|  |  |
| --- | --- |
| |  | | --- | | **编写自己的"IDE"** -- 如何在图形界面中实时捕获控制台程序的标准输出  作者：王咏刚  本文出处：[http://www.contextfree.net/wangyg/tech/myIDE.htm](http://www.contextfree.net/wangyg/tech/myIDE.htm" \t "_blank)  IDE是集成开发环境（Integrated Development Environment）的简称。印象里有很多出色的IDE，比如JBuilder和Kylix，比如Visual Studio。不知大家是否留意过，大多数IDE本身只提供代码编辑、工程管理等人机交互功能，我们在IDE中编译代码、调试程序时，IDE需要调用命令行的编译器、调试器完成相应的操作。例如，使用Visual Studio编译C++程序时，我们会在IDE下方的Output窗口中看到编译和连接的全过程，虽然我们看不到弹出的DOS窗口，但实际上是IDE先后启动了Microsoft C++编译器cl.exe和连接器link.exe这两个命令行程序，而cl.exe和link.exe的输出又实时反映到了IDE的Output窗口中。还有，我们可以在Visual Studio中配置自己需要的工具程序（比如特殊的编译器），然后让Visual Studio在适当的时候运行这些工具，并将工具程序的输出实时显示到Output窗口中。下图是我在Visual Studio 6.0的Output窗口中运行J2SDK的javac.exe编译java源程序并显示程序中语法错误的情形：  http://www.vckbase.com/document/journal/vckbase0/images/myide1.gif  也就是说，大多数IDE工具都可以在集成环境中调用特定的命令行程序（WIN32里更确切的说法是控制台程序），然后实时捕获它们的输出（这多半是输出到标准的stdout和stderr流里的东西），并将捕获到的信息显示在图形界面的窗口中。  这显然是一种具备潜在价值的功能。利用这一技术，我们至少可以  1. 编写出自己的IDE，如果我们有足够的耐心的话；  2. 在我们自己的应用程序里嵌入全文检索功能（调用Borland C++里的grep.exe工具），或者压缩和解压缩功能（调用控制台方式的压缩解压程序，比如arj.exe、pkzip.exe等）；  3. 连接其他人编写的，或者我们自己很久以前编写的控制台程序——我经常因为难以调用一个功能强大但又没有源码的控制台程序而苦恼万分。  这样好的功能是如何实现的呢？  首 先，如果我们想做的是用一个控制台程序调用另一个控制台程序，那就再简单不过了。我们只消把父进程的stdout重定向到某个匿名管道的WRITE端，然 后启动子进程，这时，子进程的stdout因为继承的关系也连在了管道的WRITE端，子进程的所有标准输出都写入了管道，父进程则在管道的另一端随时“ 侦听”——这一技术叫做输入输出的重定向。  可现在的问题是，GUI方式的Windows程序根本没有控制台，没有stdin、stdout之类的东西，子进程又是别人写好的东西无法更改，这重定向该从何谈起呢？  还 有另外一招：我们可以直接在调用子进程时用命令行中的管道指令“>”将子进程的标准输出重定向到一个文件，子进程运行完毕后再去读取文件内容。这种 方法当然可行，但它的问题是，我们很难实时监控子进程的输出，如果子进程不是随时刷新stdout的话，那我们只能等一整块数据实际写入文件之后才能看到 运行结果；况且，访问磁盘文件的开销也远比内存中的管道操作来得大。  我这里给出的方案其实很简单：既然控制台程序可以调用 另一个控制台程序并完成输入输出的重定向，那我们完全可以编写一个中介程序，这个中介程序调用我们需要调用的工具程序并随时获取该程序的输出信息，然后直 接将信息用约定的进程间通讯方式（比如匿名管道）传回GUI程序，就象下图中这样： http://www.vckbase.com/document/journal/vckbase0/images/myide2.gif 图中，工具程序和中介程序都是以隐藏的方式运行的。工具程序原本输出到stdout的信息被重定向到中介程序开辟的管道中，中介程序再利用GUI程序创建 的管道将信息即时传递到GUI程序的一个后台线程里，后台线程负责刷新GUI程序的用户界面（使用后台线程的原因是，只有这样才可以保证信息在GUI界面 中随时输出时不影响用户正在进行的其他操作，就象我们在Visual Studio中执行耗时较长的编译功能那样）。  我写的中介程序名字叫wSpawn，这个名字来自 Visual Studio里完成类似功能的中介程序VcSpawn（你可以在Visual Studio的安装目录中找到它）。我的wSpawn非常简单，它利用系统调用\_popen()同时完成创建子进程和输入输出重定向两件工作。GUI程序 则使用一种特殊的命令行方式调用wSpawn：  wspawn –h <n> <command> [arg1] [arg2] ...  