3.6.2. Cache variables

Cache variables saved in CMakeCache.txt file:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(abc "687" CACHE STRING "")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hcache-cmakecachetxt -B_builds
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> grep abc _builds/CMakeCache.txt
abc:STRING=687
```

3.6.2.1. No scope

cache 永远都是 全局的变量

Unlike regular variables CMake cache variables have no scope and are set globally:

```
# Top level CMakeLists.txt

cmake_minimum_required(VERSION 2.8)
project(foo NONE)

add_subdirectory(boo)

message("A: ${A}")
```

开辟一个新的scope

```
# CMakeLists.txt from 'boo' directory

set(A "123" CACHE STRING "")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hcache-no-scope -B_builds
A: 123
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.2.2. Double set

If variable is already in cache then command `set(... CACHE ...)` will have no effect - old variable will be used still:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(abc "123" CACHE STRING "")
message("Variable from cache: ${abc}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cp double-set/1/CMakeLists.txt double-set/
[usage-of-variables]> cmake -Hdouble-set -B_builds
Variable from cache: 123
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> grep abc _builds/CMakeCache.txt
abc:STRING=123
```

Update CMakeLists.txt (don't remove cache!):

```
--- /examples/usage-of-variables/double-set/1/CMakeLists.txt
+++ /examples/usage-of-variables/double-set/2/CMakeLists.txt
@@ -1,5 +1,5 @@
 cmake_minimum_required(VERSION 2.8)
 project(foo NONE)

-set(abc "123" CACHE STRING "")
+set(abc "789" CACHE STRING "")
 message("Variable from cache: ${abc}")
```

```
[usage-of-variables]> cp double-set/2/CMakeLists.txt double-set/
[usage-of-variables]> cmake -Hdouble-set -B_builds
Variable from cache: 123
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> grep abc _builds/CMakeCache.txt
abc:STRING=123
```

3.6.2.3. -D

Cache variable can be set by `-D` command line option. Variable that set by `-D` option take priority over `set(... CACHE ...)` command.

```
[usage-of-variables]> cmake -Dabc=444 -Hdouble-set -B_builds
Variable from cache: 444
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> grep abc _builds/CMakeCache.txt
abc:STRING=444
```

3.6.2.4. Initial cache

If there are a lot of variables to set it’s not so convenient to use `-D`. In this case user can define all variables in separate file and load it by `-C`:

```
# cache.cmake

set(A "123" CACHE STRING "")
set(B "456" CACHE STRING "")
set(C "789" CACHE STRING "")
```

```
# CMakeLists.txt

cmake_minimum_required(VERSION 2.8)
project(foo NONE)

message("A: ${A}")
message("B: ${B}")
message("C: ${C}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -C initial-cache/cache.cmake -Hinitial-cache -B_builds
loading initial cache file initial-cache/cache.cmake
A: 123
B: 456
C: 789
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.2.5. Force

If you want to set cache variable even if it’s already present in cache file you can add `FORCE`:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(A "123" CACHE STRING "" FORCE)
message("A: ${A}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -DA=456 -Hforce -B_builds
A: 123
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

This is quite surprising behavior for user and conflicts with the nature of cache variables that was designed to store variable once and globally.

⚠ Warning

`FORCE` usually is an indicator of badly designed CMake code.

3.6.2.6. Force as a workaround

`FORCE` can be used to fix the problem that described eariler:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(A "123")
set(A "456" CACHE STRING "")

message("A: ${A}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hno-force-confuse -B_builds
A: 456
-- Configuring done    这个是因为一开始没有缓存，所以就会设置456
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> cmake -Hno-force-confuse -B_builds
A: 123
-- Configuring done    这个是因为上面的操作已经造成了缓存的存在，不会再进行新的而设置
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

With `FORCE` variable will be set even it’s already present in cache, so regular variable with the same name will be unset too each time:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(A "123")
set(A "456" CACHE STRING "" FORCE)

message("A: ${A}")
```

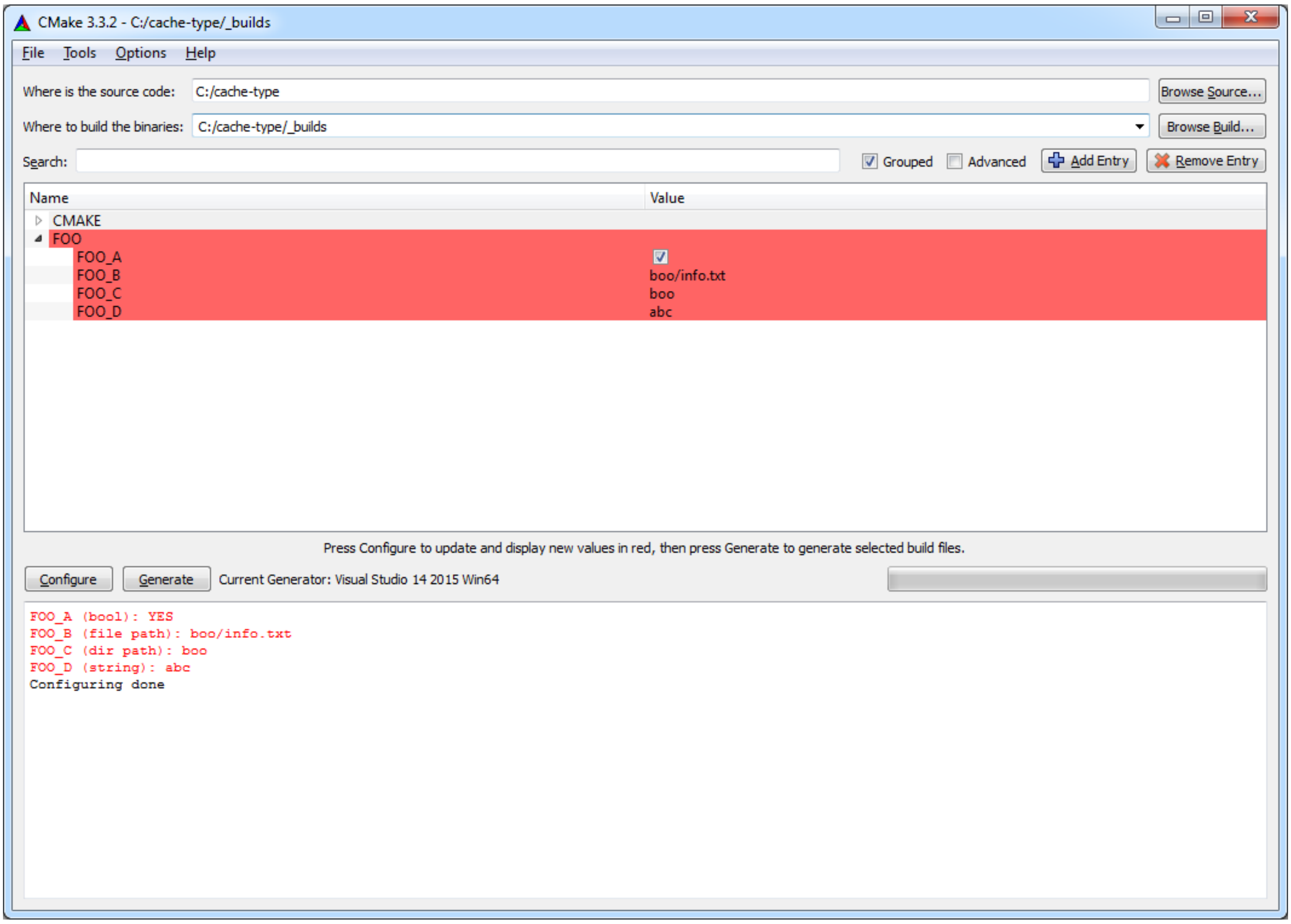
```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hforce-workaround -B_builds
A: 456
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> cmake -Hforce-workaround -B_builds
A: 456
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.2.7. Cache type

Though type of any variable is **always** string you can add some hints which will be used by CMake-GUI:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(FOO_A "YES" CACHE BOOL "Variable A")
set(FOO_B "boo/info.txt" CACHE FILEPATH "Variable B")
set(FOO_C "boo/" CACHE PATH "Variable C")
set(FOO_D "abc" CACHE STRING "Variable D")

