维普资讯 http://www.cqvip.com

计算机科学 1995 Vol. 22№. 4

中文信息处理国际化研究(下)

陈险峰 蔡希尧 金益民
(西安电子科技大学 西安 710071)

四、XDUCS Windows 的实现

XDUCS Windows 主要由输入模块、输出模块、代码转换模块和其它辅助模块组成。

1. 输入的实现

UCS 字符的输入是利用 MS-Windows 提供的系统钩子(Hook)以 DLL 的形式实现的。在 MS-Windows 中,系统钩子是安装特定类型过滤函数的共享资源,过滤函数是一种由应用程序提供的回调函数,它在事件到达任何应用程序消息循环以前处理这些事件。确切地说,Windows 把特定类型事件产生的消息发送到用同样类型钩子安装的过滤函数,在过滤函数中可以对消息进行处理或修改,然后消息才会到达应用程序的消息队列。利用 Windows 提供的 Keyboard Hook,能截获整个系统的所有键盘消息。

XDUCS Windows 的输入由总控模块和语言输入转换模块组成。下面分别介绍各模块的实现:

1.1 总控模块:主要功能是截获键盘输入消息,然后根据当前所选择的语言,调用相应的语言输入转换模块,将键盘输入转换成为 UCS 字符编码,再转发到相应的应用程序中,以实现 UCS 字符的输入。

UCS 能表示世界上几乎所有的自然语言,XDUCS Windows 输入模块也能同时支持多文种的输入。为此,我们把除英文外的所有语言输入转换模块都用动态链接库实现。对应一个键盘输入或者键盘输入序列,无论不同的语言输入转换模块在实现方法上有多大的差异,它们都按照一个统一的接口规范同总控模块接口。这个接口规范定义为一个如下的函数:

```
BOOL ToUCSChar(WORD wInKey,WORD wUCSChar);
```

这个函数具有两个参数,wInKey 是一个无符号整数,函数调用时它被赋予的值是总控模块截获的一个键盘输入消息中的输入键的虚拟键码。wUCSChar 也是一个无符号整数,当语言输入处理模块把键盘输入或输入序列转换成为一 UCS 字符时,应当把 UCS 字符的编码(两字节)赋予此参数,使 ToUCSChar=TRUE 并返回总控模块。否则,函数返回 FALSE 到总控模块。

总控模块截获到一个键盘输入消息时,根据当前用户选择的语言调用相应的语言输入转换模块。在得到一个 UCS 字符编码时,分别以这个双八位字符的高、低字节的值为虚拟键码生成两个字符消息(WM_CHAR),并将它们发送到要求输入的窗口。在 MS-Windows 中一个字符消息的出现总是同一个键的按下消息(WM_KEYDOWN)和抬起消息(WM_KEYUP)紧密相联,具体地说,系统中应出现这样的消息序列:WM_KEYDOWN、WM_CHAR、WM_KEYUP。我们用以下六条发送这两个字符消息:

```
PostMessage(hWnd,WM_KEYDOWN,0,lParam);
PostMessage (hWnd,WM_CHAR,(WORD)((UCS≫8)
            &0x00ff),lParam);
PostMessage(hWnd,WM_KEYUP,0,lParam);
PostMessage(hWnd,WM_KEYDOWN,0,lParam);
PostMessage (hWnd, WM_ CHAR, (WORD) (UCS &
            0x00ff),lParam);
PostMessage(hWnd,WM_KEYUP,0, lParam);
```

按照上面所述接口规范可以实现 UCS 支持的各种自然语言的输入转换模块,加入到系统后在总控模块配合下就可以实现对各种语言的输入支持。

1.2 语言输入转换模块:我们知道,利用西文键盘实现世界上的多数文字输入不可能做到一键一字,汉字的输入就是一个最典型的例子。对于世界上的许多文字的计算机输入都是通过输入编码实现的。另外,现有的计算机键盘结构实际上是以 ASCII 字符输入为基础的。因此,XDUCS Windows 的语言输入处理模块的主要工作是完成输入序列到一个 UCS 字符的转换。在不同的语言输入转换模块中,

