

## Untitled

**GZipStream class to compress and decompress data. The following code example creates a file (test.txt) in the current directory, fills it with text, and displays the contents of the file to the console. The code then uses the GZipStream class to create a compressed version of the file (test.txt.gz) and compares the size of the two files. Finally, the code reads in the compressed file, decompresses it, and writes out a new file (test.txt.gz.txt) to the current directory. The code then displays the contents of the decompressed file.**

**You can also use the DeflateStream class to compress and decompress data.**

```
-----  
-----  
  
using System;  
using System.Collections.Generic;  
using System.IO;  
using System.IO.Compression;  
  
public class CompressionSnippet
```

## Untitled

```
{  
    public static void Main()  
    {  
        string path = "test.txt";  
  
        // Create the text file if it doesn't already  
exist.  
        if (!File.Exists(path))  
        {  
            Console.WriteLine("Creating a new  
test.txt file");  
            string[] text = new string[] {"This is a test  
text file.",  
                "This file will be compressed and  
written to the disk.",  
                "Once the file is written, it can be  
decompressed",  
                "using various compression tools.",  
                "The GZipStream and DeflateStream  
class use the same",  
                "compression algorithms, the primary  
difference is that",  
                "the GZipStream class includes a  
cyclic redundancy check",  
                "that can be useful for detecting data  
        }  
    }  
}
```

## Untitled

corruption.",

"One other side note: both the  
GZipStream and DeflateStream",

"classes operate on streams as  
opposed to file-based",

"compression; data is read on a  
byte-by-byte basis, so it",

"is not possible to perform multiple  
passes to determine the",

"best compression method. Already  
compressed data can actually",

"increase in size if compressed with  
these classes."};

```
File.WriteAllLines(path, text);  
}
```

```
Console.WriteLine("Contents of {0}", path);  
Console.WriteLine(File.ReadAllText(path));
```

```
CompressFile(path);  
Console.WriteLine();
```

```
UncompressFile(path + ".gz");  
Console.WriteLine();
```

## Untitled

```
    Console.WriteLine("Contents of {0}", path +  
".gz.txt");  
    Console.WriteLine(File.ReadAllText(path +  
".gz.txt"));  
  
}  
  
public static void CompressFile(string path)  
{  
    FileStream sourceFile =  
File.OpenRead(path);  
    FileStream destinationFile =  
File.Create(path + ".gz");  
  
    byte[] buffer = new byte[sourceFile.Length];  
    sourceFile.Read(buffer, 0, buffer.Length);  
  
    using (GZipStream output = new  
GZipStream(destinationFile,  
        CompressionMode.Compress))  
    {  
        Console.WriteLine("Compressing {0} to  
{1}.", sourceFile.Name,  
            destinationFile.Name, false);  
    }  
}
```

## Untitled

```
    output.Write(buffer, 0, buffer.Length);  
}
```

```
    // Close the files.  
    sourceFile.Close();  
    destinationFile.Close();  
}
```

```
public static void UncompressFile(string path)  
{  
    FileStream sourceFile =  
File.OpenRead(path);  
    FileStream destinationFile =  
File.Create(path + ".txt");  
  
    // Because the uncompressed size of the  
file is unknown,  
    // we are using an arbitrary buffer size.  
    byte[] buffer = new byte[4096];  
    int n;  
  
    using (GZipStream input = new  
GZipStream(sourceFile,  
            CompressionMode.Decompress, false))
```

## Untitled

```
{  
    Console.WriteLine("Decompressing {0} to  
{1}.", sourceFile.Name,  
        destinationFile.Name);  
  
    n = input.Read(buffer, 0, buffer.Length);  
    destinationFile.Write(buffer, 0, n);  
}  
  
// Close the files.  
sourceFile.Close();  
destinationFile.Close();  
}  
}
```