



Read a single byte from stdin

1

2

Next >

**Martin Gregorie**

Guest

Is there a standard way to read single bytes from a Java program's standard input without involving AWT or Swing?

By that I mean that I need unbuffered input: I'd like to read each keystroke as the key is hit with it being echoed to standard output, since I need to control whether it gets echoed or not.

I need to do within a program that's capable of running in a headless environment and without using native code. I've looked at the standard System streams and Console, but both are doing line buffering and neither allows echo control (apart from Console.readPassword, and that's line buffered).

I have a feeling I may be missing something obvious, so if its possible to do single character, unbuffered non-echoed input from stdin, please point me at documentation, a tutorial or some example code.

--

martin@ | Martin Gregorie
gregorie. | Essex, UK
org |

Jul 12, 2009



Arne VajhÃ¸j

Guest

Martin Gregorie wrote:

- > Is there a standard way to read single bytes from a Java program's
- > standard input without involving AWT or Swing?
- >
- > By that I mean that I need unbuffered input: I'd like to read each
- > keystroke as the key is hit with it being echoed to standard output,
- > since I need to control whether it gets echoed or not.
- >
- > I need to do within a program that's capable of running in a headless
- > environment and without using native code. I've looked at the standard
- > System streams and Console, but both are doing line buffering and neither
- > allows echo control (apart from Console.readPassword, and that's line
- > buffered).
- >
- > I have a feeling I may be missing something obvious, so if its possible
- > to do single character, unbuffered non-echoed input from stdin, please
- > point me at documentation, a tutorial or some example code.

I don't think it can be done.

AFAIK then there are no universal support for this feature
in C so Java can not do it either.

A platform specific JNI solution is possible for all the most
common platforms.

Arne

Jul 12, 2009



Martin Gregorie

Guest

On Sun, 12 Jul 2009 15:15:04 -0400, Arne VajhÃ¸j wrote:

> AFAIK then there are no universal support for this feature in C so Java
> can not do it either.

>

Unless I'm much mistaken, there is a POSIX solution in C:

- use `tcsetattr()` to turn off buffering and input echoing
- use `read(0, &c, 1)` to read a single byte

> A platform specific JNI solution is possible for all the most common
> platforms.

>

Yes, but I'm keen to avoid that for portability reasons.

--

martin@ | Martin Gregorie
gregorie. | Essex, UK
org |

Jul 12, 2009



Lew

Guest

Martin Gregorie wrote:

> Is there a standard way to read single bytes from a Java program's
> standard input without involving AWT or Swing?

>

> By that I mean that I need unbuffered input: I'd like to read each
> keystroke as the key is hit with it being echoed to standard output,
> since I need to control whether it gets echoed or not.

>

> I need to do within a program that's capable of running in a headless
> environment and without using native code. I've looked at the standard
> System streams and Console, but both are doing line buffering and neither
> allows echo control (apart from Console.readPassword, and that's line
> buffered).

Are you sure about that? Is this based on tests?

There's nothing in the documentation of System.in that says that it's line buffered. There's also nothing that says it isn't. It only claims to be an 'InputStream', and as such sports a 'read()' method that takes a single byte. Which, if supported, still might not be a character.

Note also that there is no 'readLine()' or equivalent for System.in.

--

Lew

Jul 12, 2009



Martin Gregorie

Guest

On Sun, 12 Jul 2009 16:57:25 -0400, Lew wrote:

> Are you sure about that? Is this based on tests?
>

Yes it is, for both System.in and the Reader returned by Console.read(), so I assume the same goes for Console.in as well.