维普资讯 http://www.cqvip.com

由于输入规则不同，同一个键盘消息将会有不同的意义。我们实现了英文和中文字符的 UCS 输入。

1.2.1 英文输入：英文处理相对比较简单，因为英文字符的 UCS 编码同 ASCII 编码之间的对应关系简单，另外，在 ISO 10646 中规定所有的选择子集都自动包含原 ASCII 的图形字符。因此，我们把英文作为缺省的语言选择，对它的处理也包含在输入总控模块中。具体处理过程是：键盘消息被截获以后，我们从回调函数的 wParam 中得到输入键的虚拟键码，该键码等同于该键所表示字符的 ASCII 码，比如说 'A' 键被按下，我们得到虚拟键码 41H，也就得到了字符 'A' 的 UCS 编码 0041H。总控程序将产生虚拟键码为 00 和 41H 的字符消息并发送到应用程序中。这样就完成了一个英文 UCS 字符的输入。

1.2.2 中文输入：中文输入是作为 XDUCS Windows 对多种语言支持的一个具体例子来实现的。用西文键盘输入汉字要通过特定的汉字输入编码实现。目前有很多汉字输入方法，如区位码、拼音、五笔字型等，不同的汉字输入方法实际上就是按不同的方式去检索一个汉字。为了能支持多种汉字输入方法，特别是让系统支持一种新的汉字输入方法，我们把中文输入模块设计成一个汉字输入控制模块加多种汉字输入方法的形式。并设计了一个统一的接口规范，每一种汉字输入方法都符合这个规范并用一个 DLL 实现。用户可以选择一种输入方法，控制模块反复调用一个输入方法模块得到一个汉字的 UCS 编码。中文输入方法接口定义如下：

```
BOOL FAR PASCAL CHNCharInput (WORD wInKey,
        unsigned char far* lpCharCodes,
        unsigned char far* lpPromptLine,
        int nKeys);
```

当输入总控模块用一个虚拟键码调用中文输入转换模块时，中文输入转换模块的控制模块再用这个键码做为 wInKey 调用中文输入方法模块。若一个汉字输入过程结束，函数应返回 TRUE，否则返回 FALSE。

lpCharCode 在完成一个汉字输入后指向返回该汉字 UCS 编码串。lpPromptLine 在整个输入过程中指向汉字提示行串。nKeys 的值是在整个输入过程中已经输入的汉字输入码的键数。

具体的汉字输入方法：在接收到一系列键盘输入消息以后，根据自己的输入规则检索出一个汉字的 UCS 编码（两字节），然后将它们返回给中文输入转换模块，再由中文输入转换模块把这两个字节的 UCS 编码返回给输入总控模块，就实现了一个汉字的输入。

目前我们实现了区位码输入法，并增加了一种 UCS-2码输入法，这种输入法是由用户从键盘上直接键入一个汉字的十六进制 UCS 编码而唯一地检索出一个汉字。按照上面所述接口规范可以实现一种新的汉字输入方法并将其加入到系统中。

在许多已有的汉字输入方法中，汉字的检索都是通过区位码表实现的。我们构造了一个 GB2312字符集同 CJK 汉字的转换表，该表是一个大小为72×94的二维数组，数组的两个下标分别对应区位码表的区和位，数组元素是该元素在区位码表中的位置上所对应汉字的 UCS-2编码。这个转换表将 GB2312中的所有汉字对应在 ISO 10646中的 UCS 编码逐个录入计算机中而成，同时提供一个转换函数：

```
unsigned int GBtoUCS(unsigned int nGBCode);
```

该函数接受一个区位码并返回一个 UCS-2编码，这样能方便地实现 GB2312到 CJK 汉字的 UCS-2编码的转换，从而方便地实现了由汉字输入码得到汉字的 UCS-2编码。

