**C/C++ tip: How to detect the operating system type using compiler predefined macros**

January 3, 2012

Topics: [C/C++](http://nadeausoftware.com/articles/c_c)

**Table of Contents**

[How to list predefined macros](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#Howtolistpredefinedmacros)

[How to detect the operating system type](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#Howtodetecttheoperatingsystemtype)

[AIX](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#AIX)

[BSD](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#BSD)

[HP-UX](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#HPUX)

[Linux](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#Linux)

[OSX, iOS, and Darwin](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#OSXiOSandDarwin)

[Solaris](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#Solaris)

[Windows with Cygwin (POSIX)](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#WindowswithCygwinPOSIX)

[Windows, Cygwin (non-POSIX), and MinGW](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#WindowsCygwinnonPOSIXandMinGW)

[How to detect POSIX and UNIX](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#HowtodetectPOSIXandUNIX)

[POSIX](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#POSIX)

[UNIX](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#UNIX)

[Other ways to detect the operating system type](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#Otherwaystodetecttheoperatingsystemtype)

[Further reading](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#Furtherreading)

[Related articles at NadeauSoftware.com](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#RelatedarticlesatNadeauSoftwarecom)

[Web articles](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#Webarticles)

**How to list predefined macros**

See [How to list compiler predefined macros](http://nadeausoftware.com/articles/2011/12/c_c_tip_how_list_compiler_predefined_macros) for instructions on getting a list of macros for the compilers referenced here.

**How to detect the operating system type**

Throughout the following sections note:

* Red text indicates deprecated macros that don't start with an underscore. C++ compilers, and C compilers in standards compliance mode, do not define them.
* **Green text** indicates recommended macros that are well-supported and useful for detecting a specific OS.

**AIX**

|  |  |
| --- | --- |
| **Developer:** | IBM |
| **Distributions:** | [AIX](http://www-03.ibm.com/systems/power/software/aix/index.html) |
| **Processors:** | POWER |
|  |  |

#if defined(\_AIX)

/\* IBM AIX. ------------------------------------------------- \*/

#endif

|  |  |  |
| --- | --- | --- |
| ***AIX*** | | |
| **Macro** | [**GNU GCC/G++**](http://gcc.gnu.org/) | [**IBM XL C/C++**](http://www-01.ibm.com/software/awdtools/xlcpp/) |
| **\_AIX** | yes | yes |
| **\_\_unix** |  | yes |
| **\_\_unix\_\_** | yes | yes |

Notes:

* See IBM's notes on [Using the GNU C/C++ compiler on AIX.](http://www.ibm.com/developerworks/aix/library/au-gnu.html)

**BSD**

|  |  |
| --- | --- |
| **Developer:** | Open source |
| **Distributions:** | [DragonFly BSD,](http://www.dragonflybsd.org/) [FreeBSD,](http://www.freebsd.org/) [OpenBSD,](http://www.openbsd.org/) [NetBSD](http://www.netbsd.org/) |
| **Processors:** | x86, x86-64, Itanium, POWER, SPARC, etc. |
|  |  |

#if defined(\_\_unix\_\_) || (defined(\_\_APPLE\_\_) && defined(\_\_MACH\_\_))

#include <sys/param.h>

#if defined(BSD)

/\* BSD (DragonFly BSD, FreeBSD, OpenBSD, NetBSD). ----------- \*/

#endif

#endif

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***BSD*** | | | | | | | |
| **Macro** | [**Clang/LLVM**](http://llvm.org/) | | | | [**GNU GCC/G++**](http://gcc.gnu.org/) | | |
| ***DragonFly BSD*** | ***FreeBSD*** | ***NetBSD*** | ***OpenBSD*** | ***FreeBSD*** | ***NetBSD*** | ***OpenBSD*** |
| **unix** | yes | yes |  | yes | yes |  |  |
| **\_\_unix** | yes | yes |  | yes | yes |  |  |
| **\_\_unix\_\_** | yes | yes | yes | yes | yes | yes | yes |
| **\_\_DragonFly\_\_** | yes |  |  |  |  |  |  |
| **\_\_FreeBSD\_\_** |  | yes |  |  | yes |  |  |
| **\_\_NetBSD\_\_** |  |  | yes |  |  | yes |  |
| **\_\_OpenBSD\_\_** |  |  |  | yes |  |  | yes |

Notes:

* Compilers for the old BSD base for these distributions defined the **\_\_bsdi\_\_** macro, but none of these distributions define it now. This leaves no generic "BSD" macro defined by the compiler itself, but all UNIX-style OSes provide a **<sys/param.h>** file. On BSD distributions, and only on BSD distributions, this file defines a **BSD** macro that's set to the OS version. Checking for this generic macro is more robust than looking for known BSD distributions with **\_\_DragonFly\_\_**, **\_\_FreeBSD\_\_**, **\_\_NetBSD\_\_**, and **\_\_OpenBSD\_\_** macros.
* Apple's OSX for the Mac and iOS for iPhones and iPads are based in part on a fork of FreeBSD distributed as [Darwin.](http://darwinsource.sourceforge.net/) As such, OSX and iOS also define the **BSD** macro within **<sys/param.h>**. However, compilers for OSX, iOS, and Darwin do not define **\_\_unix\_\_**. To detect all BSD OSes, including OSX, iOS, and Darwin, use an **#if/#endif** that checks for **\_\_unix\_\_** along with **\_\_APPLE\_\_** and **\_\_MACH\_\_** (see the later section on OSX and iOS).

**HP-UX**

|  |  |
| --- | --- |
| **Developer:** | Hewlett-Packard |
| **Distributions:** | [HP-UX](http://h71028.www7.hp.com/enterprise/w1/en/os/hpux11i-overview.html) |
| **Processors:** | Itanium |

#if defined(\_\_hpux)

/\* Hewlett-Packard HP-UX. ----------------------------------- \*/

#endif

|  |  |  |
| --- | --- | --- |
| ***HP-UX*** | | |
| **Macro** | [**GNU GCC/G++**](http://gcc.gnu.org/) | [**HP C/aC++**](http://h21007.www2.hp.com/portal/site/dspp/menuitem.863c3e4cbcdc3f3515b49c108973a801/?ciid=6b080f1bace021100f1bace02110275d6e10RCRD) |
| **hpux** | yes |  |
| **\_\_hpux** | yes | yes |
| **unix** | yes |  |
| **\_\_unix** | yes | yes |
| **\_\_unix\_\_** | yes |  |

**Linux**

|  |  |
| --- | --- |
| **Developer:** | Open source |
| **Distributions:** | [Centos,](http://www.centos.org/) [Debian,](http://www.debian.org/) [Fedora,](http://fedoraproject.org/) [OpenSUSE,](http://www.opensuse.org/) [RedHat,](http://www.redhat.com/) [Ubuntu](http://www.ubuntu.com/) |
| **Processors:** | x86, x86-64, POWER, etc. |

#if defined(\_\_linux\_\_)

/\* Linux. --------------------------------------------------- \*/

#endif

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Linux*** | | | | | | |
| **Macro** | [**Clang/LLVM**](http://llvm.org/) | [**GNU GCC/G++**](http://gcc.gnu.org/) | [**Intel ICC/ICPC**](http://software.intel.com/en-us/articles/intel-compilers/) | [**Oracle Solaris Studio**](http://www.oracle.com/technetwork/server-storage/solarisstudio/overview/index.html) | [**Portland  PGCC/PGCPP**](http://www.pgroup.com/products/pgiworkstation.htm) | [**IBM XL C/C++**](http://www-01.ibm.com/software/awdtools/xlcpp/) |
| **linux** | yes | yes | yes | yes | yes |  |
| **\_\_linux** | yes | yes | yes | yes | yes | yes |
| **\_\_linux\_\_** | yes | yes | yes | yes | yes | yes |
| **\_\_gnu\_linux** | yes | yes | yes | yes |  |  |
| **unix** | yes | yes | yes | yes | yes |  |
| **\_\_unix** | yes | yes | yes | yes | yes | yes |
| **\_\_unix\_\_** | yes | yes | yes | yes | yes | yes |

Notes:

* Linux is available for a wide variety of processors, but it is primarily used on x86 and x86-64 processors. The above table's compilers are all for these processors. Some of them may not be available for other processors.
* There are no predefined compiler macros indicating the specific Linux distribution. At run-time you can read [**/proc/version**](http://www.kernel.org/doc/man-pages/online/pages/man5/proc.5.html) to get the distribution name and version, or invoke **[uname -a](http://www.kernel.org/doc/man-pages/online/pages/man2/uname.2.html)** from a **Makefile** then set your own macro. However, writing Linux distribution-specific code is rarely necessary due to high compatibility between distributions.
* Linux is POSIX compliant and defines the standard **\_POSIX\*** macros in **<unistd.h>**. While Linux is compliant with the latest POSIX.1-2008 specification, Linux distributions erroneously set **\_POSIX\_VERSION** to **200809L**, instead of leaving it set to **200112L**, as required by the POSIX specification. In practice, this is not a big issue since the existence, not value, of the **\_POSIX\_VERSION** macro is sufficient to detect POSIX compliance. After that, the individual **\_POSIX\_\*** feature macros provide better information about which specific POSIX features are implemented (see later in this article for POSIX discussion).

**OSX, iOS, and Darwin**

|  |  |
| --- | --- |
| **Developer:** | Apple and open source |
| **Distributions:** | [OSX,](http://www.apple.com/osx/) [iOS,](http://www.apple.com/ios/) [Darwin](http://darwinsource.sourceforge.net/) |
| **Processors:** | x86, x86-64, ARM |

#if defined(\_\_APPLE\_\_) && defined(\_\_MACH\_\_)

/\* Apple OSX and iOS (Darwin). ------------------------------ \*/

#include <TargetConditionals.h>

#if TARGET\_IPHONE\_SIMULATOR == 1

/\* iOS in Xcode simulator \*/

#elif TARGET\_OS\_IPHONE == 1

/\* iOS on iPhone, iPad, etc. \*/

#elif TARGET\_OS\_MAC == 1

/\* OSX \*/

#endif

#endif

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***OSX and Darwin*** | | | | |
| **Macro** | [**Clang/LLVM**](http://llvm.org/) | [**GNU GCC/G++**](http://gcc.gnu.org/) | [**Intel ICC/ICPC**](http://software.intel.com/en-us/articles/intel-compilers/) | [**Portland  PGCC/PGCPP**](http://www.pgroup.com/products/pgiworkstation.htm) |
| **\_\_APPLE\_\_** | yes | yes | yes | yes |
| **\_\_MACH\_\_** | yes | yes | yes | yes |

|  |  |  |
| --- | --- | --- |
| ***iOS*** | | |
| **Macro** | [**Clang/LLVM**](http://llvm.org/) | [**GNU GCC/G++**](http://gcc.gnu.org/) |
| **\_\_APPLE\_\_** | yes | yes |
| **\_\_MACH\_\_** | yes | yes |

Notes:

* For Apple's OSes, all compilers define **\_\_APPLE\_\_** and **\_\_MACH\_\_** macros. The **\_\_MACH\_\_** macro indicates the [MACH kernel](http://en.wikipedia.org/wiki/Mach_kernel) at the heart of OSX/iOS and partially derived from the obsolete [NeXTSTEP.](http://en.wikipedia.org/wiki/Nextstep) For rigor, *both* of these macros must be defined to detect OSX/iOS. If only **\_\_MACH\_\_**is defined, the OS is NeXTSTEP or one of the other OSes derived from the MACH kernel.
* All OSX compilers are available from the command-line. Apple's [Xcode](http://developer.apple.com/xcode/) IDE can be configured to invoke any of them from the GUI.
* Mac OSX and iOS include BSD UNIX components originally from [FreeBSD.](http://www.freebsd.org/) The open source parts of Mac OSX and iOS are distributed as [Darwin,](http://en.wikipedia.org/wiki/Darwin_(operating_system)) without Apple's proprietary user interface and tools. Despite being UNIX-like, Mac OSX and iOS compilers *do not define* the conventional **\_\_unix\_\_**, **\_\_unix**, or **unix**macros. They do define the **BSD** macro in **<sys/param.h>** (see later [discussion about BSD](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#BSD)).
* Some on-line lists of compiler macros (like [this one](http://beefchunk.com/documentation/lang/c/pre-defined-c/preos.html)) list **\_\_MACOSX\_\_**. Some forum comments (like [these](http://stackoverflow.com/questions/2166483/which-macro-to-wrap-mac-os-x-specific-code-in-c-c)) claim **\_\_OSX\_\_** exists. *These are incorrect.* There are no such macros predefined by OSX compilers, but they may be defined by specific project Makefiles and platform-detector scripts like [GNU **autoconf**.](http://www.gnu.org/software/autoconf/)
* Some lists (like [this one](http://sourceforge.net/apps/mediawiki/predef/index.php?title=Operating_Systems)) still include **macintosh** or **Macintosh** macros. These were only available on the obsolete Mac OS 9 discontinued back in 2002.
* Some forum advice and books (like [this one](http://books.google.com/books?id=2b_4jswA6UMC&lpg=PA72&ots=rIWbpOzMvb&pg=PA72#v=onepage&q&f=false)) claim that the **\_\_APPLE\_\_** macro is only defined by Apple's own compilers. *This is incorrect.* While it's true that compilers distributed by Apple define this macro, so do OSX distributions of [Intel's ICC,](http://software.intel.com/en-us/articles/intel-compilers/) the old [IBM XL for PowerPC,](http://publib.boulder.ibm.com/infocenter/macxhelp/v6v81/index.jsp?topic=/com.ibm.vacpp6m.doc/compiler/ref/rumacros.htm) and the latest direct downloads of [Clang](http://llvm.org/) and [GCC](http://gcc.gnu.org/) from open source web sites.
* OSX and iOS compilers do not define macros to distinguish between OSX and iOS. However, Apple's **<TargetConditionals.h>** in each platform's SDK provides **TARGET\_\*** macros that indicate the OS. All of the macros exist for all platforms, but their values change between **0** and **1**flags as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| ***TargetConditionals.h*** | | | |
|  | **Mac OSX** | **iOS** | **iOS Simulator** |
| **TARGET\_OS\_EMBEDDED** | **0** | **1** | **0** |
| **TARGET\_OS\_IPHONE** | **0** | **1** | **1** |
| **TARGET\_OS\_MAC** | **1** | **1** | **1** |
| **TARGET\_IPHONE\_SIMULATOR** | **0** | **0** | **1** |

