Compilers

Chantilly Robotics (Team 612)

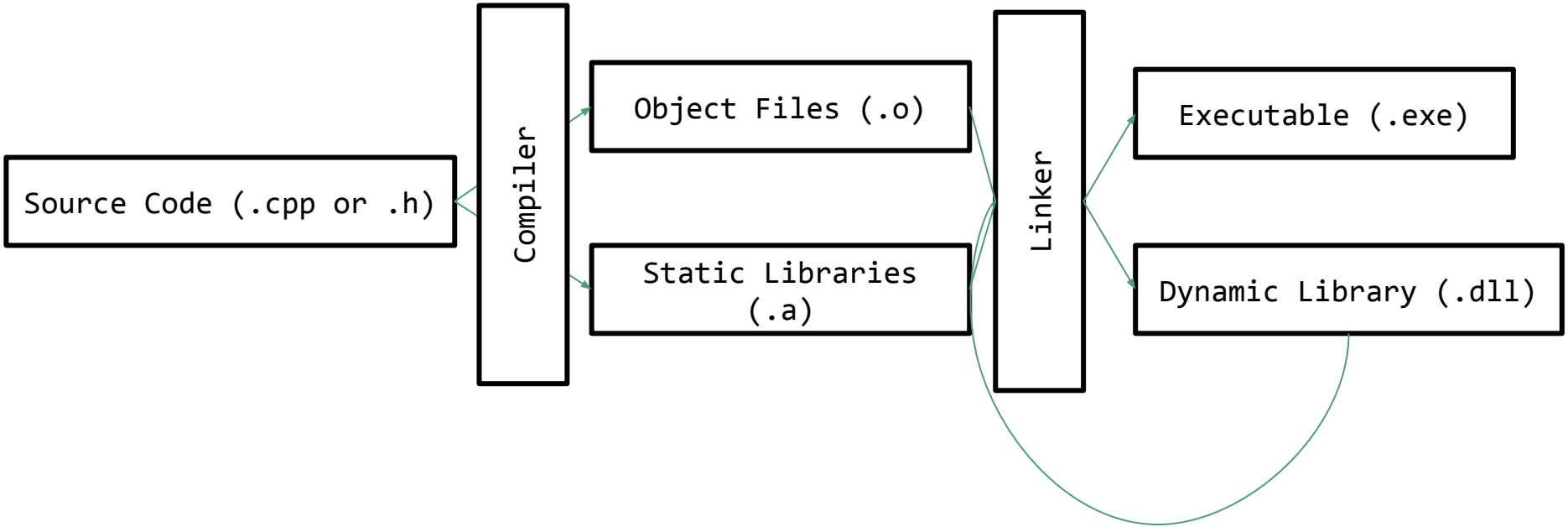
What are compilers?

- Computer program(s)
- Transforms source code to binary form
  - Binary can be executed by target CPU architecture
  - Source code is readily human readable, but binary form takes much longer time to understand
- Almost always from high level to low level
- End goal: link different parts of your code into an executable

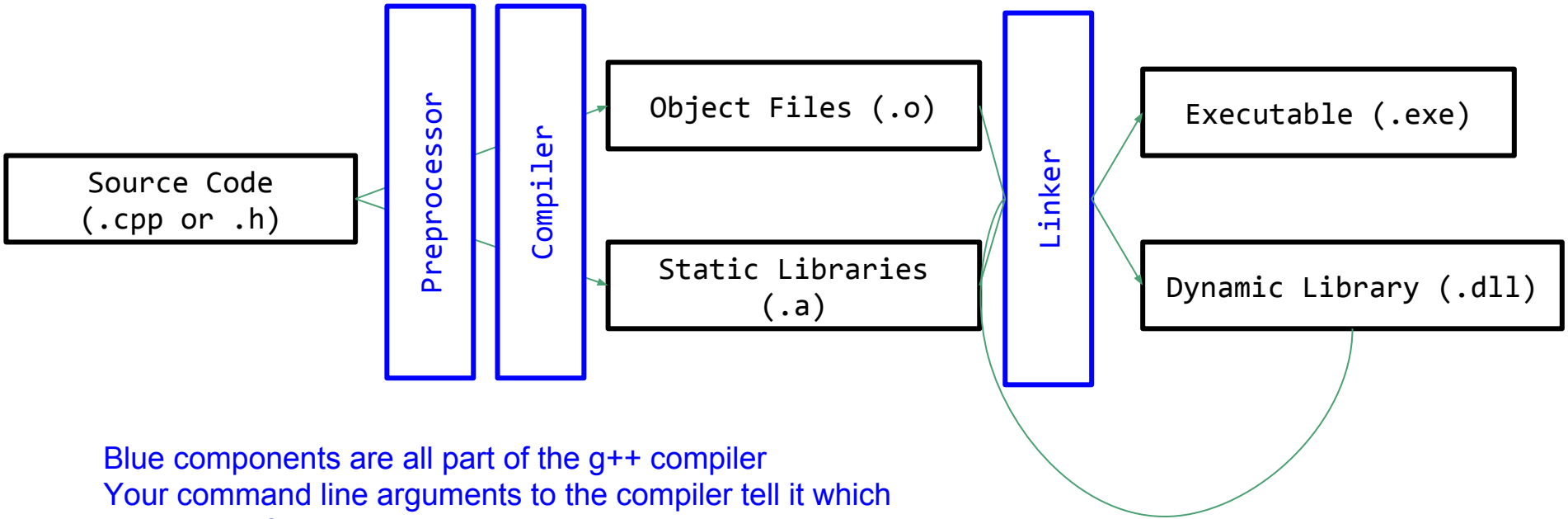
Which compiler do we use?