> There's nothing in the documentation of System.in that says that it's
> line buffered. There's also nothing that says it isn't. It only claims
> to be an 'InputStream', and as such sports a 'read()' method that takes
> a single byte.
> Which, if supported, still might not be a character.
>

In both cases you can read a character. This test program illustrates the situation for Console. Substitute System.in for the Reader and you get the same result:

```
import java.io.*;

public class TestIn
{
    public static void main(String[] args)
    {
        boolean cycle = true;
        Console cons = System.console();
        Reader rdr = cons.reader();
```

```
while(cycle)
{
try
{
char cbuf[] = new char[1];
rdr.read(cbuf, 0, 1);
System.out.println("got cbuf[0] + " + cbuf[0]);
if (cbuf[0] == 'X')
cycle = false;
}
catch (IOException e)
{
System.out.println(e.getMessage());
System.exit();
}
}
}
```

This reports individual characters as you'd expect, but not until you've hit Enter. Then the loop spins to retrieve all the characters you've entered including the final newline.

The same wait then spin occurs for the next line.

> [Note also that there is no 'readLine\(\)' or equivalent for System.in.](#)

>

Yep. InputStream and the Reader returned by Console have essentially the same set of read() operations. Console does at least have readLine() and a non-echoing readPassword().

--

martin@ | Martin Gregorie
gregorie. | Essex, UK
org |

Jul 12, 2009



Lew
Guest

Lew wrote:

>> [Are you sure about that? Is this based on tests?](#)

Martin Gregorie wrote:

> [Yes it is, for both System.in and the Reader returned by Console.read\(\),](#)
> [so I assume the same goes for Console.in as well.](#)

What is Console.in? I see no documentation for that at all.

I will take your word about the buffering of System.in, even though there's nothing to force an InputStream to be buffered.

> Yep. InputStream and the Reader returned by Console have essentially the
> same set of read() operations. Console does at least have readLine() and
> a non-echoing readPassword().

However, InputStream does not, nor does it always have to be buffered. Again, I trust that your tests show that System.in is actually line buffered.

--

Lew

Jul 12, 2009



Arne VajhÃ¸j
Guest

Martin Gregorie wrote:

> On Sun, 12 Jul 2009 15:15:04 -0400, Arne VajhÃ¸j wrote:
>> AFAIK then there are no universal support for this feature in C so Java
>> can not do it either.
>>
> Unless I'm much mistaken, there is a POSIX solution in C:
> - use tcsetattr() to turn off buffering and input echoing
> - use read(0, &c, 1) to read a single byte

Yes.

But Java is more than POSIX.

Arne

Jul 12, 2009



Arne VajhÃ¸j
Guest

Lew wrote:

> Lew wrote:
>> Yep. InputStream and the Reader returned by Console have essentially
>> the same set of read() operations. Console does at least have
>> readLine() and a non-echoing readPassword().
>
> However, InputStream does not, nor does it always have to be buffered.
> Again, I trust that your tests show that System.in is actually line
> buffered.

I would put it another way.

If System.in is line buffered on one or more systems, then a portable program can not not assume it is not line buffered.

Implementation specific does not solve his problem. He needs something that is documented to work in all compliant implementations.

Arne

Jul 12, 2009



Joshua Cranmer

Guest

Lew wrote:

> There's nothing in the documentation of System.in that says that it's
> line buffered. There's also nothing that says it isn't. It only claims
> to be an 'InputStream', and as such sports a 'read()' method that takes
> a single byte. Which, if supported, still might not be a character.

In general, buffering is actually performed by the terminal itself; one generally has to invoke special libraries (most commonly curses or ncurses) to be able to override this feature.

--

Beware of bugs in the above code; I have only proved it correct, not tried it. -- Donald E. Knuth

Jul 13, 2009



Arne VajhÃ¸

Guest

Joshua Cranmer wrote:

> Lew wrote:

>> There's nothing in the documentation of System.in that says that it's
>> line buffered. There's also nothing that says it isn't. It only
>> claims to be an 'InputStream', and as such sports a 'read()' method
>> that takes a single byte. Which, if supported, still might not be a
>> character.

>

> In general, buffering is actually performed by the terminal itself; one
> generally has to invoke special libraries (most commonly curses or
> ncurses) to be able to override this feature.

I would say terminal driver.

I believe that the echo comes from the host and not from the terminal in most contexts today.

