**串联多个字符串（C# 编程指南）**

串联是将一个字符串追加到另一个字符串末尾的过程。使用 **+** 运算符串联字符串文本或字符串常量时，编译器会创建一个字符串。串联不在运行时发生。但字符串变量只能在运行时串联，对此，您应该了解各种方法的性能含义。

下面的示例演示如何将一个长字符串拆分为几个较短的字符串，从而提高源代码的可读性。这些较短的字符串将在编译时串联成一个字符串。无论涉及到多少个字符串，都不会有运行时性能开销。

static void Main()

{

// Concatenation of literals is performed at compile time, not run time.

string text = "Historically, the world of data and the world of objects " +

"have not been well integrated. Programmers work in C# or Visual Basic " +

"and also in SQL or XQuery. On the one side are concepts such as classes, " +

"objects, fields, inheritance, and .NET Framework APIs. On the other side " +

"are tables, columns, rows, nodes, and separate languages for dealing with " +

"them. Data types often require translation between the two worlds; there are " +

"different standard functions. Because the object world has no notion of query, a " +

"query can only be represented as a string without compile-time type checking or " +

"IntelliSense support in the IDE. Transferring data from SQL tables or XML trees to " +

"objects in memory is often tedious and error-prone.";

Console.WriteLine(text);

}

若要串联字符串变量，可以使用 **+** 或 **+=** 运算符，也可以使用 [String.Concat](https://msdn.microsoft.com/zh-cn/library/system.string.concat.aspx)、[String.Format](https://msdn.microsoft.com/zh-cn/library/system.string.format.aspx) 或 [StringBuilder.Append](https://msdn.microsoft.com/zh-cn/library/system.text.stringbuilder.append.aspx) 方法。 **+** 运算符容易使用，且有利于提高代码的直观性。即使在一条语句中使用多个 + 运算符，字符串内容也将只复制一次。但是，如果重复此操作多次（如使用循环），则可能会导致出现效率问题。例如，考虑下面的代码：

static void Main(string[] args)

{

// To run this program, provide a command line string.

// In Visual Studio, see Project > Properties > Debug.

string userName = args[0];

string date = DateTime.Today.ToShortDateString();

// Use the + and += operators for one-time concatenations.

string str = "Hello " + userName + ". Today is " + date + ".";

System.Console.WriteLine(str);

str += " How are you today?";

System.Console.WriteLine(str);

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

// Example output:

// Hello Alexander. Today is 1/22/2008.

// Hello Alexander. Today is 1/22/2008\. How are you today?

// Press any key to exit.

//

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 在字符串串联操作中，C# 编译器对 null 字符串和空字符串进行相同的处理，但它不转换原始 null 字符串的值。 |

如果您串联的字符串数量不那么巨大（例如，在循环中），那么这些代码的性能成本可能不会很高。上述情况同样适用于 [String.Concat](https://msdn.microsoft.com/zh-cn/library/system.string.concat.aspx) 和 [String.Format](https://msdn.microsoft.com/zh-cn/library/system.string.format.aspx) 方法。

但如果性能的优劣很重要，则应该总是使用 [StringBuilder](https://msdn.microsoft.com/zh-cn/library/system.text.stringbuilder.aspx) 类来串联字符串。下面的代码使用 [StringBuilder](https://msdn.microsoft.com/zh-cn/library/system.text.stringbuilder.aspx) 类的 [Append](https://msdn.microsoft.com/zh-cn/library/system.text.stringbuilder.append.aspx) 方法来串联字符串，因此不会有 **+** 运算符的链接作用产生。

class StringBuilderTest

{

static void Main()

{

string text = null;

// Use StringBuilder for concatenation in tight loops.

System.Text.StringBuilder sb = new System.Text.StringBuilder();

for (int i = 0; i < 100; i++)

{

sb.AppendLine(i.ToString());

}

System.Console.WriteLine(sb.ToString());

// Keep the console window open in debug mode.

System.Console.WriteLine("Press any key to exit.");

System.Console.ReadKey();

}

}

// Output:

// 0

// 1

// 2

// 3

// 4

// ...

//