Quickly read the last line of a text file?

Asked 15 years, 9 months ago Modified 1 month ago

Viewed 108k times

70

What's the quickest and most efficient way of reading the last line of text from a [very, very large] file in Java?

java file io

Share Follow

edited Nov 4, 2013 at 20:13

Gray

117k ● 24 ● 302 ● 359

asked Mar 26, 2009 at 15:17

Jake

15.2k ● 22 ● 73 ● 86

11 Answers

Sorted by: Highest score (default)

95

Below are two functions, one that returns the last non-blank line of a file without loading or stepping through the entire file, and the other that returns the last N lines of the file without stepping through the entire file:

What tail does is zoom straight to the last character of the file, then steps backward, character by character, recording what it sees until it finds a line break. Once it finds a line break, it breaks out of the loop. Reverses what was recorded and throws it into a string and returns. 0xA is the new line and 0xD is the carriage return.

If your line endings are `\r\n` or `crlf` or some other "double newline style newline", then you will have to specify n*2 lines to get the last n lines because it counts 2 lines for every line.

```
public String tail( File file ) {
    RandomAccessFile fileHandler = null;
    try {
        fileHandler = new RandomAccessFile( file, "r"
        long fileLength = fileHandler.length() - 1;
        StringBuilder sb = new StringBuilder();

        for(long filePointer = fileLength; filePointer
            fileHandler.seek( filePointer );
            int readByte = fileHandler.readByte();

            if( readByte == 0xA ) {
                if( filePointer == fileLength ) {
                    continue;
                }
                break;

            } else if( readByte == 0xD ) {
                if( filePointer == fileLength - 1 ) {
                    continue;
                }
                break;
            }

            sb.append( ( char ) readByte );
        }
        String lastLine = sb.reverse().toString();
        return lastLine;
    } catch( java.io.FileNotFoundException e ) {
        e.printStackTrace();
        return null;
    } catch( java.io.IOException e ) {
        e.printStackTrace();
        return null;
    } finally {
        if (fileHandler != null )
            try {
                fileHandler.close();
            } catch (IOException e) {
                /* ignore */
            }
    }
}
```

But you probably don't want the last line, you want the last N lines, so use this instead:

```
public String tail2( File file, int lines) {
    java.io.RandomAccessFile fileHandler = null;
    try {
        fileHandler =
            new java.io.RandomAccessFile( file, "r" );
        long fileLength = fileHandler.length() - 1;
        StringBuilder sb = new StringBuilder();
        int line = 0;

        for(long filePointer = fileLength; filePointer
            fileHandler.seek( filePointer );
            int readByte = fileHandler.readByte();

             if( readByte == 0xA ) {
                if (filePointer < fileLength) {
                    line = line + 1;
                }
            } else if( readByte == 0xD ) {
                if (filePointer < fileLength-1) {
                    line = line + 1;
                }
                }
                if (line >= lines) {
                    break;
                }
                sb.append( ( char ) readByte );
            }

            String lastLine = sb.reverse().toString();
            return lastLine;
        } catch( java.io.FileNotFoundException e ) {
            e.printStackTrace();
            return null;
        } catch( java.io.IOException e ) {
            e.printStackTrace();
            return null;
        }
        finally {
            if (fileHandler != null )
                try {
                    fileHandler.close();
                } catch (IOException e) {
                }
        }
    }
```

Invoke the above methods like this:

```
File file = new File("D:\\stuff\\huge.log");
System.out.println(tail(file));
System.out.println(tail2(file, 10));
```

Warning In the wild west of unicode this code can cause the output of this function to come out wrong. For example "Mary?s" instead of "Mary's". Characters with hats, accents, Chinese characters etc may cause the output to be wrong because accents are added as modifiers after the character. Reversing compound characters changes the nature of the identity of the character on reversal. You will have to do full battery of tests on all languages you plan to use this with.

