  内存映射文件是由一个文件到进程地址空间的映射。

        C#提供了允许应用程序把文件映射到一个进程的函(**MemoryMappedFile.CreateOrOpen**)。内存映射文件与虚拟内存有些类似，通过内存映射文件可以保留一个地址空间的区域，同时将物理存储器提交给此区域，只是内存文件映射的物理存储器来自一个已经存在于磁盘上的文件，而非系统的页文件，而且在对该文件进行操作之前必须首先对文件进行映射，就如同将整个文件从磁盘加载到内存。由此可以看出，使用内存映射文件处理存储于磁盘上的文件时，将不必再对文件执行I/O操作，这意味着在对文件进行处理时将不必再为文件申请并分配缓存，所有的文件缓存操作均由系统直接管理，由于取消了将文件数据加载到内存、数据从内存到文件的回写以及释放内存块等步骤，使得内存映射文件在处理大数据量的文件时能起到相当重要的作用。另外，实际工程中的系统往往需要在多个进程之间共享数据，如果数据量小，处理方法是灵活多变的，如果共享数据容量巨大，那么就需要借助于内存映射文件来进行。实际上，内存映射文件正是解决本地多个进程间数据共享的最有效方法。

        共享内存是内存映射文件的一种特殊情况，内存映射的是一块内存，而非磁盘上的文件。共享内存的主语是进程（Process），操作系统默认会给每一个进程分配一个内存空间，每一个进程只允许访问操作系统分配给它的哪一段内存，而不能访问其他进程的。而有时候需要在不同进程之间访问同一段内存，怎么办呢？操作系统给出了创建访问共享内存的API，需要共享内存的进程可以通过这一组定义好的API来访问多个进程之间共有的内存，各个进程访问这一段内存就像访问一个硬盘上的文件一样。而.Net 4.0中引入了**System.IO.MemoryMappedFiles**命名空间，这个命名空间的类对windows 共享内存相关API做了封装，使.Net程序员可以更方便的使用内存映射文件。

**内存映射文件实现进程间通讯**

        内存映射文件是实现进程通讯的又一种方法，我们可以通过共享剪贴板、共享物理文件来实现进程间的数据共享，这里我们还可以通过内存映射文件来实现共享，这样，文件内的数据就可以用内存读/写指令来访问，而不是用ReadFile和WriteFile这样的I/O系统函数，从而提高了文件存取速度。这种方式更加快捷高效，最适用于需要读取文件并且对文件内包含的信息做语法分析的应用程序，如：对输入文件进行语法分析的彩色语法编辑器，编译器等。这种数据共享是让两个或多个进程映射同一文件映射对象的视图，即它们在共享同一物理存储页。这样，当一个进程向内存映射文件的一个视图写入数据时，其他的进程立即在自己的视图中看到变化。

注意：

        对文件映射对象要使用同一名字。

        是让两个或多个进程映射同一文件映射对象的视图，即它们在共享同一物理存储页。这样，当一个进程向内存映射文件的一个视图写入数据时，其他的进程立即在自己的视图中看到变化。但要注意，对文件映射对象要使用同一名字。

**内存映射文件使用实例：**

1.      在同一进程内同时读写同一内存映射文件



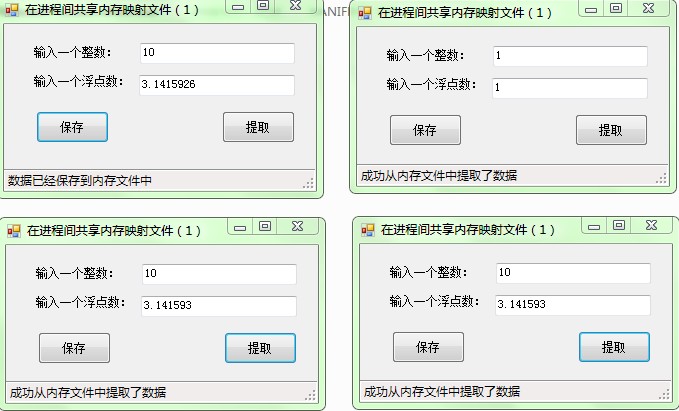
**[csharp]** [view plain](http://blog.csdn.net/onejune2013/article/details/7577152) [copy](http://blog.csdn.net/onejune2013/article/details/7577152)

[print?](http://blog.csdn.net/onejune2013/article/details/7577152)