message("FOO_A (bool): ${FOO_A}")
message("FOO_B (file path): ${FOO_B}")
message("FOO_C (dir path): ${FOO_C}")
message("FOO_D (string): ${FOO_D}")
```

Run configure:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search: ☑ Grouped ☐ Advanced

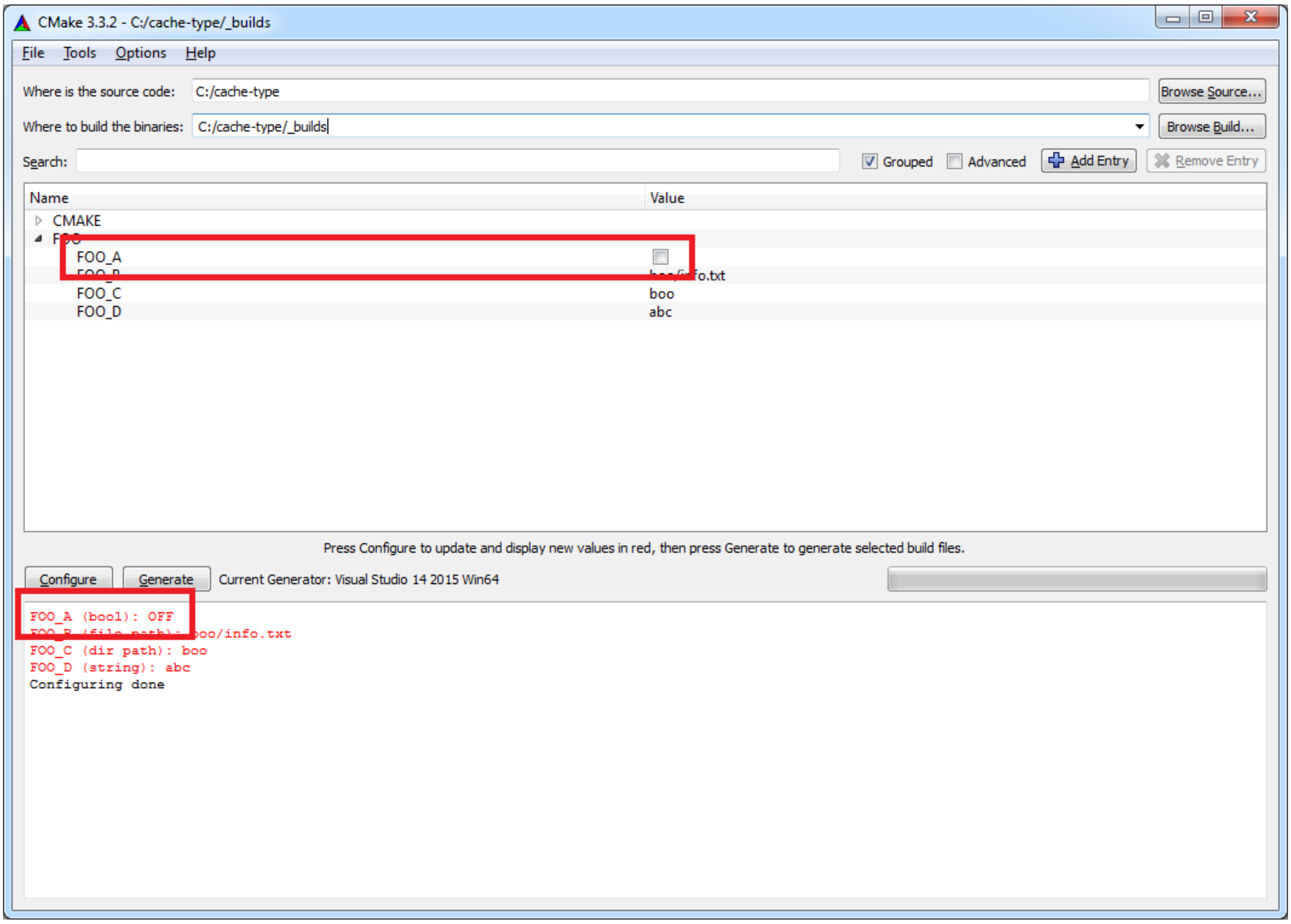
Name	Value
CMAKE	
FOO	
FOO_A	☑
FOO_B	boo/info.txt
FOO_C	boo
FOO_D	abc

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): YES
FOO_B (file path): boo/info.txt
FOO_C (dir path): boo
FOO_D (string): abc
Configuring done
```

Variable `FOO_A` will be treated as boolean. Uncheck box and run configure:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

☑ Grouped ☐ Advanced

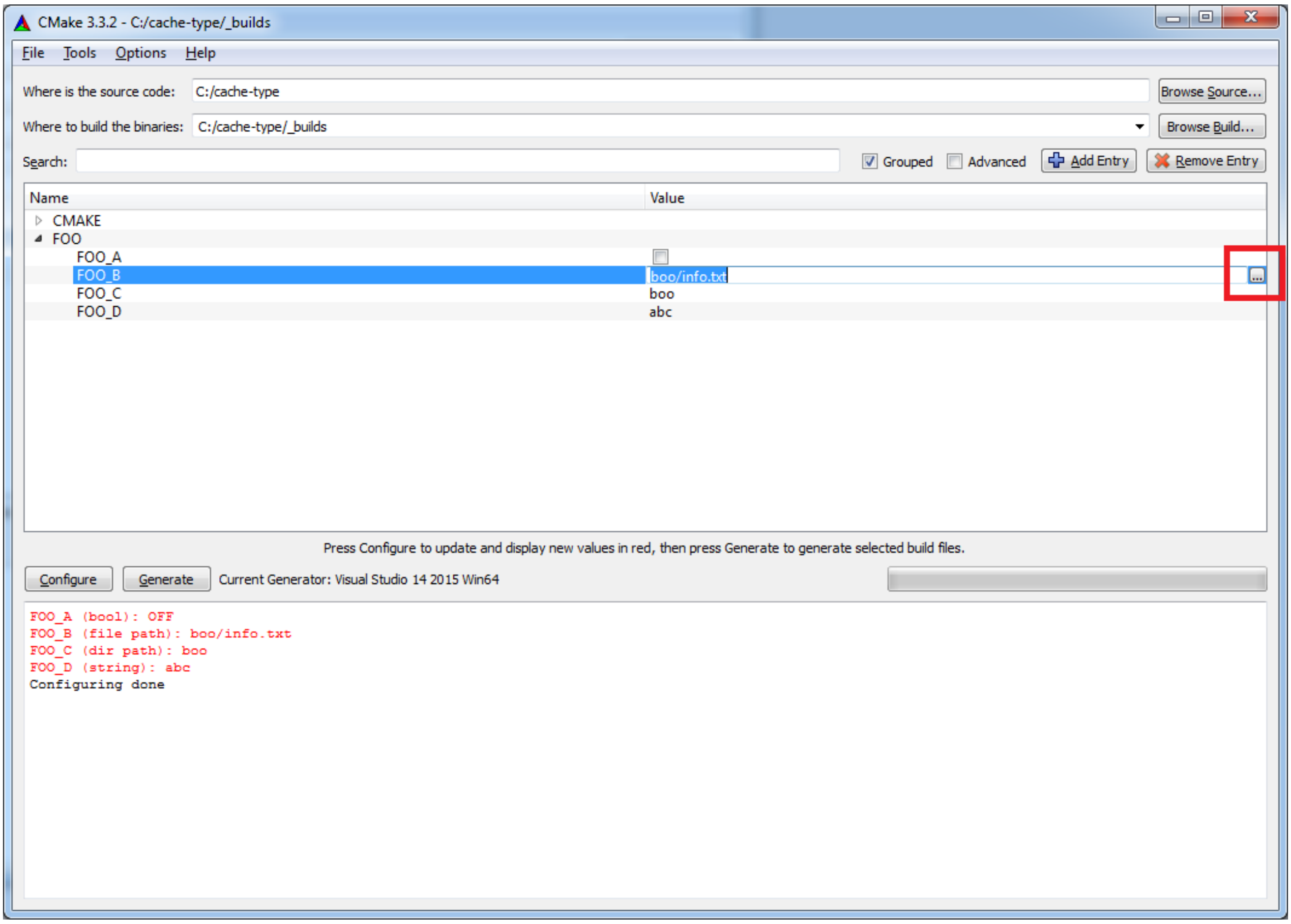
Name	Value
CMAKE	
FOO	
FOO_A	☐
FOO_B	boo/info.txt
FOO_C	boo
FOO_D	abc

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): boo/info.txt
FOO_C (dir path): boo
FOO_D (string): abc
Configuring done
```

Variable `FOO_B` will be treated as path to the file. Click on `...`:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

☑ Grouped ☐ Advanced

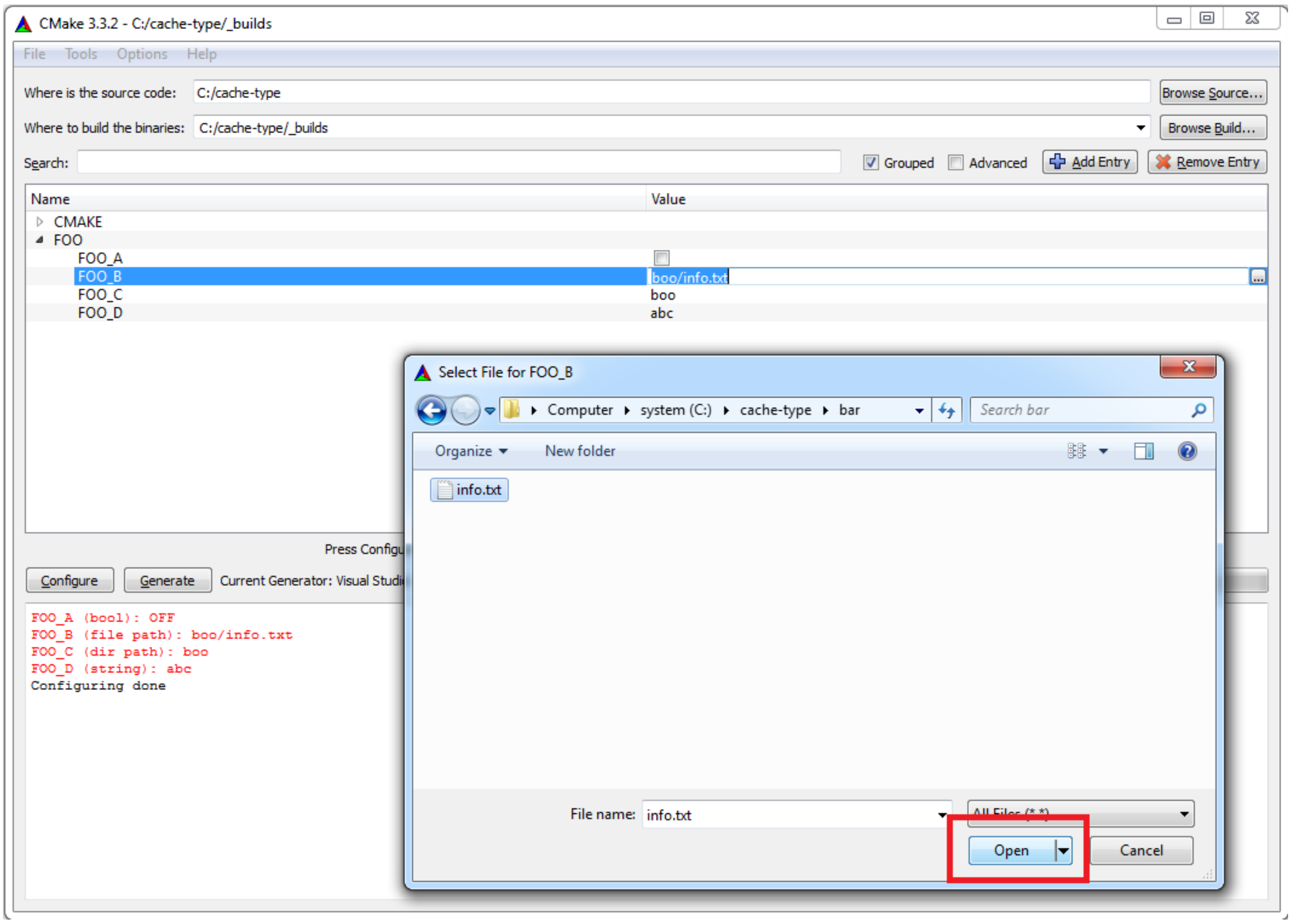
Name	Value
CMAKE	
FOO	
FOO_A	☐
FOO_B	boo/info.txt
FOO_C	boo
FOO_D	abc

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): boo/info.txt
FOO_C (dir path): boo
FOO_D (string): abc
Configuring done
```

