NAME

cmake-env-variables − CMake Environment Variables Reference

This page lists environment variables that have special meaning to CMake.

For general information on environment variables, see the Environment Variables section in the cmake−language manual.

ENVIRONMENT VARIABLES THAT CHANGE BEHAVIOR

CMAKE_PREFIX_PATH

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

The **CMAKE_PREFIX_PATH** environment variable may be set to a list of directories specifying installation *prefixes* to be searched by the **find_package()**, **find_program()**, **find_library()**, **find_file()**, and **find_path()** commands. Each command will add appropriate subdirectories (like **bin**, **lib**, or **include**) as specified in its own documentation.

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 **CMAKE_PREFIX_PATH** CMake variable.

ENVIRONMENT VARIABLES THAT CONTROL THE BUILD

CMAKE_APPLE_SILICON_PROCESSOR

New in version 3.19.2.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

On Apple Silicon hosts running macOS, set this environment variable to tell CMake what architecture to use for **CMAKE_HOST_SYSTEM_PROCESSOR**. The value must be either **arm64** or **x86_64**.

The **CMAKE_APPLE_SILICON_PROCESSOR** normal variable, if set, overrides this environment variable.

CMAKE_BUILD_PARALLEL_LEVEL

New in version 3.12.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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_BUILD_TYPE

New in version 3.22.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

The **CMAKE_BUILD_TYPE** environment variable specifies a default value for the **CMAKE_BUILD_TYPE** variable when there is no explicit configuration given on the first run while creating a new build tree.

CMAKE_CONFIGURATION_TYPES
New in version 3.22.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

The **CMAKE_CONFIGURATION_TYPES** environment variable specifies a default value for the **CMAKE_CONFIGURATION_TYPES** variable when there is no explicit configuration given on the first run while creating a new build tree.

CMAKE_CONFIG_TYPE
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

CMAKE_EXPORT_COMPILE_COMMANDS
New in version 3.17.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

The default value for **CMAKE_EXPORT_COMPILE_COMMANDS** when there is no explicit configuration given on the first run while creating a new build tree. On later runs in an existing build tree the value persists in the cache as **CMAKE_EXPORT_COMPILE_COMMANDS**.

CMAKE_GENERATOR
New in version 3.15.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Specifies the CMake default generator to use when no generator is supplied with **-G**. If the provided value doesn't name a generator known by CMake, the internal default is used. Either way the resulting generator selection is stored in the **CMAKE_GENERATOR** variable.

Some generators may be additionally configured using the environment variables:

- **CMAKE_GENERATOR_PLATFORM**
- **CMAKE_GENERATOR_TOOLSET**
- **CMAKE_GENERATOR_INSTANCE**

CMAKE_GENERATOR_INSTANCE
New in version 3.15.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Default value for **CMAKE_GENERATOR_INSTANCE** if no Cache entry is present. This value is only applied if **CMAKE_GENERATOR** is set.

CMAKE_GENERATOR_PLATFORM
New in version 3.15.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Default value for **CMAKE_GENERATOR_PLATFORM** if no Cache entry is present and no value is specified by **cmake(1) −A** option. This value is only applied if **CMAKE_GENERATOR** is set.

CMAKE_GENERATOR_TOOLSET
New in version 3.15.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Default value for **CMAKE_GENERATOR_TOOLSET** if no Cache entry is present and no value is specified by **cmake(1) −T** option. This value is only applied if **CMAKE_GENERATOR** is set.

CMAKE_INSTALL_MODE
New in version 3.22.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

The **CMAKE_INSTALL_MODE** environment variable allows users to operate CMake in an alternate mode of **file(INSTALL)** and **install()**.

The default behavior for an installation is to copy a source file from a source directory into a destination directory. This environment variable however allows the user to override this behavior, causing CMake to create symbolic links instead.

Usage Scenarios
Installing symbolic links rather than copying files can help in the following ways:

- Conserving storage space because files do not have to be duplicated on disk.
- Changes to the source of the symbolic link are seen at the install destination without having to re−run the install step.
- Editing through the link at the install destination will modify the source of the link. This may be useful when dealing with CMake project hierarchies, i.e. using **ExternalProject** and consistent source navigation and refactoring is desired across projects.

Allowed Values
The following values are allowed for **CMAKE_INSTALL_MODE**:

COPY, empty or unset
Duplicate the file at its destination. This is the default behavior.

ABS_SYMLINK
Create an *absolute* symbolic link to the source file at the destination. Halt with an error if the link cannot be created.

ABS_SYMLINK_OR_COPY
Like **ABS_SYMLINK** but fall back to silently copying if the symlink couldn't be created.

REL_SYMLINK
Create a *relative* symbolic link to the source file at the destination. Halt with an error if the link cannot be created.

REL_SYMLINK_OR_COPY
Like **REL_SYMLINK** but fall back to silently copying if the symlink couldn't be created.

