A little program to look for files with inconsistent line endings

devblogs.microsoft.com/oldnewthing/20190422-00

April 22, 2019

Raymond Chen

I wrote this little program to look for files with inconsistent line endings. Maybe you'll find it useful. Probably not, but I'm posting it anyway.

```
using System;
using System.Collections.Generic;
using System.IO;

class Program
{
    static IEnumerable<FileInfo> EnumerateFiles(string dir)
    {
        var info = new System.IO.DirectoryInfo(dir);
        foreach (var f in info.EnumerateFileSystemInfos(
                           "*.*", SearchOption.TopDirectoryOnly))
        {
            if (f.Attributes.HasFlag(FileAttributes.Hidden))
            {
                continue;
            }

            if (f.Attributes.HasFlag(FileAttributes.Directory))
            {
                switch (f.Name.ToLower())
                {
                    case "bin":
                    case "obj":
                        continue;
                }

                foreach (var inner in EnumerateFiles(f.FullName))
                {
                    yield return inner;
                }
            }
            else
            {
                yield return (FileInfo)f;
            }
        }
    }

    // Starting in the current directory, enumerate files
    // (see EnumerateFiles for criteria), and report what
    // type of line ending each file uses.

    static void Main()
    {
        foreach (var f in EnumerateFiles("."))
        {
            // Skip obvious binary files.
            switch (f.Extension.ToLower())
            {
                case ".png":
                case ".jpg":
                case ".gif":
```

```
    case ".wmv":
        continue;
}

int line = 0; // total number of lines found
int cr = 0;   // number of lines that end in CR
int crlf = 0; // number of lines that end in CRLF
int lf = 0;   // number of lines that end in LF

var stream = new FileStream(
         f.FullName, FileMode.Open, FileAccess.Read);
using (var br = new BinaryReader(stream))
{
    // Slurp the entire file into memory.
    var bytes = br.ReadBytes((int)f.Length);
    for (int i = 0; i < bytes.Length; i++)
    {
        if (bytes[i] == '\r')
        {
            if (i + 1 < bytes.Length &&
                bytes[i+1] == '\n')
            {
                line++;
                crlf++;
                i++;
            }
            else
            {
                line++;
                cr++;
            }
        }
        else if (bytes[i] == '\n')
        {
            lf++;
            line++;
        }
    }
}

if (cr == line)
{
    Console.WriteLine("{0}, {1}", f.FullName, "CR");
}
else if (crlf == line)
{
    Console.WriteLine("{0}, {1}", f.FullName, "CRLF");
}
else if (lf == line)
{
    Console.WriteLine("{0}, {1}", f.FullName, "LF");
}
```

```
            else
            {
                Console.WriteLine("{0}, {1}, {2}, {3}, {4}",
                              f.FullName, "Mixed", cr, lf, crlf);
            }
        }
    }
}
```

The `EnumerateFiles` method recursively enumerates the contents of the directory, but skips over hidden files, hidden directories, and directories with specific names.

The main program takes the files enumerated by `EnumerateFiles`, ignores certain known binary file types, and for the remaining files, counts the number of lines and how many of them use any particular line terminator.

If the file's lines all end the same way, then that line terminator is reported with the file name. Otherwise, the file is reported as *Mixed* and the number of lines of each type is reported.

I use this little program when chasing down line terminator inconsistencies. Maybe that's not something you have to deal with, in which case lucky you.

Raymond Chen

Follow