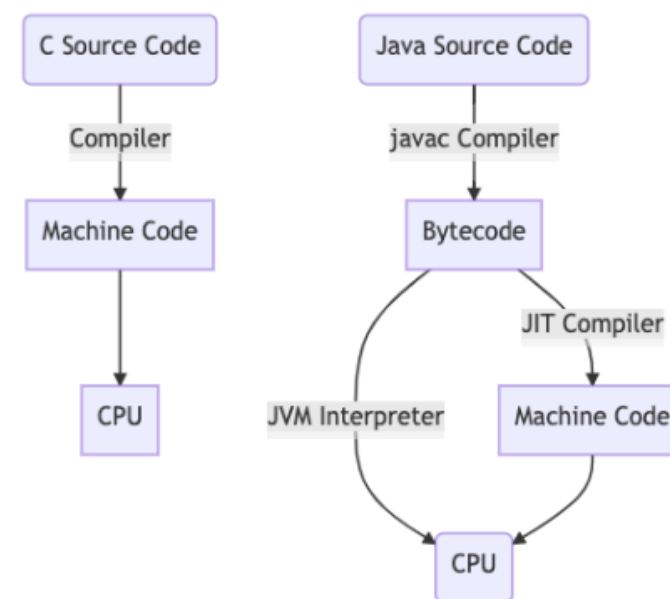
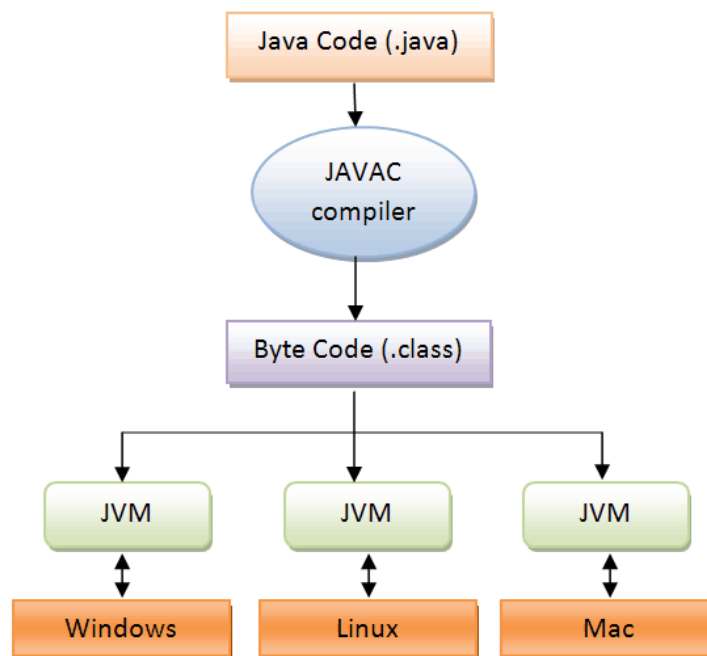
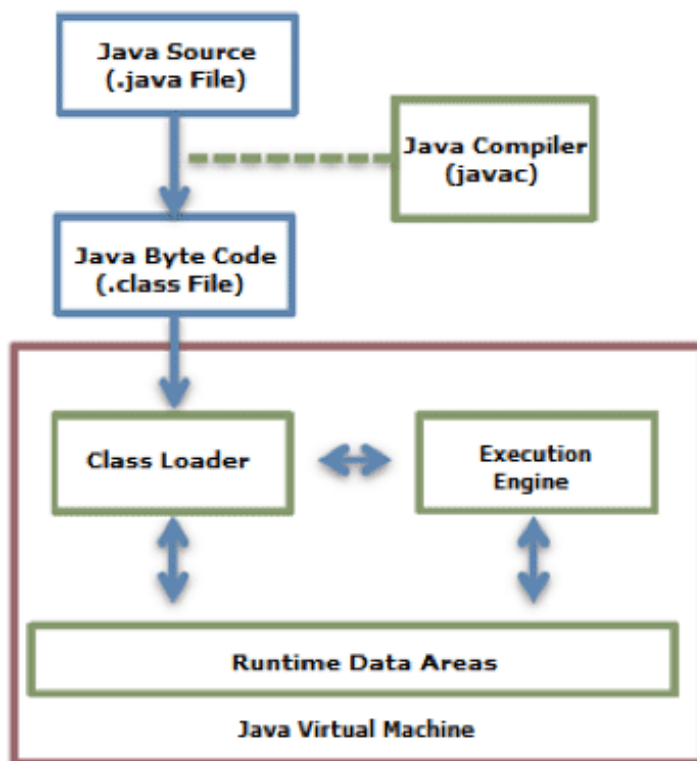
Java vs .Net vs C++ vs QT vs .Net Core

- Java
 - Java is object-oriented programming language
 - Java is a general-purpose
 - Java is cross-platform
 - Java is class-based
 - Java is managed
 - Java is free
 - Low execution speed
 - Disassembling object code
 - High development speed



Introduction to Qt >> Java

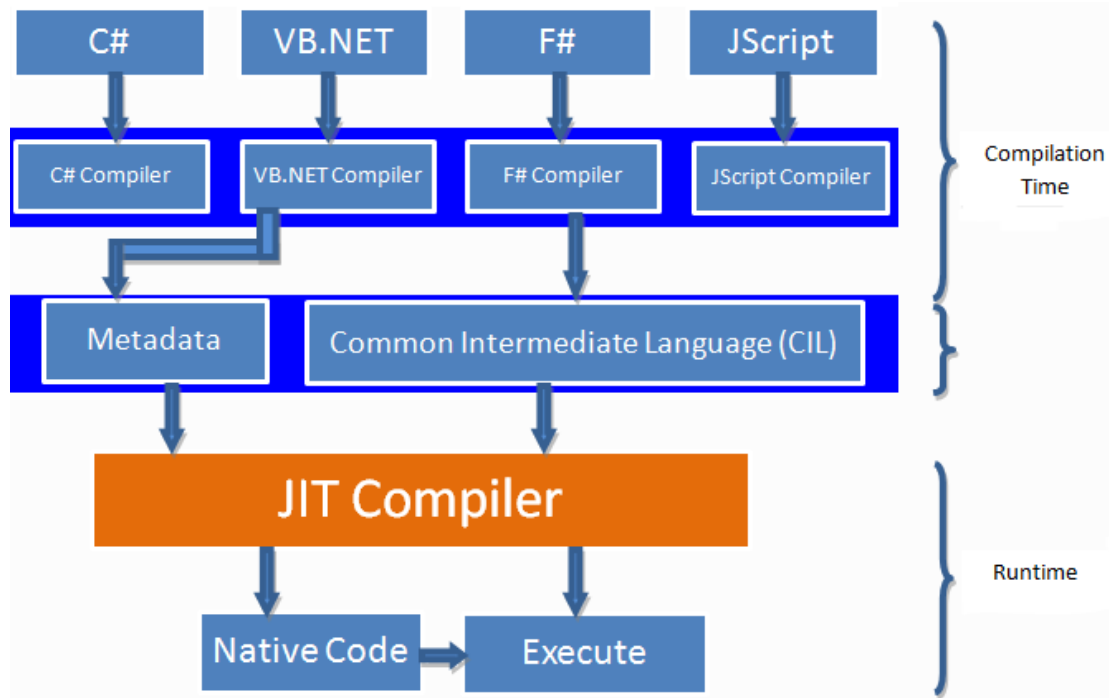
Qt



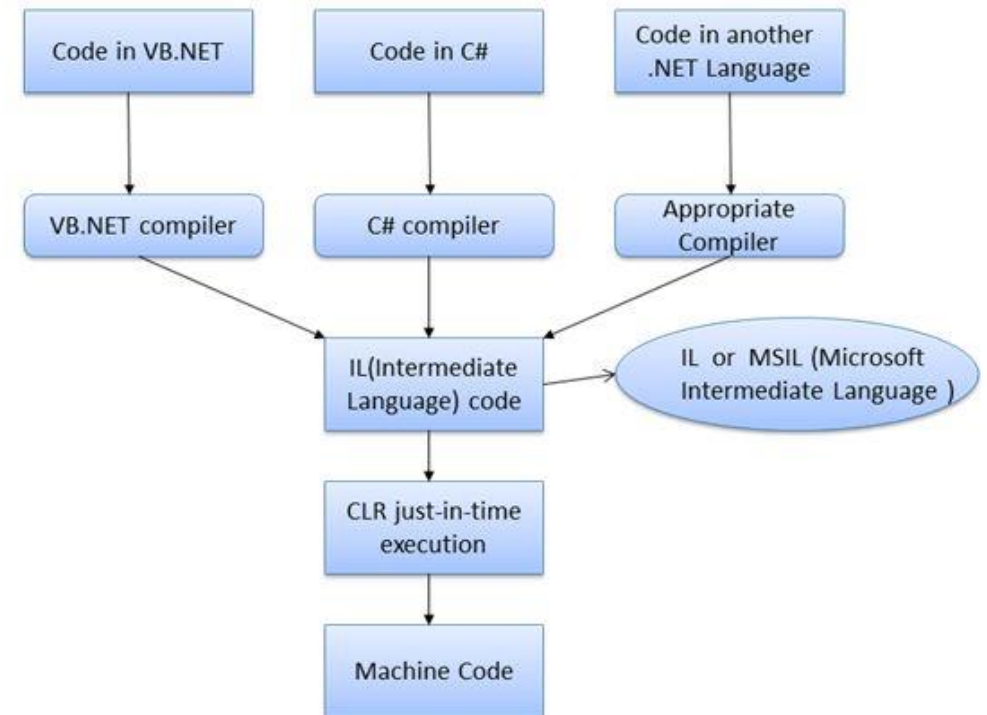
Java vs .Net vs C++ vs QT vs .Net Core

- .Net
 - Multi-language Support
 - .Net is managed
 - Automatic resource management
 - .Net is commercial (Microsoft)
 - Low execution speed
 - Disassembling object code
 - Only for windows
 - High development speed