Select file:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

Name	Value
▷ CMAKE	
◢ FOO	
FOO_A	□
FOO_B	boo/info.txt
FOO_C	boo
FOO_D	abc

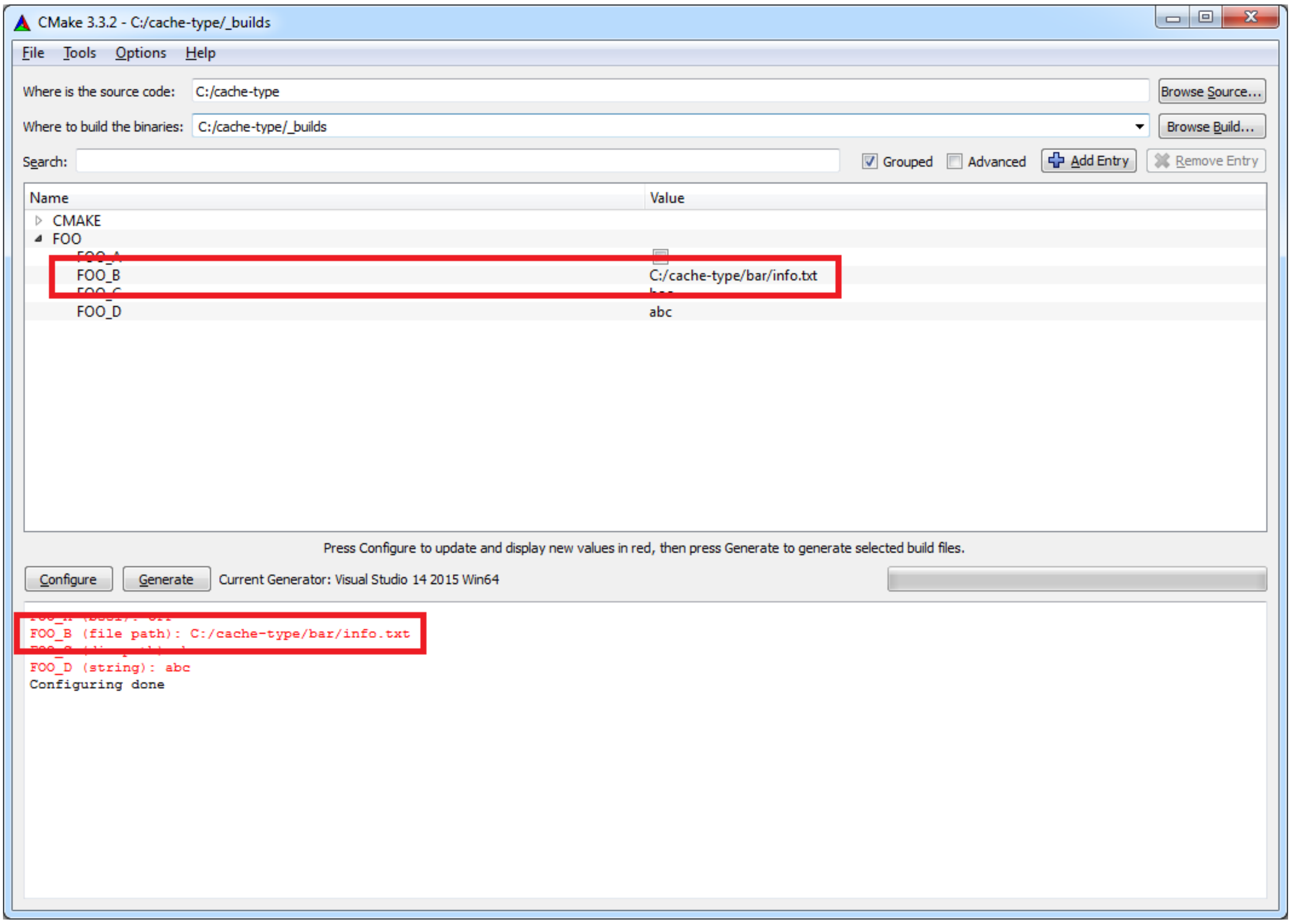
Press Configu

Current Generator: Visual Studi

File name: info.txt

```
FOO_A (bool): OFF
FOO_B (file path): boo/info.txt
FOO_C (dir path): boo
FOO_D (string): abc
Configuring done
```

Run configure:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

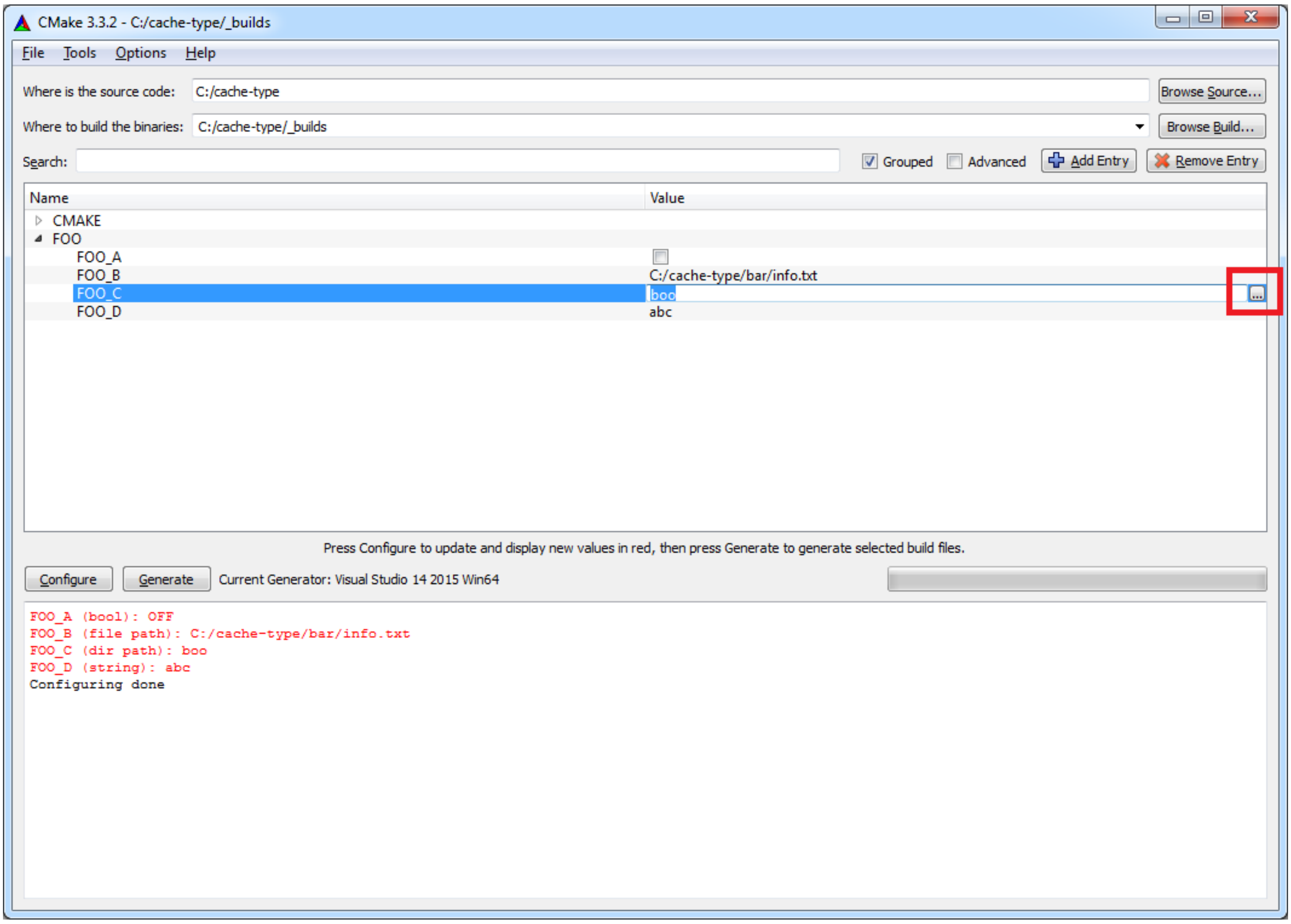
Name	Value
▷ CMAKE	
◢ FOO	
FOO_A	□
FOO_B	C:/cache-type/bar/info.txt
FOO_C	boo
FOO_D	abc