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.ComponentModel;
4. **using** System.Data;
5. **using** System.Drawing;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Windows.Forms;
9. **using** System.IO;
10. **using** System.IO.MemoryMappedFiles;

13. **namespace** UseMMFInProcess
14. {
15. **public** partial **class** frmMain : Form
16. {
17. **public** frmMain()
18. {
19. InitializeComponent();
20. CreateMemoryMapFile();
21. }
22. **private** **const** **int** FILE\_SIZE = 512;
23. /// <summary>
24. /// 引用内存映射文件
25. /// </summary>
26. **private** MemoryMappedFile memoryFile = **null**;
27. /// <summary>
28. /// 用于访问内存映射文件的存取对象
29. /// </summary>
30. **private** MemoryMappedViewAccessor accessor1, accessor2,accessor;
31. /// <summary>
32. /// 创建内存映射文件
33. /// </summary>
34. **private** **void** CreateMemoryMapFile()
35. {
36. **try**
37. {
38. memoryFile = MemoryMappedFile.CreateFromFile("MyFile.dat", FileMode.OpenOrCreate, "MyFile", FILE\_SIZE);
39. //访问文件前半段
40. accessor1 = memoryFile.CreateViewAccessor(0, FILE\_SIZE / 2);
41. //访问文件后半段
42. accessor2 = memoryFile.CreateViewAccessor(FILE\_SIZE / 2, FILE\_SIZE / 2);
43. //访问全部文件
44. accessor = memoryFile.CreateViewAccessor();
45. //InitFileContent();
46. lblInfo.Text = "内存文件创建成功";
47. ShowFileContents();
48. }
49. **catch** (Exception ex)
50. {
51. lblInfo.Text = ex.Message;
52. }
53. }
54. /// <summary>
55. /// 关闭并释放资源
56. /// </summary>
57. **private** **void** DisposeMemoryMapFile()
58. {
59. **if** (accessor1 != **null**)
60. accessor1.Dispose();
61. **if** (accessor2 != **null**)
62. accessor2.Dispose();
63. **if** (memoryFile != **null**)
64. memoryFile.Dispose();
65. }
67. **private** **void** frmMain\_FormClosing(**object** sender, FormClosingEventArgs e)
68. {
69. DisposeMemoryMapFile();
70. }
72. **private** **void** btnWrite1\_Click(**object** sender, EventArgs e)
73. {
74. **if** (textBox1.Text.Length == 0)
75. {
76. lblInfo.Text = "请输入一个字符";
77. **return**;
78. }
79. **char**[] chs = textBox1.Text.ToCharArray();
80. **char** ch = chs[0];
82. **for** (**int** i = 0; i < FILE\_SIZE / 2; i += 2)
83. accessor1.Write(i, ch);
85. lblInfo.Text = "字符“" + ch + "”已写到文件前半部份";
86. ShowFileContents();
87. }
89. **private** **void** btnShow\_Click(**object** sender, EventArgs e)
90. {
91. ShowFileContents();
92. }
94. /// <summary>
95. /// 初始化文件内容为可视的字符“0”
96. /// </summary>
97. **private** **void** InitFileContent()
98. {
99. **for** (**int** i = 0; i < FILE\_SIZE; i += 2)
100. accessor.Write(i,'0');
101. }
102. /// <summary>
103. /// 显示文件内容
104. /// </summary>
105. **private** **void** ShowFileContents()
106. {
107. StringBuilder sb = **new** StringBuilder(FILE\_SIZE);
108. sb.Append("上半段内容：\n");
110. **int** j = 0;
111. **for** (**int** i = 0; i < FILE\_SIZE / 2; i += 2)
112. {
113. sb.Append("\t");
114. **char** ch = accessor.ReadChar(i);
115. sb.Append(j);
116. sb.Append(":");
117. sb.Append(ch);
118. j++;
119. }
120. sb.Append("\n下半段内容：\n");
122. **for** (**int** i = FILE\_SIZE / 2; i < FILE\_SIZE; i += 2)
123. {
124. sb.Append("\t");
125. **char** ch = accessor.ReadChar(i);
126. sb.Append(j);
127. sb.Append(":");
128. sb.Append(ch);
129. j++;
130. }
131. richTextBox1.Text = sb.ToString();
132. }
134. **private** **void** btnWrite2\_Click(**object** sender, EventArgs e)
135. {
136. **if** (textBox2.Text.Length == 0)
137. {
138. lblInfo.Text = "请输入一个字符";
139. **return**;
140. }
141. **char**[] chs = textBox2.Text.ToCharArray();
142. **char** ch = chs[0];
144. **for** (**int** i = 0; i < FILE\_SIZE / 2; i += 2)
145. accessor2.Write(i, ch);
146. lblInfo.Text = "字符“" + ch + "”已写到文件后半部份";
147. ShowFileContents();
148. }
149. }
150. }