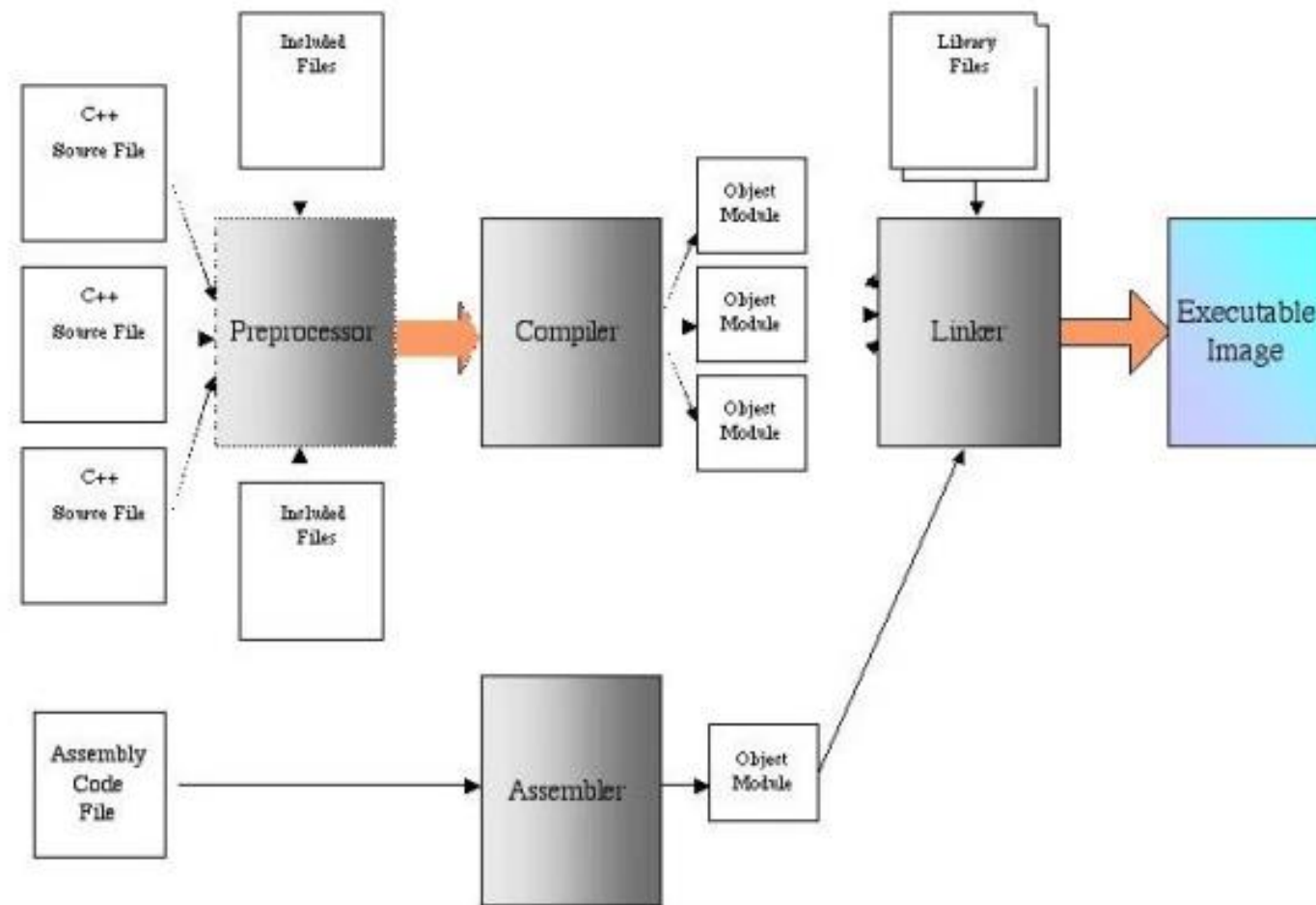
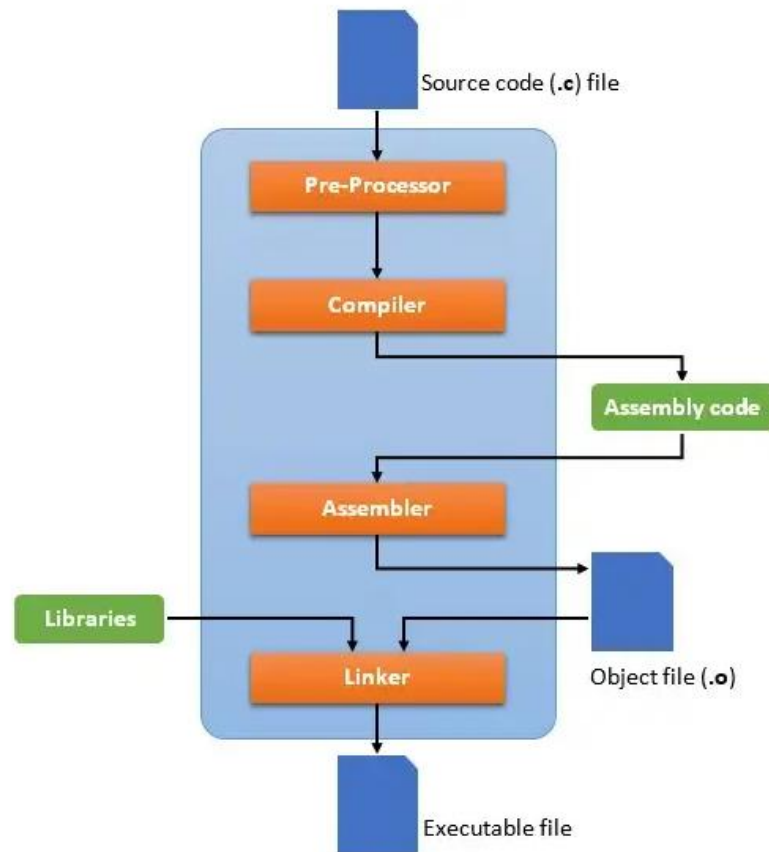
Compilation in .NET



Java vs .Net vs C++ vs QT vs .Net Core

- C/C++
 - It is native
 - Creating libraries that can be used in other languages
 - Powerful & fast
 - High security
 - Platform dependent
 - Irreversibility and reverse engineering of codes
 - It has no memory management
 - Low development speed

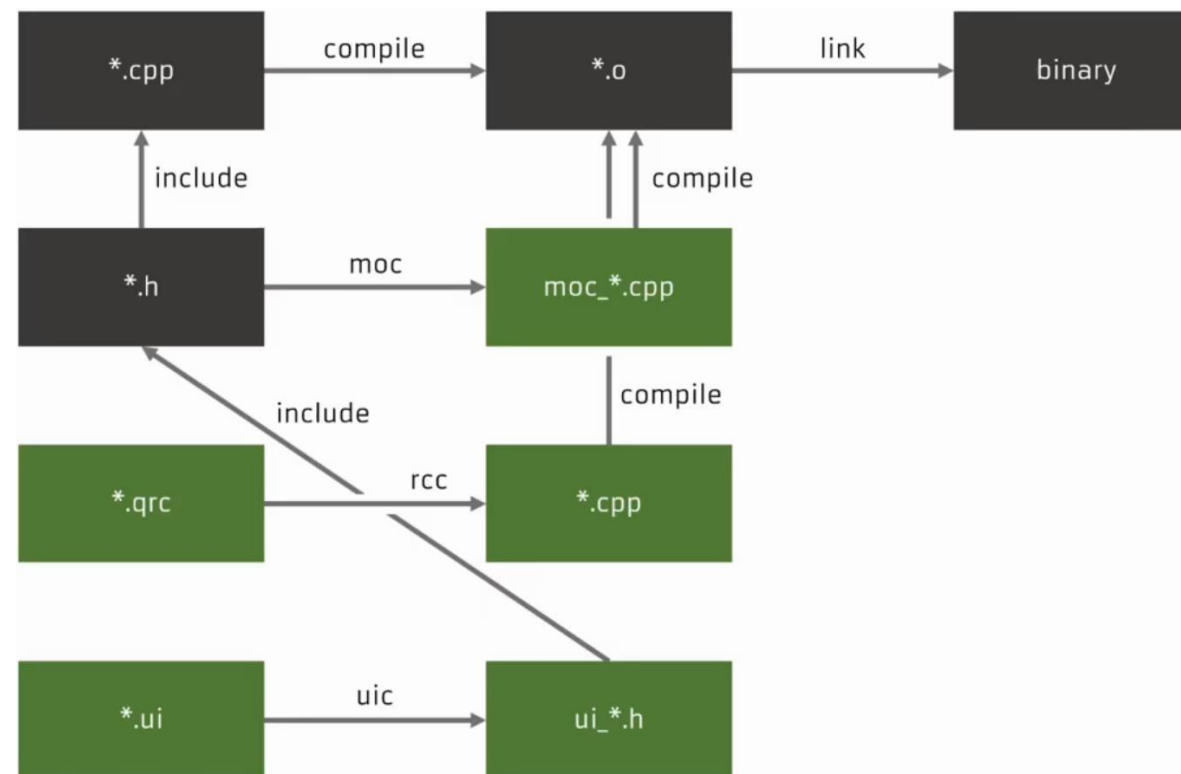
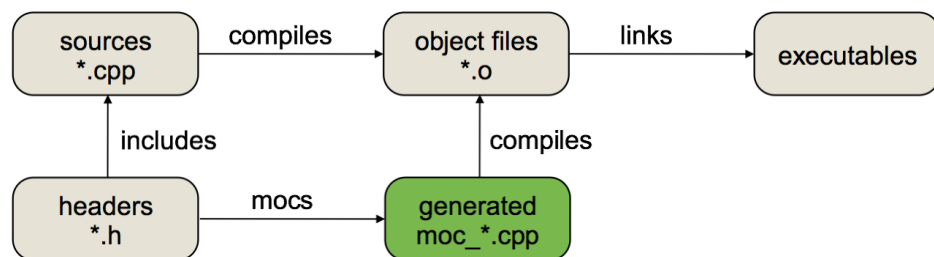


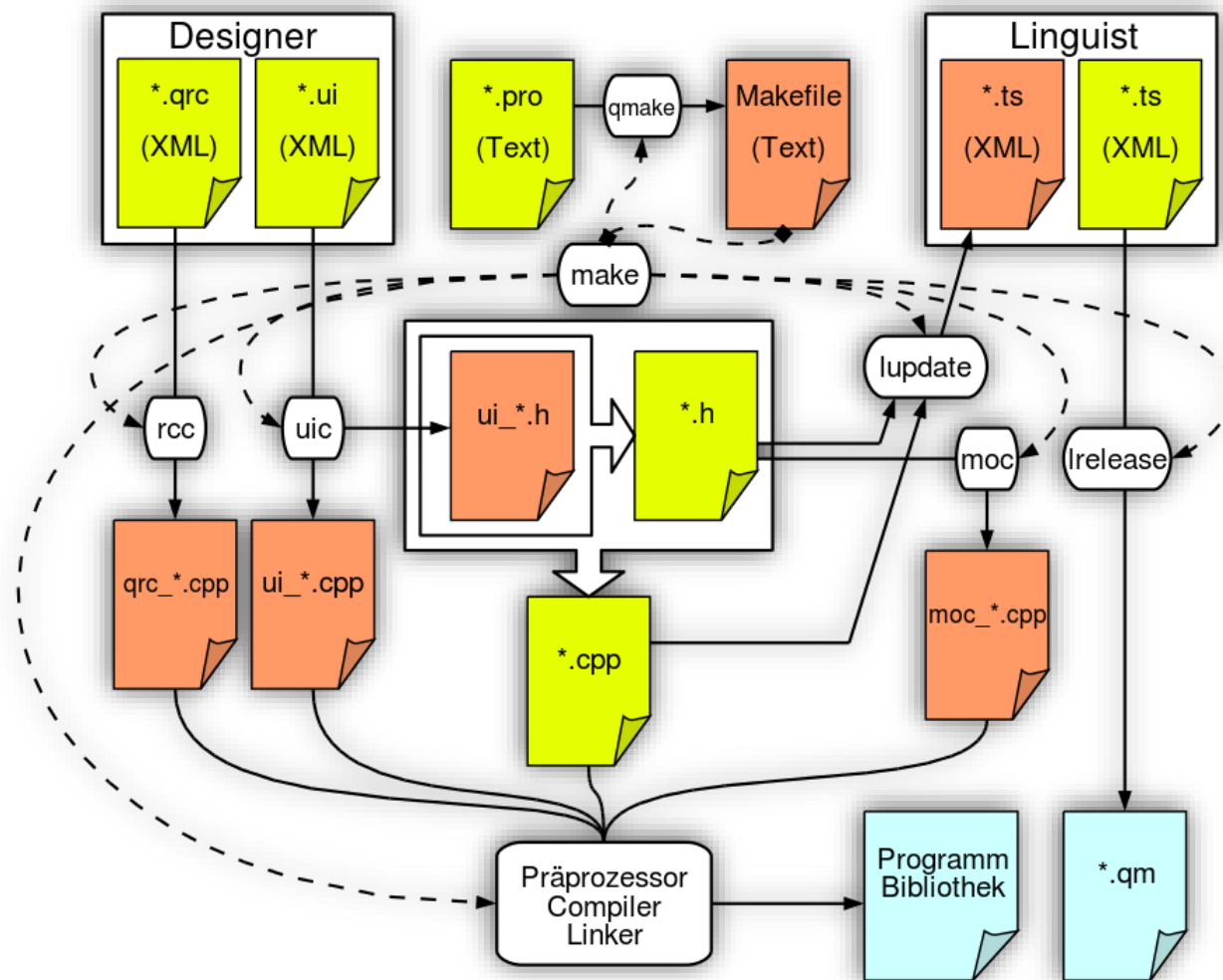


Java vs .Net vs C++ vs QT vs .Net Core

- Qt
 - It is native
 - High security
 - Cross-platform (Windows, Linux, Mac)
 - Many possibilities
 - Free and open source