(not 30-40 years ago though)

Arne

Jul 13, 2009



Mike Schilling

Guest

Arne Vajhøj wrote:

> **Lew wrote:**

>> **Lew wrote:**

>>> Yep. InputStream and the Reader returned by Console have

>>> essentially

>>> the same set of read() operations. Console does at least have

>>> readLine() and a non-echoing readPassword().

>>

>> However, InputStream does not, nor does it always have to be

>> buffered. Again, I trust that your tests show that System.in is

>> actually line buffered.

>

> I would put it another way.

>

> If System.in is line buffered on one or more systems, then

> a portable program can not not assume it is not line buffered.

>

> Implementation specific does not solve hos problem. He need

> something that is documented to work in all compliant

> implementations.

Right. In other words, if the JRE doesn't provide this (which to the best of my knowledge, it does not), the only solution is system-specific JNI. In fact, JRE support for this feature amounts to a requirement that JRE implementors provide the system-specific JNI for us.

Having said that, suppose for the sake of argument that you'd constructed separate JNI for Windows, Mac, and Linux [1], and were happy that you'd now solved the problem for all of your likely users. What's the best way to package your application? Do you need three different versions, or is there some magic that will allow the app to locate and load the proper version of the JNI library at runtime?

1. Linux on i86 processors, anyway.

Jul 13, 2009



Martin Gregorie

Guest

On Sun, 12 Jul 2009 18:36:58 -0400, Arne VajhÃj wrote:

> Martin Gregorie wrote:

>> On Sun, 12 Jul 2009 15:15:04 -0400, Arne VajhÃj wrote:

>>> AFAIK then there are no universal support for this feature in C so

>>> Java can not do it either.

>>>

>> Unless I'm much mistaken, there is a POSIX solution in C: - use

>> tcsetattr() to turn off buffering and input echoing - use read(0, &c,

>> 1) to read a single byte

>

> Yes.

>

> But Java is more than POSIX.

>

Of course. But what is the JVM and the native classes written in?

--

martin@ | Martin Gregorie

gregorie. | Essex, UK

org |

Jul 13, 2009



Roedy Green

Guest

On Sun, 12 Jul 2009 19:00:40 +0000 (UTC), Martin Gregorie

<> wrote, quoted or indirectly quoted

someone who said :

>Is there a standard way to read single bytes from a Java program's

>standard input without involving AWT or Swing?

>

>By that I mean that I need unbuffered input: I'd like to read each

>keystroke as the key is hit with it being echoed to standard output,

>since I need to control whether it gets echoed or not.

>

>I need to do within a program that's capable or running in a headless

>environment and without using native code. I've looked at the standard

>System streams and Console, but both are doing line buffering and neither
>allows echo control (apart from Console.readPassword, and that's line
>buffered).
>
>I have a feeling I may be missing something obvious, so if its possible
>to do single character, unbuffered non-echoed input from stdin, please
>point me at documentation, a tutorial or some example code.

Since some platforms support only-line-by line console input, when you read a single char unbuffered, on some platforms it will block till the user hits Enter. On some you might get the characters one at a time.

Peter van der Linden wrote a class that fielded keystroke events to simulate a console that could read a character at a time. I vaguely recall perusing the code and thinking it was quite a bit terser than I expected.

It was not his EasyIn class. If someone can track Peter down, maybe it is posted somewhere.

A raw console is pretty simple. Just how far to do go to simulate a real command processor console?

1. command recall
2. command completion
3. ability to spawn programs
4. editor
5. colour/formatted output.

--

Roedy Green Canadian Mind Products
<http://mindprod.com>

"For reason that have a lot to do with US Government bureaucracy, we settled on the one issue everyone could agree on, which was weapons of mass destruction."
~ Paul Wolfowitz 2003-06, explaining how the Bush administration sold the Iraq war to a gullible public.

Jul 13, 2009



Roedy Green
Guest

On Sun, 12 Jul 2009 20:45:45 -0400, Arne Vajhøj <>
wrote, quoted or indirectly quoted someone who said :

>(not 30-40 years ago though)

The Vucom glass TTY popular in Canada would buffer up one line, allowing local edit. Then when you had it the way you wanted, you hit

SEND.