2.      使用内存映射文件在进程间传送值类型数据



**[csharp]** [view plain](http://blog.csdn.net/onejune2013/article/details/7577152) [copy](http://blog.csdn.net/onejune2013/article/details/7577152)

[print?](http://blog.csdn.net/onejune2013/article/details/7577152)

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Text;
6. **namespace** UseMMFBetweenProcess
7. {
8. /// <summary>
9. /// 要共享的数据结构，注意，其成员不能是引用类型
10. /// </summary>
11. **public** **struct** MyStructure
12. {
13. **public** **int** IntValue
14. {
15. **get**;
16. **set**;
17. }
18. **public** **float** FloatValue
19. {
20. **get**;
21. **set**;
22. }
23. }
24. }

27. **using** System;
28. **using** System.Collections.Generic;
29. **using** System.ComponentModel;
30. **using** System.Data;
31. **using** System.Drawing;
32. **using** System.Linq;
33. **using** System.Text;
34. **using** System.Windows.Forms;
35. **using** System.IO.MemoryMappedFiles;
36. **using** System.IO;
38. **namespace** UseMMFBetweenProcess
39. {
40. **public** partial **class** frmMain : Form
41. {
42. **public** frmMain()
43. {
44. InitializeComponent();
45. InitMemoryMappedFile();
46. }
48. /// <summary>
49. /// 内存映射文件的容量
50. /// </summary>
51. **private** **const** **int** FileSize = 1024 \* 1024;
52. **private** MemoryMappedFile file = **null**;
53. **private** MemoryMappedViewAccessor accessor = **null**;
55. /// <summary>
56. /// 初始化内存映射文件
57. /// </summary>
58. **private** **void** InitMemoryMappedFile()
59. {
60. file = MemoryMappedFile.CreateOrOpen("UseMMFBetweenProcess", FileSize);
61. accessor = file.CreateViewAccessor();
62. lblInfo.Text = "内存文件创建或连接成功";
63. }
65. /// <summary>
66. /// 要共享的数据对象
67. /// </summary>
68. **private** MyStructure data;
70. /// <summary>
71. /// 显示数据到窗体上
72. /// </summary>
73. **private** **void** ShowData()
74. {
75. textBox1.Text = data.IntValue.ToString();
76. textBox2.Text = data.FloatValue.ToString();
77. }
79. /// <summary>
80. /// 根据用户输入更新数据
81. /// </summary>
82. **private** **void** UpdateData()
83. {
84. data.IntValue = **int**.Parse(textBox1.Text);
85. data.FloatValue = **float**.Parse(textBox2.Text);
86. }
88. **private** **void** btnSave\_Click(**object** sender, EventArgs e)
89. {
90. **try**
91. {
92. UpdateData();
93. accessor.Write<MyStructure>(0, **ref** data);
94. lblInfo.Text = "数据已经保存到内存文件中";
95. }
96. **catch** (Exception ex)
97. {
98. lblInfo.Text = ex.Message;
99. }
100. }
102. **private** **void** btnLoad\_Click(**object** sender, EventArgs e)
103. {
104. accessor.Read<MyStructure>(0, **out** data);
105. ShowData();
106. lblInfo.Text = "成功从内存文件中提取了数据";
107. }
109. **private** **void** frmMain\_FormClosing(**object** sender, FormClosingEventArgs e)
110. {
111. **if** (accessor != **null**)
112. accessor.Dispose();
113. **if** (file != **null**)
114. file.Dispose();
115. }
116. }
117. }

3.      利用序列化技术通过内存映射文件实现进程通讯



**[csharp]** [view plain](http://blog.csdn.net/onejune2013/article/details/7577152) [copy](http://blog.csdn.net/onejune2013/article/details/7577152)

[print?](http://blog.csdn.net/onejune2013/article/details/7577152)