其中，-h后跟的是GUI程序提供的管道句柄，由GUI程序自动将其转换为十进制数字，wSpawn运行时将信息写入该句柄中，随后的内容是GUI程序真正要执行的命令行，例如调用C++编译器cl.exe的方式大致如下：  wspawn –h 1903 cl /Id:\myInclude Test.cpp  wspawn.cpp的程序清单如下：  #include <stdlib.h>  #include <stdio.h>  #include <string.h>  #include <string>  #include <windows.h>  using namespace std;  void exit\_friendly(void)  {  puts("请不要单独运行wSpawn.");  exit(0);  }  int main( int argc, char \*argv[] )  {  HANDLE hWrite = NULL;  DWORD dwWrited;  int i = 0, ret = 0, len = 0;  char psBuffer[256];  FILE\* child\_output;  string command\_line = "";  // 检查命令行，如存在管道句柄，则将其转换为HANDLE类型  if (argc < 2)  exit\_friendly();  if (!stricmp(argv[1], "-h"))  {  if (argc < 4)  exit\_friendly();  hWrite = (HANDLE)atoi(argv[2]);  i = 3;  }  else  i = 1;  // 提取要执行的命令  for (; i < argc; i++)  {  command\_line += argv[i];  command\_line += " ";  }  // 使用\_popen创建子进程并重定向其标准输出到文件指针中  if( (child\_output = \_popen( command\_line.c\_str(), "rt" )) == NULL )  exit( 1 );  while( !feof( child\_output ) )  {  if( fgets( psBuffer, 255, child\_output ) != NULL )  {  if (hWrite)  {  // 将子进程的标准输出写入管道，提供给自己的父进程  // 格式是先写数据块长度（0表示结束），再写数据块内容  len = strlen(psBuffer);  WriteFile(hWrite, &len, sizeof(int), &dwWrited, NULL);  WriteFile(hWrite, psBuffer, len, &dwWrited, NULL);  }  else  // 如命令行未提供管道句柄，则直接打印输出  printf(psBuffer);  }  }  // 写“0”表示所有数据都已写完  len = 0;  if (hWrite)  WriteFile(hWrite, &len, sizeof(int), &dwWrited, NULL);  return \_pclose( child\_output );  }  下面，我们就利用wSpawn程序，写一个简单的“IDE”工具。我们选择Visual Studio 6.0作为开发环境（本文给出的代码也在Visual Studio.NET 7.0中做过测试）。首先，创建Visual C++工程myIDE，工程类型为MFC AppWizard(EXE)中的Dialog based类型，即创建了一个主窗口为对话框的GUI程序。工程myIDE的主对话框类是CMyIDEDlg。现在我们要在资源编辑器中为主对话框添加一个足够大的多行编辑框（Edit Box），它的控制ID是IDC\_EDIT1，必须为IDC\_EDIT1设置以下属性：  Multiline, Horizontal scroll, Auto HScroll, Vertical scroll, Auto VScroll, Want return   然后用ClassWizard为IDC\_EDIT1添加一个对应的成员变量（注意变量的类型要选CEdit型而非字符串CString型）  CEdit m\_edit1;  使用ClassWizard为“确定”按钮添加消息响应方法OnOK()，编辑该方法：  void CMyIDEDlg::OnOK()  {  AfxBeginThread(myThread, this);  InvalidateRect(NULL);  UpdateWindow();  }  也就是说，我们在“确定”按钮按下时，启动了后台线程myThread()，那么，myThread()到底做了些什么呢？