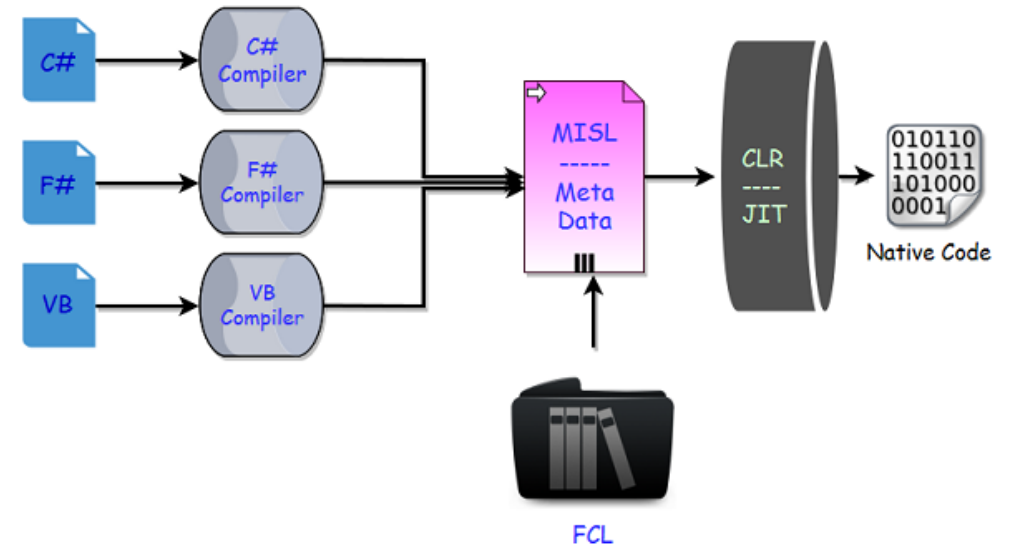
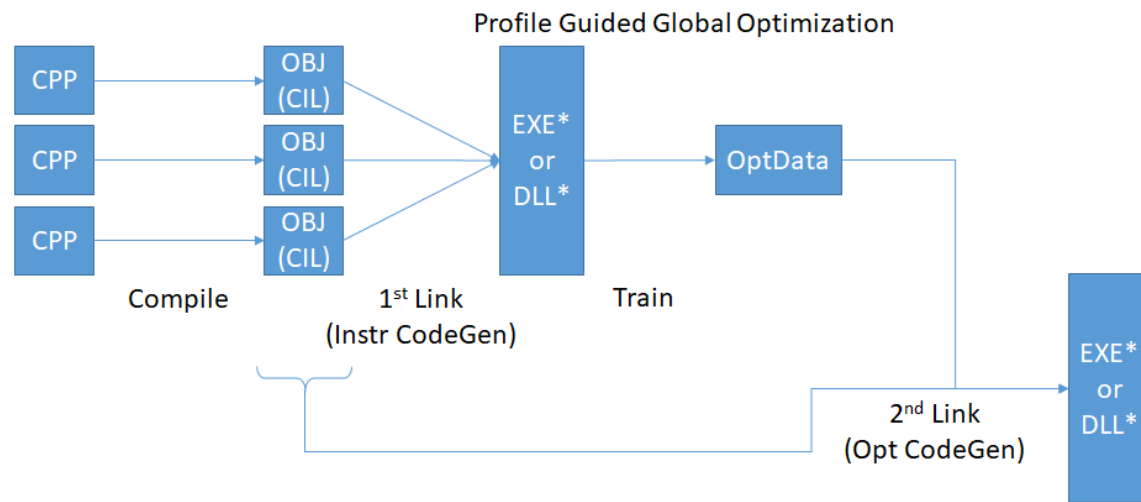


Java vs .Net vs C++ vs QT vs .Net Core

- .Net Core
 - Multi-language support
 - .Net core is managed
 - Automatic resource management
 - .Net core is free and open source (Microsoft)
 - Low execution speed
 - Disassembling object code
 - Cross-platform (Windows, Linux)
 - High development speed



Introduction to Qt >> .Net Core



Companies using Qt

Qt helps the best companies in the world deliver better user experiences faster.

Panasonic

In-flight
entertainment
systems



In-vehicle
infotainment
system

AMD

Graphics software



Anesthesia &
critical care
medical devices

ZUKEN

EDA & CAD end-
to-end
engineering
solutions

DENSO

Automotive
mobility
technology

Introduction to Qt >> Qt Versions



Version	Release date / Support until	Target (Windows)
Qt 0.90	1995	-
Qt 1.0	1996	-
Qt 2.0	1999	-
Qt 3.0	2001	-
Qt 4.0	2005	-
Qt 4.8 LTS (4.8.7)	2011	XP, 7, 8.1, 10
Qt 5.6 LTS (5.6.3)	2016 / 2019	-
Qt 5.9 LTS (5.9.9)	2017 / 2020	-
Qt 5.12 LTS (5.12.12)	2018 / 2021	-
Qt 5.15 LTS	2020 / 2025	7, 8.1, 10, 11
Qt 6.2 LTS (6.2.6)	2021 / 2024	10, 11
Qt 6.4	2022 / 2023	10, 11

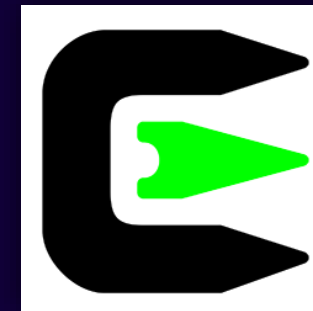
Languages that support the Qt library

Qt is developed with C++ language.

Qt can be used in several programming languages other than C++, such as Python, Javascript, C# and Rust via language bindings; many languages have bindings for Qt 5 and bindings for Qt 4.

Qt compilers

- Windows
 - MSVC
 - MinGW (Gnu Based)
 - Cygwin (BSD Based)
- Linux
 - GCC/G++
 - CLang



Qt development environments

- Qt Creator
 - Windows
 - Linux
- Visual Studio
 - Windows
- PyCharm (Only for PyQt)
 - Windows
 - Linux



Installation methods

- From installer
 - Offline installer
 - Online installer
 - From Packages (Linux)
 - From Setup wizard
- From source

Offline Installer

- It can be downloaded from the link below:
 - URL: <https://www.qt.io/offline-installers>
- It is precompiled
- Only available for limited editions
- No internet required
- The installer size is large

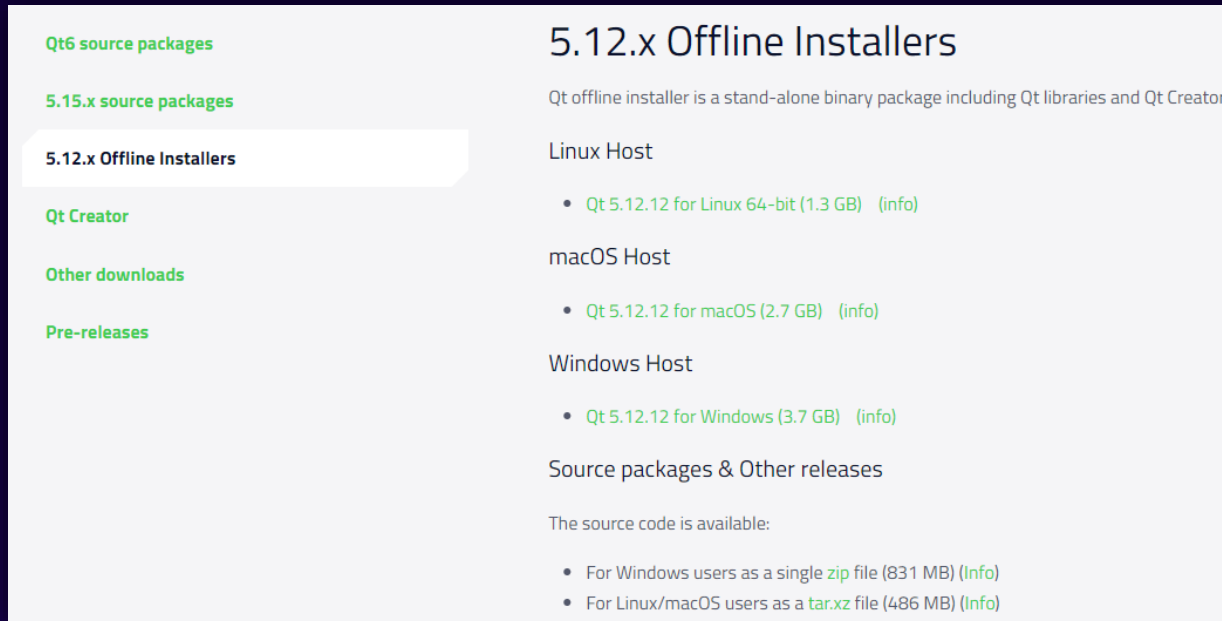
Offline Installer

On Windows

`qt-opensource-windows-x86-%VERSION%.exe`

On Linux

`qt-opensource-linux-x64-%VERSION%.run`



The screenshot shows the Qt 5.12.x Offline Installers page. On the left, there is a sidebar with links: Qt6 source packages, 5.15.x source packages, 5.12.x Offline Installers (selected), Qt Creator, Other downloads, and Pre-releases. The main content area is titled '5.12.x Offline Installers' and contains the following text: 'Qt offline installer is a stand-alone binary package including Qt libraries and Qt Creator.' Below this, there are three sections: 'Linux Host' with a link to 'Qt 5.12.12 for Linux 64-bit (1.3 GB) (info)', 'macOS Host' with a link to 'Qt 5.12.12 for macOS (2.7 GB) (info)', and 'Windows Host' with a link to 'Qt 5.12.12 for Windows (3.7 GB) (info)'. At the bottom, there is a section titled 'Source packages & Other releases' with the text 'The source code is available:' followed by two links: 'For Windows users as a single zip file (831 MB) (Info)' and 'For Linux/macOS users as a tar.xz file (486 MB) (Info)'.

[Qt6 source packages](#)

[5.15.x source packages](#)

5.12.x Offline Installers

[Qt Creator](#)

[Other downloads](#)

[Pre-releases](#)

5.12.x Offline Installers

Qt offline installer is a stand-alone binary package including Qt libraries and Qt Creator.

Linux Host

- [Qt 5.12.12 for Linux 64-bit \(1.3 GB\)](#) (info)

macOS Host

- [Qt 5.12.12 for macOS \(2.7 GB\)](#) (info)

Windows Host

- [Qt 5.12.12 for Windows \(3.7 GB\)](#) (info)

Source packages & Other releases

The source code is available:

- For Windows users as a single [zip](#) file (831 MB) ([Info](#))
- For Linux/macOS users as a [tar.xz](#) file (486 MB) ([Info](#))

Online Installer

- It can be downloaded from the link below:
 - URL: https://download.qt.io/official_releases/online_installers/
- It is precompiled
- Only available for limited editions
- Internet required
- The installer size is small

Online Installer

On Windows

qt-unified-windows-x64-%VERSION%-online.exe

On Linux

qt-unified-linux-x64-%VERSION%-online.run

Name	Last modified	Size	Metadata
↑ Parent Directory		-	
📁 qt-unified-windows-x64-online.exe	09-Nov-2022 10:52	41M	Details
📁 qt-unified-mac-x64-online.dmg	09-Nov-2022 10:52	18M	Details
📁 qt-unified-linux-x64-online.run	09-Nov-2022 10:52	55M	Details

For Qt Downloads, please visit qt.io/download

QtÅ and the Qt logo is a registered trade mark of The Qt Company Ltd and is used pursuant to a license from The Qt Company Ltd.
All other trademarks are property of their respective owners.

