就是说，cmake -H. -Bbuild 这个命令仅仅需要运行一次就够了，如果后面更改了CMakeLists.txt 以及 源代码文件，也仅仅需要在运行cmake --build就行了

1.修改源代码文件，cmake不需要重新进行configuration，只需要进行重新进行 编译；make cmake --build就可以了

2.修改了CMakeLists.txt文件，只需要进行重新cmake的configuration就可以了，不需要进行重新编译，cmake --build就可以了

综上所述，就使用了 cmake --build就可以了

3.3. Workflow

There is a nice feature in CMake that can greatly simplify developer's workflow: native build tool will watch CMake sources for changes and run re-configure step on update automatically. In command-line terms it means that you have to run `cmake -H. -B_builds` **only once**, you don't need to run configure again after modification of CMakeLists.txt - you can keep using `cmake --build`.

3.3.1. Makefile example

Back to the example with `message`:

```
# CMakeLists.txt

cmake_minimum_required(VERSION 2.8)
project(foo)

add_executable(foo foo.cpp)

message("Processing CMakeLists.txt")
```

Examples on GitHub

- Repository
- Latest ZIP

Generate Makefile:

```
[minimal-with-message]> cmake -H. -B_builds
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
Processing CMakeLists.txt
-- Configuring done
-- Generating done
-- Build files have been written to: /.../minimal-with-message/_builds
```

And run build:

```
[minimal-with-message]> cmake --build _builds
Scanning dependencies of target foo
[ 50%] Building CXX object CMakeFiles/foo.dir/foo.cpp.o
[100%] Linking CXX executable foo
[100%] Built target foo
```

Executable `foo` created from `foo.cpp` source. Make tool knows that if there are no changes in `foo.cpp` then no need to build and link executable. If you run build again there will be no compile and link stage:

可执行文件所依赖的源代码文件如果说没有发生改变，如果再次执行编译命令，实质上将不会进行编译

```
[minimal-with-message]> cmake --build _builds
[100%] Built target foo
```

Let's "modify" `foo.cpp` source:

```
[minimal-with-message]> touch foo.cpp
[minimal-with-message]> cmake --build _builds
Scanning dependencies of target foo
[ 50%] Building CXX object CMakeFiles/foo.dir/foo.cpp.o
[100%] Linking CXX executable foo
[100%] Built target foo
```

Make detects that executable `foo` is out-of-date and rebuild it. Well, that's what build systems designed for :)

Now let's "change" CMakeLists.txt. Do we need to run `cmake -H. -B_builds` again? The answer is NO - just keep using `cmake --build _builds`. CMakeLists.txt added as dependent file to the Makefile:

```
[minimal-with-message]> touch CMakeLists.txt
[minimal-with-message]> cmake --build _builds
Processing CMakeLists.txt
-- Configuring done
-- Generating done
-- Build files have been written to: /.../minimal-with-message/_builds
[100%] Built target foo
```