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): C:/cache-type/bar/info.txt
FOO_C (dir path): boo
FOO_D (string): abc
Configuring done
```

Variable `FOO_C` will be treated as path to directory. Click on `...`:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

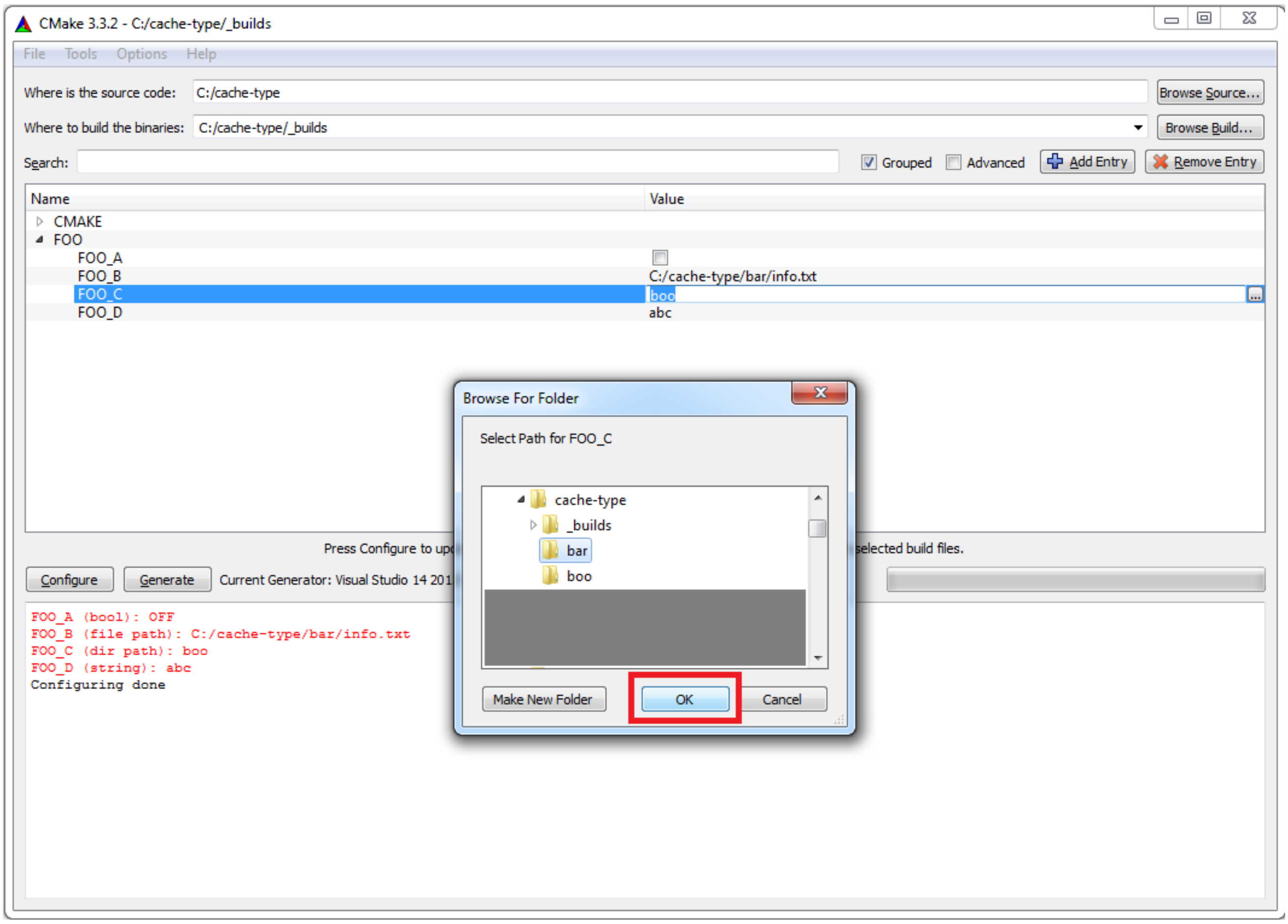
Name	Value
CMAKE	
FOO	
FOO_A	□
FOO_B	C:/cache-type/bar/info.txt
FOO_C	boo
FOO_D	abc

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): C:/cache-type/bar/info.txt
FOO_C (dir path): boo
FOO_D (string): abc
Configuring done
```

Select directory:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

Name	Value
CMAKE	
FOO	
FOO_A	□
FOO_B	C:/cache-type/bar/info.txt
FOO_C	boo
FOO_D	abc

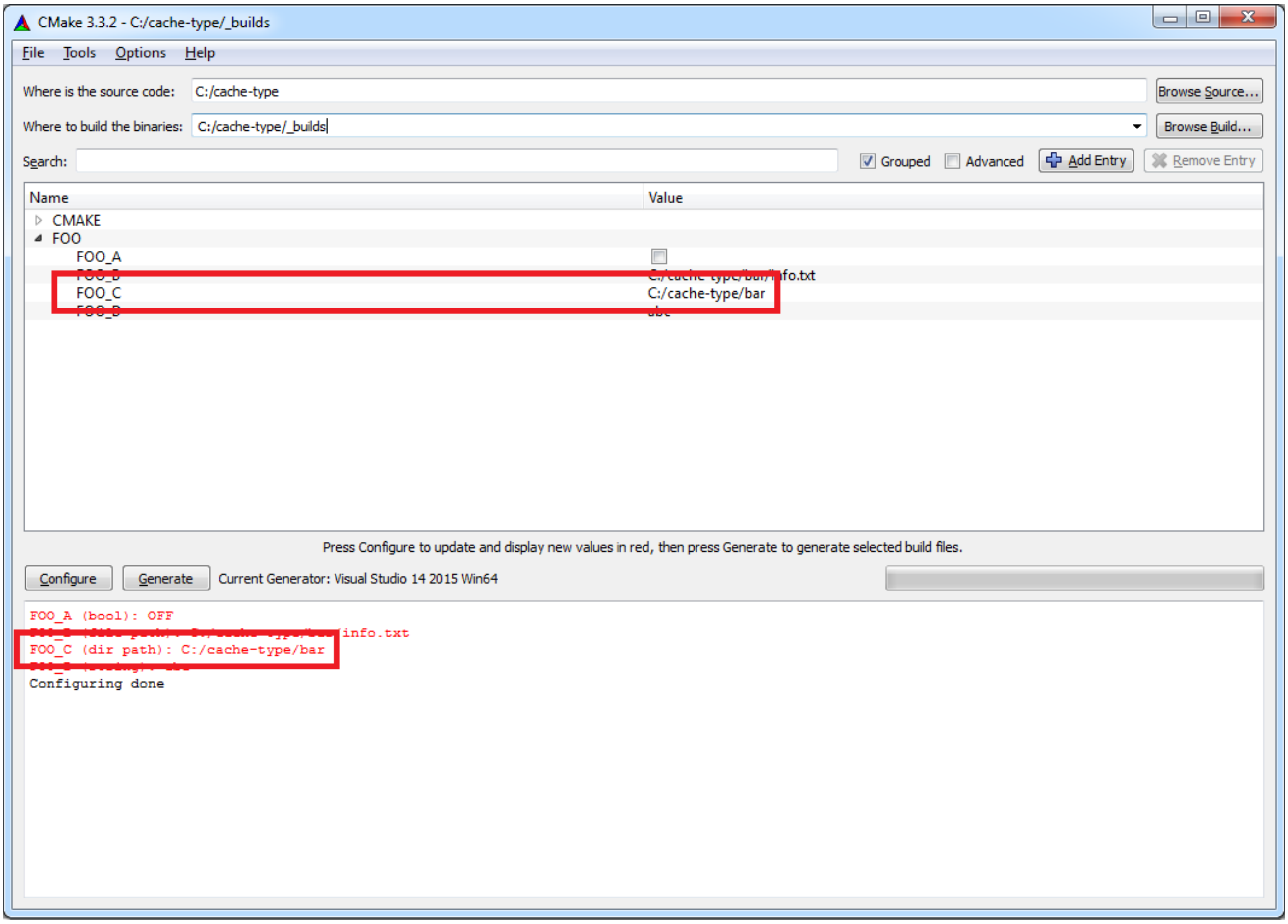
Select Path for FOO_C

Press Configure to upd… selected build files.

Current Generator: Visual Studio 14 201…

```
FOO_A (bool): OFF
FOO_B (file path): C:/cache-type/bar/info.txt
FOO_C (dir path): boo
FOO_D (string): abc
Configuring done
```

