eof() bad practice? [duplicate]

Possible Duplicate:
Why is iostream::eof inside a loop condition considered wrong?

So I've been using the eof() function in a lot of my programs that require file input, and my professor said that it is fine to use but a few people on SO have said that I shouldn't use it without really specifying the reason. So I was wondering, is there a good reason?

Thanks, Sam

c++ eof

edited Nov 12 '11 at 13:01
FredOverflow
96.9k ● 28 ● 186 ● 425

asked Apr 29 '11 at 21:35
Sam
825 ● 3 ● 16 ● 42

marked as duplicate by FredOverflow, Xeo, sbi, R. Martinho Fernandes, Dori Nov 14 '11 at 8:28

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

I take it you mean the `feof()` function? – Oliver Charlesworth Apr 29 '11 at 21:36

1 He likely means the iostream::eof() one. – Puppy Apr 29 '11 at 21:37

@Dead: Ah, yes. – Oliver Charlesworth Apr 29 '11 at 21:38

4 Answers

You can use `eof` to test for the exact condition it reports - whether you *have attempted to read past end of file*. You cannot use it to test whether there's more input to read, or whether reading succeeded, which are more common tests.

Wrong:

```
while (!cin.eof()) {
  cin >> foo;
}
```

Correct:

```
if (!(cin >> foo)) {
  if (cin.eof()) {
    cout << "read failed due to EOF\n";
  } else {
    cout << "read failed due to something other than EOF\n";
  }
}
```

answered Apr 29 '11 at 21:38
Erik
40.7k ● 2 ● 95 ● 135

but if you've read past the end of the file, shouldn't there be no remaining information? – Sam Apr 29 '11 at 21:43

2 @Sam: `eof()` does not mean "file position is at end and next read will fail". It means "You have attempted to read something *beyond* end of file and that failed" – Erik Apr 29 '11 at 21:46

I think I understand what you're saying, but what harm could come from reading past the end of file? Wouldn't you still have all data extracted? Or are you saying at times you don't want all of the data? Or.. – Sam Apr 29 '11 at 21:48

1 @Sam: The point is that in a `while (!eof()) { cin >> foo; }` loop you're not reading **until** end of file. You're reading until end of file **and once more**. And you're not testing for the other ways a read may fail. –

Erik Apr 29 '11 at 21:49

5 @Sam: `int foo; while (!cin.eof()) { cin >> foo; cout << foo << "\n" }`. This is a typical construct, which does not do what the programmer intended. It will *not* read all ints and display them again, it will *at best* read all ints and display them, then display the last one once more. If you give it e.g. "1 2 abc" as input, it'll read and display "1", "2", then loop forever displaying "2" over and over. – Erik Apr 29 '11 at 21:54

You shouldn't use it because if input fails for another reason, you can be screwed.

```
while(!file.eof()) {
    int i;
    file >> i;
    doSomething(i);
}
```

What happens in the above loop if the contents of file are "WHAARRRRRRGARBL"? It loops infinitely, because it's not the end of the file, but you still can't extract an int from it.

answered Apr 29 '11 at 21:38

Puppy
94.8k ● 8 ● 118 ● 268

just kind of thinking aloud, but couldn't you just put it in a string and then manipulate that? Or is that just inefficient? – Sam Apr 29 '11 at 21:44

Inefficient, and what's the point of using formatted input like `cin >> somedouble >> someint >> somechar;` ? – Drise Aug 1 '12 at 21:47

How are you using it? What `.eof()` will tell you is that the stream has already hit the end of file, that it has tried to read data that isn't there.

This isn't usually what you want. Usually, you want to know that you are hitting the end of file, or that you are about to hit the end of file, and not that you have already hit it. In a loop like:

```
while(!f.eof())
{
    f >> /* something */;
    /* do stuff */
}
```

you are going to attempt to read input in and fail, and then you are going to execute everything in `/* do stuff */`, and then the loop condition fails and the loop stops.

Another problem with that loop is that there's other ways input can fail. If there's an error condition on the file that isn't EOF, `.fail()` will return `true` but `.eof()` won't.

answered Apr 29 '11 at 21:43

David Thomley
40.2k ● 8 ● 59 ● 124

So are you saying that it is because my variable or whatever I'm choosing to hold the data in won't be suitable for that type? I'm not sure why this seems so confusing to me. – Sam Apr 29 '11 at 21:52

@Sam: Reading the wrong data is a possibility, but you'll have to deal with that separately. It won't trigger the `.eof()` function or any other like that. However, if there's a system error reading the file you probably also want to stop. – David Thomley Apr 29 '11 at 22:00

In case the above answers are confusing:

What people thinks it does is wrong:

- eof() does not look to see if the last portion you read included the last byte of the file.
- It does not look to see if the next byte is after the end of the file.

What it actually does:

- eof() reports whether the last read included bytes past the end of the file.

If eof() is true, you've already made a mistake. Thus explains the professors statement. Use it for

what it means, an error has occurred.

answered Jul 22 '11 at 14:43

Lee Louviere

3,417 • 11 • 37

Reading past the end of file is not a mistake but a very normal operation. Consider e.g. istringstream s("123"); int i; s >> i; Now s.eof() is true (because reading had to peek if there is a fourth digit), s.fail() is false (because the read was successful) and i contains the value 123. – Tronic Jan 20 '13 at 16:24