2. 输出的实现

MS-Windows 上的输出包括在屏幕上的输出（显示）和在打印机上的输出（打印）。在 XDUCS Windows 中我们只考虑在屏幕上的输出。

MS-Windows 核心层中的 GDI 提供了图形用户接口，所有窗口输出都是通过 GDI 完成的，其中包括所有字符的输出。GDI 提供的图形用户接口都是与具体设备无关的，而图形最终在硬件设备上的输出，则由某一个硬件设备驱动程序完成。例如在一个 VGA 屏幕上的一个划直线操作最终是由 VGA.DRV 通过对视频缓冲区操作实现的。在 MS-Windows，所有的设备驱动程序也都是以 DLL 形式实现的。

经过对 Windows 字符输出的反复跟踪分析我们发现，MS-Windows 系统中所有字符输出最终都是由显示驱动程序中的函数 ExtTextOut 实现的，其中包括采用点阵字体的输出和采用矢量字体的输出。当应用程序调用 GDI 的字符输出函数（如

维普资讯 http://www.cqvip.com

TextOut)输出字符时，GDI 把这种与设备无关的操作传递到显示设备驱动程序的函数 ExtTextOut 中，ExtTextOut 根据 GDI 传递来的物理字体信息，取得相应字符的字模或者矢量信息，经对视频缓存的操作输出字符。同时，GDI 根据应用程序的要求可以对字符的字模或矢量进行颜色改变，放大、缩小等操作。

由此我们也可以看到，UCS 字符(特别是汉字)输出的关键是如何使设备驱动程序的 ExtTextOut 识别出一个 UCS 字符并正确取得字符的字体信息。因此，XDUCS Windows 的输出模块的开发主要应从两个方面进行：

(1)显示设备驱动程序中函数 ExtTextOut 的修改，支持 UCS 字符输出的函数应能正确显示从 GDI 传送来的 UCS 字符。

(2)构造 UCS 字符特别是汉字的字模或矢量并以某种方式加入 Windows 系统中，以便 ExtTextOut 函数使用。

这里有几个问题需要解决：

(1)ExtTextOut 函数是由显示设备驱动程序提供的，在何处修改这个函数？可以通过 MS-Windows 3.1 DDK 重新构造一个支持 UCS 字符输出的设备驱动程序。但是针对不同的显示器有不同的显示设备驱动程序，重构所有这些程序工作量很大。

(2)支持 UCS 字符的 ExtTextOut 函数如何方便地从系统中取得所需的字体信息？似乎可以生成所有我们支持的 UCS 字符的字体信息，并按字符的编码顺序组织成字体文件。在英文 Windows 环境中，不仅字符的字体有固定格式，字体文件也按固定的格式组织，并以"FON"或"TTF"为后缀名存贮在 Windows 系统目录中，这些格式都是针对英文字符设计并以 ASCII 顺序排列的。由于我们不能修改 MS-Windows 核心，因而也无法重新定义一系列格式同时支持所有的 UCS 字符。所以，我们不得不采用和遵循 Windows 的字体格式。另外，采用"并存式"模块的系统中将有多种代码体制并存。具体地说，从 GDI 传送到字符输出函数 ExtTextOut 的字符数据可能采用 ASCII、GB2312、UCS 代码中任意一种。因此，我们不可能放弃原有系统的字符输出功能，而只能在此基础上扩充。

(3)Windows 的字体文件格式只适合于西文字符，英文字符的字体采用 MS-Windows 中的缺省字体，汉字的字体信息只能由我们自己构造并注册到 Windows 系统中，中文字体的字模也只能自己构造，在 ExtTextOut 中采用自己开发的方法访问。