Run configure:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

☑ Grouped □ Advanced

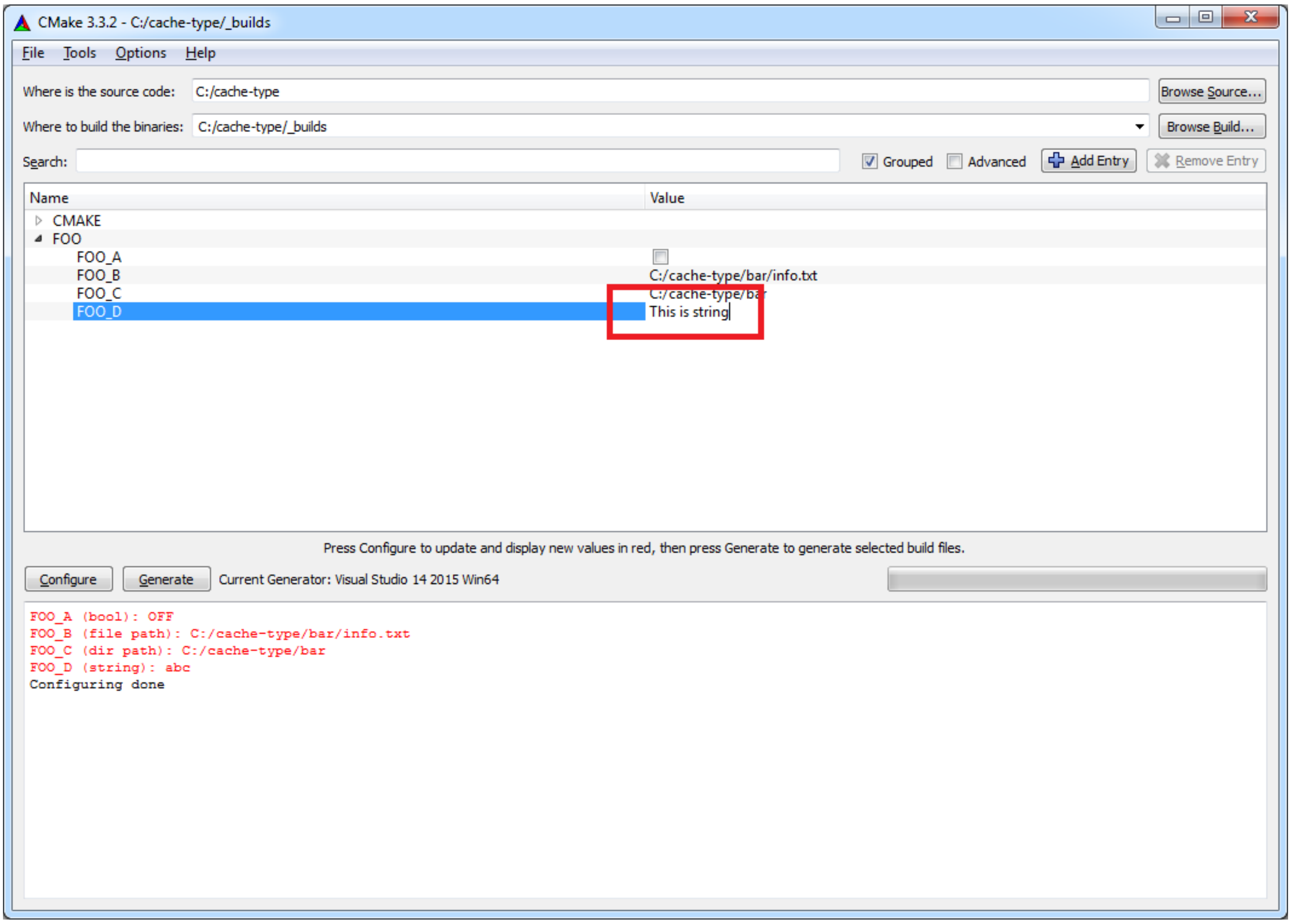
Name	Value
CMAKE	
FOO	
FOO_A	□
FOO_B	C:/cache-type/bar/info.txt
FOO_C	C:/cache-type/bar
FOO_D	abc

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): C:/cache-type/bar/info.txt
FOO_C (dir path): C:/cache-type/bar
FOO_D (string): abc
Configuring done
```

Variable `FOO_D` will be treated as string. Click near variable name and edit:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

☑ Grouped □ Advanced

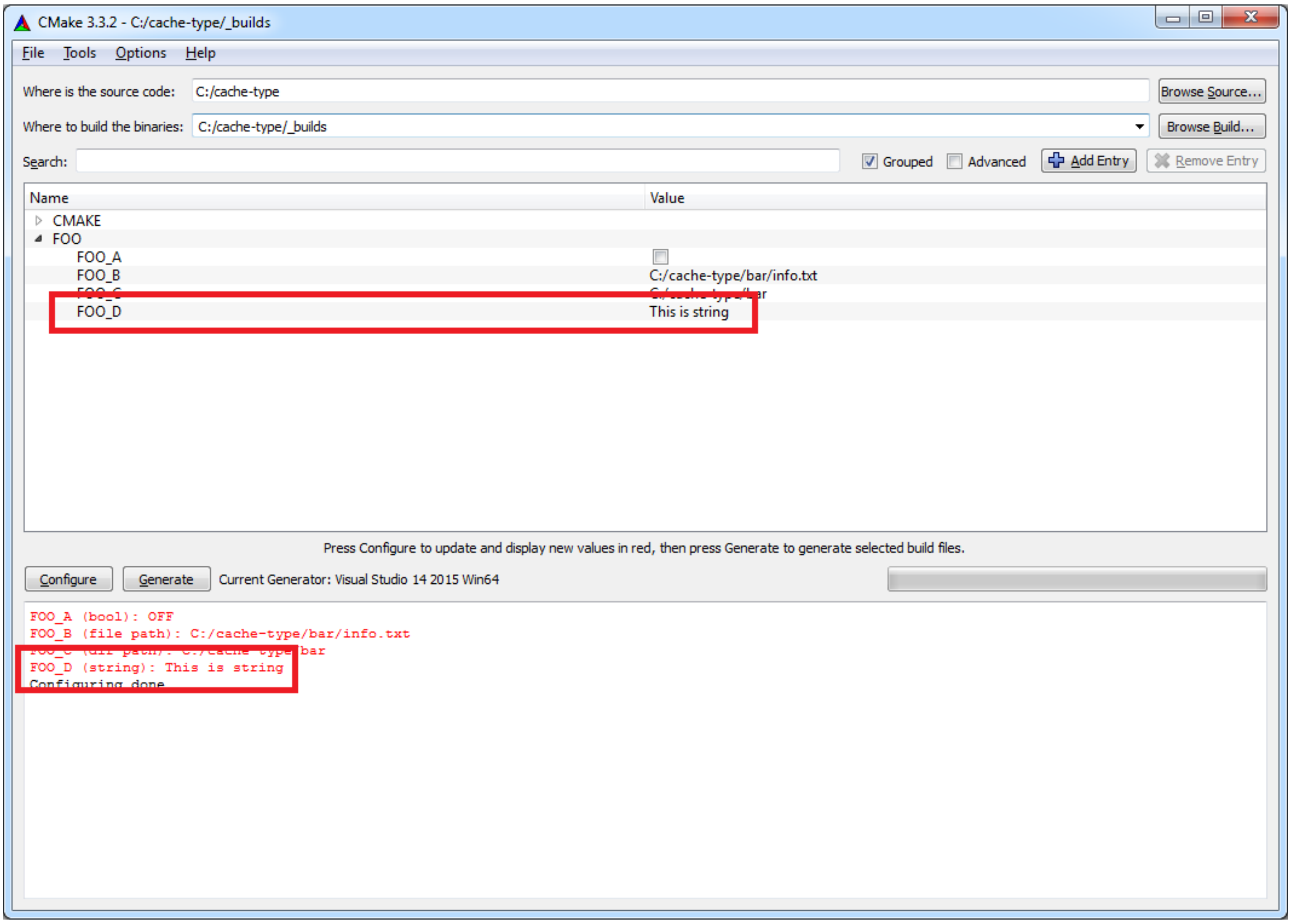
Name	Value
CMAKE	
FOO	
FOO_A	□
FOO_B	C:/cache-type/bar/info.txt
FOO_C	C:/cache-type/bar
FOO_D	This is string

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): C:/cache-type/bar/info.txt
FOO_C (dir path): C:/cache-type/bar
FOO_D (string): abc
Configuring done
```

Run configure:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search: ☑ Grouped □ Advanced

Name	Value
CMAKE	
FOO	
FOO_A	□
FOO_B	C:/cache-type/bar/info.txt
FOO_C	C:/cache-type/bar
FOO_D	This is string

Press Configure to update and display new values in red, then press Generate to generate selected build files.

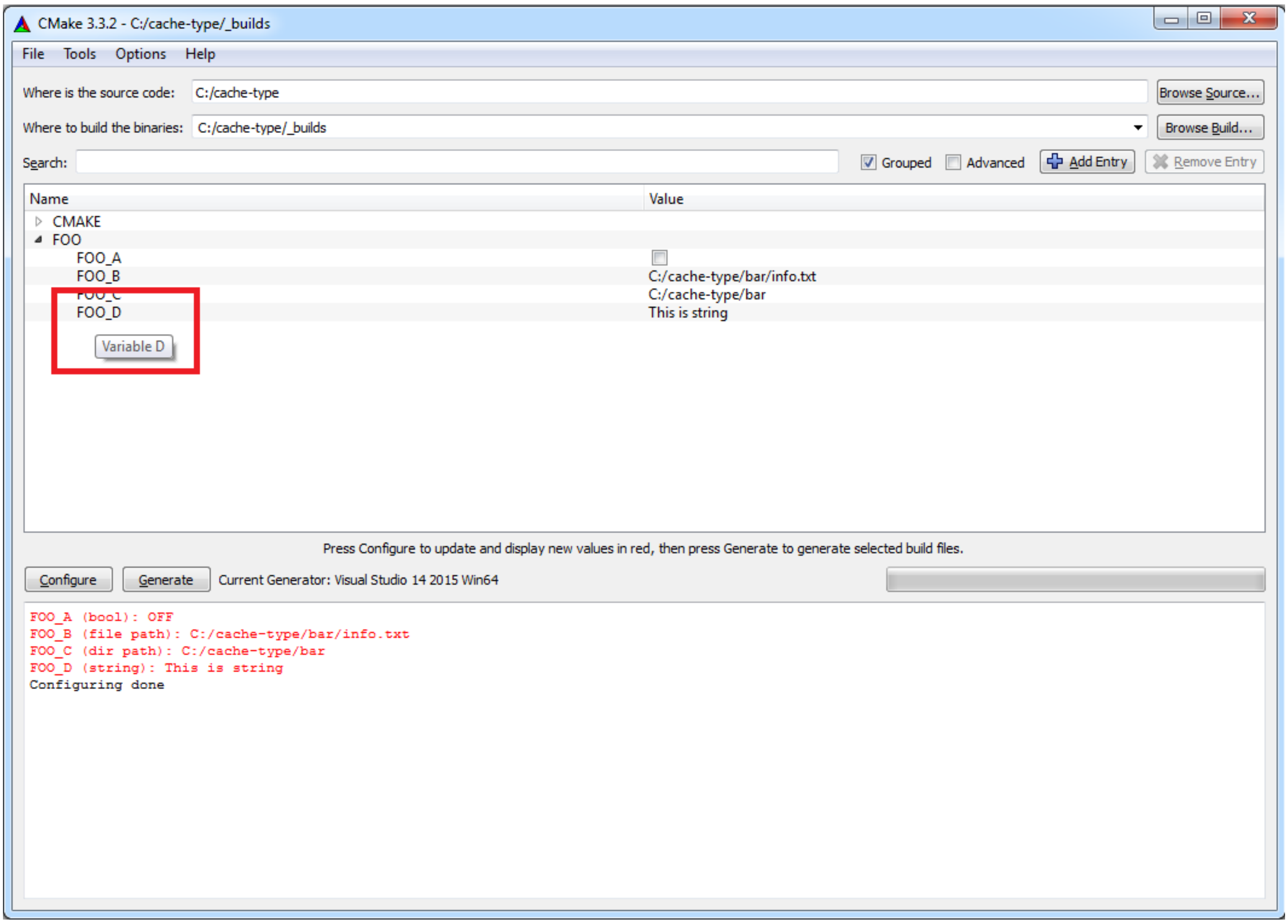
Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): C:/cache-type/bar/info.txt
FOO_C (dir path): C:/cache-type/bar
FOO_D (string): This is string
Configuring done
```

Description of variable:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(FOO_A "YES" CACHE BOOL "Variable A")
set(FOO_B "boo/info.txt" CACHE FILEPATH "Variable B")
set(FOO_C "boo/" CACHE PATH "Variable C")
set(FOO_D "abc" CACHE STRING "Variable D")

message("FOO_A (bool): ${FOO_A}")
message("FOO_B (file path): ${FOO_B}")
message("FOO_C (dir path): ${FOO_C}")
message("FOO_D (string): ${FOO_D}")
```

