Why can't g++ find iostream.h?

I'm trying to understand how to compile C++ programs from the command line using g++ and (eventually) Clang on Ubuntu.

I found a webpage which explains MakeFiles and I am following their directions. http://mrbook.org/tutorials/make/

I downloaded the four example files into their own directory.

- main.cpp
- hello.cpp
- factorial.cpp
- functions.h

I then went ahead and ran their example of how to manually compile without a MakeFile.

```
g++ main.cpp hello.cpp factorial.cpp -o hello
```

When I ran the command from above, I received the following error from g++:

```
main.cpp:1:22: fatal error: iostream.h: No such file or directory
compilation terminated.
hello.cpp:1:22: fatal error: iostream.h: No such file or directory
compilation terminated.
```

My only experience with writing c++ is using an IDE such as VS C++ Express or CodeBlocks. Isn't the compiler supposed to know what iostream.h is and where to find it?

How do I get rid of this error so the program willl compile?

Thanks for any help.

c++ g++

asked Oct 27 '12 at 18:49

quakkels
3,692 14 50 115

4 There is no `iostream.h` , it's just `iostream` . – chris Oct 27 '12 at 18:50

1 Really? So when the tutorial's files say `#include <iostream.h>` it should say `#include <iostream>` ? – quakkels Oct 27 '12 at 18:51

Well, the tutorial's probably old enough that it was valid when it was written. – chris Oct 27 '12 at 18:52

1 That tutorial links to the Make documentation for a version (3.79.1) which was released June 23rd, 2000. You might consider finding a newer tutorial. – meagar ♦ Oct 27 '12 at 18:56

1 As a tutorial for learning make, it looks fine to me. It's only the C++ that's the problem, and you should be learning that from a separate source anyway. – Benjamin Lindley Oct 27 '12 at 19:01

3 Answers

Before the C++ language was standardized by the ISO, the header file was named `<iostream.h>` , but when the C++98 standard was released, it was renamed to just `<iostream>` (without the `.h`). Change the code to use `#include <iostream>` instead and it should compile.

You'll also need to add a `using namespace std;` statement to each source file (or prefix each

reference to an iostream function/object with a `std::` specifier), since namespaces did not exist in the pre-standardized C++. C++98 put the standard library functions and objects inside the `std` namespace.

edited Oct 27 '12 at 19:23

answered Oct 27 '12 at 18:51

Adam Rosenfield
211k 51 326 456

cool. got it working. – quakkels Oct 27 '12 at 18:57

4 -1 the history lesson is complete fantasy – Cheers and hth. - Alf Oct 27 '12 at 19:02

In C++<98 edition, it was at one time named `<iostream.h>`, but that was before even namespaces were introduced. – Xeo Oct 27 '12 at 19:04

3 C++98 didn't have `<iostream.h>`. The `.h` forms of C++ header names came from the pre-standard days of cfront, Glockenspiel C++, Comeau, Zortech, and Borland. The C++98 standard removed the suffix and put all the names in namespace `std`. Some implementations still provide the `.h` headers for backward compatibility. – Pete Becker Oct 27 '12 at 19:07

3 I don't think this was worth being downvoted on. Sure, you made a mistake but everyone does. The rest of your answer is correct, just because your history is off doesn't mean I should downvote you. It's easy to fix. – Rapptz Oct 27 '12 at 19:10

`<iostream.h>` has never been a standard C++ header, because it did not make it into the C++ standard.

Instead we got `<iostream>`, in 1998.

Steer well clear of teaching material using non-standard stuff such as `<iostream.h>` or `void main`.

However, as a practical solution for your current pre-standard code, you may try to replace

```
#include <iostream.h>
```

with

```
#include <iostream>
using namespace std;
```

It's not guaranteed to work, but chances are that it will work.

answered Oct 27 '12 at 19:06

Cheers and hth. - Alf
79.8k 6 72 155

3 The better solution (imho) is to not have `using namespace std;` and instead add `std::` in front of the standard names. – Xeo Oct 27 '12 at 19:10

Another related issue that wasn't mentioned here, so I will include it for anyone's future reference, is from the command line the compiler needs the environment path variable updated to find the location of the c++ header files. In windows you can just update the path environment using the 'advanced system properties' GUI and add the location of the c++ include files. This will update the PATH environment variable in Windows cmd & Cygwin automatically upon restarting the shell.

To update your PATH from Linux or the Cygwin shell type... PATH=$PATH:/your_path_here Example:PATH=$PATH:/cygdrive/c/cygwin/lib/gcc/i686-pc-mingw32/4.7.3/include/c++ Also a good idea to add just the include directory as well: PATH=$PATH:/cygdrive/c/cygwin/lib/gcc/i686-pc-mingw32/4.7.3/include/ ...or check the proper directories for the location of your installation's include files, I recommend installing mingw for use with Cygwin, which is envoked with g++.

To install additional needed packages in Cygwin re-run the Cygwin install utility & check install from Internet to add packages from web repositories and add mingw-gcc-g++ & mingw-binutils. To

compile: g++ hello.cpp -o hello

If using the gcc utility instead compile with the command: gcc hello.cpp -o hello -lstdc++ ... to get your executable.

As long as you have either gcc or mingw installed and the path to the c++ include files is in your path environment variable, the commands will work.

answered Nov 6 '13 at 6:41

nodlehs

21 2