在对函数 ExtTextOut 所做工作进行仔细分析后，我们采用下面的方法实现 UCS 字符的输出：在程序运行时首先取得函数的入口地址，然后把函数入口处的第一条指令修改为跳转到一个我们自己开发的实现 UCS 字符输出的函数 NewExtTextOut，由这个函数完成字符输出功能。这里的关键问题是对 ExtTextOut 函数的正确修改和字符输出控制转移到函数 NewExtTextOut 后数据的正确寻址。

MS-Windows 3.1运行于保护模式下，因此，对 ExtTextOut 的修改首先必须创建一个指向 ExtTextOut 数据描述符，然后利用 ExtTextOut 原有的偏移生成一个指向 ExtTextOut 的字符指针，通过该指针实现对它的修改。当整个系统范围内有字符输出操作时，最终都会调用函数 ExtTextOut，修改后的该函数会将控制立即转向 NewExtTextOut，如果最初的字符输出请求不是直接调用函数 ExtTextOut 完成的话，比如调用函数 TextOut 或 DrawText 实现，GDI 将最终用 ExtTextOut 完成字符输出，此时函数 ExtTextOut 所访问的公共数据缺省为 Windows 显示驱动程序的数据，都限定在一个特殊的数据段内。此时，控制在函数 NewExtTextOut 中，如果也使用这个缺省的段地址的话，NewExtTextOut 将不可能正确访问到它所在的 DLL 中定义的公共数据。为解决这个问题，我们在该 DLL 初始化时首先到其数据段地址(Windows 中所有 DLL 都只能有一个数据段)，在控制转到函数 NewExtTextOut 后立即用一段嵌入的汇编指令将此时的数据段地址改为它所在的 DLL 的段地址，以保证函数 NewExtTextOut 对其所使用数据的正确寻址。

采用动态修改字符输出函数的好处不仅在于我们可以顺利截获系统的字符输出请求，在此基础上可扩充我们对 UCS 字符的输出处理；还能很好地保证应用程序的可移植性，因为采用这种动态修改的方法不修改系统的显示设备驱动程序。对我们来说，这种方法使得我们的系统也做到了与设备无关，从而也保证了系统的可移植性，节省了开发时间。

当程序控制转到 NewExtTextOut 函数时，它首先将函数 ExtTextOut 的第一条指令改回到原有状

维普资讯 http://www.cqvip.com

态，主要目的是利用它实现西文字符的输出。然后判断由GDI送来的显示字符所采用的编码形式。主要依据下面几条规则：

(1)如果所要显示字符串长度为2×n+1，其中n>0，则该字符串一定不是UCS字符串，作为ASCII串处理。

(2)如果所要显示字符串长度为2×n，其中n>0，则把n个相邻的两个字节作为一个双字节编码，看这个编码是否在我们实现的UCS子集中，即是否具有0000到007F(其中0000到001F为控制字符)或者4E00到9FFF的值。如果这n个编码都在上述范围中，则将这个字符串作为UCS字符串，否则作为ASCII字符串处理。

(3)如果字符长度为1，则保存这次输入请求的所有上下文环境，并等待下一次输出请求。如果下一次的请求中字符串长度也为1，而且两个字符的输出上下文环境可以看作两个连续的字符输出时，将两个字节合为一个编码并判断这个编码是不是一个UCS字符编码。若是，则将两者作为一个UCS字符处理。否则，把第一个字符以ASCII方式显示，后一个字符串再按(1)、(2)和(3)规则中的相应者处理。

如果所要显示的字符串为ASCII串，我们调用ExtTextOut来输出。如果所要显示的字符串为UCS英文串，先将其转换成一个ASCII串，再调用函数ExtTextOut输出。如果是UCS中文串，就调用汉字输出程序输出。如果是一个中英文混合串，则先将其中的英文字符分离出来形成一个单独的英文串，对应于原串中的汉字以两个空格代替，并按英文UCS串做输出处理。再把所有汉字分离出来，汉字的分离方法不同于英文，要分几次完成，每次分离出连续的汉字形成一个汉字串并在相应位置上显示。

