文章编号:1000-582X(2008)04-0436-06

进程间通信技术在系统集成中的应用

袁　鸿,刘　浩,廖文和
(南京航空航天大学 机电学院,江苏 南京 210016)

摘　要:集成了大型机移植到微机上的图形学算法模块,并增加 GUI(graphic user interface)界面。使用 Windows 进程间通信技术来实现 GUI 界面和控制台程序间数据通信。利用管道技术并结合某曲面造型系统开发要求,编制了一个进程间通信的中介模块,给出该模块的代码和执行流程,在系统运行时通过直接调用中介模块实现 GUI 菜单命令驱动算法执行的数据交互。中介模块经适当修改后可以解决类似场合的数据通信问题。

关键词:进程间通信;管道;中介模块;系统集成;数据交互
中图分类号:TP391　　文献标志码:A

Application of windows inter-process communication in system integration

YUAN Hong, *LIU Hao*, *LIAO Wen-he*
(Nanjing University of Aeronautics and Astronautics College of Mechanical Engineering, Nanjing 210016, P. R. China)

Abstract: In order to achieve the visualization of a graphic user interface(GUI) and the integration of a graphics arithmetic module that was transplanted from a mainframe computer to a micro-computer, we proposed a method of making possible the data communication between the GUI and console application using an inter-process communication technique. The definition, storage and establishment of Windows procesce were discussed. A general format proxy module using a pipe mode was programmed. The module fit with the development requirement of a certain surface modeling system. The program and the implementation flow of this proxy were proposed, and the data alternation between the arithmetic module and GUI menu command was realized. This proxy module holds promise for dealing with similar problems with suitable modification.

Key words: inter-process communication; pipe; proxy program; system integration; data alternation

在航空、航天、汽车等领域的复杂产品外形设计中,传统的 CAD 建模技术难以满足设计要求。近十多年来,CAD/CAGD 领域的专家学者一直试图寻找解决的方法,先后提出了许多新的曲面造型方法,在不同范围或领域内克服了现有曲面造型方法的不足。利用细分曲面造型方法在执行效率、数值稳定性、任意拓扑结构方面的优势,课题组建立了以 Nurbs 曲面与细分曲面相混合的一个曲面造型系统。

在该曲面造型系统研制过程中,算法模块直接从大型机/工作站移植到微机上,软件平台从 UNIX 系统转换到 Windows 系统,使得这些模块没有 GUI(graphic user interface 图形用户接口)界面,只有字符命令即控制台(Win32 console application)输入窗

收稿日期:2007-12-10
基金项目:国防基础科研资助项目(k1605061115)
作者简介:袁鸿(1978-),男,南京航空航天大学博士研究生,主要从事 CAD/CAGD 方向研究,(Tel)13914713920;(E-mail)yuanhong1979nuaa@yahoo.com.cn。

口,课题组需要将这些不同的程序开发平台开发的模块集成到一个统一的 Windows 系统框架中。笔者归纳了一种如何利用 Windows 的进程间通信技术来实现模块之间数据传递和通信的方法,阐述了 GUI 界面程序和编译成 Win32 控制台程序的算法模块之间是如何通过中介程序进行通信的。

1　Windows 进程

Windows 就为一个可执行程序的一次执行创建一个进程,产生一个进程句柄。即实例化句柄,操作系统和其他的进程可以通过该句柄与其进行信息交互,这种进程间的数据交换就是进程间通信(inter process communication, IPC)技术。

1.1　进程定义

进程是程序的一次执行过程;反之,程序则是进程的一种静态描述。进程在操作系统的管理下被启动,进入运行状态,并在一定条件下中止或结束。进程的运行需要使用一定的计算机资源(处理器、内存、各种外部设备等),因此,进程又是操作系统调度的基本单位。一个普通(非并发)程序被执行时,相当于产生一个新进程;执行完毕后,该进程就消亡[1]。