我们先在CMyIDEDlg类的头文件myIDEDlg.h中加上一个成员函数声明：  protected:  static UINT myThread(LPVOID pParam);  然后，在CMyIDEDlg类的实现文件myIDEDlg.cpp里添加myThread()的实现代码：  UINT CMyIDEDlg::myThread(LPVOID pParam)  {  PROCESS\_INFORMATION pi;  STARTUPINFO siStartInfo;  SECURITY\_ATTRIBUTES saAttr;  CString Output, tmp;  char command\_line[200];  DWORD dwRead;  char\* buf; int len;  HANDLE hRead, hWrite;  CMyIDEDlg\* pDlg = (CMyIDEDlg\*)pParam;  // 创建与wSpawn.exe通讯的可继承的匿名管道  saAttr.nLength = sizeof(SECURITY\_ATTRIBUTES);  saAttr.bInheritHandle = TRUE;  saAttr.lpSecurityDescriptor = NULL;  if (!CreatePipe(&hRead, &hWrite, &saAttr, 0))  {  AfxMessageBox("创建管道失败");  return 0;  }  // 准备wSpawn的命令行，在命令行给出写管道句柄和要wSpawn执行的命令  memset(π, 0, sizeof(pi));  sprintf(command\_line, "wspawn -h %d cl /?", (unsigned int)hWrite);  // 子进程以隐藏方式运行  ZeroMemory( &siStartInfo, sizeof(STARTUPINFO) );  siStartInfo.cb = sizeof(STARTUPINFO);  siStartInfo.wShowWindow = SW\_HIDE;  siStartInfo.dwFlags = STARTF\_USESHOWWINDOW;  // 创建wSpawn子进程  if (!CreateProcess( NULL, command\_line, NULL, NULL, TRUE,  0, NULL, NULL, &siStartInfo, π))  {  AfxMessageBox("调用wSpawn时失败");  return 0;  }  // 读管道，并显示wSpawn从管道中返回的输出信息  if(!ReadFile( hRead, &len, sizeof(int), &dwRead, NULL) || dwRead == 0)  return 0;  while(len)  {  buf = new char[len + 1];  memset(buf, 0, len + 1);  if(!ReadFile( hRead, buf, len, &dwRead, NULL) || dwRead == 0)  return 0;  // 将返回信息中的"\n"替换为Edit Box可识别的"\r\n"  tmp = buf;  tmp.Replace("\n", "\r\n");  Output += tmp;  // 将结果显示在Edit Box中，并刷新对话框  pDlg->m\_edit1.SetWindowText(Output);  pDlg->InvalidateRect(NULL);  pDlg->UpdateWindow();  delete[] buf;  if(!ReadFile( hRead, &len, sizeof(int), &dwRead, NULL) || dwRead == 0)  return 0;  }  // 等待wSpawn结束  WaitForSingleObject(pi.hProcess, 30000);  // 关闭管道句柄  CloseHandle(hRead);  CloseHandle(hWrite);  return 0;  }  很简单，不是吗？后台线程创建一个匿名管道，然后以隐藏方式启动wSpawn.exe并将管道句柄通过命令行传给wSpawn.exe，接下来只要从管道 里读取信息就可以了。现在我们可以试着编译运行myIDE.exe了，记住要把myIDE.exe和wSpawn.exe放在同一目录下。还有，我在 myThread()函数中写死了传给wSpawn.exe的待执行的命令行是“cl /?”，这模拟了一次典型的编译过程，如果你不打算改变这一行代码的话，那一定要注意在你的计算机上，C++编译器cl.exe必须位于环境变量PATH 指明的路径里，否则wSpawn.exe可就找不到cl.exe了。下面是myIDE程序的运行结果：  http://www.vckbase.com/document/journal/vckbase0/images/myide3.gif  补充一点，上面给出的wSpawn利用\_popen()完成子进程创建和输入输出重定向，这一方法虽然简单，但只能重定向子进程的stdout或 stdin，如果还需要重定向子进程的stderr的话（Java编译器javac就利用stderr输出结果信息），那我们就不能这么投机取巧了。根据 以上讨论，你一定可以使用传统的\_pipe()、\_dup()等系统调用，写出功能更完整的新版wSpawn来，我这里就不再罗嗦了。 | |

