断点续传和多线程下载(上)

撰文／ 行舟

现在，不要说编写专门的下载软件，在自己编写的软件中，加入下载功能有时也非常必要。如让自己的软件支持自动在线升级，或者在软件中自动下载新的数据进行数据更新，这都是很有用、而且很实用的功能。本文的主题即怎样编写一个支持“断点续传”和“多线程”的下载模块。

关键词：下载 断点续传 多线程 WinInet Socket

概述

在当今的网络时代，下载软件是使用最为频繁的软件之一。几年来，下载技术也在不停地发展。最原始的下载功能仅仅是个“下载”过程，即从WEB服务器上连续地读取文件。其最大的问题是，由于网络的不稳定性，一旦连接断开使得下载过程中断，就不得不全部从头再来一次。

随后，“断点续传”的概念就出来了，顾名思义，就是如果下载中断，在重新建立连接后，跳过已经下载的部分，而只下载还没有下载的部分。

无论“多线程下载”技术是否洪以容先生的发明，洪以容使得这项技术得到前所未有的关注是不争的事实。在“网络蚂蚁”软件流行开后，许多下载软件也都纷纷效仿，是否具有“多线程下载”技术、甚至能支持多少个下载线程都成了人们评测下载软件的要素。“多线程下载”的基础是WEB服务器支持远程的随机读取，也即支持“断点续传”。这样，在下载时可以把文件分成若干部分，每一部分创建一个下载线程进行下载。

现在，不要说编写专门的下载软件，在自己编写的软件中，加入下载功能有时也非常必要。如让自己的软件支持自动在线升级，或者在软件中自动下载新的数据进行数据更新，这都是很有用、而且很实用的功能。当然，下载的过程非常复杂，在一篇文章中难以全部阐明，所以，与下载过程关系不直接的部分基本上都忽略了，如异常处理和网络错误处理等，敬请各位读者注意。我使用的开发环境是C++ Builder 5.0，使用其他开发环境或者编程语言的朋友请自行作适当修改。

HTTP协议简介

下载文件是电脑与WEB服务器交互的过程，它们交互的“语言”的专业名称是协议。传送文件的协议有多种，最常用的是HTTP（超文本传输协议）和FTP（文件传送协议），我采用的是HTTP。

HTTP协议最基本的命令有三条：Get、Post和Head。Get从WEB服务器请求一个特定的对象，比如HTML页面或者一个文件，WEB服务器通过一个Socket连接发送此对象作为响应；Head命令使服务器给出此对象的基本描述，比如对象的类型、大小和更新时间。Post命令用于向WEB服务器发送数据，通常使把信息发送给一个单独的应用程序，经处理生成动态的结果返回给浏览器。下载文件即是通过Get命令实现。

基本的下载过程

编写下载程序，可以直接使用Socket函数，但是这要求开发人员理解、熟悉TCP/IP协议。为了简化Internet客户端软件的开发，Windows提供了一套WinInet API，对常用的网络协议进行了封装，把开发Internet软件的门槛大大降低了。我们需要使用的WinInet API函数如图1所示，调用顺序基本上是从上到下，其具体的函数原型请参考MSDN。

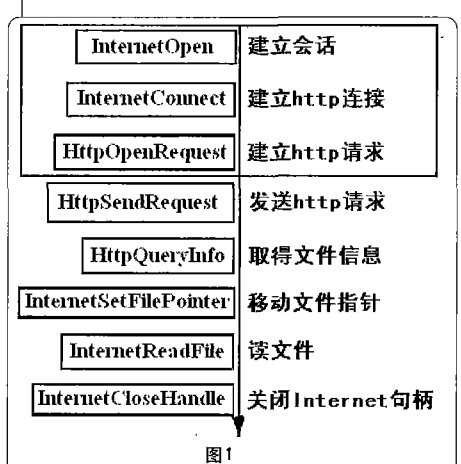

图1

在使用这些函数时，必须严格区分它们使用的句柄。这些句柄的类型是一样的，都是HINTERNET，但是作用不同，这一点非常让人迷惑。按照这些句柄的产生顺序和调用关系，可以分为三个级别，下一级的句柄由上一级的句柄得到。