程序是一段静态代码,是应用执行的蓝本。进程则是程序的一次动态执行过程。作为执行蓝本的程序可被多次加载到系统的不同内存区域而形成不同进程。并行程序中的多个进程不一定是在一开始就同时出现并一直维持到整个程序结束,而是可以根据需要动态地产生或消亡。换言之,一个进程在运行过程中可以派生出新的进程;新派生出的进程又可继续派生出下一代的进程,于是形成进程之间的"父子关系"。当一个新的子进程诞生时,它所需要的资源通常是从其父进程的资源中划分而来的。不同的进程在互相独立的内存空间中运行,这一点和"线程"是不同的。同一进程内又可分解为若干线程,它们在逻辑上可以同时运行。但是,这些线程共享同一内存空间,它们之间的调度无需操作系统的介入,该特点减小了系统开销,提高了效率[1]。

1.2　Windows 进程存储

在 Windows 操作系统中,每个进程都有 H instance,该变量在程序启动时自动生成,应用时载入程序的基地址,Windows95/98 中的典型值是 10X00400000(4194304)。事实上,每个程序所分配的地址并不是 RAM 中的物理地址,而是虚拟地址。操作系统实际只是分配一个地址范围,当需要访问这段存储空间,再把物理地址分配给它。这种把物理地址分配给虚拟存储的方法称为"映射"。一页存储空间一般为 4 KB(4 096 bit)。Win32 操作系统不停地把一页一页的虚拟存储空间映射到内存空间。映射过程中,Windows 通过索引表把虚拟地址转化为物理地址,使程序的地址看起来没有变化。在 Win32 程序中,得到的地址决不会直接指向物理存储器,是间接地址。虚拟存储使进程与进程之间不受干扰,多个同时运行的进程相互无关。这样,进程之间相互覆盖和冲突的概率就大大减小[2]。

1.3　Windows 进程创建

系统为保证多任务之间的独立性,当 Windows 各模块分开单独执行时,可使用多进程。创建进程使用 Windows 提供的 API 函数 CreateProcess 函数:

```
BOOL CreateProcess
(
LPCTSTR lpApplicationName;
//指向可执行文件名的字符串
LPTSTR lpCommandLine;
//指向将执行的应用程序命令行的字符串
LPSECURITY_ATTRIBUTES lpProcessAttributes;
//指向一个安全结构,用以说明创建进程的安全属性
LPSECURITY_ATTRIBUTES lpThreadAttributes;
//用以指定创建进程的主线程的安全属性
BOOL bInheritHandles;
//决定新进程是否从调用进程中继承句柄
DWORD dwCreationFlags;
//新进程的附加标志
LPVOID lpEnvironment;
//指向一个用于新进程的环境块
LPCTSTR lpCurrentDirectory;
//指向一个创建进程指定当前工作路径的字符串
LPSTARTUPINFO lpStartupInfo;
//指向 STARTUPINFO 结构,用于指定新进程主窗口启动时的显示
LPPROCESS_INFORMATION lpProcessInformation
//指向 PROCESS_INFORMATION 结构,用于新进程的表示信息
);
```

其中 PROCESS_INFORMATION 结构定义为:

```
typedef struct _PROCESS_INFORMATION
{
HANDLE hProcess;
//新创建进程的句柄
```

```
HANDLE hThread;
//新创建进程的主线程句柄
DWORD dwProcessID;
//新创建进程的 ID
DWORD dwThreadID;
//新创建进程的主线程 ID
}PROCESS_INFORMATION;
```

