Edit a specific Line of a Text File in C#

Ask Question

28

10

I have two text files, Source.txt and Target.txt. The source will never be modified and contain N lines of text. So, I want to delete a specific line of text in Target.txt, and replace by an specific line of text from Source.txt, I know what number of line I need, actually is the line number 2, both files.

I haven something like this:

```
string line = string.Empty;
int line_number = 1;
int line_to_edit = 2;

using (StreamReader reader = new StreamReader(@"C:\source.xml"))
{
    using (StreamWriter writer = new StreamWriter(@"C:\target.xml"))
    {
        while ((line = reader.ReadLine()) != null)
        {
            if (line_number == line_to_edit)
            {
                writer.WriteLine(line);
            }

            line_number++;
        }
    }
}
```

But when I open the Writer, the target file get erased, it writes the lines, but, when opened, the target file only contains the copied lines, the rest get lost.

c# stream filestream

edited Sep 30 '15 at 1:38

John Odom
857 2 16 33

asked Dec 28 '09 at 19:07

Luis
143 1 2 4

5 Answers

43

You can't rewrite a line without rewriting the entire file (unless the lines happen to be the same length). If your files are small then reading the entire target file into memory and then writing it out again might make sense. You can do that like this:

```
using System;
using System.IO;

class Program
{
    static void Main(string[] args)
    {
        int line_to_edit = 2; // Warning: 1-based indexing
        string sourceFile = "source.txt";
        string destinationFile = "target.txt";

        // Read the appropriate line from the file.
        string lineToWrite = null;
        using (StreamReader reader = new StreamReader(sour
        {
            for (int i = 1; i <= line_to_edit; ++i)
                lineToWrite = reader.ReadLine();
        }

        if (lineToWrite != null)
```

```
        // Read the old file.
        string[] lines = File.ReadAllLines(destinationFile

        // Write the new file over the old file.
        using (StreamWriter writer = new StreamWriter(dest
        {
            for (int currentLine = 1; currentLine <= lines
            {
                if (currentLine == line_to_edit)
                {
                    writer.WriteLine(lineToWrite);
                }
                else
                {
                    writer.WriteLine(lines[currentLine - 1
                }
            }
        }
    }
}
```

If your files are large it would be better to create a new file so that you can read streaming from one file while you write to the other. This means that you don't need to have the whole file in memory at once. You can do that like this:

```
using System;
using System.IO;

class Program
{
    static void Main(string[] args)
    {
        int line_to_edit = 2;
        string sourceFile = "source.txt";
        string destinationFile = "target.txt";
        string tempFile = "target2.txt";

        // Read the appropriate line from the file.
        string lineToWrite = null;
        using (StreamReader reader = new StreamReader(sour
        {
            for (int i = 1; i <= line_to_edit; ++i)
```

```
            if (lineToWrite == null)
                throw new InvalidDataException("Line does not

            // Read from the target file and write to a new fi
            int line_number = 1;
            string line = null;
            using (StreamReader reader = new StreamReader(dest
            using (StreamWriter writer = new StreamWriter(temp
            {
                while ((line = reader.ReadLine()) != null)
                {
                    if (line_number == line_to_edit)
                    {
                        writer.WriteLine(lineToWrite);
                    }
                    else
                    {
                        writer.WriteLine(line);
                    }
                    line_number++;
                }
            }

            // TODO: Delete the old file and replace it with t
        }
    }
```

You can afterwards move the file once you are sure that the write operation has succeeded (no excecption was thrown and the writer is closed).

Note that in both cases it is a bit confusing that you are using 1-based indexing for your line numbers. It might make more sense in your code to use 0-based indexing. You can have 1-based index in your user interface to your program if you wish, but convert it to a 0-indexed before sending it further.

Also, a disadvantage of directly overwriting the old file with the new file is that if it fails halfway through then you might permanently lose whatever data wasn't written. By writing to a third file first you only delete the original data after

so you can recover the data if the computer crashes halfway through.

A final remark: I noticed that your files had an xml extension. You might want to consider if it makes more sense for you to use an XML parser to modify the contents of the files instead of replacing specific lines.

edited Aug 22 '12 at 2:44

Matty Bear
282 1 2 12

answered Dec 28 '09 at 19:17

Mark Byers
605k 131 1375 1351

34

the easiest way is :

```
static void lineChanger(string newText, string fileName, i
{
     string[] arrLine = File.ReadAllLines(fileName);
     arrLine[line_to_edit - 1] = newText;
     File.WriteAllLines(fileName, arrLine);
}
```

usage :

```
lineChanger("new content for this line" , "sample.text" ,
```

edited Apr 8 '16 at 8:19

legrojan
352 8 16


441 4 3

Absolutely Brilliant Answer!!!!... totally working here. running in dot net 2.0... in Unity3d it's : ___ function lineChanger(newText: String, fileName :String, line_to_edit: int) { var arrLine : String[] = File.ReadAllLines(fileName); arrLine[line_to_edit - 1] = newText; File.WriteAllLines(fileName, arrLine); } – com.prehensible Mar 19 '16 at 12:09

This has 1 problem, that it rewrites the whole file. – Mayer Spitzer Jul 23 '17 at 18:20

this also loads the entire file into memory all at once. but it is probably the easiest approach and still usable with small files. – Dave Cousineau Mar 1 '18 at 18:37

why wouldn't I just create a new file with the same name as the original and save it over top? – SJ10 Feb 14 at 17:16

3

When you create a `StreamWriter` it always create a file from scratch, you will have to create a third file and copy from target and replace what you need, and then replace the old one. But as I can see what you need is XML manipulation, you might want to use `XmlDocument` and modify your file using Xpath.

edited Apr 14 '11 at 23:32


manojlds
220k 47 388 371

answered Dec 28 '09 at 19:15


Pablo Retyk
4,042 6 34 58

2

the output file.

```
StreamWriter stm = null;
fi = new FileInfo(@"C:\target.xml");
if (fi.Exists)
   stm = fi.OpenWrite();
```

Of course, you will still have to seek to the correct line in the output file, which will be hard since you can't read from it, so unless you already KNOW the byte offset to seek to, you probably really want read/write access.

```
FileStream stm = fi.Open(FileMode.OpenOrCreate, FileAccess
```

with this stream, you can read until you get to the point where you want to make changes, then write. Keep in mind that you are writing bytes, not lines, so to overwrite a line you will need to write the same number of characters as the line you want to change.

answered Dec 28 '09 at 19:43

John Knoeller
28.5k 3 45 84

2

I guess the below should work (instead of the writer part from your example). I'm unfortunately with no build environment so It's from memory but I hope it helps

```
using (var fs = File.Open(filePath, FileMode.Open, FileAcc
        {
            var destinationReader = StreamReader(fs);
            var writer = StreamWriter(fs);
            while ((line = reader.ReadLine()) != null)
            {
```

```
            }
            else
            {
                destinationReader .ReadLine();
            }
            line_number++;
        }
    }
```

edited Dec 28 '09 at 20:25

answered Dec 28 '09 at 19:42

Rune FS

18.5k 6 50 88