25

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()`:

```
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.

c buffer stdin peek

Share Edit Follow Flag

edited Oct 11, 2018 at 1:31
ShadowRanger
153k 12 203 297

asked Dec 21, 2012 at 16:29
Mike
48.9k 30 117 179

1 You can `ungetc()` after peeking. Good enough? – Daniel Fischer Dec 21, 2012 at 16:33

@DanielFischer - I guess it's not bad... What if there were a whole bunch of characters left? Can I know the length of data left on stdin? or could I `ungetc()` more then one character? – Mike Dec 21, 2012 at 16:35

"One character of pushback is guaranteed. If the `ungetc` function is called too many times on the same stream without an intervening read or file positioning operation on that stream, the operation

– 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 `getchar()` and then push it back with `ungetc()`, 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 `setvbuf`, 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 not-yet-consumed part of the buffered input begins and where it ends.

Share Edit Follow Flag

answered Jan 21, 2013 at 17:20

Daniel Fischer
183k 17 313 434

– Daniel Fischer Jan 10, 2014 at 12:46

4

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:

```
char buffer[BUFSIZ];

if (setbuf(stdin, buffer) != 0)
  // error

getchar();

printf("%15s\n", buffer);
```

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 standard-compliance, 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 `setvbuf`.

Share Edit Follow Flag

edited Dec 21, 2012 at 18:48

answered Dec 21, 2012 at 17:26
effeffe
2,871 3 23 44

4

You could set your own buffer with `setvbuf` on stdin, and peek there whenever you want.

Share Edit Follow Flag

edited Dec 22, 2012 at 16:32
user283145

answered Dec 21, 2012 at 16:36
nullpotent
9,250 1 33 44

`The setvbuf() function has no effect on stdout, stdin, or stderr.` Is what I've read. Do you have a functioning example? – Mike Dec 21, 2012 at 16:49

@Mike It does seem to work on windows. `ungetc` works as expected. Not entirely sure about how to keep track of read/left pos. But if the buff is new-lined, then it would be easy. – nullpotent Dec 21, 2012 at 18:23

1 Whilst setvbuf does indeed set a custom buffer, it is not a reliable/meaningful way to know what the input may be - the main problem being to know "where are we in the buffer" - stdin won't tell you how far it has actually consumed. Most importantly, it won't necessarily show things after the next newline (it may do, but it may not). ungetc() is the only way to reliably do this, without writing your own input functions. – Mats Petersson Dec 21, 2012 at 22:57