The Qt Company Ltd, Bertel Jungin aukio D3A, 02600 Espoo, Finland. Org. Nr. 2637805-2

[List of official Qt-project mirrors](#)

Online Installer

On Windows

Run “qt-unified-windows-x64-%VERSION%-online.exe” as administrator

On Linux

This prerequisite must be installed in the Debian operating system

```
sudo apt-get install libxcb-xinerama0
```

A file with the name qt-unified-linux-x-online.run will be downloaded, then add exec permission.

```
chmod +x qt-unified-linux-x-online.run
```

Remember to change 'x' for the actual version of the installer. Then run the installer.

```
./qt-unified-linux-x-online.run
```

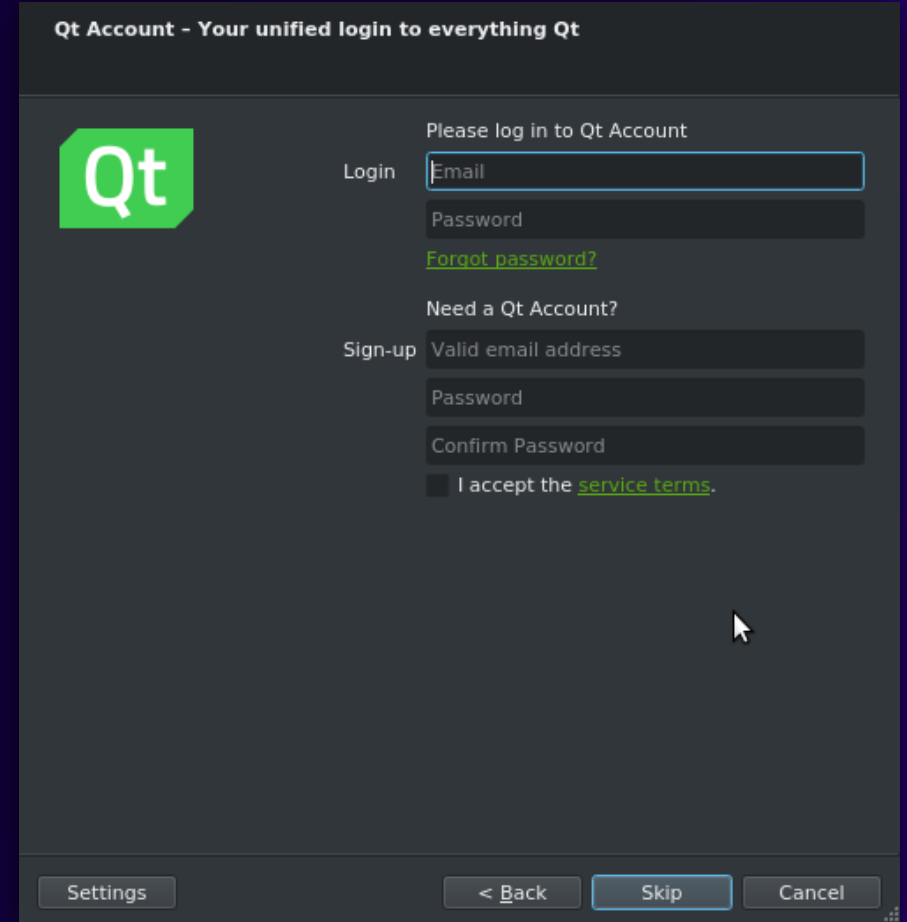

Offline/Online Installer

Install Qt in any operative system


Current sample is 4.5.0 version.

Once you've downloaded Qt and opened the installer program, the installation procedure is the same for all operative systems, although the screenshots might look a bit different. The screenshots provided here are from Linux.

Login with a existing Qt account or create a new one:



Qt Account - Your unified login to everything Qt



Please log in to Qt Account

Login

[Forgot password?](#)

Need a Qt Account?

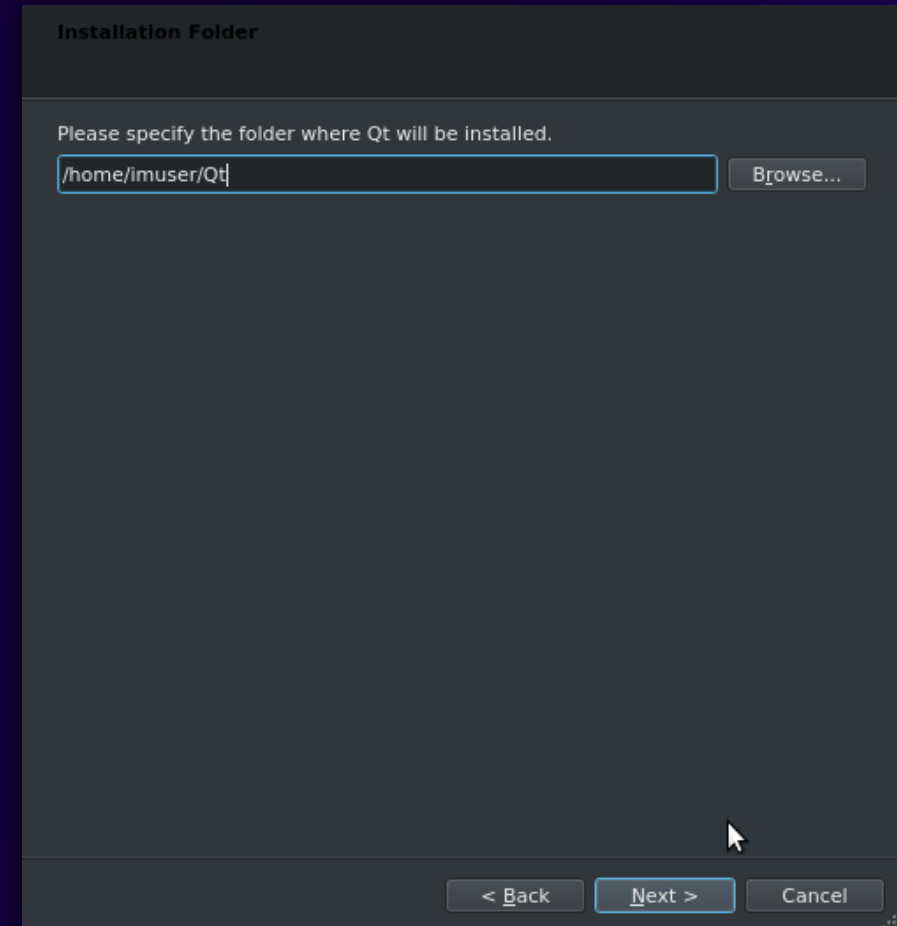
Sign-up

☐ I accept the [service terms](#).

Offline/Online Installer

Install Qt in any operative system

Select a path to install the Qt libraries and tools



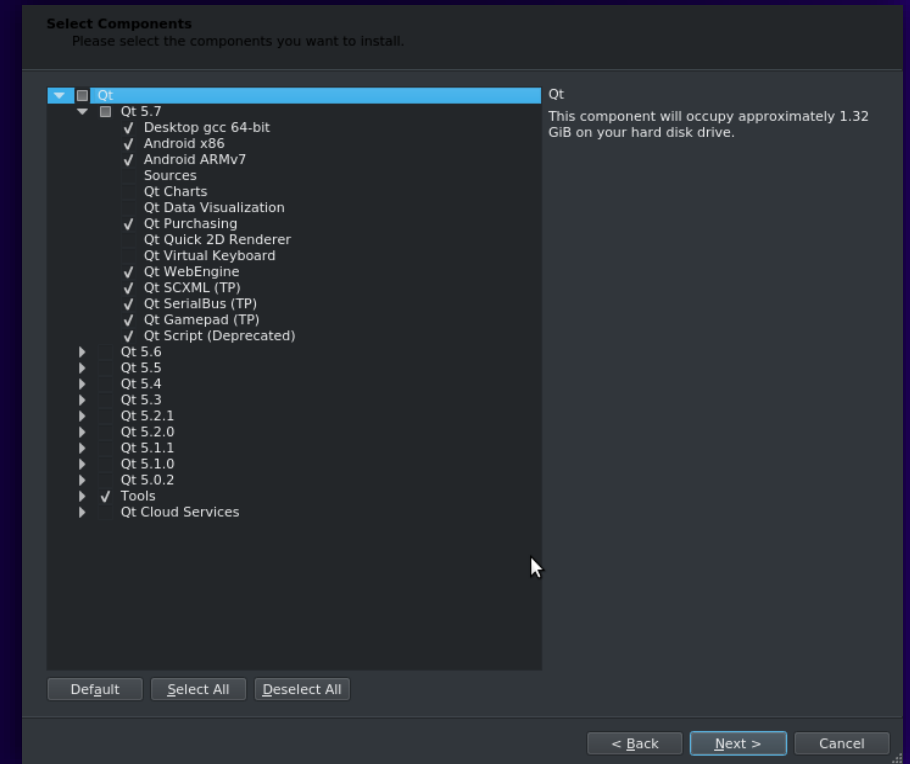
Introduction to Qt >> Installing Qt



Offline/Online Installer

Install Qt in any operative system

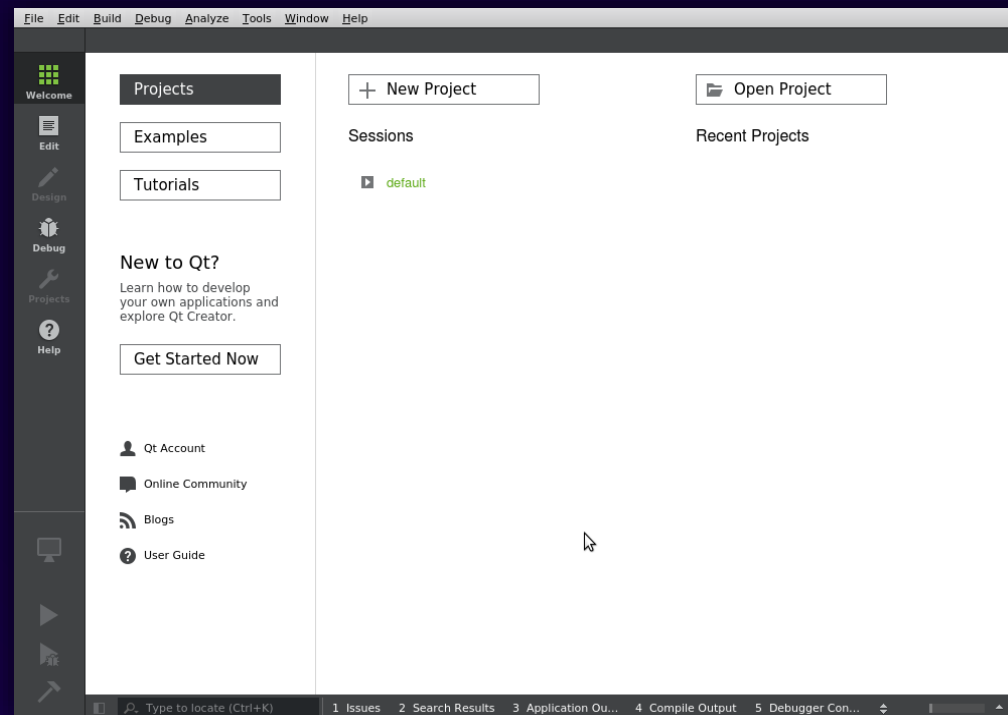
Select the library version and the features you want



Offline/Online Installer

Install Qt in any operative system

After downloading and the installation is finished, go to the Qt installation directory and launch Qt Creator or run it directly from the command line.