For more information about this unicode reversal problem read this: https://codeblog.jonskeet.uk/2009/11/02/omg-ponies-aka-humanity-epic-fail/

Share Follow

edited Apr 7, 2021 at 16:21

community wiki
20 revs, 8 users 83%
Eric Leschinski

3 The above does not take into account lines terminated with both CR and LF. – Jags Mar 17, 2014 at 3:44

1 your multiline implementation does not work in your special cases of filePointer == fileLength, line will stay the same, therefore the condition line == lines will not fire after that and the code will read the whole file. – ZPiDER Feb 16, 2015 at 10:51

The warning about the wild west of Unicode is confession of a bug. The algorithm utilized is flawed, hence the bug hence the warning. In UTF8 there is a maximum length of multibyte encoding; that length is N=4B×max-length-of-combining-chars. For any given byte position in a file, that byte might be a noninitial byte/grapheme in a nonnormalized Unicode character. When the code above thinks that it has found a fact, it must deem it a suspicious fact until the prior N bytes are examined to determine whether the suspicious fact is in fact not a fact due to being within a UTF8 multibyte seq. – Andreas ZUERCHER Apr 7, 2021 at 16:12

1 @AndreasZUERCHER I confess to the bug, as you say, Next I would like you to analyze, and write a program to solve the global timezone-problem, as described by computerphile here: youtube.com/watch?v=-5wpm-gesOY – Eric Leschinski Apr 7, 2021 at 16:28

To avoid the Unicode problems related to reverting the string (or the StringBuilder), one can read to a byte list, from the end of the file, revert it to a byte array and then create the String from the byte array. – Helder Daniel Jun 3, 2021 at 11:16

42

Apache Commons has an implementation using RandomAccessFile.

It's called ReversedLinesFileReader.

Share Follow

edited Jul 15, 2016 at 14:06

answered Feb 28, 2014 at 21:07

jaco0646
17k ● 10 ● 67 ● 95

I think this is the quickest way to read file in reverse order – Chathurika Sandarenu Sep 26, 2014 at 9:18

2 @JuanToroMarty It's possible to loop over the `readLine()` method. – Stephan May 17, 2015 at 21:18

1 This seems to me as the most elegant method. – Rauni Lillemets Jun 19, 2020 at 7:38

21

Have a look at my answer to a similar question for C#. The code would be quite similar, although the encoding support is somewhat different in Java.

Basically it's not a terribly easy thing to do in general. As MSalter points out, UTF-8 does make it easy to spot `\r` or `\n` as the UTF-8 representation of those characters is just the same as ASCII, and those bytes won't occur in multi-byte character.

So basically, take a buffer of (say) 2K, and progressively read backwards (skip to 2K before you were before, read the next 2K) checking for a line termination. Then skip to exactly the right place in the stream, create an `InputStreamReader` on the top, and a `BufferedReader` on top of that. Then just call `BufferedReader.readLine()`.

Share Follow

edited May 23, 2017 at 10:29
Community Bot
1 ● 1

answered Mar 26, 2009 at 15:22
Jon Skeet
1.5m ● 889 ● 9.3k ● 9.3k

2 UTF-8 doesn't matter - you need the last CR or LF character, which is a single byte in both ASCII and UTF-8. – MSalters Mar 26, 2009 at 15:39

6

Using FileReader or FileInputStream won't work - you'll have to use either FileChannel or RandomAccessFile to loop through the file backwards from the end. Encodings will be a problem though, as Jon said.

Share Follow

answered Mar 26, 2009 at 15:28

Michael Borgwardt
346k ● 80 ● 486 ● 723

1 Note, RandomAccessFile's performance sucks for individual operations - so do sensible size reads into a buffer.
– Tom Hawtin - tackline Mar 26, 2009 at 15:31

4

You can easily change the below code to print the last line.

