C++ Builder 控件技术在自助售票机中的应用*

金瓯 樊玮虹 朱关德

(国防科技大学电子工程学院 长沙 410073)

摘 要 用C++ Builder强有力的控件技术及多线程技术,实现一个32位串口控件,并将其应用于串行设备软件接口,走出传统的由动态链接库提供低层接口软件的方式,从而大大减轻上层应用软件的开发负担。

关键词 自助售票机,C++ Builder,控件串行通讯,多线程

分类号 TP371

C++ Builder Component Application in ATVM

Jin Ou Fan Weihong Zhu Guande

(Institute of electronic Engineering, NUDT, Changsha, 410073)

Abstract By means of the C++ Builder powerful component technology and multithread technology, a 32-bit serial communication component instead of the traditional DLL can be applied to the serial communication device in ATVM and reduce the burden in developing the application software.

Key words ATVM, C++ Builder, component serial communication, multithread

由国防科技大学、中科院软件所、沈阳铁路科研所等单位联合开发的第一台国产(现金)铁路自助售票机,集现金识别处理、IC卡磁卡、车票印制、网络通信、计算机软硬件等技术于一身,其结构复杂,设备众多,仅串口设备就达十几个。能否快速准确地对这些设备进行控制成为整个软件系统成败的关键。传统的方法是通过为每一个串口设备提供一个动态链接库来实现其接口,这种方法的缺点在于:对于许多设备的状态变化情况,应用程序是需要实时掌握的,例如对磁卡读写器,当顾客插入磁卡时,应用程序应立刻作出反应。为了实现这一目的,应用程序不得不定时地进行状态查询,由此带来的问题就是,如果查询周期太短,将会使用户操作界面反应迟钝,加长查询周期又会使得对设备的状态变化不能及时处理,这对有些设备(如非接触式IC卡)是不允许的。采用C++ Builder的控件技术能够解决动态链接库接口方式中的这些缺点,而且更符合C++面向对象的编程技术。

C++ Builder是Borland公司为编写C++应用程序而开发的新的快速应用开发产品。它不但具有Delphi快速开发创建用户界面程序的能力,而且拥有C++赋予的全部功能。然而遗憾的是,C++ Builder的控件库中并没有串口通信控件,为了用控件技术来实现串口设备的底层软件接口,首先要做的就是实现一个串口控件,然后在此基础上将各个串口设备建成控件。

1 实现串口控件

1.1 设计方案

串口控件设计的基本原则是:能将主动将收到的数据和串口的状态信息通知应用程序,尽量减少对系统资源的占用。为了使串口控件能自主地对串口的状态作出反应,必须在串口控件内部使用系统时钟或者子线程,完成对串口的读写及端口状态监视。采用定时方式时,每一个串口控件的实例将会消耗掉系统的一个时钟资源,而且无论有无数据收发,在每一个定时周期都要消耗CPU资源。这在使用串口控件比较少的应用系统中不会带来很大问题,但在串口设备较多的应用场合,如前面提到的自助售票机

* 1999年6月1日收稿
第一作者:金瓯,男,1965年生,副教授

中的串口设备达十几个之多,就可能造成耗尽所有的时钟资源,并且会降低应用程序的运行速度,此外使用定时方式还会使控件对端口状态的反应迟钝。而采用多线程技术,加上 WINDOWS 异步串口读写技术的支持,则会使控件对端口的状态作出快速反应,而且在没有数据收发时,只需极少的 CPU 资源。

1.2 串口控件的结构

串口控件的结构如图 1 所示。由消息处理、数据发送、属性及其它成员函数构成控件的主体部分,用一个子线程来完成对串口的低层操作。子线程是在串口打开时创建的,其所需的一些参数在创建时由主体部分传给它。子线程不断对端口进行监视,当收到数据或发现错误状态时,向主体部分发送一条消息(TMessage),主体部分的消息处理函数则将相应的信息通过事件(Event)通知应用程序。从多线程的技术角度来讲,控件的主体部分属于调用它的线程(Calling thread),因此在进行数据发送时,不是将数据写入端口,而是用事件通知子线程,等待其处理要发送的数据。

子线程的结构流程如图 2 所示。在正常情况下,子线程调用 WINDOWS 的一个等待函数 WaitForMultipleObjects(),处于一种等待状态,在此状态下,它只消耗极少的 CPU 资源。

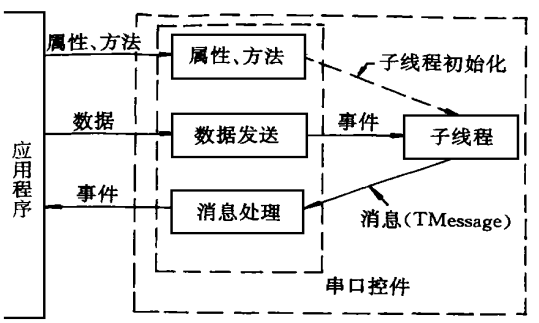

图 1 串口控件的结构

Fig. 1 structure of the communication component

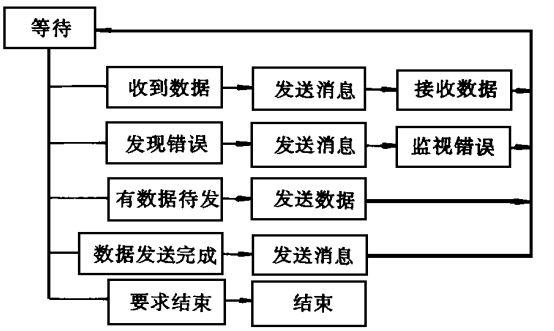

图 2 子线程的结构流程

Fig. 2 structure process of the subthread

一旦等待的事件有一个或多个发生,WaitForMultipleObjects() 就会返回,根据返回的结果进行相应的处理:

异步读(Overlapped)操作完成——向主体部分发送一个消息,将读到的数据传给它,然后开始下一次新的读操作,以等待新的数据;

发现端口错误——清除端口错误,向主体部分发送一个消息,将错误状态给它,然后继续监视端口状态;

异步写操作完成——向主体部分发送一个消息,将已写出数据传给它:

有数据待发——发送数据,等待新的数据发送任务;

要求结束——子线程结束执行。

1.3 控件实现中的技术问题

为了利用多线程技术来实现串口控件,在调用 WINDOWS 相应的 API 函数时,应采用其异步操作方式,即在调用 CreateFile, ReadFile, WriteFile, WaitCommEvent 时,采用下面的形式:

```
hCommFile = CreateFile (lpszComPortName, GENERIC_READ + GENERIC_WRITE, 0, NULL,
                        OPEN_EXISTING, FILE_FLAG_OVERLAPPED, NULL);
ReadFile(hCommFile, szInputBuffer, dwSizeofBuffer, &NumberOfBytesRead, lpOverlappedRead);
WriteFile (hCommFile, lpDataToWrite, dwNumberDataToWrite, &dwNumberOfBytesWritten,
           lpOverlappedWrite));
WaitCommEvent(hCommFile, lpdwCommEventMask, lpOverlappedEvent))。
```

2 以控件方式来提供串口设备的低层接口

设计好一个串口控件后,就可在其基础上将串口设备实现成一个控件。以自助售票机中的现金处理

设备入钞器(Banknote Acceptor) 为例, 假定串口控件为 TCommPort, 其实现的步骤如下:
第一步: 由 TCommPort 派生出新的控件 TAcceptor

```
class TAcceptor: public TCommport
    {

    }
```

第二步: 增加新的属性和事件及方法(Method)
例如在入钞器中, 现金投入、退回以及压入钱箱都是应用程序所关心的, 因此增加下面的事件以在相应的事件发生时能主动通知应用程序:

```
typedef void_ fastcall (_closure * TOnBillAccepted)(WORD nBankNote, WORD nType);
typedef void_ fastcall (_closure * TOnBillRefunded)(WORD nRefundedNotes);
typedef void_ fastcall (_closure * TOnBillEncashed)(WORD nEncashedNotes);
class Tacceptor: public Tcommport{
_published:
        _property TOnBillAccepted OnBillAccepted= {read= FOnBillAccepted, write= FOnBillAc-
                                                    cepted};
        _property TOnBillRefunded OnBillRefunded= {read= FOnBillRefunded, write=
                                                    FOnBillRefunded};
        _property TOnBillEncashed OnBillEncashed= {read= FOnBillEncashed, write=
                                                    FOnBillEncashed};
}
```

第三步: 重载 TCommport 的相关方法(Method)
假定 TCommport 在接收到数据后是通过一个函数 HandleReceivedData(char * szBuffer) 来处理的, 在 TAcceptor 内对它进行重载:

```
void Tacceptor::HandleReceivedData(char * szBuffer)
{
  DecodeTheReceivedData()
  Switch(DecodeResult)
  {
  case Bill_Inserted:
    if(FOnBillAccepted) FonBillAccepted(nBankNote, nType);
    break;
    case Bill_Refunded:

  }
}
```

完成以上这些步骤, 一个 Tacceptor 就创建好了。

3 在 C+ + Builder 中使用控件和动态链接库的比较

以上述的入钞器为例, 假设入钞器可接收 5、10、50、100 元人民币, 顾客购买一张车票需付 150 元现金。用控件和动态库的代码如下:
用控件:

```
MyAcceptor-> Enabled= true;
TmyForm:: MyAcceptor BillAccepted(nBankNote, nType)
```

```
{
nTotalInserted+ = nBankNote* nType
if(nTotalInserted> = 150)
  {
  MyAcceptor- > Enabled= false
  //Continue
  }
}
```

用动态链接库:

```
EnableAcceptor();
TmyForm:: MyTimerTimer()
{
int nBill5, nBill10, nBill50, nBill100;
GetInsertedBill( &nBill5, &nBill10, &nBill50, &nBill100);
nTotalInserted+ = nBill5* 5+ nBill10* 10+ nBill50* 50+ nBill100* 100;
if(nTotalInserted> = 150)
  {
  DisableAcceptor();
  //Continue
}
  }
```

从代码的长度来看区别不是很大,但在代码的风格上,采用控件时是标准的面向对象编程风格,用动态链接库时却没有这种特点。从代码在运行中的效率来看,在控件方式下,由于它对入钞事件的响应,因此它的运行效率是100%。采用动态链接库方式时,如果要求应用程序在100ms内对入钞事件作出反应,那么时钟的定时周期应≤100ms。在实际过程中,顾客投入一张钞票需要3～5s,因此代码运行的效率小于3.33%。

4 小结

用控件的方式来实现串口设备的低层接口,给上层应用系统的开发带来了很大的方便。控件方式使应用系统开发人员能很好地脱离具体的物理设备,集中精力去进行系统的功能设计。因为对于开发人员来讲,一个串口设备和一个屏幕上的按钮没有多大区别。在具有众多设备需要控制的系统中,如铁路自助售票机,采用这种方式是一个很好的选择,这在实际应用中也得到了证实。

参考文献

1 Charlie Calvert. Charlie Calvert's C++ Builder Unleashed, 1996

2 Matt Tellers. High Performance Borland C++ Builder, 1997