- We are using the g++ compiler, part of the GNU Compiler Collection
  - The GNU organization makes compilers for other languages too, hence the compiler collection
- The specific g++ we are using is called a cross compiler
  - A cross compiler executes on one computer architecture but builds an executable for another computer architecture
  - Example: Your intel x86 computer builds an executable for a ARM processor, such as one used in a cell phone or on the RoboRIO
- This is why the FRC competition makes you install a special compiler package
  - `arm-frc-linux-gnueabi-g++`

How does C++ compiling work? // How do?

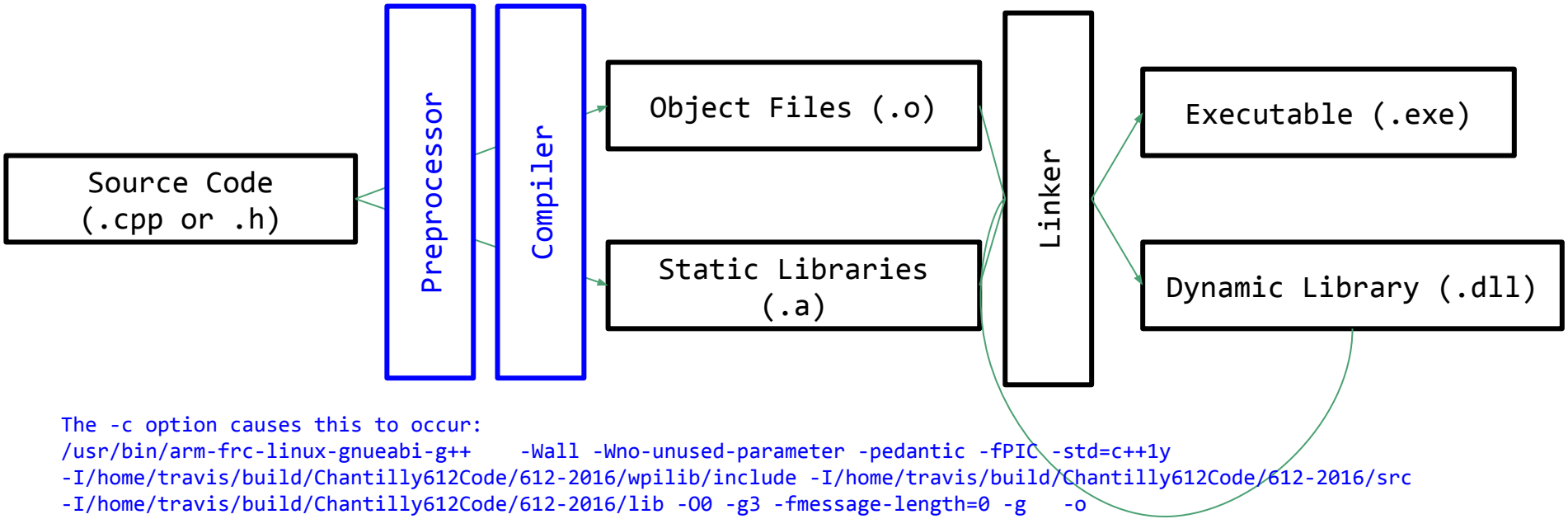


How does C++ compiling work? // How do?