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.ComponentModel;
4. **using** System.Data;
5. **using** System.Drawing;
6. **using** System.Linq;
7. **using** System.Text;
8. **using** System.Windows.Forms;
9. **using** System.IO;
10. **using** System.IO.MemoryMappedFiles;
11. **using** System.Runtime.Serialization;
12. **using** System.Runtime.Serialization.Formatters.Binary;
14. **namespace** UseMMFBetweenProcess2
15. {
16. **public** partial **class** frmMain : Form
17. {
18. **public** frmMain()
19. {
20. InitializeComponent();
21. InitMemoryMappedFile();
22. }
24. /// <summary>
25. /// 图片
26. /// </summary>
27. **private** Image bmp
28. {
29. **get**
30. {
31. **return** pictureBox1.Image;
32. }
33. **set**
34. {
35. pictureBox1.Image = value;
36. }
37. }
39. /// <summary>
40. /// 图片说明
41. /// </summary>
42. **private** **string** info
43. {
44. **get**
45. {
46. **return** txtImageInfo.Text;
47. }
48. **set**
49. {
50. txtImageInfo.Text = value;
51. }
52. }
54. **private** MemoryMappedFile memoryFile = **null**;
56. **private** MemoryMappedViewStream stream = **null**;
58. /// <summary>
59. /// 最大容量:10M
60. /// </summary>
61. **private** **const** **int** FileSize = 1024 \* 1024 \* 10;
63. /// <summary>
64. /// 创建内存映射文件,获取其读写流
65. /// </summary>
66. **private** **void** InitMemoryMappedFile()
67. {
68. **try**
69. {
70. memoryFile = MemoryMappedFile.CreateOrOpen("UseMMFBetweenProcess2", FileSize);
71. stream = memoryFile.CreateViewStream();
72. }
73. **catch** (Exception ex )
74. {
75. MessageBox.Show(ex.Message);
76. Close();
77. }
78. }
79. /// <summary>
80. /// 释放相关资源
81. /// </summary>
82. **private** **void** DisposeMemoryMappedFile()
83. {
84. **if** (stream != **null**)
85. stream.Close();
86. **if** (memoryFile != **null**)
87. memoryFile.Dispose();
88. }
90. **private** **void** btnLoadPic\_Click(**object** sender, EventArgs e)
91. {
92. ChooseImageFile();
93. }
95. //选择图片
96. **private** **void** ChooseImageFile()
97. {
98. **if** (openFileDialog1.ShowDialog() == DialogResult.OK)
99. {
100. bmp = **new** Bitmap(openFileDialog1.FileName);
101. }
102. }
103. //根据用户设定的信息创建对象
104. **private** MyPic CreateMyPicObj()
105. {
106. MyPic obj = **new** MyPic();
107. obj.pic = bmp;
108. obj.picInfo = info;
109. **return** obj;
110. }
112. /// <summary>
113. /// 将MyPic对象保存到内存映射文件中
114. /// </summary>
115. **private** **void** SaveToMMF()
116. {
117. **try**
118. {
119. MyPic obj = CreateMyPicObj();
120. IFormatter formatter = **new** BinaryFormatter();
121. stream.Seek(0, SeekOrigin.Begin);
122. formatter.Serialize(stream, obj);
123. MessageBox.Show("对象已保存到内存映射文件中");
124. }
125. **catch** (Exception ex)
126. {
127. MessageBox.Show(ex.Message);
128. }
129. }
131. **private** **void** LoadFromMMF()
132. {
133. **try**
134. {
135. // CreateMyPicObj();
136. IFormatter formatter = **new** BinaryFormatter();
137. stream.Seek(0, SeekOrigin.Begin);
138. MyPic obj =   formatter.Deserialize(stream) **as** MyPic;
139. **if** (obj != **null**)
140. {
141. bmp = obj.pic;
142. info = obj.picInfo;
143. }
144. }
145. **catch** (Exception ex)
146. {
147. MessageBox.Show(ex.Message);
148. }
149. }
151. **private** **void** btnExit\_Click(**object** sender, EventArgs e)
152. {
153. Close();
154. }
156. **private** **void** frmMain\_FormClosing(**object** sender, FormClosingEventArgs e)
157. {
158. DisposeMemoryMappedFile();
159. }
161. **private** **void** btnSaveToMMF\_Click(**object** sender, EventArgs e)
162. {
163. SaveToMMF();
164. }
166. **private** **void** btnLoadFromMMF\_Click(**object** sender, EventArgs e)
167. {
168. LoadFromMMF();
169. }
170. }
171. }