* Note that the above macros are *not mutually exclusive*: **TARGET\_OS\_MAC** is set to **1** for *all* platforms, and **TARGET\_OS\_IPHONE** is **1** for iOS *and* the simulator. To detect OSX vs. iOS vs. the iOS simulator you have to check the macro values in a specific order (see below).
* There are no macros to distinguish at compile time between an iPhone, iPad, or other Apple devices using the same OSes.

**Solaris**

|  |  |
| --- | --- |
| **Developer:** | Oracle and open source |
| **Distributions:** | [Oracle Solaris,](http://en.wikipedia.org/wiki/Solaris_%28operating_system%29) [Open Indiana](http://openindiana.org/) |
| **Processors:** | x86, x86-64, SPARC |
|  |  |

#if defined(\_\_sun) && defined(\_\_SVR4)

/\* Solaris. ------------------------------------------------- \*/

#endif

|  |  |  |  |
| --- | --- | --- | --- |
| ***Solaris*** | | | |
| **Macro** | [**Clang/LLVM**](http://llvm.org/) | [**GNU GCC/G++**](http://gcc.gnu.org/) | [**Oracle Solaris Studio**](http://www.oracle.com/technetwork/server-storage/solarisstudio/overview/index.html) |
| **sun** | yes | yes | yes |
| **\_\_sun** | yes | yes | yes |
| **\_\_sun\_\_** | yes | yes |  |
| **\_\_SunOS** |  |  | yes |
| **\_\_svr4\_\_** | yes | yes |  |
| **\_\_SVR4** | yes | yes | yes |
| **unix** | yes | yes | yes |
| **\_\_unix** | yes | yes | yes |
| **\_\_unix\_\_** | yes | yes |  |

Notes:

* To conform with long-standing convention, Solaris compilers define **\_\_sun** and **sun**, despite Oracle's acquisition of Sun Microsystems in 2010. Clang and GCC also define **\_\_sun\_\_**, but Solaris Studio does not.
* Oracle compilers do not define an "oracle" macro or a "solaris" macro.
* Checking for **\_\_sun** is not sufficient to identify Solaris. Compilers for the obsolete BSD-based SunOS also defined **\_\_sun** (and Solaris Studio still defines **\_\_SunOS**, even on System V-based Solaris). To identify Solaris specifically, the **\_\_sun** *and* **\_\_SVR4** macros must be defined. Also note that you need to check for upper-case **\_\_SVR4** instead of the lower-case **\_\_svr4** that's only defined by GCC and not by Solaris Studio.