Blue components are all part of the g++ compiler
Your command line arguments to the compiler tell it which steps to perform

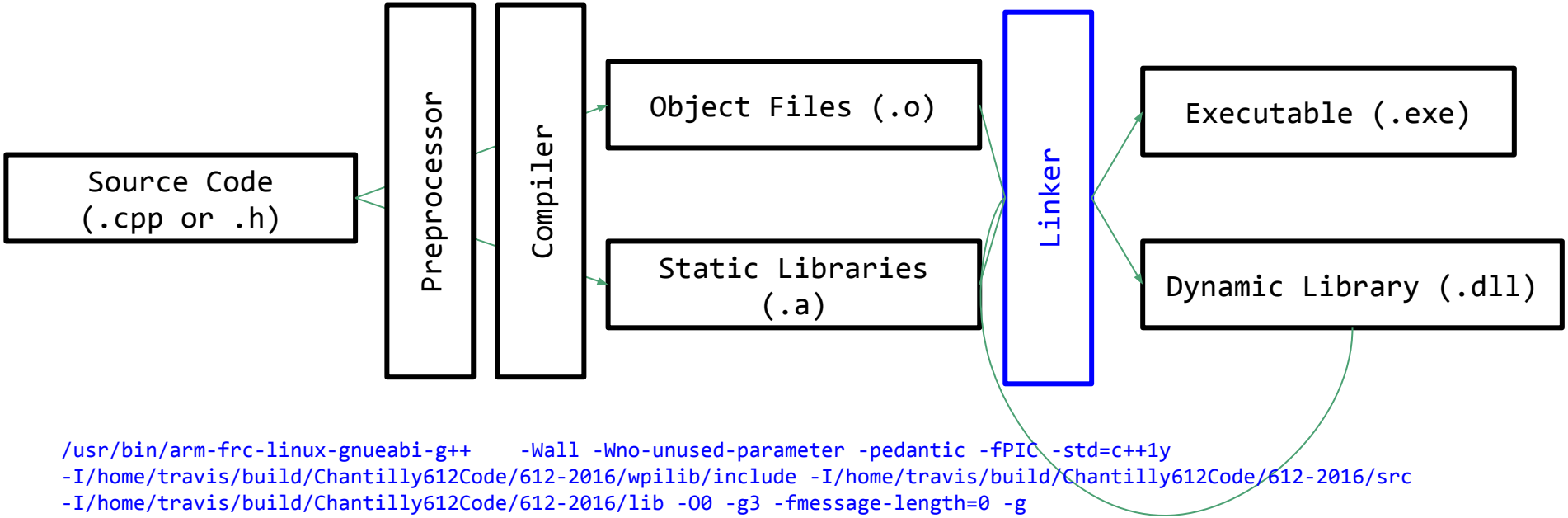
How does C++ compiling work? // How do?



The -c option causes this to occur:
/usr/bin/arm-frc-linux-gnueabi-g++ -Wall -Wno-unused-parameter -pedantic -fPIC -std=c++1y
-I/home/travis/build/Chantilly612Code/612-2016/wpilib/include -I/home/travis/build/Chantilly612Code/612-2016/src
-I/home/travis/build/Chantilly612Code/612-2016/lib -O0 -g3 -fmessage-length=0 -g -o
CMakeFiles/FRCProgram.dir/src/Commands/Drive/DriveSet.cpp.o -c
/home/travis/build/Chantilly612Code/612-2016/src/Commands/Drive/DriveSet.cpp

Note input file is .cpp and output file is .o

How does C++ compiling work? // How do?




```
/usr/bin/arm-frc-linux-gnueabi-g++    -Wall -Wno-unused-parameter -pedantic -fPIC -std=c++1y
-I/home/travis/build/Chantilly612Code/612-2016/wpilib/include -I/home/travis/build/Chantilly612Code/612-2016/src
-I/home/travis/build/Chantilly612Code/612-2016/lib -O0 -g3 -fmessage-length=0 -g
-L/home/travis/build/Chantilly612Code/612-2016/wpilib/lib -Wl,-rpath,/opt/GenICam_v2_3/bin/Linux_armv7-a
DriveJoystick.cpp.o DriveSet.cpp.o SetGear.cpp.o DriveDistance.cpp.o Shooter/SetShooter.cpp.o SetShooterAngle.cpp.o
AlignToShoot.cpp.o Shoot.cpp.o ShooterManualControl.cpp.o ShooterJoystick.cpp.o HorizontalFind.cpp.o
... many more .o files listed out here...
-o FRCProgram -rdynamic -lwpi

FRCProgram is the .exe file shown in the diagram
wpi is the static library that is linked from -lwpi (actual file is libwpi.a)
```

High Level vs Low Level

Pros:

- Easier to read
- Faster development time

Con:

- Slower
- Less direct access to hardware

Pros:

- Faster
- More direct access to hardware

Con:

- Hard to read
- Harder to debug

Program entry point

- The point at which a program begins to execute
- In C++, it's the `main` function (or array, or class, or random other thing, but just use a function. Don't even think about it.)
  - This means this is the first function to be executed (usually)