|  |  |
| --- | --- |
| http://www.vckbase.com/SYS/script/image/toplogo.gif最新评论 [[发表评论]](http://www.vckbase.com/SYS/script/viewcomment.asp?gclsid=100&itemid=1097) [[文章投稿]](http://www.vckbase.com/support/contribute.html) | http://www.vckbase.com/document/viewdoc/image/rec1.gif[查看所有评论](http://www.vckbase.com/SYS/script/viewcomment.asp?gclsid=100&itemid=1097) http://www.vckbase.com/document/viewdoc/image/rec1.gif[推荐给好友](http://www.vckbase.com/SYS/script/writemail.asp?gclsid=100&itemid=1097&title=%b1%e0%d0%b4%d7%d4%bc%ba%b5%c4IDE) http://www.vckbase.com/document/viewdoc/image/rec1.gif[打印](javascript:window.print();) |

|  |
| --- |
| http://www.vckbase.com/document/image/doc2.gifhttp://www.contextfree.net/wangyg/b/tech/myide.html 补充：相反方向的信息传递 ( flyingxu 发表于 2006-9-15 15:35:00)   http://www.vckbase.com/document/image/doc2.gif::CloseHandle(hSTDOUTWrite);    ::CloseHandle(hSTDINRead);  char strData[1024] = {0, }; DWORD dwBytes; while(::ReadFile(hSTDOUTRead, strData, sizeof(strData), &dwBytes, NULL)) {    strData[dwBytes] = '\0'; SetMessage(strData); }     ::WaitForSingleObject(pi.hProcess, INFINITE);    ::CloseHandle(hSTDOUTRead);    ::CloseHandle(hSTDINWrite);    ::CloseHandle(pi.hProcess);    ::CloseHandle(pi.hThread);   }// end of function ( robin\_fox\_nan 发表于 2006-8-27 20:52:00)   http://www.vckbase.com/document/image/doc2.gifPROCESS\_INFORMATION  pi;    ZeroMemory(&pi, sizeof(pi));    STARTUPINFO  si;  GetStartupInfo(&si);  si.cb = sizeof(STARTUPINFO);     si.dwFlags = STARTF\_USESTDHANDLES   |   STARTF\_USESHOWWINDOW; si.wShowWindow = SW\_HIDE; si.hStdInput   =   hSTDINRead;      //重定向子进程输入        si.hStdOutput   =   hSTDOUTWrite;   //重定向子进程输入     si.hStdError = hSTDOUTWrite; //GetStdHandle( STD\_ERROR\_HANDLE );   char cmd[20] = {0, }; ::strcpy(cmd, "c:\\fport.exe");  if( !::CreateProcess(NULL, cmd, NULL, NULL, TRUE, NORMAL\_PRIORITY\_CLASS, NULL, NULL, &si, &pi) )    {    int errorNo = ::GetLastError(); AfxMessageBox("create process failed");    return;    }  ( robin\_fox\_nan 发表于 2006-8-27 20:51:00)   http://www.vckbase.com/document/image/doc2.gif有那么费劲么？真的累死了，其实可以这样做的： void function() { HANDLE   hSTDINWrite, hSTDINRead;       // 用于重定向子进程输入的句柄    HANDLE   hSTDOUTWrite, hSTDOUTRead;     // 用于重定向子进程输出的句柄    SECURITY\_ATTRIBUTES   sa;         sa.bInheritHandle = TRUE;    sa.lpSecurityDescriptor = NULL;    sa.nLength = sizeof(sa);     // 创建子进程输出匿名管道    if( !CreatePipe(&hSTDOUTRead, &hSTDOUTWrite, &sa, 0) )    {    AfxMessageBox("Create   STDOUT   pipe   failed");    return;    }    // 创建子进程输入匿名管道    if( !CreatePipe(&hSTDINRead, &hSTDINWrite, &sa, 0) )    {    AfxMessageBox("Create   STDIN   pipe   failed");    return;    } ( robin\_fox\_nan 发表于 2006-8-27 20:50:00)   http://www.vckbase.com/document/image/doc2.gif其他的还行，不过建议把刷新改为pDlg->m\_edit1.LineScroll(pDlg->m\_edit1.GetLineCount()); ( fanged 发表于 2004-7-22 12:06:00)   http://www.vckbase.com/document/image/doc2.gif请教各位：我刚刚接触管道，想实现进程交互通信，我在父程序中创建子进程和两个匿名管道，怎么输入的数据传递给子进程叫子进程来处理？我将标准输入设为一 个管道的读句柄，然后由另一个管道传出来，怎么没反应？谢谢！ ( chebyshe 发表于 2004-5-4 21:41:00)   http://www.vckbase.com/document/image/doc2.gif唉, 还用中间程序啊. 直接在STARTUPINFO中指定一个管道的句柄就行了. 即不必用到临时文件,也不必用到中介程序.  ( 我酷我用弱智王 发表于 2004-5-1 19:23:00)   http://www.vckbase.com/document/image/doc2.gif这样是不是麻烦了一些?我觉得,既然CreateProcess中的STARTUPINFO结构体中可以指定标准输入输出流,那么,只需要把这个流传给子 进程(即是那些命令行程序),然后再通过readfile来读取这个流,那不是更简捷吗? ( 童梦 发表于 2004-4-26 18:58:00)   http://www.vckbase.com/document/image/doc2.gif发现牛的程序员都很有文采。 可见知识是相通的。。。。。。  尔曰浅显，彼曰艰深，唯其深入，方可浅出。 ( superdai 发表于 2004-4-22 10:25:00)   http://www.vckbase.com/document/image/doc2.gif作者就是经常在《程序员》上发表文章的那个王咏刚。 我阅读作者的多篇技术文章，对作者的技术功底简直崇拜的要死：） ( wuhuaqiang 发表于 2004-4-21 17:51:00)   ....................................................... [More...](http://www.vckbase.com/SYS/script/viewcomment.asp?gclsid=100&itemid=1097) |

