用 VC++ 多线程技术 实现 PC 机全双工串行通讯

章　玲

(南通供电公司, 南通, 226006)

[摘要]　讨论了用 VC++ 中的多线程技术, 及用多线程实现 PC 机串口全双工通讯的方法, 并进一步探讨了全双工通讯在滑动窗口协议中的应用

[关键词]　多线程, 全双工通讯, 滑动窗口
[中图分类号] TP31;　[文献标识码] B;　[文章编号] 1672- 1292(2002) 01- 0038- 04

Win32 支持争先式多任务和多线程编程, 因而 Windows 应用程序的编写有了很大的变化。相对单工和半双工特性而言, 全双工串行通讯的管理要复杂性得多. 本文就应用 VC++ 的多线程技术实现全双工串行通讯问题进行探讨.

1　关于线程的三个问题

1.1　线程的概念

线程　是指程序指令顺序的执行. Windows 98 或 Windows NT 下程序可以启动一个或几个辅助线程, 每个线程独立执行程序代码中的一系列指令. 从用户或应用程序编程人员的角度看, 进程中各线程是同时运行的. 操作系统在线程与线程之间快速切换实现宏观视觉的同时. 程序在某个时间需要完成多个任务时, 将每个任务放在不同的线程中, 使程序功能的实现更加有效, 还能简化开发工作.

1.2　Windows 中的两种线程

Windows 提供了两种线程: 辅助线程和用户界面线程。Microsoft Foundation Class(MFC) 库对两种线程都支持. 一个用户界面线程有窗口, 所以它有自己的消息循环, 使程序能迅速响应命令和其它事件; 辅助线程没有消息窗口, 所以它不需要处理消息, 而是用于完成费时的工作, 例如磁盘的操作和串行口的通讯

1.3　多线程间的通讯

可以在一个用户界面线程中启动一个辅助线程. 辅助线程一旦启动, 两个线程就独立运行. 进程中的所有代码和数据空间被进程内的线程所利用, 进程内所有的线程可以访问同一个全局变量. 因此, 主线程与辅助线程间的通讯, 最简单的方法是利用全局变量. 较为复杂的方法是利用 事件 (event).

辅助线程与主线程的通讯可以采用消息队列的方式. 前面提到, 主用户界面线程有自己的消息队列, 所以采用从辅助线程向主线程发送 Windows 消息的方法, 通知主线程一段数据已发送完毕或已收到对方发来的数据. 发送消息有两种方法: SendMessage 和 PostMessage. SendMessage 强迫接收线程立即响应, 但是容易引起重入. PostMessage 函数只将消息放在目标窗口消息队列的队尾, 不会引起重入. 所以使用 PostMessage 函数给窗口发送一个用户自定义消息比较可靠.

收稿日期: 2001- 10- 12
作者简介: 章玲, 女, 1962- , 江苏省南通供电公司苏源自动化设备有限公司高级工程师, 从事电力系统自动化通讯产品的研发工作

2 用多线程实现全双工通讯的必要性

2.1 防止用户界面线程的阻塞

一个线程被阻塞,是指该线程被停止执行.在主用户界面线程里,要避免阻塞调用.如果通过串口发送大量数据的程序段被放在主线程里,主线程就不能及时响应鼠标或键盘命令而处理消息了,这样会使程序显得反应迟缓.

2.2 使程序简洁有效

在 DOS 或 WIN16 方式下,大多采用半双工通讯.程序员必须在程序中确定应该在什么时候发送,什么时候接收,以什么样的频率监测接收缓冲区.现在,有了 Windows95 抢先式多任务操作系统中,收、发工作的时间片切换由系统负责,程序员只需把管理收、发的代码放在不同的线程中.在程序员看来,两者是同时进行的,真正的全双工通讯变得简单多了.当然,如果计算机有多个 CPU,则系统可以直接同时执行多个线程.

2.3 调整辅助线程的优先级灵活性

在函数 AfxBeginThread(, int nPriority= THREAD _ PRIORITY _ NORMAL,)中,nPriority 指定本线程的优先级.线程的优先级确定操作系统在线程与线程间切换控制时,本线程运行的频繁程度.如果某线程要迅速地完成任务,应当赋予其相对较高的优先级.反之,如果线程完成比较不重要的任务,允许在其它线程不活动期间完成,则可以赋予其较低的优先级。程序员可以跟据需要灵活的调整优先级.相比之下,在 VB 中,对串口的操作都是在后台进行的,如果想让操作系统给串口操作更多的时间片,则不得不在程序中加上无数个 DoEvents().因此,使用 VC++ 多线程技术,可以最有效的利用系统现有的资源.