SYMLINK
Try as if through **REL_SYMLINK** and fall back to **ABS_SYMLINK** if the referenced file cannot be expressed using a relative path. Halt with an error if the link cannot be created.

SYMLINK_OR_COPY
Like **SYMLINK** but fall back to silently copying if the symlink couldn't be created.

NOTE:
A symbolic link consists of a reference file path rather than contents of its own, hence there are two ways to express the relation, either by a *relative* or an *absolute* path.

When To Set The Environment Variable
For the environment variable to take effect, it must be set during the correct build phase(s).

- If the project calls **file(INSTALL)** directly, the environment variable must be set during the configuration phase.
- In order to apply to **install()**, the environment variable must be set during installation. This could be during a build if using the **install** or **package** build targets, or separate from the build when invoking an install or running **cpack** from the command line.
- When using **ExternalProject**, it might be required during the build phase, since the external project's own configure, build and install steps will execute during the main project's build phase.

Given the above, it is recommended to set the environment variable consistently across all phases (configure, build and install).

Caveats
Use this environment variable with caution. The following highlights some points to be considered:

- **CMAKE_INSTALL_MODE** only affects files, not directories.
- Symbolic links are not available on all platforms.
- The way this environment variable interacts with the install step of **ExternalProject** is more complex. For further details, see that module's documentation.
- A symbolic link ties the destination to the source in a persistent way. Writing to either of the two affects both file system objects. This is in contrast to normal install behavior which only copies files as they were at the time the install was performed, with no enduring relationship between the source and destination of the install.
- Combining **CMAKE_INSTALL_MODE** with **IOS_INSTALL_COMBINED** is not supported.
- Changing **CMAKE_INSTALL_MODE** from what it was on a previous run can lead to unexpected results. Moving from a non−symlinking mode to a symlinking mode will discard any previous file at the destination, but the reverse is not true. Once a symlink exists at the destination, even if you switch to a non−symlink mode, the symlink will continue to exist at the destination and will not be replaced by an actual file.

CMAKE_<LANG>_COMPILER_LAUNCHER
New in version 3.17.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Default compiler launcher to use for the specified language. Will only be used by CMake to initialize the variable on the first configuration. Afterwards, it is available through the cache setting of the variable of the same name. For any configuration run (including the first), the environment variable will be ignored if the **CMAKE_<LANG>_COMPILER_LAUNCHER** variable is defined.

CMAKE_<LANG>_LINKER_LAUNCHER
New in version 3.21.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Default launcher to use when linking a target of the specified language. Will only be used by CMake to initialize the variable on the first configuration. Afterwards, it is available through the cache setting of the variable of the same name. For any configuration run (including the first), the environment variable will be ignored if the **CMAKE_<LANG>_LINKER_LAUNCHER** variable is defined.

CMAKE_MSVCIDE_RUN_PATH
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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_NO_VERBOSE
New in version 3.14.

Disables verbose output from CMake when **VERBOSE** environment variable is set.

Only your build tool of choice will still print verbose output when you start to actually build your project.

CMAKE_OSX_ARCHITECTURES
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

CMAKE_TOOLCHAIN_FILE
New in version 3.21.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

The **CMAKE_TOOLCHAIN_FILE** environment variable specifies a default value for the **CMAKE_TOOLCHAIN_FILE** variable when there is no explicit configuration given on the first run while creating a new build tree. On later runs in an existing build tree the value persists in the cache as **CMAKE_TOOLCHAIN_FILE**.

DESTDIR
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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
New in version 3.12.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Calls to **find_package(<PackageName>)** 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 **<PackageName>_ROOT** CMake variable.

VERBOSE
New in version 3.14.

Activates verbose output from CMake and your build tools of choice when you start to actually build your project.

Note that any given value is ignored. It's just checked for existence.

See also Build Tool Mode and **CMAKE_NO_VERBOSE** environment variable

ENVIRONMENT VARIABLES FOR LANGUAGES

ASM<DIALECT>
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Preferred executable for compiling a specific dialect of assembly language files. **ASM<DIALECT>** can be **ASM**, **ASM_NASM** (Netwide Assembler), **ASM_MASM** (Microsoft Assembler) or **ASM−ATT** (Assembler AT&T). 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<DIALECT>_COMPILER**. For subsequent configuration runs, the environment variable will be ignored in favor of **CMAKE_ASM<DIALECT>_COMPILER**.

NOTE:
Options that are required to make the compiler work correctly can be included; they can not be changed.

```
$ export ASM="custom-compiler --arg1 --arg2"
```

ASM<DIALECT>FLAGS
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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<DIALECT>_FLAGS <CMAKE_<LANG>_FLAGS>**. For any configuration run (including the first), the environment variable will be ignored, if the **CMAKE_ASM<DIALECT>_FLAGS <CMAKE_<LANG>_FLAGS>** variable is defined.

See also **CMAKE_ASM<DIALECT>_FLAGS_INIT**.