其中 STARTUPINFO 结构定义为：

```
typedef struct _STARTUPINFO
{
DWORD cb;
//本结构大小,sizeof(STARTUPINFO)
LPTSTR lpReserved;
//保留字,在调用 CreateProcess 前应置为空
LPTSTR lpDesktop;
//指向桌面名,只在 WindowsNT 下有效
LPTSTR lpTitle;
//用于设置控制台进程的窗口标题
DWORD dwX;
//新建窗口位置的 X 向坐标值
DWORD dwY;
//新建窗口位置的 Y 向坐标值
DWORD dwXSize;
//新建窗口的 X 尺寸
DWORD dwYSize;
//新建窗口的 Y 尺寸
DWORD dwXCountChars;
//用于设置控制台进程的一行的总字符数
DWORD dwYCountChars;
//用于设置控制台进程的一列的总字符数
DWORD dwFillAttribute;
//用于设置控制台进程的字符颜色
DWORD dwFlags;
//标志位,指定本结构各个成员的使用方式
WORD wShowWindow;
//创建新进程的 ShowWindow 的参数
WORD cbReserved2;
//保留字段,应为 0
LPBYTE lpReserved2;
//保留字段,应为 NULL
HANDLE hStdInput;
//新进程标准输入句柄
HANDLE hStdOutput;
//新进程标准输出句柄
HANDLE hStdError;
//新进程标准错误句柄
} STARTUPINFO, *LPSTARTUPINFO;
```

系统创建的新进程完全独立于调用进程。但是,如果用 CreateProcess 函数创建进程成功,新进程的相关句柄和 ID 值可通过 PROCESS_INFORMATION 结构返回,通过进程句柄可获得新创建进程的完全控制权。终止一个进程可调用 ExitProcess 函数或 TerminateProgress 函数,ExitProcess 函数可终止此进程的所有附属的 DLL 和线程。

2 进程间通信

Win32 所提供的进程间通信方式有 7 种:COM/DCOM、File Mapping 文件映射方式、WM_COP・ATA 系统消息方式、Pipe 管道方式、Mailslots 邮件槽、RPC 远程过程调用方式和 Windows Sockets 网络套接字方式[3]。

在研究中根据实际需要采用了 Pipe 管道来实现进程间通信。Pipe 是一种以先进先出的方式保存一定数量数据的特殊文件。Pipe 通信是高层的、基于内存的通信系统;通信中,由系统提供再执行写操作和读操作的进程之间的同步。在默认情况下,如果一个进程试图写入一个已填满的管道,系统会自动阻塞该进程,直到管道能够接收数据;如果试图读一个空管道,进程会阻塞,直到有可读数据出现为止;如果一个进程以读方式打开一个管道,而没有另外的进程以写方式打开该管道,则同样会造成该进程阻塞。

Pipe 是由系统 Server 端创建的：

```
Pipe=CreateNamedPipe(pPipeName,
PIPE_ACCESS_DUPLEX, PIPE_TYPE_MESSAGE|PIPE_READMODE_MESSAGE|PIPE_WAIT,
PIPE_UNLIMITED_INSTANCES,
BUFSIZE, BUFSIZE, 20000, NULL);
```

创建完成后,系统 Client 端就可通过一般的文件 API 读写数据,代码为

```
Pipe = CreateFile(pPipeName,
GENERIC_READ|GENERIC_WRITE,
0, NULL, OPEN_EXISTING,0,NULL);
success = Write File(pipe, pMessage,
strlen(pMessage) + 1,
&bytesWritten, NULL);
```

Pipe 分 2 种,即命名 Pipe 和匿名 Pipe。匿名 Pipe 是以句柄而不是以名字标识的,因此限制它只能在同一台机器上通信,不能应用于网络;Pipe 既可单向通信也可双向通信,较为可靠。

3　应用实例

3.1　系统集成中的中介程序

研究的曲面造型系统中的细分基础理论算法模块是在 Visual C++ 6 平台上对部分代码进行修改、编译而成的 Win32 控制台程序(Win32 console application),用户可以通过控制台字符命令运行这个模块,对算法进行实时参数控制与调整。将其集成到曲面造型系统的主要任务是增加 Windows 风格的 GUI(graphic user interface 图形用户界面)程序,用户通过界面程序向后台算法模块发送命令,算法模块进行命令解释后,在其自有的图形窗口或者系统的图形窗口显示执行结果,并将反馈信息发送到终端控制台窗口上[4]。