MemoryMappedFile for printing last 5 lines:

```
private static void printByMemoryMappedFile(File file)
FileNotFoundException, IOException{
        FileInputStream fileInputStream=new FileInputS
        FileChannel channel=fileInputStream.getChannel
        ByteBuffer buffer=channel.map(FileChannel.MapM
channel.size());
        buffer.position((int)channel.size());
        int count=0;
        StringBuilder builder=new StringBuilder();
        for(long i=channel.size()-1;i>=0;i--){
            char c=(char)buffer.get((int)i);
            builder.append(c);
            if(c=='\n'){
                if(count==5)break;
                count++;
                builder.reverse();
                System.out.println(builder.toString())
                builder=null;
                builder=new StringBuilder();
            }
        }
        channel.close();
    }
```

RandomAccessFile to print last 5 lines:

```
private static void printByRandomAcessFile(File file)
FileNotFoundException, IOException{
        RandomAccessFile randomAccessFile = new Random
        int lines = 0;
        StringBuilder builder = new StringBuilder();
        long length = file.length();
        length--;
        randomAccessFile.seek(length);
        for(long seek = length; seek >= 0; --seek){
            randomAccessFile.seek(seek);
            char c = (char)randomAccessFile.read();
            builder.append(c);
            if(c == '\n'){
                builder = builder.reverse();
                System.out.println(builder.toString())
                lines++;
                builder = null;
                builder = new StringBuilder();
                if (lines == 5){
                    break;
                }
            }

        }
    }
```

Share Follow

answered Sep 25, 2013 at 11:43

Trying

14.3k ● 10 ● 71 ● 113

worked for me . thanks. is that way have any inconvenient ?
– Omar B. Nov 20, 2018 at 21:24

2

as far as I know The fastest way to read the last line of a text file is using FileUtils Apache class which is in "org.apache.commons.io". I have a two-million-line file and by using this class, it took me less than one second to find the last line. Here is the my code:

```
LineIterator lineIterator = FileUtils.lineIterator(new
String lastLine="";
while (lineIterator.hasNext()){
 lastLine=  lineIterator.nextLine();
}
```

Share Follow

answered Sep 17, 2018 at 4:27

arash nadali
598 ● 4 ● 7

1 Same comment above by Lorenzo also applies here: This works, but is probably not the most efficient solution.
– Martin Wunderlich Jul 21, 2021 at 6:12

1

```
try(BufferedReader reader = new BufferedReader(new Fil

    String line = null;

    System.out.println("==============================

    line = reader.readLine();       //Read Line ONE
    line = reader.readLine();       //Read Line TWO
    System.out.println("first line : " + line);

    //Length of one line if lines are of even length
    int len = line.length();

    //skip to the end - 3 lines
    reader.skip((reqFile.length() - (len*3)));

    //Searched to the last line for the date I was loo

    while((line = reader.readLine()) != null){

        System.out.println("FROM LINE : " + line);
        String date = line.substring(0,line.indexOf(",

        System.out.println("DATE : " + date);      //B
    }

    System.out.println(reqFile.getName() + " Read(" +
"KB)");
    System.out.println("==============================
} catch (IOException x) {
    x.printStackTrace();
}
```

Share Follow

edited Apr 3, 2018 at 7:49

Ahmed Ashour
5,521 ● 10 ● 39 ● 62

answered May 8, 2015 at 4:30

Ajai Singh
57 ● 2

1

Code is 2 lines only

```
     // Please specify correct Charset
     ReversedLinesFileReader rlf = new ReversedLinesFi
StandardCharsets.UTF_8);
```

```
      // read last 2 lines
      System.out.println(rlf.toString(2));
```

Gradle:

```
implementation group: 'commons-io', name: 'commons-io'
```

Maven:

```
   <dependency>
        <groupId>commons-io</groupId><artifactId>commo
<version>2.11.0</version>
   </dependency>
```

