**Windows上C++使用命名管道进行进程间通讯**

命名管道是通过网络来完成进程间的通信 , 它屏蔽了底层的网络协议细节 ; 我们在不了解网络协议的情况下 , 也可以利用命名管道来实现进程间的通信 ; 与 Socket 网络通信相比 , 命名管道不再需要编写身份验证的代码 ; 将命名管道作为一种网络编程方案时 , 它实际上建立了一个 C/S 通信体系 , 并在其中可靠的传输数据 ; 命名管道服务器和客户机的区别在于 : 服务器是唯一一个有权创建命名管道的进程 , 也只有它能接受管道客户机的连接请求 , 而客户机只能同一个现成的命名管道服务器建立连接 ; 示例代码如下 :

**服务端代码 :**

#include "stdafx.h"

#include "windows.h"

#include <iostream>

using namespace std;

#define BUF\_SIZE 4096

// 定义管道名 , 如果是跨网络通信 , 则在圆点处指定服务器端程序所在的主机名

#define EXAMP\_PIPE L"\\\\.\\pipe\\ReadPipe"

int \_tmain(int argc, TCHAR\* argv[], TCHAR\* envp[])

{

// 创建命名管道

HANDLE hPipe = NULL;

hPipe = CreateNamedPipe(

EXAMP\_PIPE,

PIPE\_ACCESS\_DUPLEX,

PIPE\_TYPE\_MESSAGE |

PIPE\_READMODE\_MESSAGE |

PIPE\_WAIT,

PIPE\_UNLIMITED\_INSTANCES,

BUF\_SIZE,

BUF\_SIZE,

0,

NULL);

if (hPipe==INVALID\_HANDLE\_VALUE)

{

cout<<"Create Read Pipe Error"<<endl;

return FALSE;

}

// 等待客户端的连接

if (!ConnectNamedPipe(hPipe, NULL))

{

cout<<"Connect Failed"<<endl;

return FALSE;

}

DWORD dwReturn = 0;

char szBuffer[BUF\_SIZE] = {0};

// 向客户端发送数据

cin>>szBuffer;

if (!WriteFile(hPipe,szBuffer,strlen(szBuffer),&dwReturn,NULL))

{

cout<<"Write Failed"<<endl;

}

// 读取客户端数据

memset(szBuffer, 0, BUF\_SIZE);

if (ReadFile(hPipe,szBuffer,BUF\_SIZE,&dwReturn,NULL))

{

szBuffer[dwReturn] = '\0';

cout<<szBuffer<<endl;

}

else

{

cout<<"Read Failed"<<endl;

}

DisconnectNamedPipe(hPipe);

CloseHandle(hPipe);

return 0;

}

https://csdnimg.cn/release/blogv2/dist/pc/img/newCodeMoreWhite.png

**客户端代码 :**

#include "stdafx.h"

#include "windows.h"

#include <iostream>

using namespace std;

#define BUF\_SIZE 4096

// 定义管道名 , 如果是跨网络通信 , 则在圆点处指定服务器端程序所在的主机名

#define EXAMP\_PIPE L"\\\\.\\pipe\\ReadPipe"

int \_tmain(int argc, TCHAR\* argv[], TCHAR\* envp[])

{

HANDLE hPipe = NULL;

char szBuffer[BUF\_SIZE] = {0};

DWORD dwReturn = 0;

// 判断是否有可以利用的命名管道

if (!WaitNamedPipe(EXAMP\_PIPE, NMPWAIT\_USE\_DEFAULT\_WAIT))

{

cout<<"No Read Pipe Accessible"<<endl;

return 0;

}

// 打开可用的命名管道 , 并与服务器端进程进行通信

hPipe = CreateFile(EXAMP\_PIPE,GENERIC\_READ | GENERIC\_WRITE,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,

NULL,OPEN\_EXISTING,0,NULL);

if (hPipe==INVALID\_HANDLE\_VALUE)

{

cout<<"Open Read Pipe Error"<<endl;

return 0;

}

// 读取服务端发来的数据

if (ReadFile(hPipe,szBuffer,BUF\_SIZE,&dwReturn,NULL))

{

szBuffer[dwReturn] = '\0';

cout<<szBuffer<<endl;

}

else

{

cout<<"Read Failed"<<endl;

}

// 向服务端发送数据

memset(szBuffer, 0, BUF\_SIZE);

cin>>szBuffer;

if (!WriteFile(hPipe,szBuffer,strlen(szBuffer),&dwReturn,NULL))

{

cout<<"Write Failed"<<endl;

}

CloseHandle(hPipe);

return 0;

}

https://csdnimg.cn/release/blogv2/dist/pc/img/newCodeMoreWhite.png

*个人网站:*[*Github*](https://github.com/tojohnonly)*, 欢迎点击给星*