http://www.vckbase.com/SYS/script/image/doc.gif龙仪:GOOD  
(发表于2004-4-18 22:52:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifxiaojin:好，建议放到在线杂志中去。  
(发表于2004-4-19 12:35:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifMadlee:收藏先，不过我想应该可以不通过中间的那个进程直接读写子进程的io的。msdn里有个例子，我一直希望能把他封装成标准的stream，可惜水平不够。:(  
  
(发表于2004-4-19 18:44:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifwuhuaqiang:作者就是经常在《程序员》上发表文章的那个王咏刚。  
我阅读作者的多篇技术文章，对作者的技术功底简直崇拜的要死：）  
(发表于2004-4-21 17:51:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifsuperdai:发现牛的程序员都很有文采。  
可见知识是相通的。。。。。。  
  
尔曰浅显，彼曰艰深，唯其深入，方可浅出。  
(发表于2004-4-22 10:25:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gif童梦:这样是不是麻烦了一些?我觉得,既然CreateProcess中的STARTUPINFO结构体中可以指定标准输入输出流,那么,只需要把这个流传给子进程(即是那些命令行程序),然后再通过readfile来读取这个流,那不是更简捷吗?  
(发表于2004-4-26 18:58:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gif我酷我用弱智王:唉, 还用中间程序啊. 直接在STARTUPINFO中指定一个管道的句柄就行了. 即不必用到临时文件,也不必用到中介程序.  
  