**Windows with Cygwin (POSIX)**

|  |  |
| --- | --- |
| **Developer:** | Open source |
| **Distributions:** | [Cygwin](http://www.cygwin.com/) |
| **Processors:** | x86 |
|  |  |

#if defined(\_\_CYGWIN\_\_) && !defined(\_WIN32)

/\* Cygwin POSIX under Microsoft Windows. -------------------- \*/

#endif

|  |  |  |
| --- | --- | --- |
| ***Cygwin building for POSIX*** | | |
| **Macro** | [**Clang/LLVM**](http://llvm.org/) | [**GNU GCC/G++**](http://gcc.gnu.org/) |
| **\_\_CYGWIN\_\_** | yes | yes |
| **\_\_CYGWIN32\_\_** | yes | yes |
| **unix** | yes | yes |
| **\_\_unix** | yes | yes |
| **\_\_unix\_\_** | yes | yes |

Notes:

* [Cygwin](http://www.cygwin.com/) provides a POSIX development environment for Windows, including shells, command-line tools, and compilers. Using Cygwin's libraries, POSIX applications can be built and run under Windows without any Windows-specific code.
* Cygwin POSIX libraries are 32-bit-only, so 64-bit POSIX applications cannot be built. Some code found on-line references **\_\_CYGWIN64\_\_**. However, there is no 64-bit Cygwin, so this macro is never defined. It exists only in forum discussions about a possible future 64-bit Cygwin.
* Clang/LLVM and GCC both can build POSIX or Windows applications. The table above shows macros when building POSIX applications. See the Windows section later in this article for macros when building Windows applications. Comparing the macro set for both types of applications note that **\_\_CYGWIN\_\_** and the standard **\_\_unix\_\_** macros are *always* defined by GCC, even when building a Windows application. For this reason, detecting POSIX builds under Cygwin must use an **#if/#endif** that checks that **\_\_CYGWIN\_\_** is defined, but **\_WIN32** is not.
* Checking for Cygwin POSIX builds probably isn't necessarily at all. The whole point of Cygwin is to run standard POSIX applications under Windows, so checking for Cygwin explicitly shouldn't be needed.
* Intel's compilers are not supported under Cygwin, but users have hacked running them from a Cygwin bash command line. However, the compilers still build Windows applications, not POSIX applications.
* Portland Group's compilers for Windows come with a Cygwin install that enables the compilers to be run from a bash command-line, but they still build Windows applications, not POSIX applications.

**Windows, Cygwin (non-POSIX), and MinGW**

|  |  |
| --- | --- |
| **Developer:** | Microsoft |
| **Distributions:** | [Windows XP, Vista, 7, 8](http://windows.microsoft.com/) |
| **Processors:** | x86, x86-64 |

#if defined(\_WIN64)

/\* Microsoft Windows (64-bit). ------------------------------ \*/

#elif defined(\_WIN32)

/\* Microsoft Windows (32-bit). ------------------------------ \*/

#endif

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Windows*** | | | | | | | | | | | | | |
| **Macro** | [**Clang/LLVM *(Windows target)***](http://llvm.org/) | | [**Clang/LLVM *(MinGW target)***](http://llvm.org/) | | [**GNU GCC/G++ *(Windows target)***](http://gcc.gnu.org/) | [**GNU GCC/G++ *(MinGW target)***](http://gcc.gnu.org/) | | [**Intel ICC/ICPC**](http://software.intel.com/en-us/articles/intel-compilers/) | | [**Portland  PGCC/PGCPP**](http://www.pgroup.com/products/pgiworkstation.htm) | | [**Microsoft  Visual Studio**](http://www.microsoft.com/visualstudio/en-us) | |
| ***32-bit*** | ***64-bit*** | ***32-bit*** | ***64-bit*** | ***32-bit*** | ***32-bit*** | ***64-bit*** | ***32-bit*** | ***64-bit*** | ***32-bit*** | ***64-bit*** | ***32-bit*** | ***64-bit*** |
| **\_\_CYGWIN\_\_** |  |  |  |  | yes |  |  |  |  |  |  |  |  |
| **\_\_CYGWIN32\_\_** |  |  |  |  | yes |  |  |  |  |  |  |  |  |
| **\_\_MINGW32\_\_** |  |  | yes | yes |  | yes | yes |  |  |  |  |  |  |
| **\_\_MINGW64\_\_** |  |  |  | yes |  |  | yes |  |  |  |  |  |  |
| **unix** |  |  |  |  | yes |  |  |  |  |  |  |  |  |
| **\_\_unix** |  |  |  |  | yes |  |  |  |  |  |  |  |  |
| **\_\_unix\_\_** |  |  |  |  | yes |  |  |  |  |  |  |  |  |
| **WIN32** |  |  | yes |  | yes | yes | yes |  |  |  |  |  |  |
| **\_WIN32** | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| **\_\_WIN32** |  |  | yes |  | yes | yes | yes |  |  | yes | yes |  |  |
| **\_\_WIN32\_\_** |  |  | yes |  | yes | yes | yes |  |  | yes | yes |  |  |
| **WIN64** |  |  |  | yes |  |  | yes |  |  |  |  |  |  |
| **\_WIN64** |  | yes |  | yes |  |  | yes |  | yes |  | yes |  | yes |
| **\_\_WIN64** |  |  |  | yes |  |  | yes |  |  |  | yes |  |  |
| **\_\_WIN64\_\_** |  |  |  | yes |  |  | yes |  |  |  | yes |  |  |
| **WINNT** |  |  | yes |  | yes | yes | yes |  |  |  |  |  |  |
| **\_\_WINNT** |  |  | yes |  |  | yes | yes |  |  |  |  |  |  |
| **\_\_WINNT\_\_** |  |  | yes |  |  | yes | yes |  |  |  |  |  |  |