Installation From Packages (Linux)

Installation in Debin (11.x) distribution

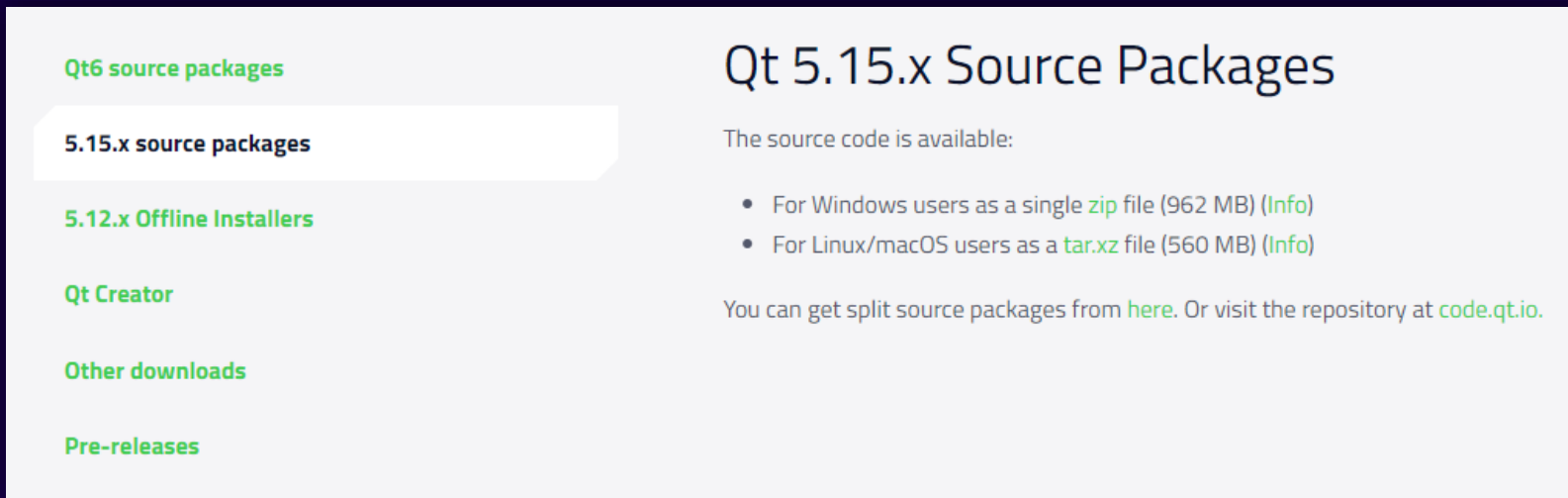
```
$ sudo apt-get update && sudo apt-get upgrade  
$ sudo apt install qtbase5-dev qt5-qmake qtbase5-dev-tools  
$ sudo apt-get install qtcreator  
$ qtcreator
```

Type of build

- Dynamic
 - Dynamic plug-in is basically a shared library which is loaded at runtime.
 - ✓ Ability to update and patch Qt libraries.
 - ✓ The Qt libraries should be included with the final executable file
- Static
 - Static plug-in is built into your executable (like a static lib).
 - ✓ The libraries are combined with the executable file and a final file is created

Build in Windows

- It can be downloaded from the link below (windows version 5.15.2):
 - URL: <https://www.qt.io/offline-installers>
- Ability to customize modules
- The ability to change the source code
- Ability to compile statically



The screenshot shows the Qt 5.15.x Source Packages page. On the left, there is a sidebar with links: "Qt6 source packages", "5.15.x source packages" (which is highlighted), "5.12.x Offline Installers", "Qt Creator", "Other downloads", and "Pre-releases". The main content area is titled "Qt 5.15.x Source Packages" and contains the text "The source code is available:" followed by two bullet points: "For Windows users as a single zip file (962 MB) (Info)" and "For Linux/macOS users as a tar.xz file (560 MB) (Info)". Below this, it says "You can get split source packages from here. Or visit the repository at code.qt.io."

Qt6 source packages

5.15.x source packages

5.12.x Offline Installers

Qt Creator

Other downloads

Pre-releases

Qt 5.15.x Source Packages

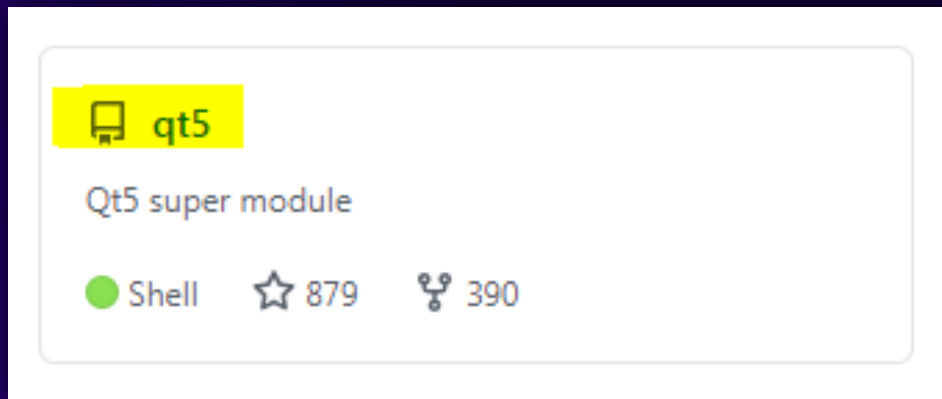
The source code is available:

- For Windows users as a single zip file (962 MB) (Info)
- For Linux/macOS users as a tar.xz file (560 MB) (Info)

You can get split source packages from [here](#). Or visit the repository at code.qt.io.

Build in Windows

- Enter the following website (Qt requirements to compile):
 - URL: <https://github.com/qt>



System requirements

- CMake 3.18 or later
- Perl 5.8 or later
- Python 2.7 or later
- C++ compiler supporting the C++17 standard

It's recommended to have ninja 1.8 or later installed.

For other platform specific requirements, please see section "Setting up your machine" on: http://wiki.qt.io/Get_The_Source

Build in Windows

- Prerequisites for compiling Qt on the windows platform

Windows:

1. Open a command prompt.
2. Ensure that the following tools can be found in the path:
 - Supported compiler (Visual Studio 2019 or later, or MinGW-builds gcc 8.1 or later)
 - Perl version 5.12 or later [<http://www.activestate.com/activeperl/>]
 - Python version 2.7 or later [<http://www.activestate.com/activepython/>]
 - Ruby version 1.9.3 or later [<http://rubyinstaller.org/>]

```
cd <path>\<source_package>
configure -prefix %CD%\qtbase
cmake --build .
```

Build in Windows

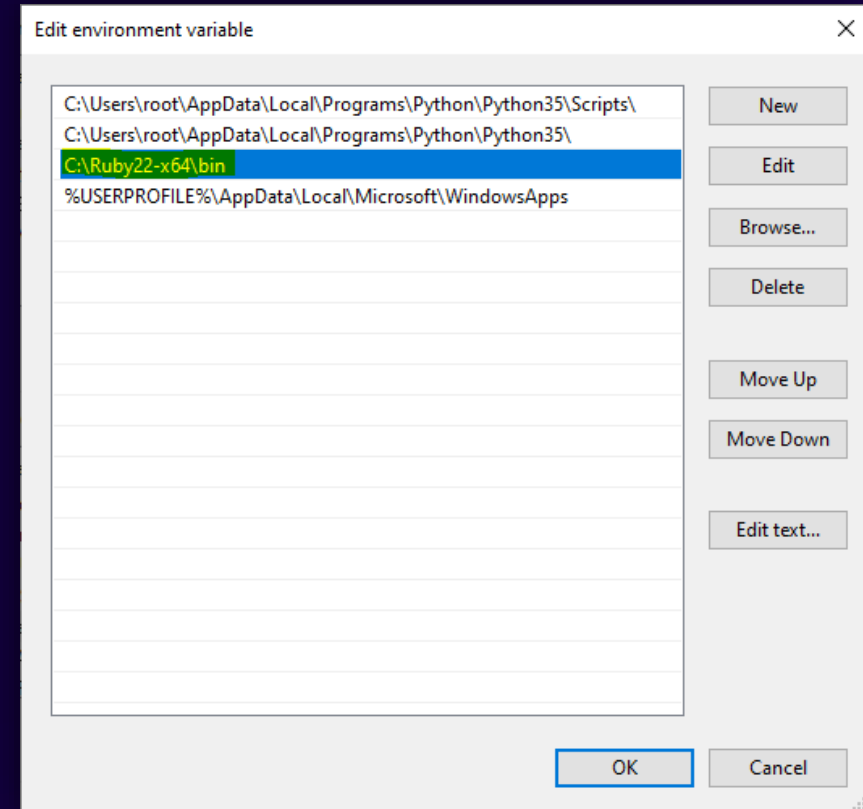
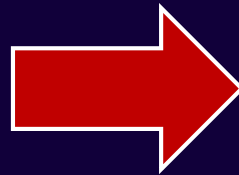
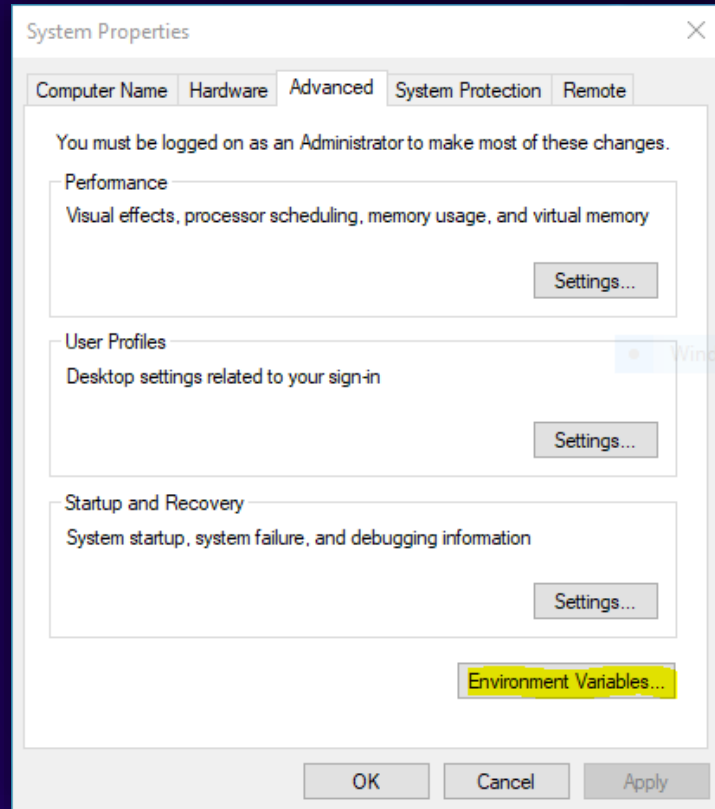
- Compiler:
 - Visual Studio 2019 or later
 - MinGW-builds gcc 8.1 or later
- Tools:
 - Perl version 5.12 or later [<http://www.activestate.com/activeperl/>]
 - Python version 2.7 or later [<http://www.activestate.com/activepython/>]
 - Ruby version 1.9.3 or later [<http://rubyinstaller.org/>]

Note: “Python” and “Perl” installation requires internet.

Note: Compilation requires Windows version 7 or higher.

