

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY**

**Individual Work № : 7**

**Topic:** **C program to get the device information**

**Subject: OS2**

**Teacher: Mamed Shahmaliyev**

**Group: 1459i Student: Yusifov Rəvan**

**Date: Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**­ Baku 2022**

To check the operating system of the host in a C or C++ code, we need to check the macros defined by the compiler (GNU GCC or G++). For example, on Windows platform, the compiler has a special macro named \_WIN32 defined. So, if the macro \_WIN32 is defined, we are on Windows. Similarly, other operating systems have specific macro defined by the compiler.

Example of checking Windows OS:

*#ifdef \_WIN32*

*printf("You have Windows Operating System");*

*#endif*

Following is the list of macros defined based on the Operating System:

| OPERATING SYSTEM | MACRO PRESENT | NOTES |
| --- | --- | --- |
| Windows 32 bit + 64 bit | \_WIN32 | for all Windows OS |
| Windows 64 bit | \_WIN64 | Only for 64 bit Windows |
| Apple | \_\_APPLE\_\_ | for all Apple OS |
| Apple | \_\_MACH\_\_ | alternative to above |
| iOS embedded | TARGET\_OS\_EMBEDDED | include TargetConditionals.h |
| iOS stimulator | TARGET\_IPHONE\_SIMULATOR | include TargetConditionals.h |
| iPhone | TARGET\_OS\_IPHONE | include TargetConditionals.h |
| MacOS | TARGET\_OS\_MAC | include TargetConditionals.h |
| Android | \_\_ANDROID\_\_ | subset of linux |
| Unix based OS | \_\_unix\_\_ |  |
| Linux | \_\_linux\_\_ | subset of unix |
| POSIX based | \_POSIX\_VERSION | Windows with Cygwin |
| Solaris | \_\_sun |  |
| HP UX | \_\_hpux |  |
| BSD | BSD | all BSD flavors |
| DragonFly BSD | \_\_DragonFly\_\_ |  |
| FreeBSD | \_\_FreeBSD\_\_ |  |
| NetBSD | \_\_NetBSD\_\_ |  |
| OpenBSD | \_\_OpenBSD\_\_ |  |

Note that the macros are valid for GNU GCC and G++ and may vary for other compilers.

We will go through some basic examples and explore the use of this features in real life as well.

## Simple example

In the following example, we are focused on detecting the flavor of Windows we are running which can be a 64 bit or 32 bit. For Windows, our table will be:

| OPERATING SYSTEM | MACRO PRESENT |
| --- | --- |
| Windows OS 32 bit + 64 bit | \_WIN32 |
| Windows OS 64 bit | \_WIN64 |

As \_WIN32 is present in both 32 bit and 64 bit Windows Operating System, we need to first check the presence of \_WIN32 to confirm that it is a Windows Operating System and then, check the presence of \_WIN64 to confirm whether it is a 64 bit Windows Operating System or a 32 bit Windows Operating System.

Following is the code to check your Windows OS:

*#include <stdio.h>*

*int main()*

*{*

*#ifdef \_WIN32 // Includes both 32 bit and 64 bit*

*#ifdef \_WIN64*

*printf("Windows 64 bit\n");*

*#else*

*printf("Windows 32 bit\n");*

*#endif*

*#else*

*printf("Not a Windows OS\n");*

*#endif*

*return 0;*

*}*

Output:

***Windows 32 bit***

## Example of Apple OS

In this example, we use the macros of Apple OS to detect which Apple OS like MacOS or iPhone is being used.

*#include <stdio.h>*

*int main()*

*{*

*#if \_\_APPLE\_\_*

*#include "TargetConditionals.h"*

*#if TARGET\_OS\_IPHONE && TARGET\_IPHONE\_SIMULATOR*

*printf("iPhone stimulator\n");*

*#elif TARGET\_OS\_IPHONE*

*printf("iPhone\n");*

*#elif TARGET\_OS\_MAC*

*printf("MacOS\n");*

*#else*

*printf("Other Apple OS\n");*

*#endif*

*#else*

*printf("Not an Apple OS\n");*

*#endif*

*return 0;*

*}*

Output:

***MacOS***

## General example

In this example, we are detecting if we have a Windows OS or a Linux based OS.

*#include <stdio.h>*

*int main() {*

*#ifdef \_WIN32*

*printf("Windows\n");*

*#elif \_\_linux\_\_*

*printf("Linux\n");*

*#elif \_\_unix\_\_*

*printf("Other unix OS\n");*

*#else*

*printf("Unidentified OS\n");*

*#endif*

*return 0;*

*}*

## Usage

With the power to detect the operating system in the language (C and C++ in our case\_, we can write a single cross platform code which works on all platforms by separating platform dependant code.

Example:

*#include <stdio.h>*

*int main()*

*{*

*#if \_\_APPLE\_\_*

*// apple specific code*

*#elif \_WIN32*

*// windows specific code*

*#elif \_\_LINUX\_\_*

*// linux specific code*

*#elif BSD*

*// BSD specific code*

*#else*

*// general code or warning*

*#endif*

*// general code*

*return 0;*

*}*

At the same time, we can write code optimized for specific platforms. For example, a function call may be supported on all platforms but we can optimize it greatly for a specific platform say Linux but this new code will raise errors in other platforms. In this case, we can use the macros to detect if it is Linux and for this case, we can easily use other alternative optimized code.

Example:

*#include <stdio.h>*

*int main()*

*{*

*#if \_\_linux\_\_*

*// linux optimized code (will fail in other platforms)*

*#else*

*// general code for all platforms*

*#endif*

*// general code*

*return 0;*

*}*