汉字显示的关键是正确取得汉字的字体信息，这要求先必须构造所有汉字的字体信息并将其加入到Windows中。由于汉字数量很多，构造汉字字体信息的工作量很大，因此暂时借助于由北京新天地电子信息技术研究所研制的软件"中文之星"来实现。"中文之星"是一个西文Windows汉化工具，它采用外挂方式，以应用程序方式在MS-Windows 3.1英文版上方便地实现了对汉字的支持。在运行了"中文之星"的Windows系统的ExtTextOut函数支持采用GB2312编码的汉字显示和打印，在"中文之星"支持下的UCS汉字输出过程也就变得比较简单，我们只需将用UCS表示的汉字串转换成用GB2312表示的汉字串，然后调用函数ExtTextOut就可实现汉字的输出，在此基础上一个用UCS表示的中英文混合串的显示也可以简化成为将这个UCS串转换成为一个用ASCII表示英文和用GB2312表示的混合串，然后调用函数ExtTextOut即可完成。

每当一次字符输出完成以后，我们再把显示驱动程序的ExtTextOut函数入口处的第一条指令修改为跳转到函数NewExtTextOut，准备为下一个字符输出请求服务。

为了实现CJK汉字到GB2312汉字的转换，我们构造了一个从CJK汉字到GB2312汉字的转换表。这个转换表用一个一维数组实现；数组大小为20992；元素的排列次序同CJK汉字的排列次序相同。每个数组元素是一个无符号整数，元素的取值要么是该元素对应位置的CJK汉字在GB2312中的区位码(如果这个汉字在GB2312中有定义)，要么为零(这个汉字未在GB2312中定义)。这个转换表占用了50K左右内存，利用一个转换工具通过对GB2312到CJK汉字转换表操作生成。同时，提供一个转换函数：

```
unsigned int UCS toGB(unsigned int nUCSCode);
```

该函数接受一个UCS-2编码并返回一个区位码。目前，我们只实现了GB2312中的所有汉字，因此，这个表中有36K左右内存未加利用。我们采用DLL实现这个转换表以及在输入模块中使用的GB2312汉字到CJK汉字的转换表，在Windows管理下，只在必要时才把它们调入内存中，其他时候可以将其对换到硬盘上，从而保证它们对系统内存的使用上不造成较大的浪费。

五、小　结

XDUCS Windows全部使用C语言在Borland C++ for Windows 3.1环境下开发，在MS-Windows 3.1英文环境下运行。所完成的工作，归纳起来有：

(1)实现了对一个UCS子集的输入/显示的支持，该子集定义为所有ASCII图形字符和GB2312中的所有汉字。

(2)汉字的输入方式有区位码输入和一种新的汉字输入方法——UCS-2编码汉字输入方法，并给出了一个开发其他汉字输入方法的接口规范。

(3)提出了一种采用UCS的多种自然语言输入/显示的模式，并给出了开发输入模块的接口规范。

(4)实现了CJK汉字同GB2312标准之间的相互转换，从而保证了同已有软件系统间的兼容性。

维普资讯 http://www.cqvip.com

由于XDUCS Windows以应用程序形式实现和运行，特别是Windows系统的任何键盘事件和字符输出都经由该应用程序判断和处理，因此，对Windows系统的运行效率或多或少地都会产生影响。XDUCS Windows中最主要的工作是多个转换，因此，转换的效率和转换表的大小是两个对Windows系统运行效率产生影响的主要因素。

在XDUCS Windows中我们实现了GB2312标准同CJK汉字间的相互转换。用线性数组构造了两个转换表，共占用54K左右内存，转换时除计算元素在数组中的偏移外，不用进行额外的比较操作和计算。但是，由于GB2312标准只是CJK汉字的子集，在从CJK汉字到GB2312标准的转换表中存在40K左右的无用空间。采用这种方法的主要原因是为了追求转换的速度而不吝惜内存空间，因为54K空间将不会对Windows系统及应用程序的运行造成多大的影响。

