What is the difference between cout, cerr, clog of iostream header in c++? When to use which one?



We have 2 open jobs ♥

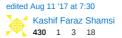
Delivering EnergyLink – the leading smart energy management platform.

Learn more

I tried researching the difference between <code>cout</code> , <code>cerr</code> and <code>clog</code> on the internet but couldn't find a perfect answer. I still am not clear on when to use which. Can anyone explain to me, through simple programs and illustrate a perfect situation on when to use which one?

I visited this site which shows a small program on cerr and clog , but the output obtained over there can also be obtained using cout . So, I'm confused over each one's exact use.

c++ iostream cout clog



asked May 27 '13 at 12:04

Arlene Batada

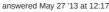
457 1 6 9

4 Each one has a computer-recognized stream, stdout, stdin (for cin), and stderr that it uses by default. I believe clog is just cerr with a buffering change. — chris May 27 '13 at 12:08

7 Answers

stdout and stderr are different streams, even though they both refer to console output by default. Redirecting (piping) one of them (e.g. program.exe >out.txt) would not affect the other.

Generally, stdout should be used for actual program output, while all information and error messages should be printed to stderr, so that if the user redirects output to a file, information messages are still printed on the screen and not to the output file.





2,514 10 23

You don't even talk about clog – void pointer Aug 11 '17 at 15:27



We have 2 open jobs 🔻

Delivering EnergyLink – the leading smart energy management platform.

Learn more

Generally you use std::cout for normal output, std::cerr for errors, and std::clog for "logging" (which can mean whatever you want it to mean).

The major difference is that std::cerr is not buffered like the other two.

In relation to the old C stdout and stderr, std::cout corresponds to stdout, while std::cerr and std::clog both corresponds to stderr (except that std::clog is buffered).

answered May 27 '13 at 12:09



Some programmer dude **255k** 22 199 334

4 I think this is the best answer. – Ren Nov 30 '15 at 11:18

I've read that clog also outputs to cerr. So based on that, which one do you choose? If clog is normally for "logging", why would I want that to go to the error stream? Logs seem more like "normal logs" (a.k.a. cout) than errors. – void.pointer Aug 11 '17 at 15:27

@void.pointer As I said in my answer, both cerr and clog uses the standard "error" output, but clog is buffered which might be why it seems more like cout . Which one to pick for error output? Depends I guess, on more reasons than I can list and it has to be decided from case to case. – Some programmer dude Aug 11 '17 at 15:48

cerr does not require a buffer, so it is faster than the other ones and does not use the memory

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up

OR SIGN IN WITH

G Google

Facebook



- · Use cerr to show errors.
- Use clog for logging.

answered Jun 7 '15 at 15:41

David Vargas Carrillo

99 1 5

Standard output stream (cout): cout is the instance of the ostream class. cout is used to produce output on the standard output device which is usually the display screen. The data needed to be displayed on the screen is inserted in the standard output stream (cout) using the insertion operator (<<).

Un-buffered standard error stream (cerr): cerr is the standard error stream which is used to output the errors. This is also an instance of the <code>ostream</code> class. As <code>cerr</code> is **un-buffered** so it is used when we need to display the error message immediately. It does not have any buffer to store the error message and display later.

Buffered standard error stream (clog): This is also an instance of ostream class and used to display errors but unlike cerr the error is first inserted into a **buffer** and is stored in the buffer until it is not fully filled.

further reading: basic-input-output-c



cout is usually used to display some statements on user screen. excout<<"Arlene Batada":

output:

Arlene Batada



Both **cout** and **clog** are buffered but **cerr** is un-buffered and all of these are predefined objects which are instances of class ostream. The basic use of these three are **cout** is used for standard input whereas **clog** and **cerr** is used for showing errors. The main point why **cerr** is un-buffered is may be because suppose you have several outputs in the buffer and an error exception is mentioned in the code then you need to display that error immediately which can be done by **cerr** effectively.

Please correct me if I am wrong.



The difference of these 3 streams is buffering.

- 1. With cerr, the output flushs
 - immediately (because cerr does not use buffer).
- 2. With clog, the output flushs
 - after you finish your current function.
 - explicitly call the function flush.
- 3. With cout, the output flushs
 - after you have call to any output streams (cout, cerr, clog).
 - after you finish your current function.
 - explicitly call the function flush.

Please check the following code, and run DEBUG through 3 lines: f(std::clog), f(std::cerr), f(std::out), then open 3 output files to see what happened. You can swap these 3 lines to see what will happen.

#include <iostream>
#include <fstream>
#include <string>

Join Stack Overflow to learn, share knowledge, and build your career.

Email Sign Up OR SIGN IN WITH

G Google

Facebook

```
std::cin.seekg(0, std::cin.beg); // seek to begin
     std::string line;
    while(std::getline(std::cin, line)) //input from the file in.txt
    os << line < "\n"; //output to the file out.txt</pre>
void test()
std::cin.rdbuf(in.rdbuf()); //redirect std::cin to in.txt!
std::cout.rdbuf(out.rdbuf()); //redirect std::cout to out.txt!
     std::cerr.rdbuf(err.rdbuf());
     std::clog.rdbuf(log.rdbuf());
    f(std::clog);
f(std::cerr);
    f(std::cout);
    std::cin.rdbuf(cinbuf);
     std::cout.rdbuf(coutbuf);
    std::cerr.rdbuf(cerrbuf);
std::clog.rdbuf(clogbuf);
int main()
     test();
     std::cout << "123";
```

answered Nov 1 '17 at 4:47