CC
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

NOTE:
Options that are required to make the compiler work correctly can be included; they can not be changed.

```
$ export CC="custom−compiler −−arg1 −−arg2"
```

CFLAGS
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

See also **CMAKE_C_FLAGS_INIT**.

CSFLAGS
New in version 3.9.2.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Default compilation flags to be used when compiling **CSharp** 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.

See also **CMAKE_CSharp_FLAGS_INIT**.

CUDAARCHS
New in version 3.20.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Value used to initialize **CMAKE_CUDA_ARCHITECTURES** on the first configuration if it's not already

defined. Subsequent runs will use the value stored in the cache.

This is a semicolon−separated list of architectures as described in **CUDA_ARCHITECTURES**.

CUDACXX
New in version 3.8.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

NOTE:
Options that are required to make the compiler work correctly can be included; they can not be changed.

```
$ export CUDACXX="custom−compiler −−arg1 −−arg2"
```

CUDAFLAGS
New in version 3.8.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

See also **CMAKE_CUDA_FLAGS_INIT**.

CUDAHOSTCXX
New in version 3.8.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

NOTE:
Ignored when using Visual Studio Generators.

New in version 3.13: The **FindCUDA** module will use this variable to initialize its **CUDA_HOST_COMPILER** setting.

CXX
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

NOTE:
Options that are required to make the compiler work correctly can be included; they can not be changed.

```
$ export CXX="custom-compiler --arg1 --arg2"
```

CXXFLAGS
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

See also **CMAKE_CXX_FLAGS_INIT**.

FC
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

NOTE:
Options that are required to make the compiler work correctly can be included; they can not be changed.

```
$ export FC="custom-compiler --arg1 --arg2"
```

FFLAGS
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

See also **CMAKE_Fortran_FLAGS_INIT**.

HIPCXX
New in version 3.21.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

be ignored if the **CMAKE_HIP_COMPILER** variable is defined.

HIPFLAGS
New in version 3.21.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

See also **CMAKE_HIP_FLAGS_INIT**.

ISPC
New in version 3.19.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

ISPCFLAGS
New in version 3.19.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

See also **CMAKE_ISPC_FLAGS_INIT**.

OBJC
New in version 3.16.7.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

If **OBJC** is not defined, the **CC** environment variable will be checked instead.

OBJCXX
New in version 3.16.7.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

If **OBJCXX** is not defined, the **CXX** environment variable will be checked instead.

RC

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

NOTE:

Options that are required to make the compiler work correctly can be included; they can not be changed.

```
$ export RC="custom-compiler --arg1 --arg2"
```

RCFLAGS

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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.

See also **CMAKE_RC_FLAGS_INIT**.

SWIFTC

New in version 3.15.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

NOTE:

Options that are required to make the compiler work correctly can be included; they can not be changed.

```
$ export SWIFTC="custom-compiler --arg1 --arg2"
```

ENVIRONMENT VARIABLES FOR CTEST

CTEST_INTERACTIVE_DEBUG_MODE

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Environment variable that will exist and be set to **1** when a test executed by **ctest(1)** is run in interactive mode.

CTEST_OUTPUT_ON_FAILURE
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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 f ailed tests.

CTEST_PARALLEL_LEVEL
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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

CTEST_PROGRESS_OUTPUT
New in version 3.13.

This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

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
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Initializes the **CTEST_USE_LAUNCHERS** variable if not already defined.

DASHBOARD_TEST_FROM_CTEST
This is a CMake Environment Variable. Its initial value is taken from the calling process environment.

Environment variable that will exist when a test executed by **ctest(1)** is run in non−interactive mode. The value will be equal to **CMAKE_VERSION**.

ENVIRONMENT VARIABLES FOR THE CMAKE CURSES INTERFACE

CCMAKE_COLORS
New in version 3.18.

Determines what colors are used by the CMake curses interface, when run on a terminal that supports colors. The syntax follows the same conventions as **LS_COLORS**; that is, a list of key/value pairs separated by **:**.

Keys are a single letter corresponding to a CMake cache variable type:

- **s**: A **STRING**.
- **p**: A **FILEPATH**.
- **c**: A value which has an associated list of choices.
- **y**: A **BOOL** which has a true−like value (e.g. **ON**, **YES**).
- **n**: A **BOOL** which has a false−like value (e.g. **OFF**, **NO**).

Values are an integer number that specifies what color to use. **0** is black (you probably don't w ant to use

that). Others are determined by your terminal's color support. Most (color) terminals will support at least 8 or 16 colors. Some will support up to 256 colors. The colors will likely match *this chart*, although the first 16 colors may match the original *CGA color palette*. (Many modern terminal emulators also allow their color palette, at least for the first 16 colors, to be configured by the user.)

Note that fairly minimal checking is done for bad colors (although a value higher than what curses believes your terminal supports will be silently ignored) or bad syntax.

For example:

```
CCMAKE_COLORS='s=39:p=220:c=207:n=196:y=46'
```

COPYRIGHT