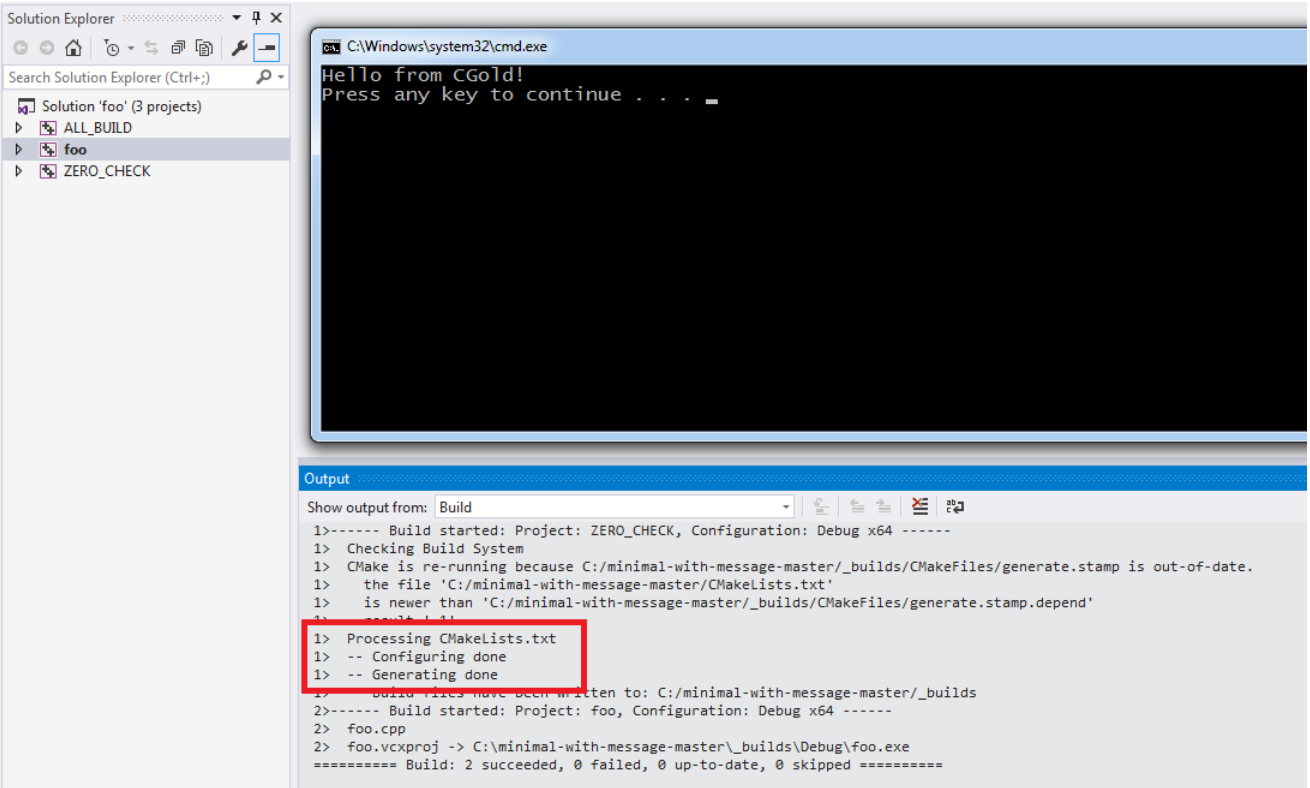
You see `Processing CMakeLists.txt`, `Configuring done` and `Generating done` indicating that CMake code parsed again and new Makefile generated. Since we don't change the way target `foo` is built (like adding new build flags or compile definitions) there is no compile/link stages.

If you “modify” both CMake and C++ code you will see the full configure/generate/build stack of commands:

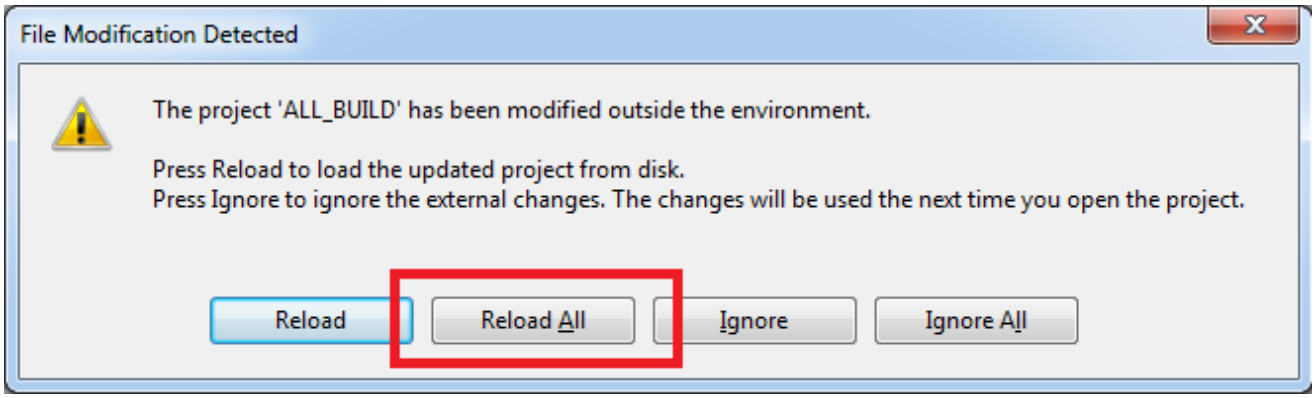
```
[minimal-with-message]> touch CMakeLists.txt foo.cpp
[minimal-with-message]> cmake --build _builds
Processing CMakeLists.txt
-- Configuring done
-- Generating done
-- Build files have been written to: /.../minimal-with-message/_builds
Scanning dependencies of target foo
[ 50%] Building CXX object CMakeFiles/foo.dir/foo.cpp.o
[100%] Linking CXX executable foo
[100%] Built target foo
```