在前面的系统集成过程中需要解决的问题是:GUI 方式的 Windows 程序没有控制台的 stdin、stdout(标准输入输出)之类的接口,如何将 GUI 方式的 Windows 菜单命令和对话框中的参数传给 1 个控制台程序。

在研究中,利用"控制台程序可调用另一个控制台程序并完成输入输出的重定向"这一特点[5-6],编写了一个中介程序。这个中介程序调用细分算法模块程序并定向该程序的输入接口,GUI 界面程序将该中介程序以后台进程方式启动,通过 Pipe 管道将用户对界面程序的操作(鼠标和键盘的输入)信息即时传给中介程序,中介程序再将信息发送到被重定向的后台模块的 stdin 中,同时由界面信息提示面板读取中介程序写入到反馈信息文件中的操作反馈信息,从而实现用户和后台算法模块之间的人机交互。使用后台进程可以保证信息在从 GUI 界面经中介程序传到核心程序中时不影响用户正在进行的其他操作[7](如图 1-2)。

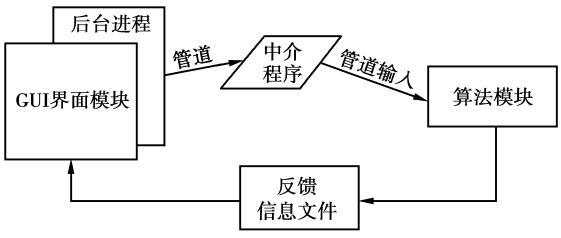

图 1　造型系统模块结构

中介程序是用 C++ 语言编写的 Win32 控制台程序。通过系统调用_popen()函数,它可同时完成创建子进程(细分理论算法模块)和输入输出重定向

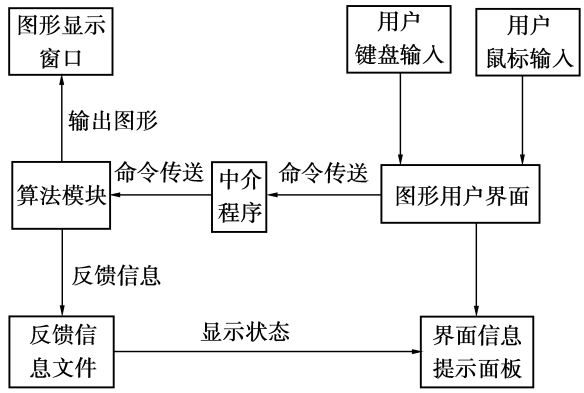

图 2　中介程序工作流程

两件工作。GUI 界面程序则使用一种特殊的命令行方式调用中介程序 proxy

proxy - h <n> <command> [arg1] [arg2]…

其中:-h 后跟的是 GUI 界面程序提供的管道句柄,由 GUI 界面程序自动将其转换为十进制数字;proxy 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。

1)中介程序 proxy.exe 代码。此文件需要与主界面执行程序和算法模块执行程序在同一文件夹中。

```
//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();
  hRead = (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入
if((child_input = _popen( command_line.c_str(),
"wt" )) == NULL)
```

```
  exit( 1 );
if(hRead)
{
  while(1)
  {
    memset(psBuffer, 0, BUFFER_LEN);
    if(ReadFile(
    hRead, psBuffer, BUFFER_LEN,
    &dwReaded, NULL)&& dwReaded > 0)
    {
      puts(psBuffer, child_input);
      flush(child_input);
      psBuffer[4] = 0;
      if (! stricmp(psBuffer, "quit"))
    break;
    }
  }
}
return _pclose(child_input);
```

