Building Access to Retro

Table of Contents

1. Dependencies

There are three dependencies that need to be installed for Access to Retro to compile:

- CMake
- SDL2
- Qt6

For macOS this guide will assume that Homebrew is installed on your system. Homebrew is package manager similar to those found on Linux systems, if you are using macOS and do not have Homebrew installed then please see https://brew.sh/.

1.1. CMake

1.1.1. Windows

If Visual Studio is installed on your computer there is a high chance that CMake is installed already, however, if it's not then there are two days to install it:

- By installing Visual Studio and making sure "C++ Desktop Development" component is checked to be installed (you can also modify existing Visual Studio installation using Visual Studio Installer program)
- By using installer from https://cmake.org/download/

1.1.2. MacOS/Linux

It is very common for Linux distributions to have CMake pre-installed however it is not always the case and can be installed using your distribution's package manager, the package name will most likely be 'cmake'. If you are using macOS then it can be installed using homebrew using command:

"brew install cmake"

It is strongly discouraged to use installer/binaries from https://cmake.org/download/ for Unix based systems.

1.2. SDL2

1.2.1. Windows

To install SDL2 on Windows first you need to download the library from https://github.com/libsdl-org/SDL/releases/tag/release-2.26.5. Make sure to choose "SDL2-devel-2.26.5-VC.zip" if you are using Visual C++ Compiler that comes with Visual Studio.

You need to then unpack this zip file to wherever you keep your libraries, for example: if you are keeping libraries on the root of C drive then the path to the SDL2 library would be C:/SDL2. It is very important to remember where the library was unpacked as it will be needed for later.

1.2.2. MacOS/Linux

On Unix systems a package manager should be used, the package name will be different based on your distribution, however, it will generally be one of the following: "libsdl2-dev", "libsdl2", "sdl2". On some distributions you might need to manually install gfx module for example "libsdl2-gfx". On macOS you can use Homebrew with this command:

"brew install sdl2 sdl2_gfx"

1.3. Qt6

1.3.1. Windows

To install Qt you can use an open-source installer from https://www.qt.io/download-open-source. It requires a free account and allows you to choose the components that you want installed, in this case you want to install version 6.5.0. Once again you need to remember where Qt was installed to.

1.3.2. MacOS/Linux

On Unix systems the installation can be done, just like above, using the open-source installer, however, once again a package manager is strongly recommended to avoid issues, the package name will be different based on your distribution, however, it will generally be one of the following: "qt6", "libqt6" "qt6-dev" "libqt6-dev". On macOS you can use Homebrew with this command:

"brew install qt@6"

Warning: As of writing this document Homebrew have reverted its Qt version back to 6.4.3. This should not cause any issues however, if you want, you can use open-source installer to install Qt6.5.0 but using Homebrew is still recommended on macOS.

2. Building the Project

First, the project needs to be cloned using Git. To do that you can use this command:

“git clone https://gitlab.kingston.ac.uk/K2028420/access-to-retro.git”

You can also use this if you have GitLab SSH token configured:

“git clone git@gitlab.kingston.ac.uk:K2028420/access-to-retro.git”

You can then navigate to the project directory using:

“cd access-to-retro/access-to-retro”

And then create a new build folder and navigate to it using:

“mkdir build && cd build”

It is now time to use CMake to configure the project and prepare MSBuild project (on windows) or Unix Makefile (on unix), the command used for this will be different depending on your operating system.

2.1. Windows

Please note that for the commands to work you need to use “Developer Command Prompt” provided by visual studio, this can be usually found by searching using your start menu.

To run CMake on windows you should use following command:

“cmake .. -A x64 -DCMAKE_BUILD_TYPE=Release -DSDL2_DIR=SDL_PATH -DQt6_DIR=QT_PATH”

Please notice the parts of the command in red. You need to replace SDL_PATH with a path to “cmake” subfolder inside your SDL installation folder, for example:

If you installed SDL2 to the root of C drive (C:/SDL2) then you would replace SDL_PATH with

“C:/SDL2/cmake”

For QT_PATH you need to replace it with a path to where you installed Qt6 however you need to add following sub-path to it:

“VERSION/COMPILER/lib/cmake/Qt6”

Where VERSION is the version number, for example: 6.5.0. and COMPILER with the name of the compiler folder, for example (in the case of Visual Studio C++ Compiler 2019) “msvc2019_64”. For those examples, and assuming Qt has been installed to the root of C drive (C:/Qt) then you would replace QT_PATH with:

“C:/Qt/6.5.0/msvc2019_64/lib/cmake/Qt6”

An example of full command:

“cmake .. -A x64 -DCMAKE_BUILD_TYPE=Release -DSDL2_DIR=C:/SDL2/cmake -DQt6_DIR=C:/Qt/6.5.0/msvc2019_64/lib/cmake/Qt6”

It is now time to build the project, this can be done using following command:

“cmake --build . --target ALL_BUILD --config RELEASE”

After the build completes you will be able to find following parts of the Access to Retro projects (paths are relative to build folder):

- Frontend: frontend/Release/AccessToRetro.exe
- Developer Library: developer-library/Release/access-to-retro.lib
- CHIP8 Emulator Virtual Console: chip8-virtual-console/Release/access-to-retro-chip8.windows.vc

While the project has been built there is still one more thing that needs to be done on windows specifically – the .DLL files from SDL2 and Qt6 need to be copied to the location of AccessToRetro.exe (frontend/Release/AccessToRetro.exe [relative from build folder]).

For SDL2 you need to navigate to the install location of the library, then go to "lib" folder and then to "x64" folder, for example: "C:/SDL2/lib/x64" – there you will find a file named "SDL2.dll" that needs to be copied to the previously mentioned directory where AccessToRetro.exe is.

For Qt6 there is a tool that can do that automatically, you need to open a Qt specific command prompt for your compiler (you can use start menu search and search for "Qt 6.5.0 (MSVC 2019 64-bit)" for Visual Studio C++ compiler specific command prompt, you should then navigate, using that terminal, to the location of AccessToRetro.exe and execute following command:

"windeployqt AccessToRetro.exe"

The Access to Retro frontend can now be started if the command is successful and the CHIP8 virtual console that was also built can be installed using the "Manager" option in the frontend.

2.2 MacOS/Linux

On Unix systems the process is much easier, to run CMake you can use the following command:

"cmake .. -DCMAKE_BUILD_TYPE=Release"

You can then use this command to build the project:

"make"

Warning: some Linux distributions have pre-installed CMake that defaults to Ninja build system, if the "make" command does not work then try "ninja".

After the build completes you will be able to find following parts of the Access to Retro projects (paths are relative to build folder):

- Frontend: frontend/AccessToRetro (or frontend/AccessToRetro.app on macOS)
- Developer Library: developer-library/access-to-retro.a
- CHIP8 Emulator Virtual Console: chip8-virtual-console /access-to-retro-chip8.X.vc (where X is the name of the operating system)

Please note that, on macOS, you might need to disable Gatekeeper to run the application due to Apple rules. Please see this: https://lucidgen.com/en/how-to-disable-gatekeeper-sip-on-mac/. Depending on your macOS version you might need to only allow the app to run in the privacy settings instead of turning Gatekeeper off.