3.3.2. Visual Studio example

Same is true for other generators as well. For example when you touch CMakeLists.txt and try to run `foo` target in Visual Studio:

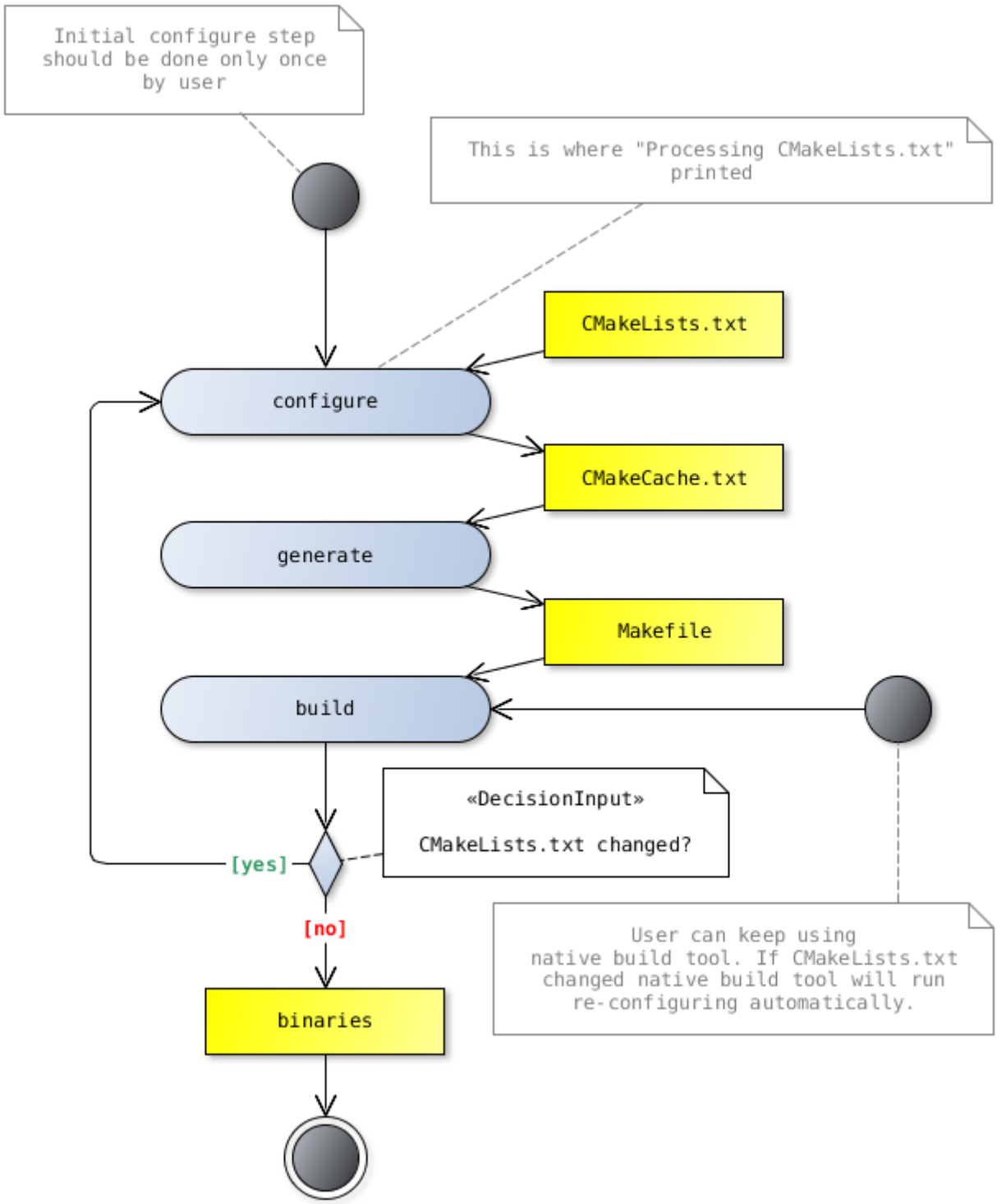


IDE will notify you about update of the project. You can click “Reload All” to reload new configuration:



3.3.3. UML activity diagram

Activity diagram for workflow described above:



3.3.4. Suspicious behavior

If your workflow doesn't match configure-once approach then it may be a symptom of wrongly written CMake code. Especially when you have to run `cmake -H. -B_builds` twice or when `cmake --build _builds` doesn't catch updates from CMake code.

CMake issue

- XCode: Real targets do not depend on ZERO_CHECK