InternetOpen是最先调用的函数，它返回的HINTERNET句柄级别最高，我习惯定义为hSession，即会话句柄。

InternetConnect使用hSession句柄，返回的是http连接句柄，我把它定义为hConnect。

HttpOpenRequest使用hConnect句柄，返回的句柄是http请求句柄，定义为hRequest。

HttpSendRequest、HttpQueryInfo、InternetSetFilePointer和InternetReadFile都使用HttpOpenRequest返回的句柄，即hRequest。

当这几个句柄不再使用时，应该用函数InternetCloseHandle把它关闭，以释放其占用的资源。

首先建立一个名为THttpGetThread、创建后自动挂起的线程模块，我希望线程在完成后自动销毁，所以在构造函数中设置：

```
FreeOnTerminate = True;  // 自动删除
```

并增加以下成员变量：

```
char Buffer[HTTPGET_BUFFER_MAX+4];        // 数据缓冲区
AnsiString FURL;          // 下载对象的URL
AnsiString FOutFileName;  // 保存的路径和名称
HINTERNET FhSession;      // 会话句柄
HINTERNET FhConnect;      // http连接句柄
```

```
HINTERNET FhRequest;        // http请求句柄
bool  FSuccess;             // 下载是否成功
int iFileHandle;            // 输出文件的句柄
```

1、建立连接

按照功能划分，下载过程可以分为4部分，即建立连接、读取待下载文件的信息并分析、下载文件和释放占用的资源。建立连接的函数如下，其中ParseURL的作用是从下载URL地址中取得主机名称和下载的文件的WEB路径，DoOnStatusText用于输出当前的状态：

```
//初始化下载环境
void THttpGetThread::StartHttpGet(void)
{
    AnsiString HostName,FileName;
    ParseURL(HostName, FileName);
    try
    {
        // 1.建立会话
        FhSession = InternetOpen("http-get-demo",
                    INTERNET_OPEN_TYPE_PRECONFIG,
                    NULL,NULL,
                    0);          // 同步方式
        if( FhSession==NULL)throw(Exception("Error: InterOpen"));
        DoOnStatusText("ok: InterOpen");
        // 2.建立连接
        FhConnect=InternetConnect(FhSession,
                    HostName.c_str(),
                    INTERNET_DEFAULT_HTTP_PORT,
                    NULL,NULL,
                    INTERNET_SERVICE_HTTP, 0, 0);
        if(FhConnect==NULL)throw(Exception("Error: InternetConnect"));
        DoOnStatusText("ok: InternetConnect");
        // 3.初始化下载请求
        const char *FAcceptTypes = "*/*";
        FhRequest = HttpOpenRequest(FhConnect,
                    "GET",              // 从服务器获取数据
                    FileName.c_str(),   // 想读取的文件的名称
                    "HTTP/1.1",         // 使用的协议
                    NULL,
                    &FAcceptTypes,
                    INTERNET_FLAG_RELOAD,
                    0);
        if( FhRequest==NULL)throw(Exception("Error: HttpOpenRequest"));
        DoOnStatusText("ok: HttpOpenRequest");
        // 4.发送下载请求
        HttpSendRequest(FhRequest, NULL, 0, NULL, 0);
        DoOnStatusText("ok: HttpSendRequest");
    }catch(Exception &exception)
    {
        EndHttpGet();  // 关闭连接，释放资源
        DoOnStatusText(exception.Message);
    }
}
// 从URL中提取主机名称和下载文件路径
void THttpGetThread::ParseURL(AnsiString &HostName,AnsiString &FileName)
{
    AnsiString URL=FURL;
    int i=URL.Pos("http://");
    if(i>0) URL.Delete(1, 7);
    i=URL.Pos("/");
    HostName = URL.SubString(1, i-1);
    FileName = URL.SubString(i, URL.Length());
}
```