通过使用MS-Windows 3.1中的资源监视工具Windows Resource Monitor对XDUCS Windows运行时对系统资源（包括对Kernel、User和GDI资源）的占用情况的监视和分析，我们认为XDUCS Windows对系统资源的要求是微不足道的，对Windows系统运行效率的影响完全可以忽略。

系统的文件构成、资源使用，以及存在的问题如下：

1. 文件构成

XDUCS Windows目标系统由七个文件构成，全部源程序共6000行，目标文件共178K，运行时各模块间的调用关系如图5所示。

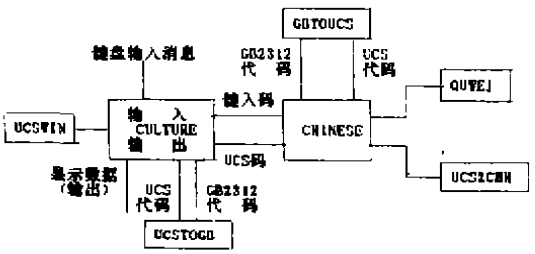

图5 XDUCS Windows运行时各模块间的调用关系

七个文件如下：

(1)UCSWIN.EXE——XDUCS Windows的主控程序，负责整个系统界面和同用户的交互以实现用户对系统功能的选择，由UCSWIN.C，UCSWIN.RC和UCSWIN.DEF三个文件编译生成。

(2)CULTURE.DLL——UCS字符输入总控模块（包括英文字符输入处理）和UCS字符输出模块。由CULTURE.C，CULTURE.RC，CULTURE.DEF三个文件编译生成。

(3)CHINESE.DLL——CJK汉字输入总控模块。由CHINESE.C，CHINESE.RC和CHINESE.DEF三个文件编译生成。

(4)GBTOUCS.DLL——GB2312标准到CJK汉字的转换模块。由GBTOUCS.C，GBTOUCS.DEF两个文件编译生成。

(5)UCSTOGB.DLL——CJK汉字到GB2312标准的转换模块。由UCSTOGB.C，UCSTOGB.DEF两个文件编译生成。

(6)QUWEI.DLL——区位码汉字输入方法模块。由QUWEI.C，QUWEI.DEF两个文件编译生成。

(7)UCS2CHN.DLL——UCS-2编码汉字输入模块。由UCS2CHN.C，UCS2CHN.DEF两个文件编译生成。

2. XDUCS Windows对Windows资源的使用情况

我们使用MS-Windows 3.1提供的实用程序System Resource Monitor对XDUCS Windows的资源使用情况在一台带有4M内存的386DX微机做了一个测试，结果如下表所示，表中"CStar"为"中文之星V1.1"，"BCW"为"Borland C++ for Windows 3.1"。

应用程序运行情况	User	GDI	Memory
Program Manager	93% Free	85% Free	88% Free
Program Manager + UCSWIN	93% Free	85% Free	87% Free
Program Manager + CStar	85% Free	85% Free	83% Free
Program Manager + CStar+UCSWIN	85% Free	85% Free	82% Free
Program Manager + BCW	87% Free	75% Free	80% Free
Program Manager + BCW + UCSWIN	87% Free	75% Free	78% Free
Program Manager + CStar + BCW	77% Free	73% Free	78% Free
Program Manager + CStar+BCW+UCSWIN	77% Free	73% Free	77% Free

