**山东师范大学**

**实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 院系部所 | 信息科学与工程学院 | | 课程名称 | | 计算机操作系统 |
| 所在班级 | 计工本1702 | 学号 | 201711010202 | 姓名 | 王汝芸 |
| 实验编号 | 04 | 题目 | 实验四：使用命名管道实现进程通信 | | |

一、实验目的和要求

（1）了解Windows系统环境下的进程通信机制。

（2）熟悉Windows系统提供的进程通信API。

二、实验环境

VC++6.0

Windows 10

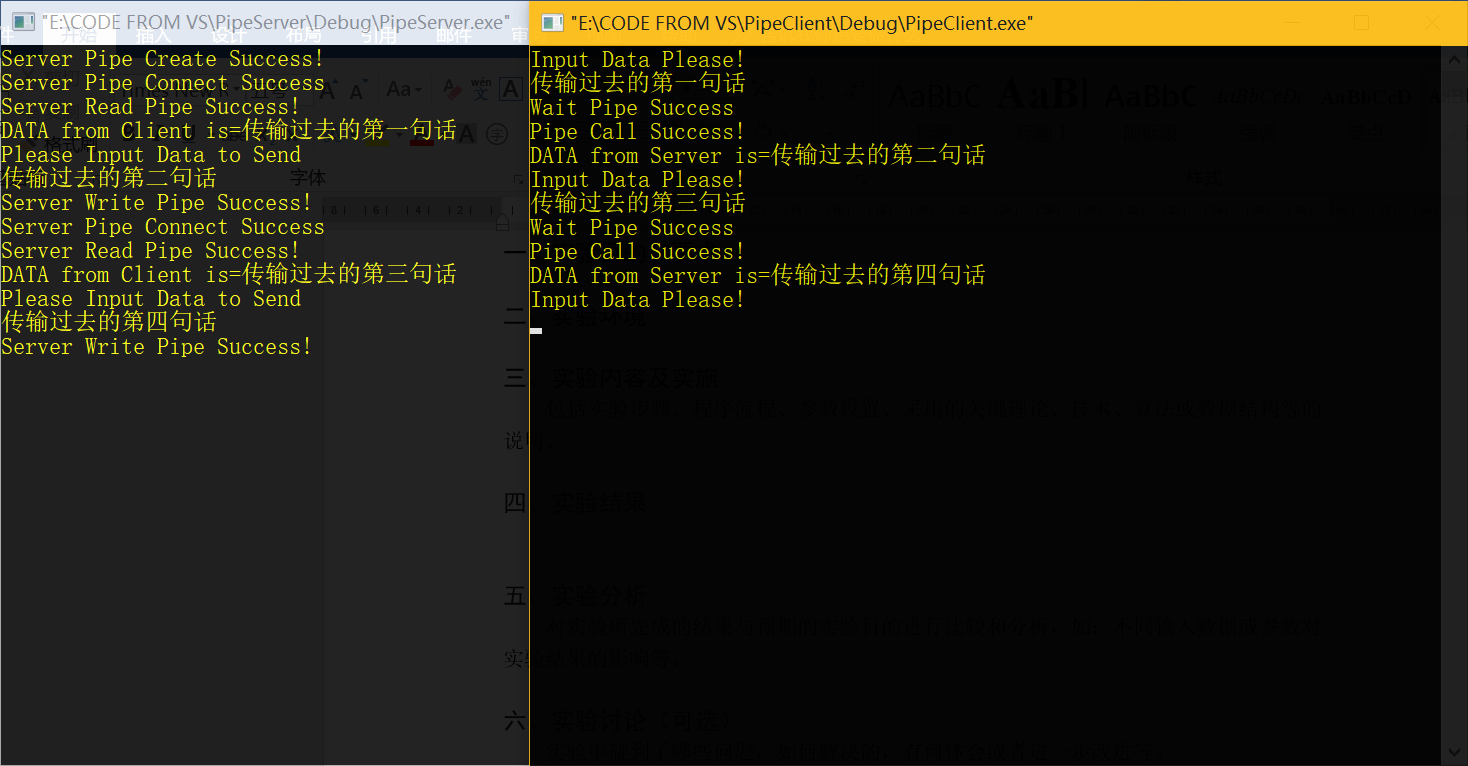
三、实验内容及实施

实验内容：

使用命名管道完成两个进程之间的通信。

实验指导：

完成两个进程之间的通信，需要建立两个工程文件，在Microsoft Visual C++ 6.0环境下建立服务器工程文件PipeServer和客户端工程文件PipeClient。在服务器程序中，首先使用CreateNamedPipe()创建一个命名管道，之后使用ConnectNamedPipe()连接命名管道，如果命名管道连接成功，可以使用读文件函数ReadFile()从命名管道中读数据，并可使用写文件函数WriteFile()向命名管道中写数据，命名管道使用完毕后可以使用DisconnextNamedPipe()拆除与命名管道的连接。在客户端程序中，可以先使用WaitNamedPipe()等待服务器建立好命名管道，然后使用CallNamedPipe()与服务器建立命名管道的连接，并同时得到服务器发来的数据或向服务器发送数据。