Share Follow

answered Jan 21, 2023 at 23:54

grep
5,623 ● 12 ● 65 ● 114

0

In **C#**, you should be able to set the stream's position:

From: http://bytes.com/groups/net-c/269090-streamreader-read-last-line-text-file

```
using(FileStream fs = File.OpenRead("c:\\file.dat"))
{
    using(StreamReader sr = new StreamReader(fs))
    {
        sr.BaseStream.Position = fs.Length - 4;
        if(sr.ReadToEnd() == "DONE")
            // match
    }
}
```

Share Follow

edited Mar 19, 2013 at 13:43

Eric Leschinski
153k ● 96 ● 421 ● 335

answered Mar 26, 2009 at 15:24

rball
6,955 ● 7 ● 50 ● 78

In Java's FileInputStream (which FileReader is based on), you cannot set the position; you can only skip forward, which probably does not read the parts you skip, but is still a one-way operation and thus not suited to looking for a linebreak at an unknown offset from the end. – Michael Borgwardt Mar 26, 2009 at 15:32

You can use mark() to get around that problem, depending on what the streams markLimit() is. – James Schek Mar 26, 2009 at 16:08

0

To avoid the Unicode problems related to reverting the string (or the StringBuilder), as discussed in Eric Leschinski excellent answer, one can read to a byte list, from the end of the file, revert it to a byte array and then create the String from the byte array.

Below are the changes to Eric Leschinski answer's code, to do it with a byte array. The code changes are below the commented lines of code:

```
static public String tail2(File file, int lines) {
    java.io.RandomAccessFile fileHandler = null;
    try {
        fileHandler = new java.io.RandomAccessFile( fi
            long fileLength = fileHandler.length() - 1;
            //StringBuilder sb = new StringBuilder();
            List<Byte> sb = new ArrayList<>();
            int line = 0;

            for(long filePointer = fileLength; filePointer
                fileHandler.seek( filePointer );
                int readByte = fileHandler.readByte();

                if( readByte == 0xA ) {
                    if (filePointer < fileLength) {
                        line = line + 1;
                    }
                } else if( readByte == 0xD ) {
                    if (filePointer < fileLength-1) {
                        line = line + 1;
                    }
                }
                if (line >= lines) {
                    break;
                }
                //sb.add( (char) readByte );
                sb.add( (byte) readByte );
            }

            //String lastLine = sb.reverse().toString();
            //Revert byte array and create String
            byte[] bytes = new byte[sb.size()];
            for (int i=0; i<sb.size(); i++) bytes[sb.size(
            String lastLine = new String(bytes);
            return lastLine;
        } catch( java.io.FileNotFoundException e ) {
            e.printStackTrace();
            return null;
        } catch( java.io.IOException e ) {
            e.printStackTrace();
            return null;
        }
        finally {
            if (fileHandler != null )
                try {
                    fileHandler.close();
                } catch (IOException e) {
                }
```

```
    }
}
```

Share Follow

edited Jun 3, 2021 at 11:16

answered Jun 3, 2021 at 11:11

Helder Daniel

411 ● 4 ● 10

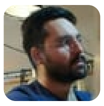
0

I know you asked for java but this is a kotlin version of it (some one might need)

```
    fun getLastLine(storageFolder: File): String {
        // Use RandomAccessFile to read from end
        RandomAccessFile(storageFolder, "r").use { raf
            // Go to end of file minus some buffer (as
            val bufferSize = 200 // Adjust based on yo
            val fileLength = raf.length()
            val startPosition = max(0, fileLength - bu

            raf.seek(startPosition)

            // Read the buffer into string
            val buffer = ByteArray(bufferSize)
            val bytesRead = raf.read(buffer)
            val bufferStr = String(buffer, 0, bytesRea

            // Get last complete line
            return bufferStr.trim().split("\n").last()
        }
    }
```

Share Follow

answered Oct 23 at 17:35



MohammadBaqer
1,376 ● 1 ● 21 ● 57