维普资讯 http://www.cqvip.com

3. 存在的主要问题

(1)MS-Windows 英文版系统采用单字节内核及单字节字符编码，在某种程度上我们可以将这个系统看作一个对控制字符敏感的系统。也就是说，范围从00到1F之间的一个字符在系统中流动时，系统将会对这些字符产生某种反应。比如说，值为00的字符将表示一个字符串的结束，值为08的字符将引起系统的编辑控制类产生回删一个字符的动作等等。XDUCS Windows 是通过对 MS-Windows 英文版目标码系统进行修改来达到对 UCS 的支持，因为不能修改系统，一个 UCS 字符在系统中将以两个单独的连续字节流动。单从 UCS 字符一个八位的编码范围来看，它的取值范围可以从0到 FF，也就是说，UCS 字符的单个八位的编码同以往的单字节字符编码间存在冲突。因此，一个 UCS 字符的某一个取值为00到1F 的八位在系统中流动时，有可能被原有系统做为一个控制字符看待，这不仅会引起系统的误操作，还使 UCS 字符的传送出现错误。一个典型的例子是一个文件的文件名采用 UCS 字符，比如所说这个文件名是一个英文名字，而英文字符的 UCS 编码的第一个八位值为0，以这样一个文件名存储文件时，Windows 将认为这个名字非法。这样的问题在这种通过对目标码系统修改实现对 UCS 支持的系统中都会出现。要避免这个问题的出现，只能通过对系统源码的修改得到一个新系统。

(2)另一个不容易解决的问题是采用 UCS 的应用程序中对 UCS 字符的正确操作，这个问题应该由编程语言及其编译器解决。由于目前的程序语言大多只支持单字节字符，特别是没有支持宽字节字符的编译器，因此，应用程序要对 UCS 字符进行正确处理。用户不得不自己专门实现对 UCS 字符的处理。虽然 XDUCS Windows 能支持采用 UCS 的应用程序的运行，但必须在用户的专门参与下才能完成。

中文信息处理国际化是一个重要的发展方向，我们所作的工作只是初步的，还有更多的工作有待继续努力，例如：

(1)实现独立的汉字输出功能。这需要构造汉字字体信息，并实现汉字点阵和矢量的输出操作。

(2)进一步研究更有效的 CJK 汉字同 GB2312 汉字的相互转换方法。

(3)支持更多的汉字输入方法。

(4)增加对更多子集的支持，从而增加对更多汉字(如繁体字)的支持以及对其它自然语言的支持。

参考文献

[1] ISO/IEC 10646-1.2中文译稿(V1.0)，ACCC秘书处翻译1992年

[2] ISO 8859信息处理一八位单字节编码图形字符集

[3] 陈明源，通用字符集 UCS 及其支撑环境——中文数据类型，计算机及信息处理标准化，№.3，1992

[4] 孙玉方，开放系统国标准化与国际化，计算机及信息处理标准化，№.3，1992

[5] 孙玉方，中文电脑开放平台规范的研制，计算机及信息处理标准化，№.3，1992

[6] 岳晋生，Microsoft Windows 中文化工具 GWCWinTool 的研制与开发，小型微型计算机系统，Vol.14，№.5，1993

[7] Jeffrey Richten, Simulating Keyboard Input Between Programs Requires a(Key)Stroke of Genius, Microsoft System Journal, Dec,1992

[8] William S. Hall, Adapt Your Program for Worldwide Use with Windows Internationalization Support, Microsoft System Journal, Nov-Dec,1991

[9] Gordon S. Smith, Embedded Device Drivers Simplify the Support of Unusual Devices Under Windows, May, 1991

[10] Helen Custer, Inside Windows NT, Microsoft Press,1993

[11] Keith Weiskamp, Windows 31 Insider, John Wiley & Sons,Inc.1993

[12] Charles Petzold, Programming Windows, Microsoft Press,1987

[13] 郭平欣、张淞芝，汉字信息处理技术，国防工业出版社，1985年

[14] 钱培德，计算机中文信息处理技术，电子科技大学出版社，1992年

[15] 熊桂喜、钟宁等译，Windows 3.0软件开发指南(一)、(二)，清华大学出版社，1992年

[16] 熊桂喜、钟宁等译，Windows 3.0程序员参考手册，清华大学出版社，1992年