2)界面程序通过中介程序与算法模块通信的代码

```
//后台进程的启动界面设置
siStartInfo.cb=sizeof(STARTUPINFO);
siStartInfo.lpReserved=NULL;
siStartInfo.lpDesktop=NULL;
siStartInfo.lpTitle="Command Input Mode";
siStartInfo.dwX=0;
siStartInfo.dwY=0;
siStartInfo.cbReserved2=0;
siStartInfo.lpReserved2=NULL;
siStartInfo. wShowWindow = SW_SHOWNOAC-
TIVATE;
saAttr.nLength=0;
sizeof(SECURITY_ATTRIBUTES);
saAttr.bInheritHandle=TRUE;
saAttr.lpSecurit·escriptor=NULL;
if (! CreatePipe(&hRead, &hWrite, &saAttr,
0))
{
    ShowMessage("创建管道失败,无法与后台程
序建立连接,程序将退出!");
    Form1->Close();
}
//准备 proxy.exe 的命令行,在命令行给出写管道
句柄和要 proxy.exe 执行的命令
memset(&pi, 0, sizeof(pi));
sprintf(command_line,
  "proxy-h %d background.exe",
  (unsigned int)hRead);
ZeroMemory( &siStartInfo,
  sizeof(STARTUPINFO));.
//启动后台 back 程序/
if(! CreateProcess( NULL, command_line,
NULL, NULL, TRUE,
  0, NULL, NULL, &siStartInfo, &pi))
{
  ShowMessage("调用 proxy.exe 时失败,无法与
后台程序建立连接,程序将退出!");
  Form1->Close();
}
hProcess = pi.hProcess;
SendCommand("…");
//程序执行完毕,关闭管道
SendCommand("quit\n");
CloseHandle(hRead);
CloseHandle(hWrite);
//发送用户命令函数
void SendCommand(const char* command)
{
  DWORD dwWritten;
  WriteFile(hWrite, command, strlen(command),
&dwWritten, NULL);
}
```

3.2 应用实例

以“View”命令为例说明代码实现过程[8-10]。界面按钮“View”的作用是用户点击后在图形显示窗口显示细分几何模型。代码编写中,在“View”按钮的OnClick事件中添加代码 SendCommand(“view\n”),消息提示面板从文件“warning.txt”中读取反馈信息并显示即可,命令的传送、消息反馈等均自动实现。采用中介程序的好处在于今后的细分算法模块程序在扩充功能和执行效率优化后,只要保持其命令关键字就可以在界面增加实现模块功能的菜单项,程序的可维护性和可扩充性较好。图3是造型系统中的系统集成流程。图4是课题组研制的细分曲面造型系统界面截图,图5是采用此中介程序的方法实现的另一个造型系统的后处理模块界面截图,2个系统均已通过项目验收。

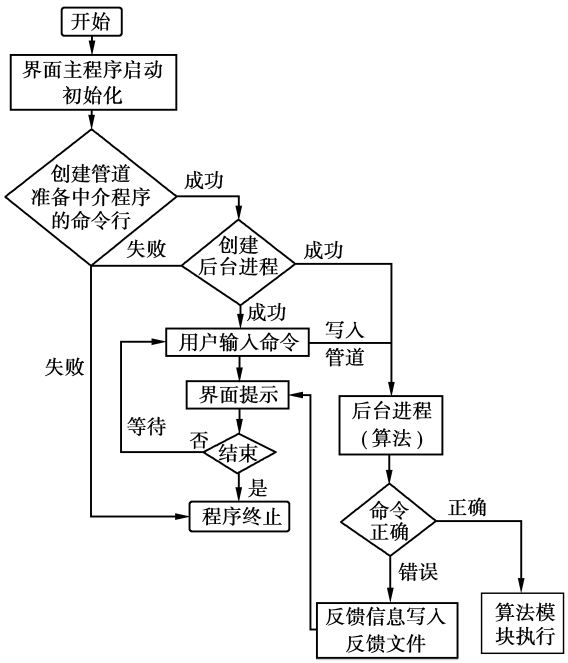

图 3　造型系统流程图(系统集成部分)

