cmake-env-variables(7)¶

Contents

Environment Variables that Control the Build

- CMAKE_BUILD_PARALLEL_LEVEL
- CMAKE_CONFIG_TYPE
- CMAKE_MSVCIDE_RUN_PATH
- CMAKE_OSX_ARCHITECTURES
- DESTDIR
- LDFLAGS
- MACOSX_DEPLOYMENT_TARGET
- _ROOT

Environment Variables for Languages

- ASM
- ASMFLAGS
- CC
- CFLAGS
- CSFLAGS
- CUDACXX
- CUDAFLAGS
- CUDAHOSTCXX
- CXX
- CXXFLAGS
- FC
- FFLAGS
- RC
- RCFLAGS

Environment Variables for CTest

- CMAKE_CONFIG_TYPE
- CTEST_INTERACTIVE_DEBUG_MODE
- CTEST_OUTPUT_ON_FAILURE
- CTEST_PARALLEL_LEVEL
- CTEST_PROGRESS_OUTPUT
- CTEST_USE_LAUNCHERS_DEFAULT

- DASHBOARD_TEST_FROM_CTEST

Environment Variables that Control the Build

CMAKE_BUILD_PARALLEL_LEVEL

Specifies the maximum number of concurrent processes to use when building using the `cmake --build` command line Build Tool Mode.

If this variable is defined empty the native build tool's default number is used.

CMAKE_CONFIG_TYPE

The default build configuration for Build Tool Mode and `ctest` build handler when there is no explicit configuration given.

CMAKE_MSVCIDE_RUN_PATH

Extra PATH locations for custom commands when using `Visual Studio 9 2008` (or above) generators.

The `CMAKE_MSVCIDE_RUN_PATH` environment variable sets the default value for the `CMAKE_MSVCIDE_RUN_PATH` variable if not already explicitly set.

CMAKE_OSX_ARCHITECTURES

Target specific architectures for macOS.

The `CMAKE_OSX_ARCHITECTURES` environment variable sets the default value for the `CMAKE_OSX_ARCHITECTURES` variable. See `OSX_ARCHITECTURES` for more information.

DESTDIR

On UNIX one can use the `DESTDIR` mechanism in order to relocate the whole installation. `DESTDIR` means DESTination DIRectory. It is commonly used by makefile users in order to install software at non-default location. It is usually invoked like this:

```
make DESTDIR=/home/john install
```

which will install the concerned software using the installation prefix, e.g. `/usr/local` prepended with the `DESTDIR` value which finally gives `/home/john/usr/local`.

WARNING: `DESTDIR` may not be used on Windows because installation prefix usually contains a drive letter like in `C:/Program Files` which cannot be prepended with some other prefix.

LDFLAGS

Will only be used by CMake on the first configuration to determine the default linker flags, after which the value for `LDFLAGS` is stored in the cache as `CMAKE_EXE_LINKER_FLAGS_INIT`, `CMAKE_SHARED_LINKER_FLAGS_INIT`, and `CMAKE_MODULE_LINKER_FLAGS_INIT`. For any configuration run (including the first), the environment variable will be ignored if the equivalent `CMAKE_<TYPE>_LINKER_FLAGS_INIT` variable is defined.

MACOSX_DEPLOYMENT_TARGET

Specify the minimum version of macOS on which the target binaries are to be deployed.

The `MACOSX_DEPLOYMENT_TARGET` environment variable sets the default value for the `CMAKE_OSX_DEPLOYMENT_TARGET` variable.

<PackageName>_ROOT

Calls to `find_package()` will search in prefixes specified by the `<PackageName>_ROOT` environment variable, where `<PackageName>` is the name given to the `find_package` call and `_ROOT` is literal. For example, `find_package(Foo)` will search prefixes specified in the `Foo_ROOT` environment variable (if set). See policy `CMP0074`.

This variable may hold a single prefix or a list of prefixes separated by `:` on UNIX or `;` on Windows (the same as the `PATH` environment variable convention on those platforms).

See also the `_ROOT` CMake variable.

Environment Variables for Languages

ASM<DIALECT>

Preferred executable for compiling a specific dialect of assembly language files. `ASM<DIALECT>` can be `ASM`, `ASM_NASM`, `ASM_MASM` or `ASM-ATT`. Will only be used by CMake on the first configuration to determine `ASM<DIALECT>` compiler, after which the value for `ASM<DIALECT>` is stored in the cache as `CMAKE_ASM_COMPILER`. For subsequent configuration runs, the environment variable will be ignored in favor of `CMAKE_ASM_COMPILER`.

ASM<DIALECT>FLAGS

Default compilation flags to be used when compiling a specific dialect of an assembly language. `ASM<DIALECT>FLAGS` can be `ASMFLAGS`, `ASM_NASMFLAGS`, `ASM_MASMFLAGS` or `ASM-ATTFLAGS`. Will only be used by CMake on the first configuration to determine `ASM<DIALECT>` default compilation flags, after which the value for `ASM<DIALECT>FLAGS` is stored in the cache as `CMAKE_ASM_FLAGS`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_ASM_FLAGS` variable is defined.

