Let’s first discuss some definitions:

Macro: A macro is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro.

Preprocessor: is a program that converts a .cpp source file containing # directives (such as #include, #ifndef etc) into a source file that contains no such directives. Generally compiler calls it automatically.

Compiler: turns source code (.cpp) into object code (.o)

Linker: turns object code (.o) files with libraries into raw executable (platform specific, for example .exe files)

Building: is the sequence composed of Compiling and Linking with possibly other tasks such as installer creation.

Makefile: Simply a technique to explicitly tell the Compiler and the Linker all the information they need to Compile the source code and link those object codes into executables properly.

qmake tool: qmake will add relevant libraries to linked against and ensure that build lines for moc and uic are included in the generated Makefiles.

Meta-object-compiler (**moc**): The **moc** tool reads a C++ header file. If it finds one or more class declarations that contain the **Q_OBJECTS** macro, it produces a C++ source file containing the meta-object code for those classes.

The C++ source file generated by **moc** must be compiled and linked with the implementation of the class.

When Qt creator is used **qmake** is called automatically to create the Makefiles, build rules will be included that call the **moc** when required, so it is not needed to call **moc** directly.

Static vs. Dynamic Library: Static lib are directly put into the executable files as though they were .o file which increases application size, Where dynamic or shared libraries are located in a specific location in user’s system which are loaded automatically on application startup.

Now we try to understand the process of Compiling a C++ program, which not surprisingly varies from platform to platform. Qt provides tools like qmake, moc etc that make it easy to build applications on all platforms.

So from project creation to running an application; the whole process works like this in Qt/C++ especially when we are using Qt Creator IDE:

The .pro files defines all the attributes of the Qt projects like what are the

- .cpp files,
- .h files,
- what libraries to use,
- what kind of application is this application, libraries etc (APP, lib)

Additionally a .user.pro file is also created to store user system specific data.

qmake use this .pro file to generate Makefiles (Debug or Release). **Preprocessor** (to expand the .h files into cpp codes), **moc** (to expand Q_OBJECT like macros), uic (ui compiler) and rcc (resource compiler) supports appropriately to generate this native Makefile.

Then we invoke our native build system (either GNU **make**, or MS **nmake** or whatever), which will call our native compiler (**g++/gcc** or whatever) to use this Makefile. Which will subsequently using this makefile compile the .cpp to .o files and then link them together (all the .cpp files and static or dynamic libraries (.dll files) to generate the executable. These processes are called building an application. Then we run our application