可以看到，程序按照图1中的顺序，依次调用InternetOpen、InternetConnect、HttpOpenRequest函数得到3个相关的句柄，然后通过HttpSendRequest函数把下载请求发送给WEB服务器。

InternetOpen的第一个参数是无关的，最后一个参数如果设置为INTERNET_FLAG_ASYNC，则将建立异步连接，这很有实际意义，考虑到本文的复杂程度，我没有采用。但是对于需要更高下载要求的读者，强烈建议采用异步方式。

HttpOpenRequest打开一个请求句柄，命令是“GET”，表示下载文件，使用的协议是“HTTP/1.1”。

另外一个需要注意的地方是HttpOpenRequest的参数FAcceptTypes，表示可以打开的文件类型，设置为“*/*”表示可以打开所有文件类型，可以根据实际需要改变它的值。

2、读取待下载文件的信息并分析

在发送请求后，可以使用HttpQueryInfo函数获取文件的有关信息，或者取得服务器的信息以及服务器支持的相关操作。对于下载程序，最常用的是传递HTTP_QUERY_CONTENT_LENGTH参数取得文件的大小，即文件包含的字节数。模块如下所示：

```
// 取得待下载文件的大小
int __fastcall THttpGetThread::GetWEBFileSize(void)
{
    try
    {
        DWORD BufLen=HTTPGET_BUFFER_MAX;
        DWORD dwIndex=0;
        bool RetQueryInfo=HttpQueryInfo(FhRequest,
                HTTP_QUERY_CONTENT_LENGTH,
                Buffer, &BufLen,
                &dwIndex);
        if( RetQueryInfo==false) throw(Exception("Error: HttpQueryInfo"));
        DoOnStatusText("ok: HttpQueryInfo");
        int FileSize=StrToInt(Buffer); // 文件大小
        DoOnGetFileSize(FileSize);
    }catch(Exception &exception)
    {
        DoOnStatusText(exception.Message);
    }
    return FileSize;
}
```

模块中的DoOnGetFileSize是发出取得文件大小的事件。取得文件大小后，对于采用多线程的下载程序，可以按照这个值把文件进行合适的分块，确定每个文件块的起点和大小。

3、下载文件的模块

开始下载前，还应该先安排怎样保存下载结果。方法很多，我直接采用了C++ Builder提供的文件函数打开一个文件句柄。当然，也可以采用Windows本身的API，对于小文件，全部缓冲到内存中也可以考虑。

```
// 打开输出文件，以保存下载的数据
DWORD THttpGetThread::OpenOutFile(void)
{
    try
    {
        if(FileExists(FOutFileName))
            DeleteFile(FOutFileName);
        iFileHandle=FileCreate(FOutFileName);
        if(iFileHandle==-1) throw(Exception("Error: FileCreate"));
        DoOnStatusText("ok:CreatFile");
    }catch(Exception &exception)
    {
        DoOnStatusText(exception.Message);
    }
    return 0;
}
// 执行下载过程
void THttpGetThread::DoHttpGet(void)
{
```

```
DWORD dwCount=OpenOutFile();
try
{
    // 发出开始下载事件
    DoOnStatusText("StartGet: InternetReadFile");
    // 读取数据
    DWORD dwRequest;        // 请求下载的字节数
    DWORD dwRead;           // 实际读出的字节数
    dwRequest=HTTPGET_BUFFER_MAX;
    while(true)
    {
        Application->ProcessMessages();
        bool ReadReturn = InternetReadFile(FhRequest,
                (LPVOID)Buffer,
                dwRequest,
                &dwRead);
        if(!ReadReturn)break;
        if(dwRead==0)break;
        // 保存数据
        Buffer[dwRead]='\0';
        FileWrite(iFileHandle, Buffer, dwRead);
        dwCount = dwCount + dwRead;
        // 发出下载进程事件
        DoOnProgress(dwCount);
    }
    Fsuccess=true;
}catch(Exception &exception)
{
    Fsuccess=false;
    DoOnStatusText(exception.Message);
}
FileClose(iFileHandle);
DoOnStatusText("End: InternetReadFile");
}
```