3 全双工串行通讯实现方法

以下是实现串口全双工通讯的例程

3.1 初始化串口

在 VC++ 中,对串口操作与对文件操作一样,所以可以从 Cfile 派生新类 CSerialPort,创建 CSerialPort 类的成员函数 OpenCom,用类似于打开文件的方法打开并初始化串口.应该注意的是,在打开串口之前,要检查一下串口是否已经打开.如果串口已经打开,必须先关闭,然后再重新打开(有关程序略)

3.2 发送数据

在发送前,需在头文件中定义结构体 SENDDATA,并添加一个 SENDDATA 型的私有数据成员 SENDDATA m _ SendData.

```
typedef struct t _ SENDDATA
{
    CFile *  pFile; // 串口
    int*  lpBuf;    // 发送数据缓冲区指针
    int len;        // 数据长度
} SENDDATA;
```

然后,创建 CSerialPort 类的成员函数 Send,发送数据

```
void CSerialPort: : Send( int*  lpBuf, int len)
{
    // 初始化待发送数据 m _ SendData 结构
    m _ SendData. pFile= this;
    m _ SendData. lpBuf= lpBuf
```

```
    m _ SendData. len= len;

    // 启动名为 SendThread 辅助线程, 发送 m _ SendData
    AfxBeginThread(SendThread, &m _ SendData);
}
unsigned SendThread(LPVOID pParam)
{
    SENDDATA * pSend= (SENDDATA * ) pParam;
    // 实际发送数据
    pSend- > pFile- > Write(pSend- > lpBuf, pSend- > len);
    return(0);
}
```

3.3 接收数据

聆听线程用于监视对端是否有数据发送过来. 在这段程序中, 借用了 VC++ Socket 通讯中聆听的概念. 应用程序一起动, 就开始聆听对端是否有报文送过来, 如有报文发来就先接收一个固定长度的报头, 并从报头中的某个位置得到等待接收的报文体的长度, 最后接收报文体. 用这种方法, 每次接收或发送报文体的长度可变, 比较灵活.

如果接收方不能以足够快的速度处理报文, 还要建立一个报文队列, 将新来的报文添加到队列的尾部. 这样, 报文处理程序就可以按照先来先处理的顺序逐个处理报文. 为此, 应从 CObject 中派生出用于存放报文的类 CPacket, 从 CObList 类中派生出一个存放报文队列的类 CQueue.

```
// CQueue. cpp
CPacket: : CPacket( int id, int* pHdr, int* pBody, int len, int error);
{
    m _ nID= id;
    m _ pHdr= pHdr;
    m _ pBody= pBody;
    m _ len= len;
    m_error= error;
}
CPacket: : ~ CPacket()
{
    delete[ ]m _ pHdr;
    delete[ ]m _ pBody;
}
void CQueue: : Add( CPacket * pPacket)
{
    CSingleLock slock( &m _ mutex);
    if(slock. Lock(1000)) // timeout in millseconds, default= INFINITE
    {
        AddTail(pPacket);
    }
}
CPacket * CQueue: : Remove()
{
```

```
    CSingleLock slock(&m_mutex);
    if(slock.Lock(1000))
    {
        if(!IsEmpty())
            return(CPacket*)RemoveHead();
    }
    return NULL;
}
```

3.4 串口全双工通讯的应用

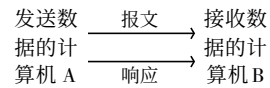

图1 滑动窗口通信

计算机串口全双工通讯可被用于物理层采用 RS232 协议, 传输差错处理采用滑动窗口协议的通讯工作, 如图 1 所示

设计算机 A 发送的数据序列号为 n, 计算机 B 发送的响应号为 r. 发送方可以一次连续发送多块数据, 最大数据块数的限制 m 称为窗口尺寸(window size), 即 $n = 0, 1, \quad m - 1$; 接收方在接到第一个数据块时启动计时器, 随后对收到的每一块数据块进行差错分析, 如果发现错误, 立即反馈发送方, 指出出错的数据块 r, 并重置计时器; 否则接收方等待计时器超时, 对接收到的多个正确的数据块进行一次性确认; 发送方根据反馈的结果, 即可以重发指定的数据块, 也可以重发指定数据块及其后的所有数据块, 或者连续发送后继的数据块.

实际上, 滑动窗口协议是等 停协议的改进, 与等 停协议比较, 滑动窗口协议的效率较高, 尤其是在合理的选择了接收方的计时值之后, 效果更为明显. 很明显, 滑动窗口协议需要全双工信道的支持. 并且, 发送方在发送完某个数据块, 但未接收到最终的一次性响应之前, 需要保存该数据块, 以便重发, 所以管理发送数据队列和接收数据队列比半双工要复杂. VC++ 提供了 CObList 类, 可以充分利用它管理收发队列.

PC 机的串行通讯的文章中, 可以用 Mscomm 实现, 也可以用文件操作实现的, 但大多是关于半双工通讯, 停 等协议的通讯. 本文在全双工通讯方面做了一些研究, 并在此基础上改进停 等协议, 实现了滑动窗口协议.

[参考文献]

[1] 李大友. 微型计算机接口技术[M]. 北京: 清华大学出版社, 1999.
[2] 吴国新. 计算机网络[M]. 南京: 东南大学出版社, 2000.
[3] David J. Kruglinski. Visual C++ 技术内幕[M]. (第四版). 北京: 清华大学出版社, 1999.
[4] John E. Swanke. Visual C++ MFC 扩展编程实例[M]. 北京: 机械工业出版社, 1999.

Multithread Technology for Serial Port Dual Communication

Zhan Lin

(Nangtong Power Compary, 226006, Nangtong, PRC)

Abstract: This serial port dual communication by adopting multifhread techmology in VC++ programming is disucssed. The appliation of dual communication to glide windows protocal is further explored.

Key words: multithread, dual communication, glide windows

[责任编辑: 严海琳]