四、实验结果

五、实验分析

该实验完成了两个进程的通信。首先在客户端输入数据，按Enter键后可以看到客户端输入的数据已经在服务器端显示出来了。同样在服务器端输入数据，在客户端也会显示出来，这说明建立的命名管道已经做到了双向通信。

六、实验讨论（可选）

一开始用VS，建立两个工程但是并不能出结果，总是显示err。后来用VC++6.0运行，在项目默认的.cpp文件下实现目标结果。

七、附录 （可选）

//服务器程序

// PipeServer.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "PipeServer.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// The one and only application object

CWinApp theApp;

using namespace std;

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

{

int nRetCode = 0;

int err;

BOOL rc;

HANDLE hPipeHandle1;

char lpName[] = "\\\\.\\pipe\\myPipe";

char InBuffer[50] = "";

char OutBuffer[50] = "";

DWORD BytesRead, BytesWrite;

hPipeHandle1 = CreateNamedPipe(

(LPCTSTR)lpName,

PIPE\_ACCESS\_DUPLEX | FILE\_FLAG\_OVERLAPPED | WRITE\_DAC,

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_BYTE | PIPE\_WAIT,

1, 20, 30, NMPWAIT\_USE\_DEFAULT\_WAIT,

(LPSECURITY\_ATTRIBUTES)NULL

);

if ((hPipeHandle1 == INVALID\_HANDLE\_VALUE) || (hPipeHandle1 == NULL))

{

err = GetLastError();

printf("Server Pipe Create Fail!err=%d\n", err);

exit(1);

}

else

printf("Server Pipe Create Success!\n");

while (1)

{

rc = ConnectNamedPipe(hPipeHandle1, (LPOVERLAPPED)NULL);

if (rc == 0)

{

err = GetLastError();

printf("Server Pipe Connect Fail!err=%d\n", err);

exit(2);

}

else

printf("Server Pipe Connect Success\n");

strcpy(InBuffer, "");

strcpy(OutBuffer, "");

rc = ReadFile(hPipeHandle1, InBuffer, sizeof(InBuffer), &BytesRead,

(LPOVERLAPPED)NULL);

if (rc == 0 && BytesRead == 0)

{

err = GetLastError();

printf("Server Read Pipe Fail!err=%d\n", err);

exit(3);

}

else

printf("Server Read Pipe Success!\nDATA from Client is=%s\n", InBuffer);

rc = strcmp(InBuffer, "end");

if (rc == 0)

break;

printf("Please Input Data to Send\n");

scanf("%s", OutBuffer);

rc = WriteFile(hPipeHandle1, OutBuffer, sizeof(OutBuffer), &BytesWrite,

(LPOVERLAPPED)NULL);

if (rc == 0)

printf("Server Write Pipe Fail!\n");

else

printf("Server Write Pipe Success!\n");

DisconnectNamedPipe(hPipeHandle1);

rc = strcmp(OutBuffer, "end");

if (rc == 0)

break;

}

printf("Now Server be END!\n");

CloseHandle(hPipeHandle1);

return nRetCode;

}

//客户端程序

// PipeClient.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "PipeClient.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

/////////////////////////////////////////////////////////////////////////////

// The one and only application object

CWinApp theApp;

using namespace std;

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

{

int nRetCode = 0;

int err = 0;

BOOL rc = 0;

char lpName[] = "\\\\.\\pipe\\myPipe";

char InBuffer[50] = "";

char OutBuffer[50] = "";

DWORD BytesRead;

while (1)

{

strcpy(InBuffer, "");

strcpy(OutBuffer, "");

printf("Input Data Please!\n");

scanf("%s", InBuffer);

rc = strcmp(InBuffer, "end");

if (rc = 0)

{

rc = CallNamedPipe(lpName, InBuffer, sizeof(InBuffer), OutBuffer,

sizeof(OutBuffer), &BytesRead, NMPWAIT\_USE\_DEFAULT\_WAIT);

break;

}

rc = WaitNamedPipe(lpName, NMPWAIT\_WAIT\_FOREVER);

if (rc == 0)

{

err = GetLastError();

printf("Wait Pipe Fail!err=%d\n", err);

exit(1);

}

else

printf("Wait Pipe Success\n");

rc = CallNamedPipe(lpName, InBuffer, sizeof(InBuffer), OutBuffer,

sizeof(OutBuffer), &BytesRead, NMPWAIT\_USE\_DEFAULT\_WAIT);

rc = strcmp(OutBuffer, "end");

if (rc == 0)

break;

if (rc == 0)

{

err = GetLastError();

printf("Pipe Call Fail!err=%d\n", err);

exit(1);

}

else

printf("Pipe Call Success!\nDATA from Server is=%s\n", OutBuffer);

}

printf("Now Client be END!\n");

return nRetCode;

}