Build in Windows

- After installing the tools (Perl, Python, Ruby), their path should be added to the “System Environment”



Build in Windows

- First, download the source code
 - Example: qt-everywhere-src-5.15.2.zip
- Decompress the source code in the desired path, for example "c:\"

Build in Windows >> Compile with MSVC

- Install Visual Studio 2019
- Depending on the required architecture (32-bit or 64-bit version), we run the Visual Studio command line from start menu:
 - x86 Native Tools Command Prompt for VS 2019
 - x64 Native Tools Command Prompt for VS 2019
- Enter the following commands in the command line:

```
> SET _ROOT=C:\Qt\qt-everywhere-src-5.15.2
> SET PATH=%_ROOT%\qtbase\bin;%_ROOT%\gnuwin32\bin;%PATH%
> SET PATH=%_ROOT%\qtrepotools\bin;%PATH%
```

Build in Windows >> Compile with MSVC

- Enter the Qt source code path

```
> Cd C:\Qt\qt-everywhere-src-5.15.2
```

- Configure the compiler

```
> configure -debug-and-release -platform win32-msvc -developer-build  
-prefix "C:\Qt\5.15.2-x86" -nomake examples -nomake tests  
-skip qtwebengine -opensource -mp
```

- Start the compilation with the following command:
 - nmake
- install Qt with the following command
 - nmake install
- Clean source code
 - nmake clean

Build in Windows >> Compile with MinGW

- Install MinGW 8.1 or later
 - URL: <http://mingw-w64.org/>
- Run mingw command line or run it in the "cmd" and set mingw path in system environment (Path variable).
- Enter the following commands in the command line:

```
> SET _ROOT=C:\Qt\qt-everywhere-src-5.15.2
> SET PATH=%_ROOT%\qtbase\bin;%_ROOT%\gnuwin32\bin;%PATH%
> SET PATH=%_ROOT%\qtrepotools\bin;%PATH%
```

- Enter the Qt source code path

```
> Cd C:\Qt\qt-everywhere-src-5.15.2
```

Build in Windows >> Compile with MinGW

- Configure the compiler

```
> configure -debug-and-release -platform win32-g++ -developer-build  
-prefix "C:\Qt\5.15.2-mingw-x86" -nomake examples -nomake tests  
-skip qtwebengine -opensource -no-angle -mp
```

- Start the compilation with the following command:

```
> mingw32-make -jn (n is number of cpu for parallel build)
```

- Install Qt with the following command

```
> mingw32-make install
```

Build in Windows

- Clean source code

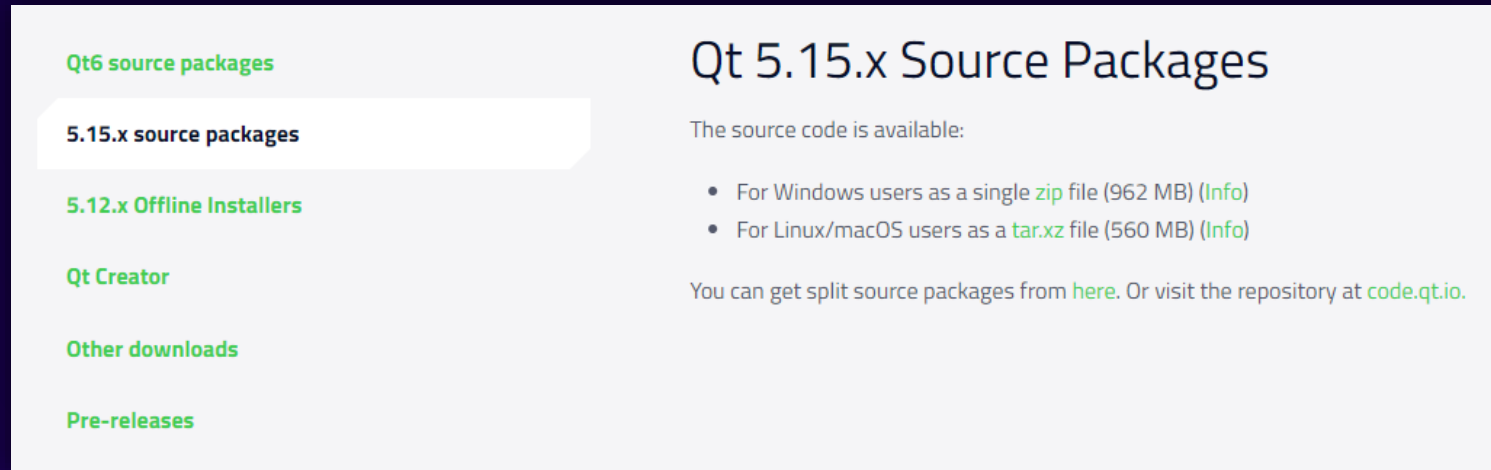
```
> mingw32-make clean
```

- Remove source code directory

```
> del /q "C:\Qt\qt-everywhere-src-5.15.2\*"
FOR /D %%p IN ("C:\Qt\qt-everywhere-src-5.15.2\*.*) DO rmdir "%%p" /s /q
```

Build in Linux

- It can be downloaded from the link below (linux/mac version 5.15.2):
 - URL: <https://www.qt.io/offline-installers>
- Ability to customize modules
- The ability to change the source code
- Ability to compile statically



The screenshot shows the Qt 5.15.x Source Packages page. On the left, there is a sidebar with links: "Qt6 source packages", "5.15.x source packages" (which is highlighted), "5.12.x Offline Installers", "Qt Creator", "Other downloads", and "Pre-releases". The main content area is titled "Qt 5.15.x Source Packages" and contains the text "The source code is available:" followed by two bullet points: "For Windows users as a single zip file (962 MB) (Info)" and "For Linux/macOS users as a tar.xz file (560 MB) (Info)". Below this, it says "You can get split source packages from here. Or visit the repository at code.qt.io."

Build in Linux

- Compiler:
 - GCC/g++
 - Clang
- Tools and library:
 - Perl (>=5.14)
 - Python (>=2.6.x)
 - build-essential
 - Libxcb

Build in Linux / Dependency Packages

- Build Essentials

Distribution	Packages
Ubuntu and/or Debian:	<pre>\$ sudo apt-get install build-essential perl python3 git</pre>
Fedora 30:	<pre>\$ su - -c "dnf install perl-version git gcc-c++ compat- openssl10-devel harfbuzz-devel double-conversion-devel libzstd-devel at-spi2-atk-devel dbus-devel mesa-libGL- devel"</pre>
OpenSUSE:	<pre>\$ sudo zypper install git-core gcc-c++ make</pre>

Build in Linux / Dependency Packages

- Libxcb

Distribution	Packages
Ubuntu and/or Debian:	<pre>\$ sudo apt-get install '^libxcb.*-dev' libx11-xcb-dev libglu1-mesa-dev libxrender-dev libxi-dev libxkbcommon-dev libxkbcommon-x11-dev</pre>
Fedora 30:	<pre>\$ su - -c "dnf install libxcb libxcb-devel xcb-util xcb-util-devel xcb- util-*-devel libX11-devel libXrender-devel libxkbcommon-devel libxkbcommon-x11-devel libXi-devel libdrm-devel libXcursor-devel libXcomposite-devel"</pre>
OpenSUSE 12+:	<pre>\$ sudo zypper in xorg-x11-libxcb-devel xcb-util-devel xcb-util-image- devel xcb-util-keysyms-devel xcb-util-renderutil-devel xcb-util-wm-devel xorg-x11-devel libxkbcommon-x11-devel libxkbcommon-devel libXi-devel</pre>
Centos 7:	<pre>\$ yum install libxcb libxcb-devel xcb-util xcb-util-devel mesa-libGL- devel libxkbcommon-devel</pre>

Build in Linux / Dependency Packages

- OpenGL support
 - For Qt Quick 2, a graphics driver with native OpenGL 2.0 support is highly recommended.
- Accessibility
 - It is recommended to build with accessibility enabled, install libatspi2.0-dev and libdbus-1-dev packages.

Build in Linux / Dependency Packages

- Qt WebKit

Distribution	Packages
Ubuntu and/or Debian:	<pre>\$ sudo apt-get install flex bison gperf libicu-dev libxslt-dev ruby</pre>
Fedora 30:	<pre>\$ su - -c "dnf install flex bison gperf libicu-devel libxslt-devel ruby"</pre>
OpenSUSE:	<pre>\$ sudo zypper install flex bison gperf libicu-devel ruby</pre>
Mandriva/ROSA/Unity:	<pre>\$ urpmi gperf</pre>

Build in Linux / Dependency Packages

- Qt WebEngine

Distribution	Packages
Ubuntu and/or Debian:	<pre>\$ sudo apt-get install libxcursor-dev libxcomposite-dev libxdamage-dev libxrandr-dev libxtst-dev libxss-dev libdbus-1-dev libevent-dev libfontconfig1-dev libcap-dev libpulse-dev libudev-dev libpci-dev libnss3-dev libasound2-dev libegl1-mesa-dev gperf bison nodejs</pre>
Fedora/RHEL:	<pre>\$ sudo dnf install freetype-devel fontconfig-devel pciutils-devel nss- devel nspr-devel ninja-build gperf cups-devel pulseaudio-libs-devel libcap-devel alsa-lib-devel bison libXrandr-devel libXcomposite-devel libXcursor-devel libXtst-devel dbus-devel fontconfig-devel alsa-lib- devel rh-nodejs12-nodejs rh-nodejs12-nodejs-devel</pre>
OpenSUSE:	<pre>\$ sudo zypper install alsa-devel dbus-1-devel libXcomposite-devel libXcursor-devel libXrandr-devel libXtst-devel mozilla-nspr-devel mozilla-nss-devel gperf bison nodejs10 nodejs10-devel</pre>

Build in Linux / Dependency Packages

- Qt Multimedia
 - You'll need at least alsa-lib ($\geq 1.0.15$) and gstreamer ($\geq 0.10.24$) with the base-plugins package.

Distribution	Packages
Ubuntu and/or Debian:	<pre>\$ sudo apt-get install libasound2-dev libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-good1.0-dev libgstreamer-plugins-bad1.0-dev</pre>
Fedora 30:	<pre>\$ dnf install pulseaudio-libs-devel alsa-lib-devel gstreamer1-devel gstreamer1-plugins-base-devel wayland-devel</pre>

Build in Linux / Dependency Packages

- QDoc Documentation Generator Tool

Distribution	Packages
Ubuntu and/or Debian:	<code>\$ sudo apt install clang libclang-dev</code>
Fedora 30:	<code>\$ su -c 'dnf install llvm-devel'</code>

Build in Linux

- Enter the following website (Qt requirements to compile):
 - URL: <https://doc.qt.io/qt-5/linux-building.html>
 - URL: https://wiki.qt.io/Building_Qt_5_from_Git#Linux.2FX11
- First, download the source code
 - Example: qt-everywhere-src-5.15.2.tar.xz
- Installing the License File (Commercially Licensed Qt Only)
- Decompress the source code in the desired path, for example “/tmp”

```
$ cd /tmp
$ tar -xvf qt-everywhere-src-5.15.2.tar.xz
Or
$ cd /tmp
$ gunzip qt-everywhere-opensource-src-%VERSION%.tar.gz      # uncompress the archive
$ tar xvf qt-everywhere-opensource-src-%VERSION%.tar        # unpack it
```

Build in Linux

- Installing the license file (Commercially licensed Qt only)
- Enter the Qt source code path

```
$ cd /tmp/qt-everywhere-opensource-src-%VERSION%
```

- Compile with GCC/g++
 - Configure the compiler

```
$ ./configure -platform linux-g++ -developer-build -prefix  
"/opt/qt5.15.2-x64" -nomake examples -nomake tests -skip qtwebengine  
-opensource -mp
```

- Start the compilation with the following command:

```
$ gmake
```

Build in Linux

- Compile with GCC/g++
 - Compile Qt Docs

```
$ make docs
```

- Install Qt Docs

```
$ sudo gmake install_docs    # Need to root privileges
```

- Remove source code directory

```
$ rm -rf /tmp/*
```

Build in Linux

- Compile with GCC/g++
 - Install Qt with the following command

```
$ sudo gmake install # Need to root privileges
```

- Clean source code

```
$ gmake clean
```

WARNING: -debug-and-release is only supported on Darwin and Windows platforms. Qt can be built in release mode with separate debug information, so -debug-and-release is no longer necessary.