In theory, Java could be implemented on a mainframe with such terminals as the console.

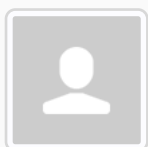
--

Roedy Green Canadian Mind Products
<http://mindprod.com>

"For reason that have a lot to do with US Government bureaucracy, we settled on the one issue everyone could agree on, which was weapons of mass destruction."

~ Paul Wolfowitz 2003-06, explaining how the Bush administration sold the Iraq war to a gullible public.

Jul 13, 2009



Martin Gregorie

Guest

On Sun, 12 Jul 2009 18:19:14 -0400, Lew wrote:

> [What is Console.in?](#) I see no documentation for that at all.

>

My mistake - I thought it was an attribute of Console, but it isn't. At a guess Console sits over System and the reader(), readLine() and readPassword() methods all map onto System.in.

> [However, InputStream does not, nor does it always have to be buffered.](#)

> [Again, I trust that your tests show that System.in is actually line](#)

> [buffered.](#)

>

Here's how my TestIn class behaves:

type 'abc' - keys echo, no program output

hit ENTER - get output:

got [a]

got

got [c]

got [

]

hit X - 'X' is echoed, no program output

hit Enter - get output:

got [X]

\$

That looks like line buffering to me.

--

martin@ | Martin Gregorie

Jul 13, 2009



Martin Gregorie
Guest

On Sun, 12 Jul 2009 20:45:45 -0400, Arne VajhÃj wrote:

> I believe that the echo comes from the host and not from the terminal in
> most contexts today.
>

That's true for all full-duplex physical terminals I've used, e.g. Wyse 120, and the X-term console. AFAIK it was only the old half-duplex terminals (or half-duplex compatibility mode on newer ones) that don't require a software echo from the host.

There's a useful little OS/9 utility, codes, that shows what byte(s) a terminal keystroke generates that I've re-implemented for DOS and *nixen without any problem. Here's what it does:

Keystroke Output

=====

a a = 97 0x61

^A . = 1 0x01 (ctrl-a)

right arrow . = 27 0x1b (Esc)

[= 91 0x5b

C = 67 0x43

'Right arrow' is the '->' key with an X-term console running. 'codes' is a good test case for this stuff, since to work correctly it needs unbuffered terminal input with echoing suppressed. Non-printable characters are output as '.' to avoid messing up the display so echo must be off.

Implementing this has involved three C dialects: K&R under OS/9, Borland C for DOS/Windows and the Gnu ANSI/POSIX C for Linux. All have methods for suppressing line buffering and echo though the system calls to do it differ. I haven't tried compiling it with the DJGPP DOS/Windows port of the Gnu C compiler, but the Linux version should 'just work' there too.

--

martin@ | Martin Gregorie
gregorie. | Essex, UK
org |

Jul 13, 2009



Martin Gregorie

Guest

On Mon, 13 Jul 2009 03:52:02 -0700, Roedy Green wrote:

> On Sun, 12 Jul 2009 20:45:45 -0400, Arne VajhÃj <> wrote,

> quoted or indirectly quoted someone who said :

>

>>(not 30-40 years ago though)

>

> The Vucom glass TTY popular in Canada would buffer up one line, allowing

> local edit. Then when you had it the way you wanted, you hit SEND.

>

That was advanced. The only glass teletypes I remember seeing then (1972/73) was literally just that. IIRC it was exactly compatible with an ASR-33 teletype. I remember we plugged one onto our mainframe (an ICL 1900 running George 3) and it behaved just like, errm, a glass teletype. It had no scroll buffer, so at least emulating a slow teletype meant you could read the output before it scrolled off the screen and was lost forever. Its chief benefit was that it was quiet. I don't remember who made it.