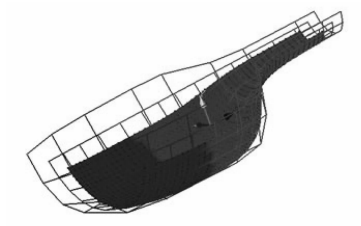

图 4　曲面造型系统

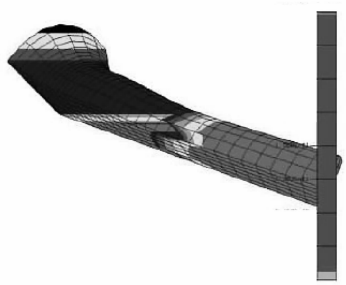

图 5　处理后模块

4　结　语

研究提到的细分曲面造型系统是针对航空航天领域复杂曲面的特殊设计要求而开发的,采用的是多进程系统,通过管道、共享内存等进程间通信技术,灵活地解决了为某些控制台程序方式模块而增加Windows GUI界面的问题。该应用具有实时性、灵活性和可扩展性等特性,可以方便地进行系统集成和扩充。笔者编写的中介模块代码经过适当修改后即可应用于类似的控制台和 GUI 程序之间的交互。

参考文献:

[1] Borland/Inprise 公司著. C++Builder5 开发人员指南[M]. 梁志刚,汪浩,译. 北京:机械工业出版社, 2000.

[2] 杜明芳,程红,周晓阳,等. 基于 OPC 和多进程的集成系统网关[J]. 电气应用,2005(10):123-126.
DU MIN-FANG, CHEN HONG, ZHOU XIAO-YANG. Design and realization of gateway based on OPC and multi-process[J]. Electro-technical Application, 2005(10):123-126.

[3] XIANG DONG, WANG RUN-XIAO. Research and implementation of multi-tier and OPC-based process control system[J]. Journal of Computer Applications, 2003(2):68-70.

[4] 梁庚, 白焰. Windows 下进程间通信方式探讨[J]. 微型电脑应用, 2006(12):58-60.
LIANG GEN, BAI YAN. The modes of communications between processess in windows applications[J]. Microcomputer Application, 2006(12):58-60.

[5] 夏季, 陈国华. Windows 并行开发及其在实时监测系统中的应用[J]. 兵工自动化, 2002(4):44-46.
XIA JI, CHEN GUO-HUA. Parallel development on windows and its application in real-time supervision and testing system[J]. Ordnance Industry Automation, 2002(4):44-46.

[6] 刘恒, 朱正为. 基于 MFC 的 Windows 下多进程消息通信技术及应用[J]. 西南科技大学学报, 2003(3):1-5.
LIU HENG, ZHU ZHENG-WEI. Message communication techonology between processes based on MFC in windows and the application[J]. Journal of Southwest University of Science and Technology, 2003(3):1-5.

[7] 谭夏梅, 何宝新. Windows 环境下 CAD 系统进程通信的实现[J]. 微处理机, 2003(3):27-28.
TAN XIA-MEI, HE BAO-XIN. Design CAD inter-process communication & environment on MS-windows platform[J]. Micro-Processors, 2003(3):27-28.

[8] 梁金千, 管晓宏. 基于进程的访问控制模型[J]. 计算机工程, 2007(2):25-27.
LIANG JIN-QIAN, GUAN XIAO-HONG. Process-based access control model[J]. Computer Engineering, 2007(2):25-27.

[9] 郑培余, 姚绍文. 一种基于进程外窗口挂钩回调的软件集成方法[J]. 计算机应用, 2004(5):109-112.
ZHEN PEI-YU, YAO SHAO-WEN, A method of software Integration based on out-process windows hook callback function[J]. Journal of Computer Application, 2004(5):109-112.

[10] WANG DONG-SHENG. Checkpointing and roolback recovery for windows nt applications[J]. Journal of Computer Research and Development, 2001(1):33-36.

(编辑　侯　湘)