  
(发表于2004-5-1 19:23:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifchebyshe:请教各位：我刚刚接触管道，想实现进程交互通信，我在父程序中创建子进程和两个匿名管道，怎么输入的数据传递给子进程叫子进程来处理？我将标准输入设为一个管道的读句柄，然后由另一个管道传出来，怎么没反应？谢谢！  
(发表于2004-5-4 21:41:00)  
  
http://www.vckbase.com/SYS/script/image/doc.giffanged:其他的还行，不过建议把刷新改为pDlg->m\_edit1.LineScroll(pDlg->m\_edit1.GetLineCount());  
(发表于2004-7-22 12:06:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifrobin\_fox\_nan:有那么费劲么？真的累死了，其实可以这样做的：  
void function()  
{  
HANDLE   hSTDINWrite, hSTDINRead;       // 用于重定向子进程输入的句柄     
HANDLE   hSTDOUTWrite, hSTDOUTRead;     // 用于重定向子进程输出的句柄     
SECURITY\_ATTRIBUTES   sa;     
      
sa.bInheritHandle = TRUE;     
sa.lpSecurityDescriptor = NULL;     
sa.nLength = sizeof(sa);     
  
// 创建子进程输出匿名管道     
if( !CreatePipe(&hSTDOUTRead, &hSTDOUTWrite, &sa, 0) )     
{     
AfxMessageBox("Create   STDOUT   pipe   failed");     
return;     
}    
  
// 创建子进程输入匿名管道     
if( !CreatePipe(&hSTDINRead, &hSTDINWrite, &sa, 0) )     
{     
AfxMessageBox("Create   STDIN   pipe   failed");     
return;     
}  
(发表于2006-8-27 20:50:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifrobin\_fox\_nan: PROCESS\_INFORMATION  pi;     
ZeroMemory(&pi, sizeof(pi));     
STARTUPINFO  si;   
GetStartupInfo(&si);  
  
si.cb = sizeof(STARTUPINFO);  
    si.dwFlags = STARTF\_USESTDHANDLES   |   STARTF\_USESHOWWINDOW;  
si.wShowWindow = SW\_HIDE;  
si.hStdInput   =   hSTDINRead;      //重定向子进程输入     
    si.hStdOutput   =   hSTDOUTWrite;   //重定向子进程输入      
si.hStdError = hSTDOUTWrite; //GetStdHandle( STD\_ERROR\_HANDLE );   
  
char cmd[20] = {0, };  
::strcpy(cmd, "c:\\fport.exe");  
  
if( !::CreateProcess(NULL, cmd, NULL, NULL, TRUE, NORMAL\_PRIORITY\_CLASS, NULL, NULL, &si, &pi) )     
{     
int errorNo = ::GetLastError();  
AfxMessageBox("create process failed");     
return;     
}  
  
  
(发表于2006-8-27 20:51:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifrobin\_fox\_nan:::CloseHandle(hSTDOUTWrite);     
::CloseHandle(hSTDINRead);  
  
char strData[1024] = {0, };  
DWORD dwBytes;  
while(::ReadFile(hSTDOUTRead, strData, sizeof(strData), &dwBytes, NULL))  
{     
strData[dwBytes] = '\0';  
SetMessage(strData);  
}     
  
::WaitForSingleObject(pi.hProcess, INFINITE);     
::CloseHandle(hSTDOUTRead);     
::CloseHandle(hSTDINWrite);     
::CloseHandle(pi.hProcess);     
::CloseHandle(pi.hThread);    
}// end of function  
(发表于2006-8-27 20:52:00)  
  
http://www.vckbase.com/SYS/script/image/doc.gifflyingxu:http://www.contextfree.net/wangyg/b/tech/myide.html  
补充：相反方向的信息传递  
(发表于2006-9-15 15:35:00)  
  
..........................................................................