--

martin@ | Martin Gregorie
gregorie. | Essex, UK
org |

Jul 13, 2009



Joshua Cranmer

Guest

Martin Gregorie wrote:

> Of course. But what is the JVM and the native classes written in?

s/classes/methods, but in any case, they're all written in C++. I'm not sure how array classes are handled in the JVM (that's a nice piece of black magic), but I suspect they are not really constructed classes with any attached bytecode but rather magic oops.

--

Beware of bugs in the above code; I have only proved it correct, not tried it. -- Donald E. Knuth

Jul 13, 2009

Martin Gregorie

Guest



On Mon, 13 Jul 2009 03:49:35 -0700, Roedy Green wrote:

> on some platforms it will block till the
> user hits Enter. On some you might get the characters one at a time.
>

That's exactly what I want.

> Peter van der Linden wrote a class that fielded keystroke events to
> simulate a console that could read a character at a time. I vaguely
> recall perusing the code and thinking it was quite a bit terser than I
> expected.

>

Earlier today I was wondering how possible it would be to do exactly that
via the KeyEvent mechanism.

It seems likely that I'll have to define an invisible AWT or Swing window
the same size as my screen in order to define the area where key presses
would generate KeyEvents. The idea is that this whole mess should run on
a headless server that's being accessed via ssh, so does anybody know if
a program that creates invisible AWT or Swing objects can start correctly
on a headless system that doesn't have either an X-server running locally
or access to a remote X-term via ssh X.11 forwarding?

> It was not his EasyIn class. If someone can track Peter down, maybe it
> is posted somewhere.
>

Found an e-mail address and pinged him. Thanks for the hint.

--

martin@ | Martin Gregorie
gregorie. | Essex, UK
org |

Jul 13, 2009



Arne VajhÃj
Guest

Martin Gregorie wrote:

> On Sun, 12 Jul 2009 18:36:58 -0400, Arne VajhÃj wrote:

>> Martin Gregorie wrote:

>>> On Sun, 12 Jul 2009 15:15:04 -0400, Arne VajhÃj wrote:

>>>> AFAIK then there are no universal support for this feature in C so

>>>> Java can not do it either.

>>>>

>>> Unless I'm much mistaken, there is a POSIX solution in C: - use
>>> tcsetattr() to turn off buffering and input echoing - use read(0, &c,
>>> 1) to read a single byte
>> Yes.
>>
>> But Java is more than POSIX.
>>
> Of course. But what is the JVM and the native classes written in?

I don't think it is required to be in any specific languages.

I would expect it to be coded in C (with possible a bit of C++)
on all platforms - both POSIX complaint and non POSIX compliant.

But JVM's are very platform specific, so they do not have a
problem with platform specific code.

Arne

Jul 13, 2009

Want to reply to this thread or ask your own question?

It takes just 2 minutes to sign up (and it's free!). Just click the sign up button to choose a username and then you can ask your own questions on the forum.

Sign Up Now!

Similar Threads



Single byte addressable, multiple byte readout.

Andreas, in forum: VHDL

Replies: 1

May 4, 2004



read/write data byte-per-byte to and from a socket

crash.test.dummy, in forum: Java

Replies: 1

Feb 17, 2006

peek at stdin, flush stdin



Johnathan Doe, in forum: C Programming
Replies: 5

May 17, 2013



Reading stdin once confuses second stdin read
Charlie Zender, in forum: C Programming
Replies: 6

Jun 21, 2004



how to read/write a characters stream which is either of one byte/2 byte
Deep, in forum: C Programming
Replies: 6

Feb 28, 2007



How to pass stdin of a C++ program to the stdin of a process createdwith
ShellExecute()
Ben, in forum: C Programming
Replies: 2

Aug 29, 2009



read text file byte by byte
daved170, in forum: Python
Replies: 30

Dec 16, 2009



STDIN, OUT, ERR and \$stdin, out, err - Differences?
Terry Cooper, in forum: Ruby
Replies: 7


Jun 9, 2009

About Us

The Coding Forums is a place to seek help and ask questions relating to coding and programming languages.

[Privacy Policy](#) [Terms and Rules](#) [Help](#)

Connect With Us

[Register](#) 

[Contact Us](#) 