Will pop-up as a hint for users:



Where is the source code: C:/cache-type

Where to build the binaries: C:/cache-type/_builds

Search:

Name	Value
CMAKE	
FOO	
FOO_A	□
FOO_B	C:/cache-type/bar/info.txt
FOO_C	C:/cache-type/bar
FOO_D	This is string

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
FOO_A (bool): OFF
FOO_B (file path): C:/cache-type/bar/info.txt
FOO_C (dir path): C:/cache-type/bar
FOO_D (string): This is string
Configuring done
```

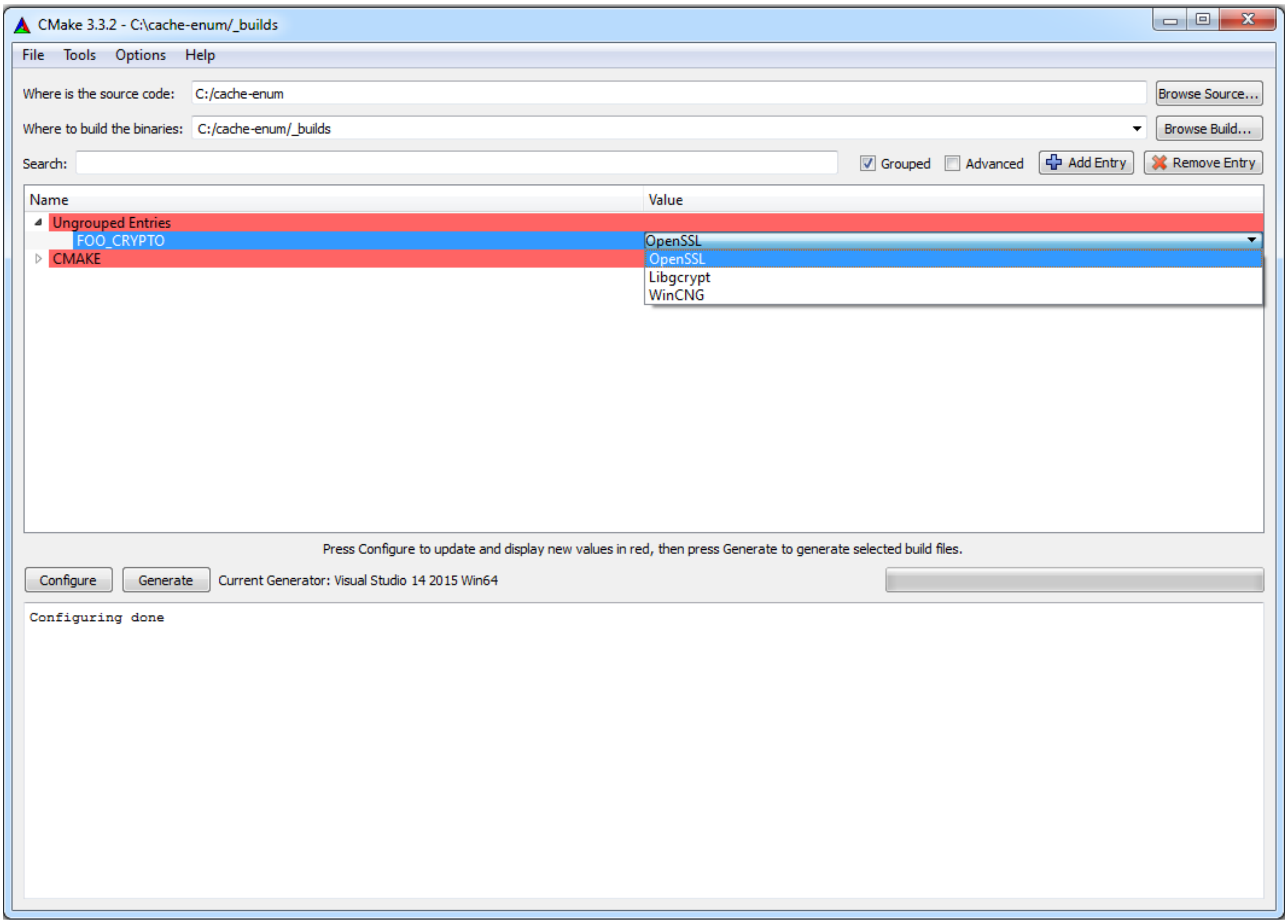
CMake documentation

- Cache entry

3.6.2.8. Enumerate

Selection widget can be created for variable of string type:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(FOO_CRYPTO "OpenSSL" CACHE STRING "Backend for cryptography")   变量名 value 类型 hint

set_property(CACHE FOO_CRYPTO PROPERTY STRINGS "OpenSSL;Libgcrypt;WinCNG")
```



Where is the source code: C:/cache-enum

Where to build the binaries: C:/cache-enum/_builds

Search:

☑ Grouped □ Advanced

Name	Value
Ungrouped Entries	
FOO_CRYPTO	OpenSSL
CMAKE	

OpenSSL
Libgcrypt
WinCNG

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
Configuring done
```

CMake documentation

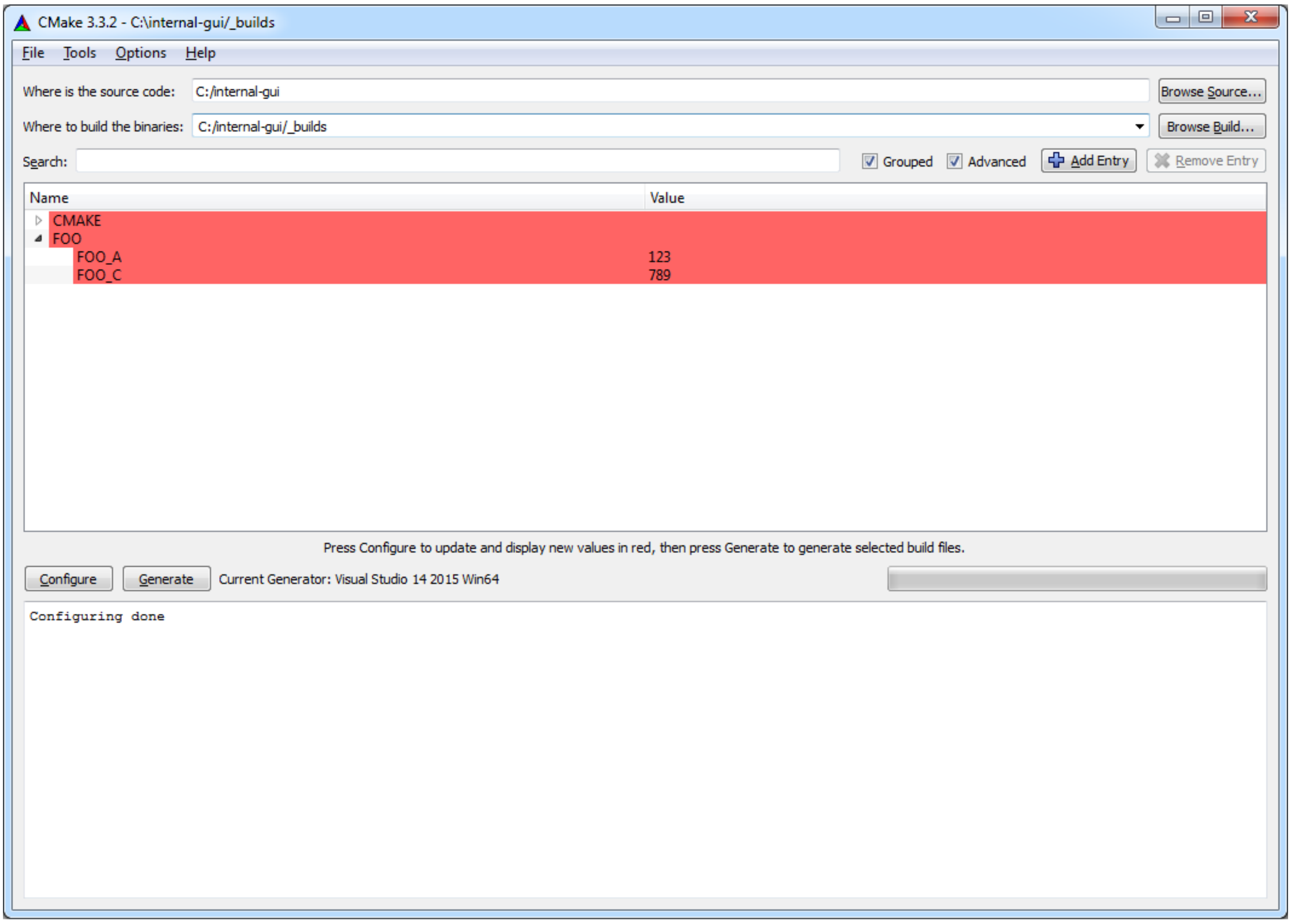
- STRINGS property

3.6.2.9. Internal

Variable with type `INTERNAL` will not be shown in CMake-GUI (again, real type is a string still):

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(FOO_A "123" CACHE STRING "")
set(FOO_B "456" CACHE INTERNAL "")
set(FOO_C "789" CACHE STRING "")
```