Build Options

- Compiler Options: Compile with GCC/g++ (32 bit)

```
-platform linux-g++-32
```

- Compiler Options: Compile with Clang

```
-platform linux-clang
```

- Compiler Options: Cross-Compilation Options. To configure Qt for cross-platform development and deployment, the development toolchain for the target platform needs to be set up. This set up varies among the Supported Platforms.

```
-xplatform
```

Build Options

- Install Directories

```
-prefix /opt/Qt
```

- Excluding Qt Modules. Configure's `-skip` option allows top-level source directories to be excluded from the Qt build.

```
./configure -skip qtconnectivity
```

- Including or Excluding Features. The `-feature-<feature>` and `-no-feature-<feature>` options include and exclude specific features, respectively.

```
./configure -no-feature-accessibility
```

Build Options

- **Third-Party Libraries.** The Qt source packages include third-party libraries. To set whether Qt should use the system's versions of the libraries or to use the bundled version, pass either `-system` or `-qt` before the name of the library to configure.

```
./configure -no-zlib -qt-libjpeg -qt-libpng -system-xcb
```

Library Name	Bundled in Qt	Installed in System
zlib	-qt-zlib	-system-zlib
libjpeg	-qt-libjpeg	-system-libjpeg
libpng	-qt-libpng	-system-libpng
freetype	-qt-freetype	-system-freetype
PCRE	-qt-pcre	-system-pcre
HarfBuzz -NG	-qt-harfbuzz	-system-harfbuzz

Build Options

- OpenGL Options for Windows

- Dynamic: With the dynamic option, Qt will try to use native OpenGL first. If that fails, it will fall back to ANGLE and finally to software rendering in case of ANGLE failing as well.

```
configure.bat -opengl dynamic
```

- Desktop: With the desktop option, Qt uses the OpenGL installed on Windows, requiring that the OpenGL in the target Windows machine is compatible with the application. The -opengl option accepts two versions of OpenGL ES, es2 for OpenGL ES 2.0 or es1 for OpenGL ES Common Profile.

```
configure.bat -opengl desktop
```

- You can also use -opengl dynamic, which enable applications to dynamically switch between the available options at runtime. For more details about the benefits of using dynamic GL-switching, see Graphics Drivers.

```
configure.bat -opengl es2
```

Note: For a full list of options, consult the help with `configure -help`.

Checking Build and Run Settings

- The Qt Installer attempts to auto-detect the installed compilers and Qt versions. If it succeeds, the relevant kits will automatically become available in Qt Creator.

Adding Kits

- Qt Creator groups settings used for building and running projects as kits to make cross-platform and cross-configuration development easier. Each kit consists of a set of values that define one environment, such as a device, compiler, Qt version, and debugger command to use, and some metadata, such as an icon and a name for the kit. Once you have defined kits, you can select them to build and run projects.

Specifying Kit Settings

- Select Edit > Preferences > Kits > Add
- Specify kit settings. The settings to specify depend on the build system and device type.

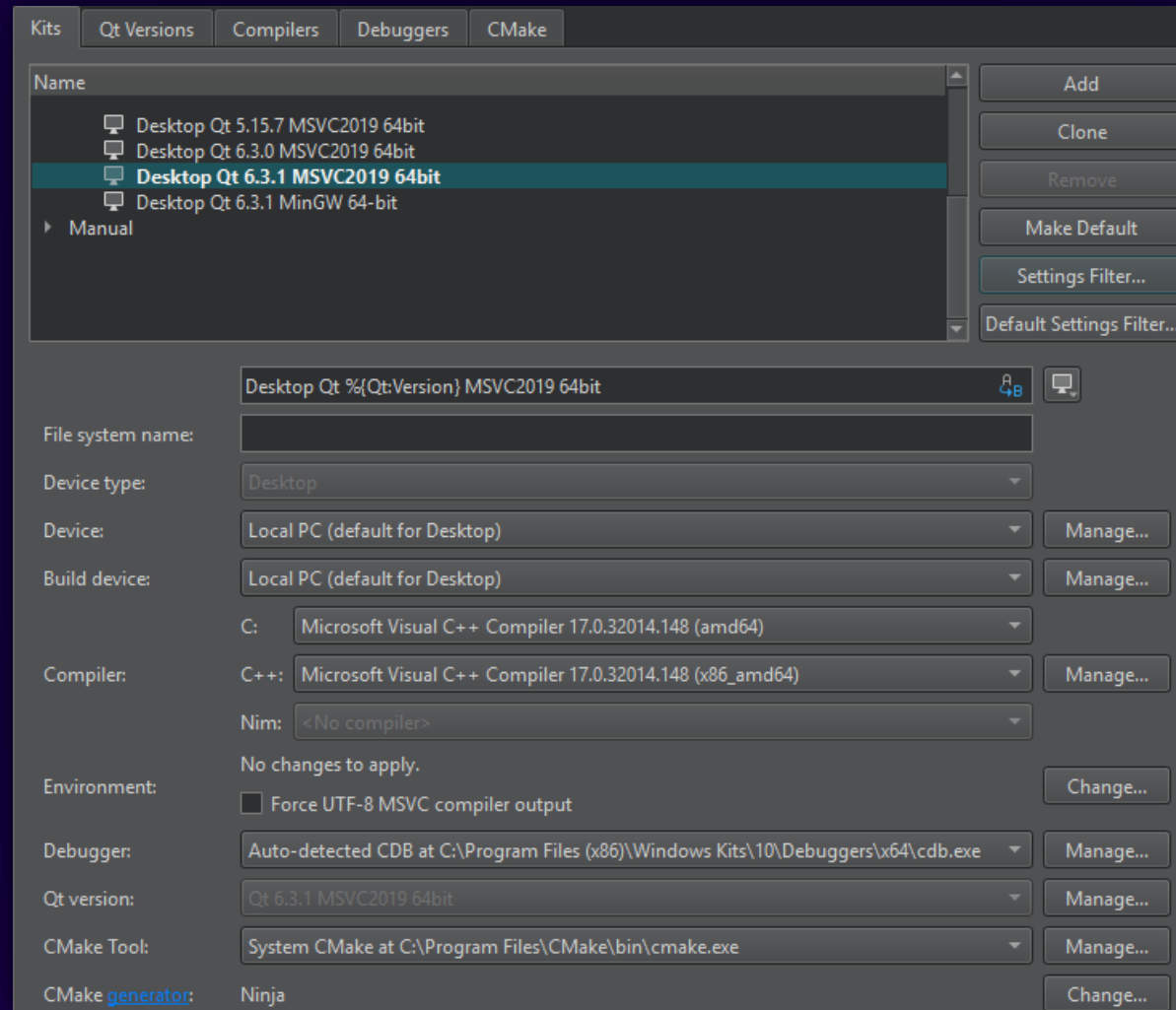
Kit Settings

- Name:
 - Name of the kit.
- Device type:
 - The device to build applications on.
- Device:
 - The device to run applications on.
- Compiler:
 - C or C++ compiler that you use to build the project. You can add compilers to the list if they are installed on the development PC.

Kit Settings

- Debugger:
 - Debugger to debug the project on the target platform. Qt Creator automatically detects available debuggers and displays a suitable debugger in the field. You can add debuggers to the list.
- Qt version
 - Qt version to use for building the project. You can add Qt versions to the list if they are installed on the development PC.

Kits:



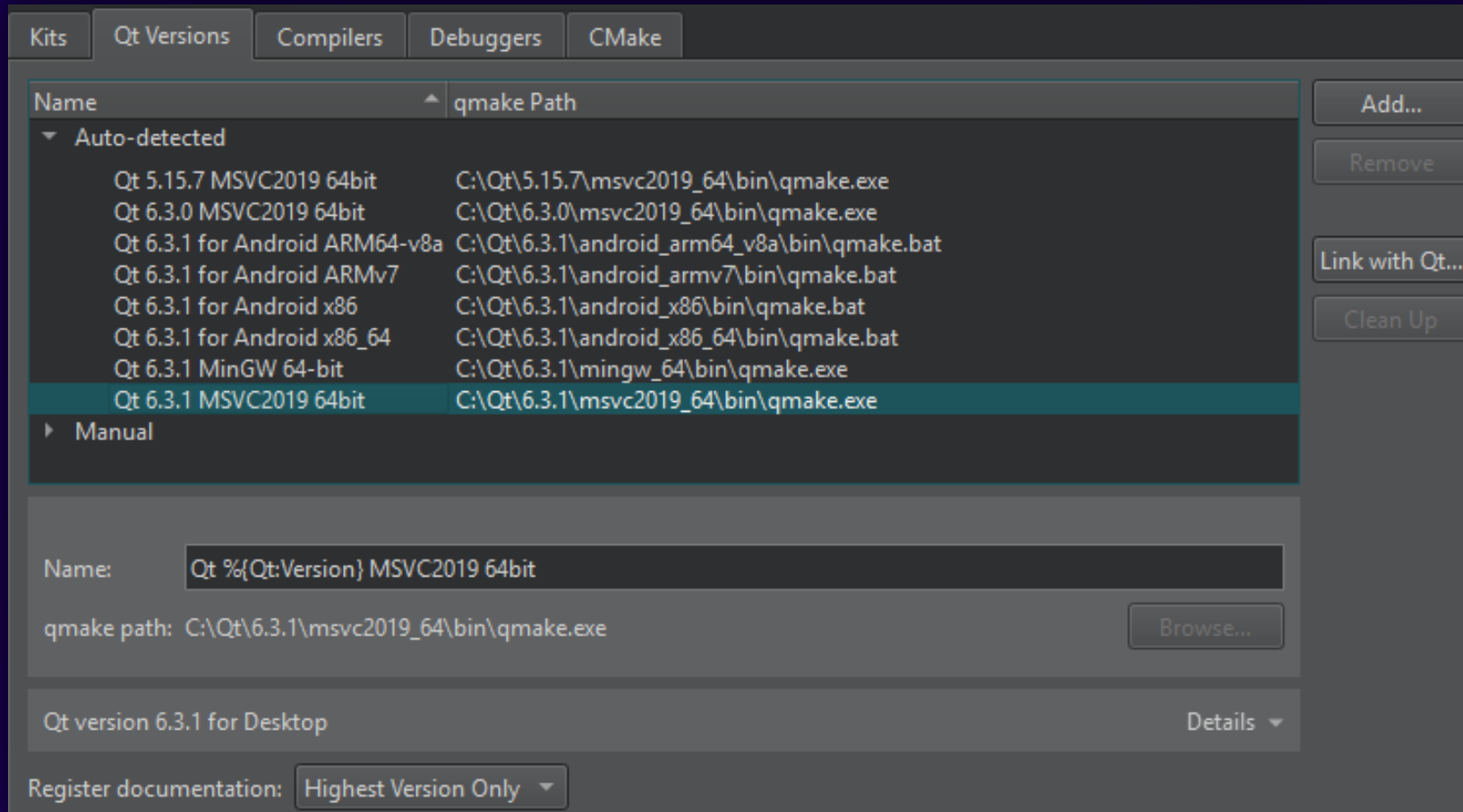
Adding Qt Versions

- Qt Creator allows you to have multiple versions of Qt installed on your development PC and use different versions to build your projects.

Setting Up New Qt Versions

- Select Edit > Preferences > Kits > Qt Versions > Add.
- Select the qmake executable for the Qt version that you want to add.