Notes:

* Clang/LLVM and GCC under Windows run within the [Cygwin](http://www.cygwin.com/) POSIX environment or the [MinGW](http://mingw.org/) minimal GNU environment. Both provide a bash shell and assorted command-line utilities. Cygwin also provides POSIX libraries while MinGW does not.
* Based on command-line options, the Clang/LLVM and GCC compilers can build Windows applications or POSIX applications that run under Windows using POSIX compatibility libraries. Predefined macros for POSIX applications are described in the previous section of this article. The table above is strictly for compiling Windows applications.
* Clang/LLVM can build Windows applications using the Windows target (e.g. "**-ccc-host-triple i386-pc-win32**") or the MinGW target (e.g. "**-ccc-host-triple i386-pc-mingw32**"). The "**-m32**" option builds 32-bit applications and "**-m64**" builds 64-bit.
* GCC under Cygwin can build Windows applications using the "**-mwin32**" command-line option. While GCC is capable of building 64-bit applications, Cygwin is 32-bit only and the version of GCC included with it only builds 32-bit applications.
* Oddly enough, GCC under Cygwin predefines UNIX macros even when building Windows applications.
* Some on-line code references **\_\_CYGWIN64\_\_**. Since there is no 64-bit Cygwin, this macro is never defined. It exists only in forum discussions about a possible future 64-bit Cygwin.
* GCC under MinGW can build Windows applications using the "**-mwin32**" command-line option. The "**-m32**" and "**-m64**" options build 32-bit and 64-bit applications.
* While Clang/LLVM, GCC, and Portland Group compilers define a lot of WIN32 and WIN64 macros with various numbers of underscores, the only macros that matter are those that are compatible with Microsoft's Visual Studio: **\_WIN32** and **\_WIN64**.
* Some on-line advice recommends checking for **\_MSC\_VER**. The macro is defined with the compiler version number for Clang/LLVM, ICC, and Visual Studio, but it isn't defined by GCC or Portland Group compilers.
* Some lists of predefined macros (like [this one](http://sourceforge.net/apps/mediawiki/predef/index.php?title=Operating_Systems)) include additional macros for discontinued products, such as **\_\_TOS\_WIN\_\_** for IBM's XL compiler on Windows (XL is still available for AIX and Linux), and **\_\_WINDOWS\_\_** for the discontinued but open sourced [Watcom compiler.](http://en.wikipedia.org/wiki/Watcom_C/C++_compiler)

**How to detect POSIX and UNIX**

POSIX and UNIX are not operating systems. Rather they are formal or de facto standards followed to some degree by all UNIX-style OSes.