下载过程并不复杂，与读取本地文件一样，执行一个简单的循环。当然，如此方便的编程还是得益于微软对网络协议的封装。

4、释放占用的资源

这个过程很简单，按照产生各个句柄的相反的顺序调用InternetCloseHandle函数即可。

```
void THttpGetThread::EndHttpGet(void)
{
    if(FConnected)
    {
        DoOnStatusText("Closing:InternetConnect");
        try
        {
            InternetCloseHandle(FhRequest);
            InternetCloseHandle(FhConnect);
            InternetCloseHandle(FhSession);
        }catch(...){}
        FhSession=NULL;
        FhConnect=NULL;
        FhRequest=NULL;
        FConnected=false;
        DoOnStatusText("Closed:InternetConnect");
    }
}
```

我觉得，在释放句柄后，把变量设置为NULL是一种良好的编程习惯。在这个示例中，还出于如果下载失败，重新进行下载时需要再次利用这些句柄变量的考虑。

5、功能模块的调用

这些模块的调用可以安排在线程对象的Execute方法中，如下所示：

```
void __fastcall THttpGetThread::Execute()
{
    int FrepeatCount=5;
    for(int i=0;i<FRepeatCount;i++)
    {
        StartHttpGet();
        GetWEBFileSize();
        DoHttpGet();
        EndHttpGet();
        if(FSuccess)break;
    }
    // 发出下载完成事件
    if(FSuccess)DoOnComplete();
    else DoOnError();
}
```

这里执行了一个循环，即如果产生了错误自动重新进行下载，实际编程中，重复次数可以作为参数自行设置。

实现断点续传功能

在基本下载的代码上实现断点续传功能并不是很复杂，主要的问题有两点：

1、检查本地的下载信息，确定已经下载的字节数。所以应该对打开输出文件的函数作适当修改。我们可以建立一个辅助文件保存下载的信息，如已经下载的字节数等。我处理得较为简单，先检查输出文件是否存在，如果存在，再得到其大小，并以此作为已经下载的部分。由于Windows没有直接取得文件大小的API，我编写了GetFileSize函数用于取得文件大小。注意，与前面相同的代码被省略了。

```
DWORD THttpGetThread::OpenOutFile(void)
{
    ......
    if(FileExists(FOutFileName))
    {
        DWORD dwCount=GetFileSize(FOutFileName);
        if(dwCount>0)
        {
            iFileHandle=FileOpen(FOutFileName,fmOpenWrite);
            FileSeek(iFileHandle,0,2);   // 移动文件指针到末尾
            if(iFileHandle==-1) throw(Exception("Error: FileCreate"));
            DoOnStatusText("ok:OpenFile");
            return dwCount;
        }
        DeleteFile(FOutFileName);
    }
    ......
}
```

2、在开始下载文件（即执行InternetReadFile函数）之前，先调整WEB上的文件指针。这就要求WEB服务器支持随机读取文件的操作，有些服务器对此作了限制，所以应该判断这种可能性。对DoHttpGet模块的修改如下，同样省略了相同的代码：

```
void THttpGetThread::DoHttpGet(void)
{
    DWORD dwCount=OpenOutFile();
    if(dwCount>0)   // 调整文件指针
    {
        dwStart = dwStart + dwCount;
        if(!SetFilePointer())   // 服务器不支持操作
        {
            // 清除输出文件
            FileSeek(iFileHandle,0,0);   // 移动文件指针到头部
        }
    }
    ......
}
```

断点续传到此就完成了，下一期将介绍多线程下载模块。