

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour ×

How does does ifstream eof() work?

```
#include <iist>
#include <map>
#include <set>
#include <deque>
#include <stack>
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <cctype>
#include <fstream>
```

```
using namespace std;
```

```
int main() {
```

```

    fstream inf( "ex.txt", ios::in );
    while( !inf.eof() ) {
        std::cout << inf.get() << "\n";
    }
    inf.close();
    inf.clear();
    inf.open( "ex.txt", ios::in );
    char c;
    while( inf >> c ) {
        std::cout << c << "\n";
    }
    return 0;
}

```

I'm really confused about `eof()` function. Suppose that my `ex.txt`'s content was:

abc

It always reads an extra character and shows `-1` when reading using `eof()`. But the `inf >> c` gave the correct output which was 'abc'? Can anyone help me explain this?

c++

edited May 23 '13 at 8:47



Soon
7,258 2 14 31

asked Dec 26 '10 at 7:21



Chan
2,949 6 43 95

- Why are you using `std::` if you're importing that namespace. And more important why are you importing all those libraries if you don't use them. — [razpeitia](#) Dec 26 '10 at 7:33

Thanks for pointing out that. I was competing on TopCoder, and I usually import all the namespaces in order to save time. — [Chan](#) Dec 26 '10 at 7:57

[add comment](#)

4 Answers

`-1` is `get`'s way of saying you've reached the end of file. Compare it using the `std::char_traits<char>::eof()` (or `std::istream::traits_type::eof()`) - avoid `-1`, it's a magic number. (Although the other one is a bit verbose - you can always just call `istream::eof`)

The EOF flag is only set once a read tries to read *past the end of the file*. If I have a 3 byte file, and I only read 3 bytes, EOF is `false`, because I've not tried to read past the end of the file yet. While this seems confusing for files, which typically know their size, EOF is not known until a read is attempted on some devices, such as pipes and network sockets.

The second example works as `inf >> foo` will always return `inf`, with the side effect of attempt to read something and store it in `foo`. `inf`, in an `if` or `while`, will evaluate to `true` if the file is "good": no errors, no EOF. Thus, when a read fails, `inf` evaluates to `false`, and your loop properly aborts. However, take this common error:

```

while(!inf.eof()) // EOF is false here
{
    inf >> x;      // read fails, EOF becomes true, x is not set
    // use x      // we use x, despite our read failing.
}

```

However, this:

```

while(inf >> x) // Attempt read into x, return false if it fails
{
    // will only be entered if read succeeded.
}

```

Which is what we want.

edited Dec 26 '10 at 7:52

answered Dec 26 '10 at 7:38



Thanatos
11.5k 4 32 68

Thanks Thanatos. I think I get it now ;) — [Chan](#) Dec 26 '10 at 8:04

- Probably should not use `>>` instead of `get()`. Remember that the `>>` operator always skips proceeding white space before attempting to read the next value while `get()` reads every character one at a time. — [Loki Astari](#) Dec 26 '10 at 8:09

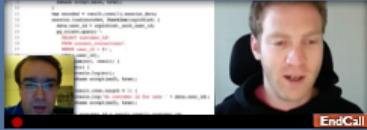
@Thanatos: Sorry to bother you but ... What about streams? You said that EOF is set when we try to read characters past the end of the file but **what happens when we try to peek() at the input stream cin when it has no characters?** The standard says that `traits::eof()` is called but `peek()` seems to wait for an input when I attempt to use it that way. – [ChosenTorture](#) Jul 11 '13 at 16:06

`peek` will return EOF if you attempt to peek past the end of the input. However, `cin` is typically a terminal that you're typing into. (This seems to be your case.) "Waiting for user input" is not the same as an EOF, it's just the source of data (you) is slow. So `peek` blocks, waiting for you to give it input. If you input EOF

(Ctrl+D on Linux), then `peek` will return, and `eof` should be true. Otherwise, it returns what you typed. – [Thanatos](#) Aug 23 '13 at 22:37

[add comment](#)

Become a better C++ coder
Pair program with an expert **NOW!**



Visit us at
airpair
www.airpair.com

`istream` doesn't know it's at the end of the file until it tries to read that first character past the end of the file.

The [sample code at cplusplus.com](#) says to do it like this: (But you shouldn't actually do it this way)

```
while (is.good())    // loop while extraction from file is possible
{
    c = is.get();    // get character from file
    if (is.good())
        cout << c;
}
```

A better idiom is to move the read into the loop condition, like so: (You can do this with *all* `istream` read operations that return `*this`, including the `<<` operator)

```
char c;
while(is.get(c))
    cout << c;
```

answered Dec 26 '10 at 7:31



[Ken Bloom](#)

25.2k 1 51 103

Should "istream read operations ... including the `<<` operator" read "...including the `>>` operator"? – [Thanatos](#) Dec 26 '10 at 7:47

@Thanatos: yeah. That's what I get for posting late at night. – [Ken Bloom](#) Dec 26 '10 at 17:09

[add comment](#)

The EOF flag is only set after a read operation attempts to read past the end of the file. `get()` is returning the symbolic constant `traits::eof()` (which just happens to equal -1) because it reached the end of the file and could not read any more data, and only at that point will `eof()` be true. If you want to check for this condition, you can do something like the following:

```
int ch;
while ((ch = inf.get()) != EOF) {
    std::cout << static_cast<char>(ch) << "\n";
}
```

edited Dec 26 '10 at 7:46

answered Dec 26 '10 at 7:29



[Justin Spahr-Summers](#)

10.4k 1 26 49

3 "The EOF flag is only set after a read operation reaches the end of the file." is exactly the statement that causes confusing about EOF. The EOF flag is only set *after a read operation attempts to read past the end of the file*. The last bit is critical: If I read 3 bytes from a file 3 bytes long, EOF is false, until I attempt another read. – [Thanatos](#) Dec 26 '10 at 7:31

@Thanatos Good catch. I'll update my answer. – [Justin Spahr-Summers](#) Dec 26 '10 at 7:32

`istream` doesn't return an EOF constant. The return value of `int istream::get()` appears to be undefined at the end of the file – you have to check `istream::eof()` – [Ken Bloom](#) Dec 26 '10 at 7:36

@Ken Bloom: Are you sure? My standard says "Returns: c if available, otherwise traits::eof() ." 27.6.1.3 (Draft 2, 2 Dec 96) – [Thanatos](#) Dec 26 '10 at 7:45

<s>You're right, but that's still a far cry from the EOF constant used by C stdio.<s> I'm wrong. Somewhere out there in the definition of char_traits<char> it says that char_traits<char>::eof() returns EOF – [Ken Bloom](#) Dec 26 '10 at 7:48

show 8 more comments

eof() checks the eofbit in the stream state.

On each read operation, if the position is at the end of stream and more data has to be read, eofbit is set to true. Therefore you're going to get an extra character before you get eofbit=1.

The correct way is to check whether the eof was reached (or, whether the read operation succeeded) *after* the reading operation. This is what your second version does - you do a read operation, and then use the resulting stream object reference (which >> returns) as a boolean value, which results in check for fail().

edited Dec 26 '10 at 8:12

answered Dec 26 '10 at 7:35



[zeuxcg](#)

3,725 9 19

Thanks for the explanation ;) – [Chan](#) Dec 26 '10 at 8:08

It checks several bits actually. – [Loki Astari](#) Dec 26 '10 at 8:11

Thanks, fixed the answer. – [zeuxcg](#) Dec 26 '10 at 8:13

add comment

Not the answer you're looking for? Browse other questions tagged [c++](#) or [ask your own question](#).