**POSIX**

|  |  |
| --- | --- |
| **Developer:** | Standard |
| **Distributions:** | All current UNIX-style OSes, including BSD, Linux, OSX, and Solaris |
| **Processors:** | x86, x86-64, ARM, POWER, SPARC, etc. |

#if !defined(\_WIN32) && (defined(\_\_unix\_\_) || defined(\_\_unix) || (defined(\_\_APPLE\_\_) && defined(\_\_MACH\_\_)))

/\* UNIX-style OS. ------------------------------------------- \*/

#include <unistd.h>

#if defined(\_POSIX\_VERSION)

/\* POSIX compliant \*/

#endif

#endif

All UNIX-style OSes (see also [UNIX](http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system#UNIX) below) have **<unistd.h>** that defines macros indicating the level of [POSIX](http://en.wikipedia.org/wiki/POSIX) compliance. The **\_POSIX\_VERSION** macro value indicates the version of the standard with which the OS is compliant. Known values are:

* **198808L** for POSIX.1-1988
* **199009L** for POSIX.1-1990
* **199506L** for ISO POSIX.1-1996
* **200112L** for ISO POSIX.1-2001
* **200809L** for ISO POSIX.1-2008

Another way to detect ISO POSIX.1-2008 compliance is with a run-time check using **sysconf**.

if ( sysconf( \_SC\_VERSION ) >= 200809L )

{

/\* POSIX.1-2008 \*/

}

else

{

/\* Pre-POSIX.1-2008 \*/

}

While the **#if/#endif** and **sysconf** call above will both work to detect broad POSIX compliance, it's more useful to check for individual POSIX features flagged by macros in **<unistd.h>**. The POSIX specification has a long list of these macros, but a few of the more useful ones include:

* **\_POSIX\_IPV6** indicates IPv6 address support.
* **\_POSIX\_MAPPED\_FILES** indicates memory mapping support.
* **\_POSIX\_SEMAPHORES** indicates semaphore support for multi-threading.
* **\_POSIX\_THREADS** indicates pthreads support. A number of **\_POSIX\_THREAD**\* macros then indicate whether thread resource usage can be reported or thread scheduling priority controlled.

**UNIX**

|  |  |
| --- | --- |
| **Developer:** | De facto standard |
| **Distributions:** | All current UNIX-style OSes, including BSD, Linux, OSX, and Solaris |
| **Processors:** | x86, x86-64, ARM, POWER, SPARC, etc. |

#if !defined(\_WIN32) && (defined(\_\_unix\_\_) || defined(\_\_unix) || (defined(\_\_APPLE\_\_) && defined(\_\_MACH\_\_)))

/\* UNIX-style OS. ------------------------------------------- \*/

#endif

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***UNIX (Clang/LLVM compilers)*** | | | | | | | | | |
| **Macro** | **Cygwin (POSIX)** | **DragonFly BSD** | **FreeBSD** | **iOS** | **Linux** | **NetBSD** | **OpenBSD** | **OSX** | **Solaris** |
| **unix** | yes | yes | yes |  | yes |  | yes |  | yes |
| **\_\_unix** | yes | yes | yes |  | yes |  | yes |  | yes |
| **\_\_unix\_\_** | yes | yes | yes |  | yes | yes | yes |  | yes |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***UNIX (GCC compilers)*** | | | | | | | | | | |
| **Macro** | **AIX** | **Cygwin (POSIX)** | **FreeBSD** | **iOS** | **HP-UX** | **Linux** | **NetBSD** | **OpenBSD** | **OSX** | **Solaris** |
| **unix** |  | yes | yes |  | yes | yes |  |  |  | yes |
| **\_\_unix** |  | yes | yes |  | yes | yes |  |  |  | yes |
| **\_\_unix\_\_** | yes | yes | yes |  | yes | yes | yes | yes |  | yes |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***UNIX (Other compilers)*** | | | | | | | | | |
|  | **AIX** | **HP-UX** | **Linux** | | | | **OSX** | | **Solaris** |
| **Macro** | [**IBM XL C/C++**](http://www-01.ibm.com/software/awdtools/xlcpp/) | [**HP C/aC++**](http://h21007.www2.hp.com/portal/site/dspp/menuitem.863c3e4cbcdc3f3515b49c108973a801/?ciid=6b080f1bace021100f1bace02110275d6e10RCRD) | [**Intel ICC/ICPC**](http://software.intel.com/en-us/articles/intel-compilers/) | [**Oracle Solaris Studio**](http://www.oracle.com/technetwork/server-storage/solarisstudio/overview/index.html) | [**Portland  PGCC/PGCPP**](http://www.pgroup.com/products/pgiworkstation.htm) | [**IBM XL C/C++**](http://www-01.ibm.com/software/awdtools/xlcpp/) | [**Intel ICC/ICPC**](http://software.intel.com/en-us/articles/intel-compilers/) | [**Portland  PGCC/PGCPP**](http://www.pgroup.com/products/pgiworkstation.htm) | [**Oracle Solaris Studio**](http://www.oracle.com/technetwork/server-storage/solarisstudio/overview/index.html) |
| **unix** |  |  | yes | yes | yes |  |  |  | yes |
| **\_\_unix** | yes | yes | yes | yes | yes | yes |  |  | yes |
| **\_\_unix\_\_** | yes |  | yes | yes | yes | yes |  |  |  |

