Welcome back! If you found this question useful, don't forget to vote both the question and the answers up.

close this message



We know that stdin is, by default, a buffered input; the proof of that is in usage of any of the mechanisms that "leave data" on stdin, such as scanf():

25

```
int main()
{
         char c[10] = {'\0'};
         scanf("%9s", c);
         printf("%s, and left is: %d\n", c, getchar());
         return 0;
}

./a.out
    hello
    hello, and left is 10
```

10 being newline of course...

I've always been curious, is there any way to "peek" at the stdin buffer without removing whatever may reside there?

EDIT

A better example might be:

```
scanf("%9[^.]", c);
```

With an input of "at.ct", now I have "data" (ct\n) left on stdin, not just a newline.

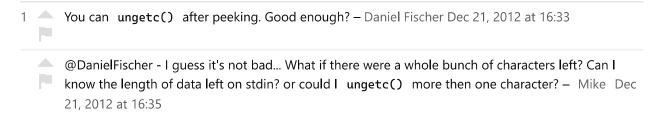


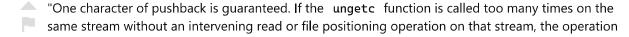
Share Edit Follow Flag



asked Dec 21, 2012 at 16:29







Welcome back! If you found this question useful, don't forget to vote both the question and the answers up.

close this message

Mike Jan 21, 2013 at 16:07

3 Answers

Sorted by: Highest score (default)

\$



16

Portably, you can get the next character in the input stream with <code>getchar()</code> and then push it back with <code>ungetc()</code>, which results in a state as if the character wasn't removed from the stream.







The ungetc function pushes the character specified by c (converted to an unsigned char) back onto the input stream pointed to by stream. Pushed-back characters will be returned by subsequent reads on that stream in the reverse order of their pushing.



Only one character of pushback is guaranteed by the standard, but usually, you can push back more.

As mentioned in the other answers resp. the comments there, in practice, you can almost certainly peek at the buffer if you provide your own buffer with <code>setvbuf</code>, although that is not without problems:

If buf is not a null pointer, the array it points to may be used instead of a buffer allocated by the setvbuf function

that leaves the possibility that the provided buffer may not be used at all.

The contents of the array at any time are indeterminate.

that means you have no guarantee that the contents of the buffer reflects the actual input (and it makes using the buffer undefined behaviour if it has automatic storage duration, if we're picky).

However, in practice the principal problem would be finding out where in the buffer the notyet-consumed part of the buffered input begins and where it ends.

Share Edit Follow Flag

answered Jan 21, 2013 at 17:20

Daniel Fischer



Welcome back! If you found this question useful, don't forget to vote both the question and the answers up.

close this message

Daniel Fischer Jan 10, 2014 at 12:46



If you want to look at the stdin buffer without changing it, you could tell it to use a another buffer with setbuf, using an array you can access:











if (setbuf(stdin, buffer) != 0) // error getchar(); printf("%15s\n", buffer);

char buffer[BUFSIZ];

This let you see something more than ungetc, but I don't think you can go further in a portable way.

Actually this is legal but is not correct for the standard, quoting from it about the setvbuf (setbuf has the same behavior):

The contents of the array at any time are indeterminate.

So this is not what you need if you're looking for complete portability and standardcompliance, but I can't imagine why the buffer should not contain what is expected. However, it seems to work on my computer.

Beware that you have to provide an array of at least BUFSIZ characters to setbuf, and you must not do any I/O operation on the stream before it. If you need more flexibility, take a look at <u>setvbuf</u>.

Share Edit Follow Flag

edited Dec 21, 2012 at 18:48

answered Dec 21, 2012 at 17:26



2,871 3 23 44

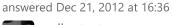


You could set your own buffer with <u>setvbuf</u> on stdin, and peek there whenever you want.



Share Edit Follow Flag

edited Dec 22, 2012 at 16:32 user283145





nullpotent

9,250 1 33 44







Welcome back! If you found this question useful, don't forget to vote both the question and the answers up.

close this message