这个将不会在cmake-gui中显示



CMake 3.3.2 - C:\internal-gui/_builds

Where is the source code: C:/internal-gui

Where to build the binaries: C:/internal-gui/_builds

Search:

☑ Grouped ☑ Advanced

Name	Value
CMAKE	
FOO	
FOO_A	123
FOO_C	789

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
Configuring done
```

Also such type of variable implies `FORCE`:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(FOO_A "123" CACHE INTERNAL "")
set(FOO_A "456" CACHE INTERNAL "")
set(FOO_A "789" CACHE INTERNAL "")

set(FOO_B "123" CACHE STRING "")
set(FOO_B "456" CACHE STRING "")
set(FOO_B "789" CACHE STRING "")

message("FOO_A (internal): ${FOO_A}")
message("FOO_B (string): ${FOO_B}")
```

Variable `FOO_A` will be set to `123` then **rewritten** to `456` because behavior is similar to variable **with FORCE**, then one more time to `789`, so final result is `789`. Variable `FOO_B` is a cache variable with **no FORCE** so first `123` will be set to cache, then since `FOO_B` is already in cache `456` and `789` **will be ignored**, so final result is `123`:

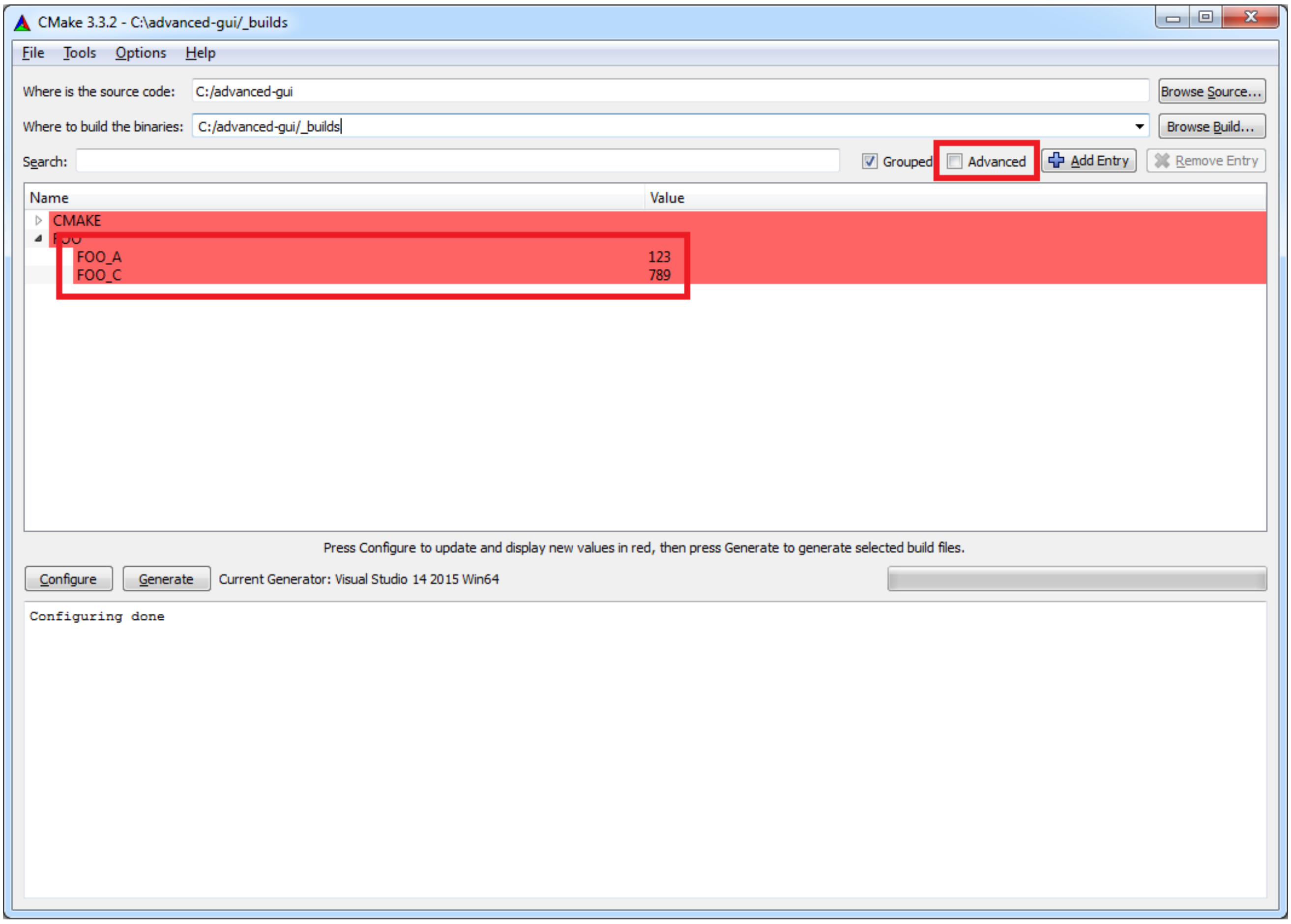
```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hinternal-force -B_builds
FOO_A (internal): 789
FOO_B (string): 123
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.2.10. Advanced

If variable is marked as advanced:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(FOO_A "123" CACHE STRING "")
set(FOO_B "456" CACHE STRING "")
set(FOO_C "789" CACHE STRING "")

mark_as_advanced(FOO_B)
```

it will not be shown in CMake-GUI if `Advanced` checkbox is not set:



Where is the source code: C:/advanced-gui

Where to build the binaries: C:/advanced-gui/_builds

Search:

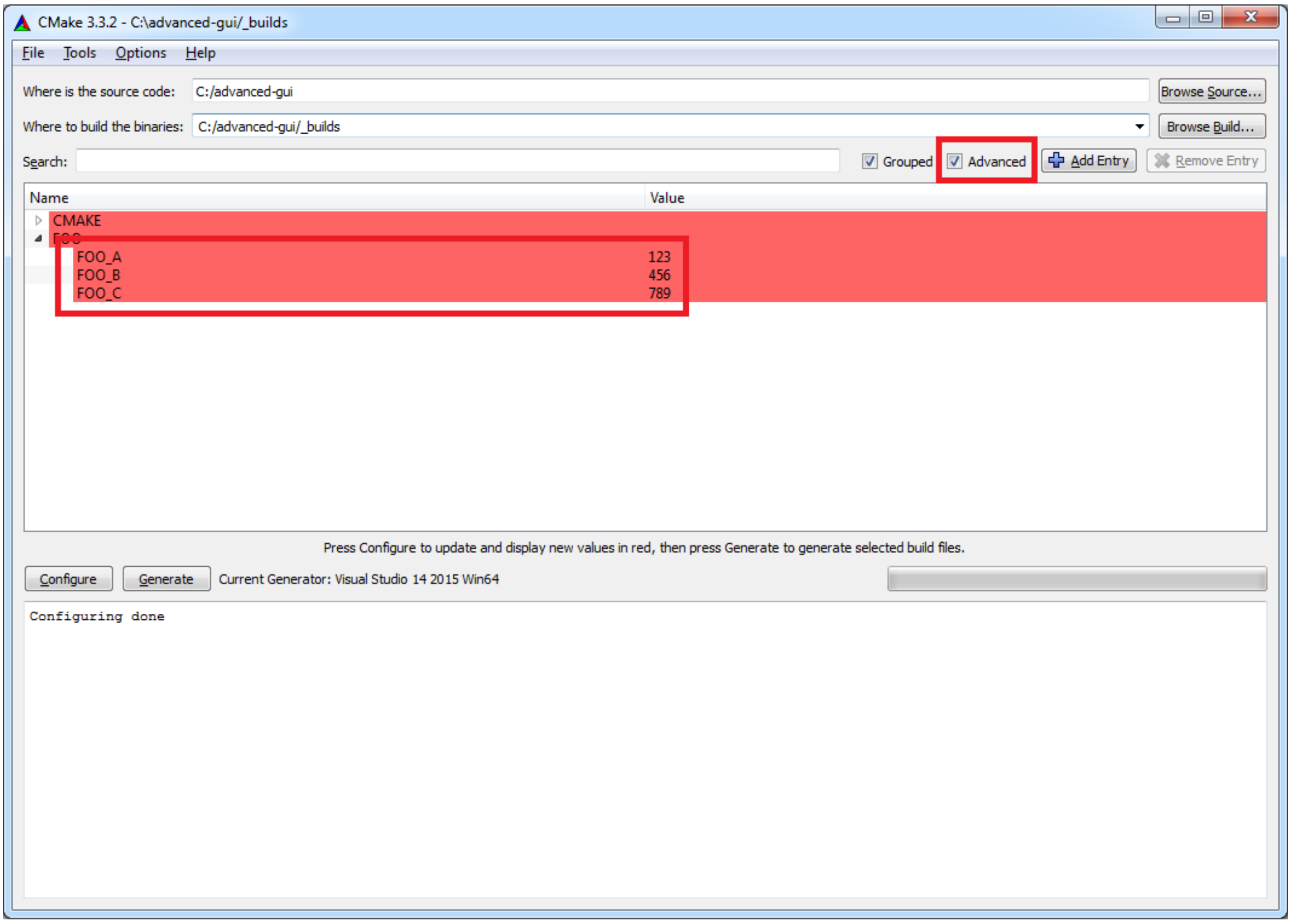
☑ Grouped □ Advanced

```
Name                    Value
▷ CMAKE
◢ FOO
    FOO_A               123
    FOO_C               789
```