CC

Preferred executable for compiling `C` language files. Will only be used by CMake on the first configuration to determine `C` compiler, after which the value for `CC` is stored in the cache as `CMAKE_C_COMPILER`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_C_COMPILER` variable is defined.

CFLAGS

Default compilation flags to be used when compiling `C` files. Will only be used by CMake on the first configuration to determine `CC` default compilation flags, after which the value for `CFLAGS` is stored in the cache as `CMAKE_C_FLAGS`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_C_FLAGS` variable is defined.

CSFLAGS

Preferred executable for compiling `CSharp` language files. Will only be used by CMake on the first configuration to determine `CSharp` default compilation flags, after which the value for `CSFLAGS` is stored in the cache as `CMAKE_CSharp_FLAGS`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_CSharp_FLAGS` variable is defined.

CUDACXX

Preferred executable for compiling `CUDA` language files. Will only be used by CMake on the first configuration to determine `CUDA` compiler, after which the value for `CUDA` is stored in the cache as `CMAKE_CUDA_COMPILER`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_CUDA_COMPILER` variable is defined.

CUDAFLAGS

Default compilation flags to be used when compiling `CUDA` files. Will only be used by CMake on the first configuration to determine `CUDA` default compilation flags, after which the value for `CUDAFLAGS` is stored in the cache as `CMAKE_CUDA_FLAGS`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_CUDA_FLAGS` variable is defined.

CUDAHOSTCXX

Preferred executable for compiling host code when compiling `CUDA` language files. Will only be used by CMake on the first configuration to determine `CUDA` host compiler, after which the value for `CUDAHOSTCXX` is stored in the cache as `CMAKE_CUDA_HOST_COMPILER`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_CUDA_HOST_COMPILER` variable is defined.

This environment variable is primarily meant for use with projects that enable `CUDA` as a first-class language. The `FindCUDA` module will also use it to initialize its `CUDA_HOST_COMPILER` setting.

CXX

Preferred executable for compiling `CXX` language files. Will only be used by CMake on the first configuration to determine `CXX` compiler, after which the value for `CXX` is stored in the cache as `CMAKE_CXX_COMPILER`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_CXX_COMPILER` variable is defined.

CXXFLAGS

Default compilation flags to be used when compiling `CXX` (C++) files. Will only be used by CMake on the first configuration to determine `CXX` default compilation flags, after which the value for `CXXFLAGS` is stored in the cache as `CMAKE_CXX_FLAGS`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_CXX_FLAGS` variable is defined.

FC

Preferred executable for compiling `Fortran` language files. Will only be used by CMake on the first configuration to determine `Fortran` compiler, after which the value for `Fortran` is stored in the cache as `CMAKE_Fortran_COMPILER`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_Fortran_COMPILER` variable is defined.

FFLAGS

Default compilation flags to be used when compiling `Fortran` files. Will only be used by CMake on the first configuration to determine `Fortran` default compilation flags, after which the value for `FFLAGS` is stored in the cache as `CMAKE_Fortran_FLAGS`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_Fortran_FLAGS` variable is defined.

RC

Preferred executable for compiling `resource` files. Will only be used by CMake on the first configuration to determine `resource` compiler, after which the value for `RC` is stored in the cache as `CMAKE_RC_COMPILER`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_RC_COMPILER` variable is defined.

RCFLAGS

Default compilation flags to be used when compiling `resource` files. Will only be used by CMake on the first configuration to determine `resource` default compilation flags, after which the value for `RCFLAGS` is stored in the cache as `CMAKE_RC_FLAGS`. For any configuration run (including the first), the environment variable will be ignored if the `CMAKE_RC_FLAGS` variable is defined.

Environment Variables for CTest

CMAKE_CONFIG_TYPE

The default build configuration for Build Tool Mode and `ctest` build handler when there is no explicit configuration given.

CTEST_INTERACTIVE_DEBUG_MODE¶

Environment variable that will exist and be set to `1` when a test executed by CTest is run in interactive mode.

CTEST_OUTPUT_ON_FAILURE

Boolean environment variable that controls if the output should be logged for failed tests. Set the value to 1, True, or ON to enable output on failure. See `ctest(1)` for more information on controlling output of failed tests.

CTEST_PARALLEL_LEVEL

Specify the number of tests for CTest to run in parallel. See `ctest(1)` for more information on parallel test execution.

CTEST_PROGRESS_OUTPUT

Boolean environment variable that affects how `ctest` command output reports overall progress. When set to 1, TRUE, ON or anything else that evaluates to boolean true, progress is reported by repeatedly updating the same line. This greatly reduces the overall verbosity, but is only supported when output is sent directly to a terminal. If the environment variable is not set or has a value that evaluates to false, output is reported normally with each test having its own start and end lines logged to the output.

The `--progress` option to `ctest` overrides this environment variable if both are given.

CTEST_USE_LAUNCHERS_DEFAULT

Initializes the `CTEST_USE_LAUNCHERS` variable if not already defined.

DASHBOARD_TEST_FROM_CTEST

Environment variable that will exist when a test executed by CTest is run in non-interactive mode. The value will be equal to `CMAKE_VERSION`.