Notes:

* There is no single UNIX macro defined by all compilers on all UNIX-style OSes. An **#if/#endif** that checks multiple macros is required.
* Compilers for Apple's OSX and iOS don't define *any* UNIX macros. An **#if/#endif** that checks **\_\_APPLE\_\_** and **\_\_MACH\_\_** is required (see the earlier section OSX and iOS).
* GCC under Cygwin defines UNIX macros *even when building Windows applications*. An **#if/#endif** that excludes **\_WIN32** is required to detect UNIX builds on Cygwin (see the earlier section on Windows).

**Other ways to detect the operating system type**

On UNIX-style OSes a common way to detect OS features is to use [GNU's **autoconf**.](http://www.gnu.org/software/autoconf/) This tool builds configuration shell scripts that automatically check for OS and compiler features, build Makefiles, and set compiler flags. For code that only targets UNIX-style OSes, this works well. But**autoconf** doesn't work for code that must compile on non-UNIX-style OSes (e.g. Windows) or within an IDE. And it's way overkill for many projects where a simple **#if/#endif** set will do.

From the command line there are several ways to detect the OS. On UNIX-style OSes, the **[uname](http://en.wikipedia.org/wiki/Uname)** command reports the OS name. On Windows, the **[ver](http://en.wikipedia.org/wiki/Ver_%28command%29)** and **[winver](http://en.wikipedia.org/wiki/Ver_(command))** commands report the OS name. On Linux, the [**/proc/version**](http://en.wikipedia.org/wiki/Procfs) virtual file reports the Linux kernel version. But using any of these to automatically configure code requires scripts and Makefiles. And those have the same problems as **autoconf**.

The most elegant solution is to eschew all detection scripts and simply use the above predefined macros *already available on every OS* and designed specifically for use in **#if/#endif** sets for OS-specific code. Don't reinvent the wheel.

**Further reading**

**Related articles at NadeauSoftware.com**

* [C/C++ tip: How to list compiler predefined macros](http://nadeausoftware.com/articles/2011/12/c_c_tip_how_list_compiler_predefined_macros) explains how to get a compiler's macros by using command-line options and other methods.
* [C/C++ tip: How to detect the compiler name and version using compiler predefined macros](http://nadeausoftware.com/articles/2012/10/c_c_tip_how_detect_compiler_name_and_version_using_compiler_predefined_macros) provides **#if/#endif** sets for detecting common compilers.
* [C/C++ tip: How to detect the processor type using compiler predefined macros](http://nadeausoftware.com/articles/2012/02/c_c_tip_how_detect_processor_type_using_compiler_predefined_macros) provides **#if/#endif** sets for detecting desktop and server processors using compiler macros.

**Web articles**

* [Operating Systems](http://sourceforge.net/apps/mediawiki/predef/index.php?title=Operating_Systems) at Sourceforge.net provides a list of OSes and their compiler predefined macros. However the list is cluttered with obsolete OSes (OS/2? Palm OS?), doesn't note macros shared across many OSes (like **\_\_unix\_\_**), and doesn't include discussion.
* [Pre-defined C/C++ Compiler Macros](http://beefchunk.com/documentation/lang/c/pre-defined-c/preos.html) at beefchunk.com has a list of OSes and predefined macros. Like the Sourceforge list, the information is a bit cluttered with obsolete OSes (DG/UX? Unicos?) and lacks discussion.
* [Operating System Resources](http://wiki.apache.org/stdcxx/OperatingSystems) at Apache.org has a good list of OSes, links to vendor documentation, and some tables of compiler predefined macros.

**Article source**

<http://nadeausoftware.com/articles/2012/01/c_c_tip_how_use_compiler_predefined_macros_detect_operating_system>