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
Configuring done
```



Where is the source code: C:/advanced-gui

Where to build the binaries: C:/advanced-gui/_builds

Search:

☑ Grouped ☑ Advanced

Name	Value
CMAKE	
FOO	
FOO_A	123
FOO_B	456
FOO_C	789

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
Configuring done
```

CMake documentation

- mark_as_advanced

3.6.2.11. Use case

The ability of cache variables respect user’s settings fits perfectly for creating project’s customization option:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(FOO_A "Default value for A" CACHE STRING "")
set(FOO_B "Default value for B")

message("FOO_A: ${FOO_A}")
message("FOO_B: ${FOO_B}")
```

FOO_A是cache变量

FOO_B不是cache变量

Default value:

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hproject-customization -B_builds
FOO_A: Default value for A
FOO_B: Default value for B
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

User's value:

```
[usage-of-variables]> cmake -DFOO_A=User -Hproject-customization -B_builds
FOO_A: User
FOO_B: Default value for B
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

Note that such approach doesn't work for regular CMake variable `FOO_B`:

```
[usage-of-variables]> cmake -DFOO_B=User -Hproject-customization -B_builds
FOO_A: User
FOO_B: Default value for B
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.2.12. Option

Command `option` can be used for creating boolean cache entry:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

option(FOO_A "Option A" OFF)
option(FOO_B "Option A" ON)

message("FOO_A: ${FOO_A}")
message("FOO_B: ${FOO_B}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hoption -B_builds
FOO_A: OFF
FOO_B: ON
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> grep FOO_ _builds/CMakeCache.txt
FOO_A:BOOL=OFF
FOO_B:BOOL=ON
```

CMake documentation

- option

3.6.2.13. Unset

If you want to remove variable `X` from cache you need to use `unset(X CACHE)`. Note that `unset(X)` will remove regular variable from current scope and have no effect on cache:

区分开 cache变量 以及 regular 变量

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(X "123" CACHE STRING "X variable")   cache变量，已经写入到CMakeCache.txt文件中去了
set(X "456")
message("[0] X = ${X}")

unset(X)
message("[1] X = ${X}")   当前scope中已经不存在变量x了，所以就从cache中去读取变量

unset(X CACHE)   会将变量从cache文件中去除
message("[2] X = ${X}")

option(Y "Y option" ON)
set(Y OFF)
message("[0] Y = ${Y}")

unset(Y)
message("[1] Y = ${Y}")

unset(Y CACHE)
message("[2] Y = ${Y}")
```

When we have both cache and regular `X` variables regular variable has higher priority and will be printed:

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hunset-cache -B_builds
[0] X = 456
[1] X = 123
[2] X =
[0] Y = OFF
[1] Y = ON
[2] Y =
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

Command `unset(X)` will remove regular variable so cache variable will be printed:

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hunset-cache -B_builds
[0] X = 456
[1] X = 123
[2] X =
[0] Y = OFF
[1] Y = ON
[2] Y =
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

Command `unset(X CACHE)` will remove cache variable too. Now no variables left:

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hunset-cache -B_builds
[0] X = 456
[1] X = 123
[2] X =
[0] Y = OFF
[1] Y = ON
[2] Y =
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

Since `option` do modify cache same logic applied here:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(X "123" CACHE STRING "X variable")
set(X "456")
message("[0] X = ${X}")

unset(X)
message("[1] X = ${X}")

unset(X CACHE)
message("[2] X = ${X}")

option(Y "Y option" ON)
set(Y OFF)
message("[0] Y = ${Y}")

unset(Y)
message("[1] Y = ${Y}")

unset(Y CACHE)
message("[2] Y = ${Y}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Hunset-cache -B_builds
[0] X = 456
[1] X = 123
[2] X =
[0] Y = OFF
[1] Y = ON
[2] Y =
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.2.14. Recommendation

Because of the global nature of cache variables and options (well it's cache too) you should do prefix it with the name of the project to avoid clashing in case several projects are mixed together by `add_subdirectory`:

因为cache变量具有全局性，当存在多的项目时，通过add_subdirectory引入别的项目的时候，就应该小心地命令，否则很可能造成混乱

```
# top-level CMakeLists.txt

cmake_minimum_required(VERSION 2.8)
project(zoo)

add_subdirectory(boo)
add_subdirectory(foo)
```

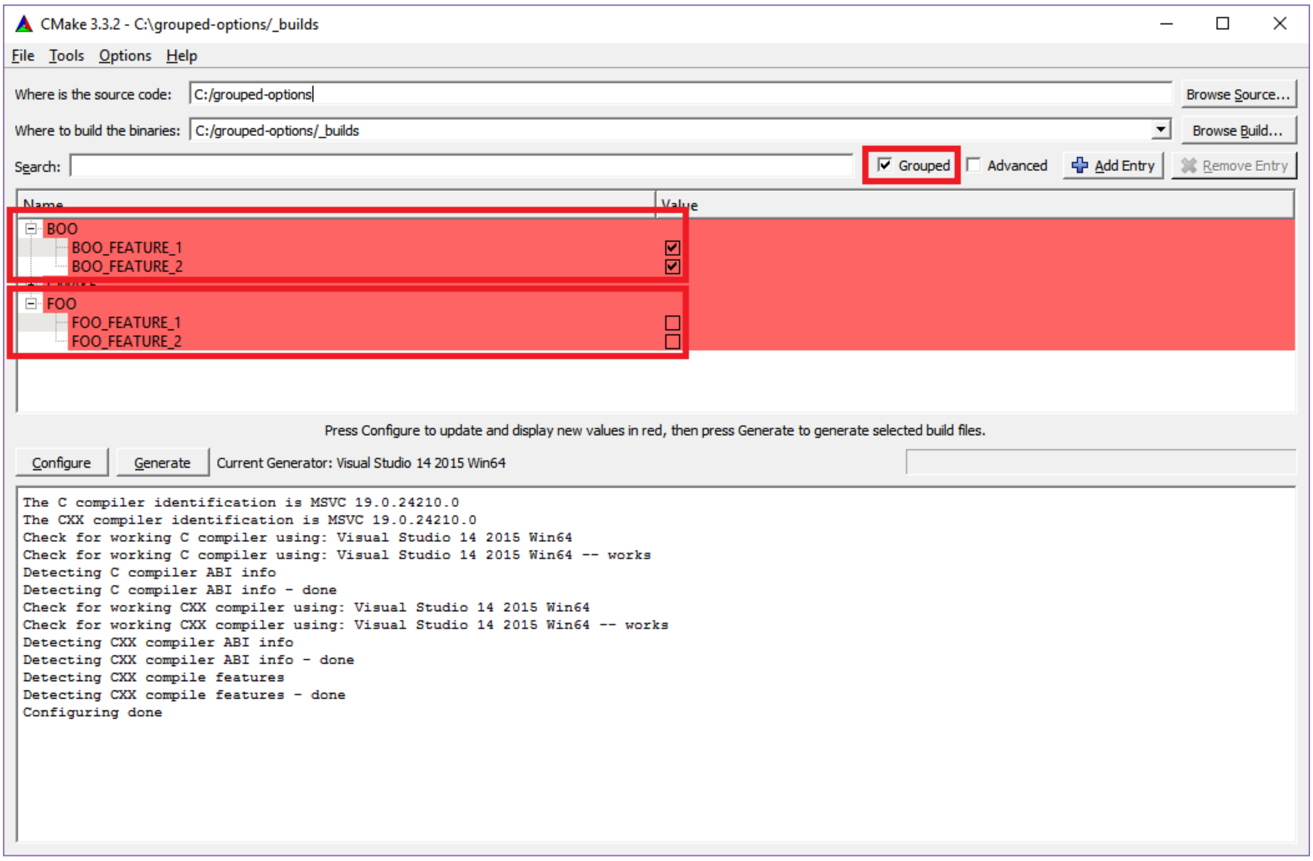
```
# foo/CMakeLists.txt

cmake_minimum_required(VERSION 2.8)
project(foo)

option(FOO_FEATURE_1 "Enable feature 1" OFF)
option(FOO_FEATURE_2 "Enable feature 2" OFF)
```

```
# boo/CMakeLists.txt

cmake_minimum_required(VERSION 2.8)
project(boo)

option(BOO_FEATURE_1 "Enable feature 1" ON)
option(BOO_FEATURE_2 "Enable feature 2" ON)
```

See also

- Module names
- Function names

Besides the fact that both features can be set independently now also CMake-GUI will group them nicely:



Where is the source code: C:/grouped-options

Where to build the binaries: C:/grouped-options/_builds

Search:

☑ Grouped □ Advanced

Name	Value
BOO	
BOO_FEATURE_1	☑
BOO_FEATURE_2	☑
FOO	
FOO_FEATURE_1	□
FOO_FEATURE_2	□

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 14 2015 Win64

```
The C compiler identification is MSVC 19.0.24210.0
The CXX compiler identification is MSVC 19.0.24210.0
Check for working C compiler using: Visual Studio 14 2015 Win64
Check for working C compiler using: Visual Studio 14 2015 Win64 -- works
Detecting C compiler ABI info
Detecting C compiler ABI info - done
Check for working CXX compiler using: Visual Studio 14 2015 Win64
Check for working CXX compiler using: Visual Studio 14 2015 Win64 -- works
Detecting CXX compiler ABI info
Detecting CXX compiler ABI info - done
Detecting CXX compile features
Detecting CXX compile features - done
Configuring done
```

3.6.2.15. Summary

所以，cmake中如何使用全局变量，那么使用cache变量就可以了

- Use cache to set **global** variables
- Cache variables fits perfectly for expressing customized options: default value and respect user’s value
- Type of cache variable helps CMake-GUI users
- Prefixes should be used to avoid clashing because of the global nature of cache variables