Qt Versions:



Adding Compilers

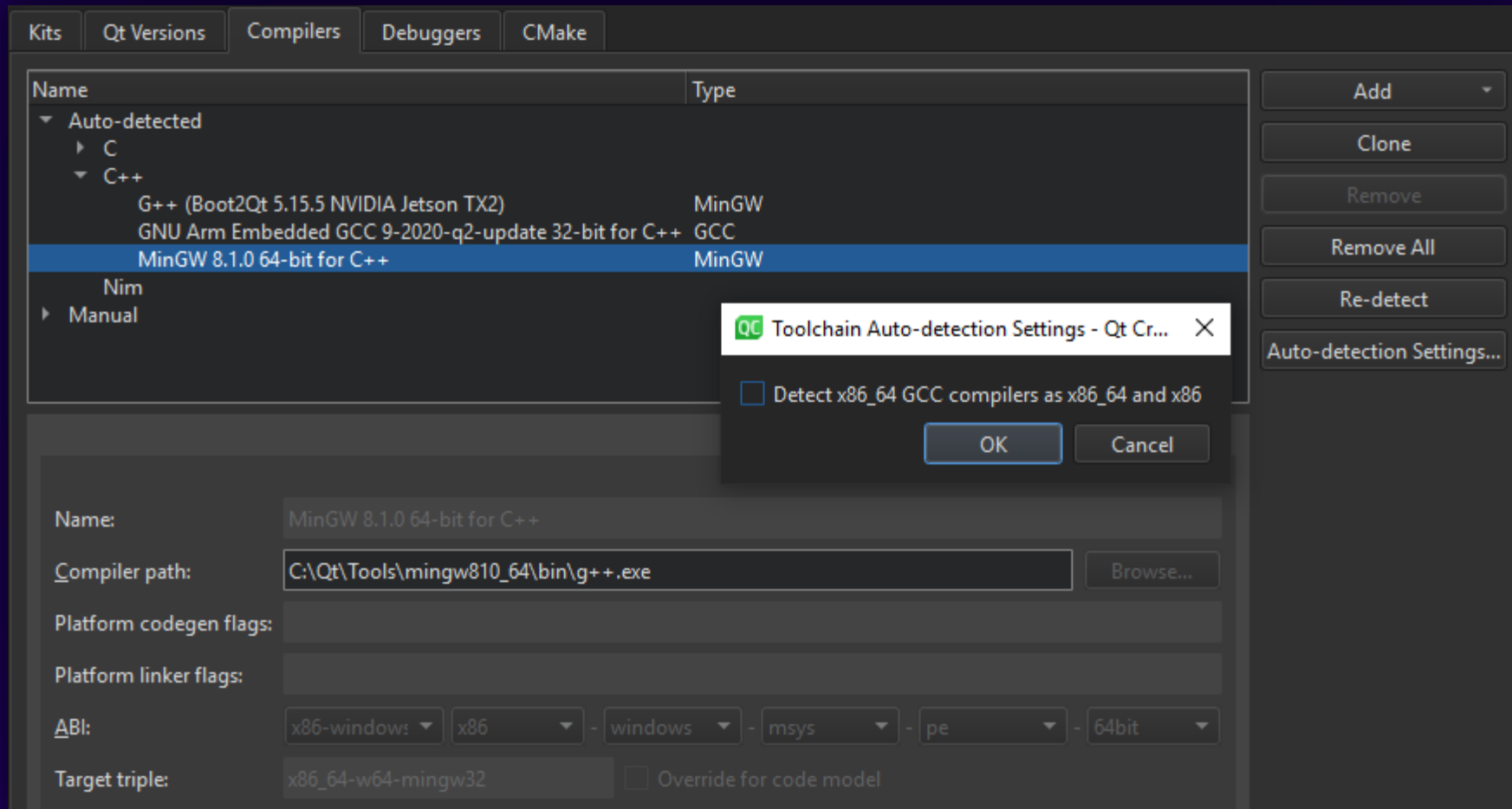
- Qt is supported on a variety of 32-bit and 64-bit platforms, and can usually be built on each platform with GCC, a vendor-supplied compiler, or a third party compiler.

You can add the following compilers to build applications

- **Clang** is a C, C++, Objective C, and Objective C++ front-end for the LLVM compiler for Windows, Linux, and macOS.
- **clang-cl** is an alternative command-line interface to Clang that is compatible with the Visual C++ compiler, cl.exe.
- **GNU Compiler Collection (GCC)** is a compiler for Linux and macOS.
- **ICC (Intel C++ Compiler)** is a group of C and C++ compilers. Only the GCC-compatible variant, available for Linux and macOS, is currently supported by Qt Creator.
- **MinGW (Minimalist GNU for Windows)** is a native software port of GCC and GNU Binutils for use in the development of native Microsoft Windows applications on Windows. MinGW is distributed together with Qt Creator and Qt for Windows.
- **MSVC (Microsoft Visual C++ Compiler)** is a C++ compiler that is installed with Microsoft Visual Studio.
- **Nim** is the Nim Compiler for Windows, Linux, and macOS.
- **QCC** is the interface for compiling C++ applications for QNX.

Note: MSVC compiler exists in Windows Software Development Kit (SDK)

Qt Compilers:

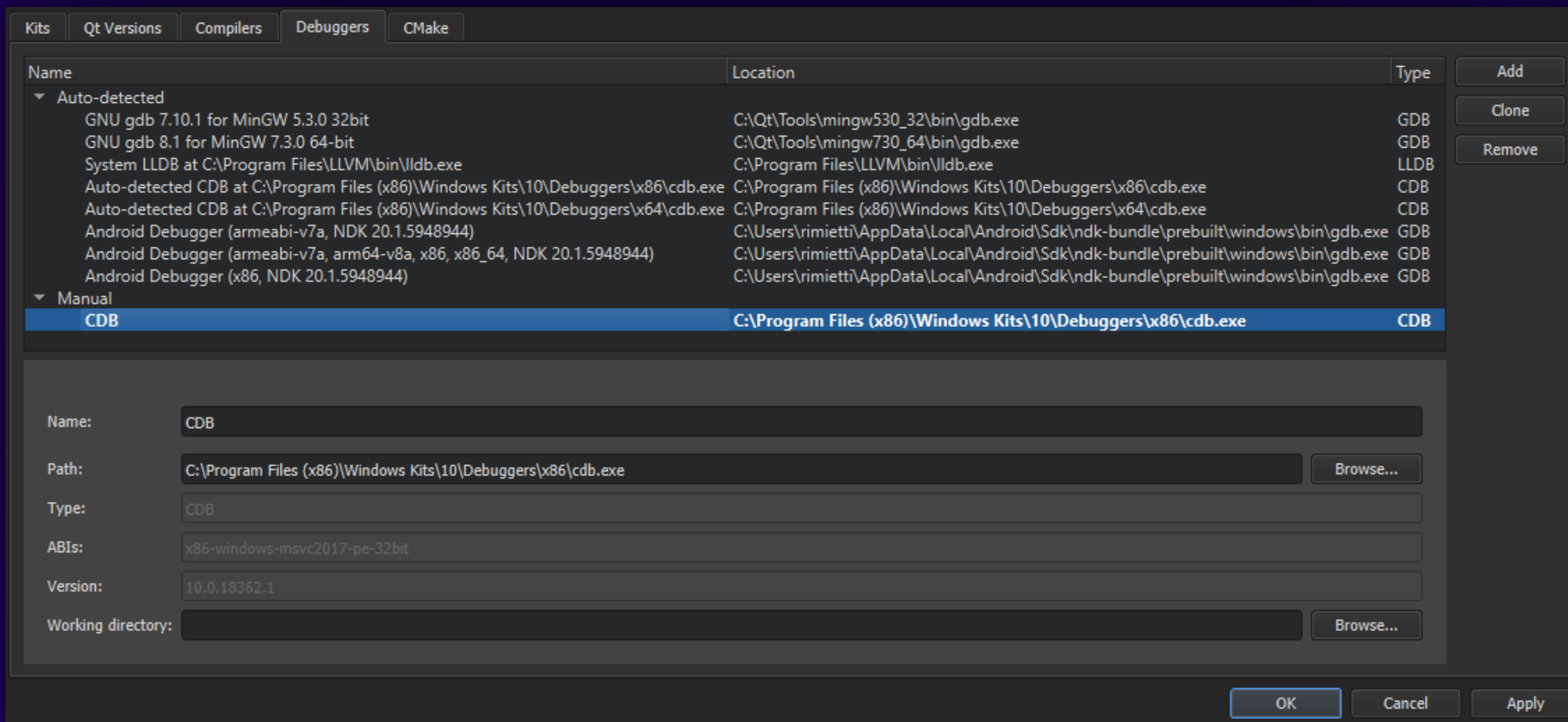


Adding Debuggers

- The Qt Creator debugger plugin acts as an interface between the Qt Creator core and external native debuggers such as the GNU Symbolic Debugger (GDB), the Microsoft Console Debugger (CDB), a QML/JavaScript debugger, and the debugger of the low level virtual machine (LLVM) project, LLDB.
- Select Edit > Preferences > Kits > Debuggers > Add.
- In the Path field, specify the path to the debugger binary:
 - **For CDB** (Windows only), specify the path to the Windows Console Debugger executable.
 - **For GDB**, specify the path to the GDB executable. The executable must be built with Python scripting support enabled.
 - **For LLDB** (experimental), specify the path to the LLDB executable.

Note: CDB compiler exists in Windows Software Development Kit (SDK)

Qt Debuggers:



Concepts

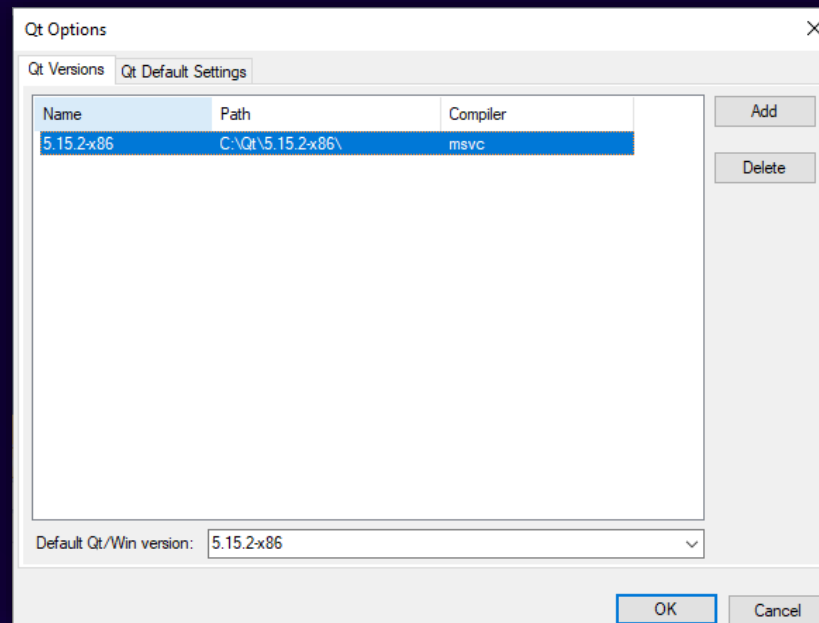
- **Shadow building** means building a project in a separate directory, the build directory. The build directory is different from the source directory. One of the benefits of shadow building is that it keeps your source directory clean, which makes it faster to switch between build configurations.
- Abi
- llvm

Prerequisite

- Install Visual Studio (Example: Visual Studio 2019)
- Install Visual Studio addin (Example: qt-vsaddin-msvc2019-2.6.0-rev.07.vsix)

Adding Kits

- Select Extensions > Qt VS Tools > Qt Options > Add (qmake file from Compiled Qt for MSVC)



Qt

Thank You

By Ali Panahi

