第九讲

汉字键盘输入系统

苏州大学计算机科学与技术学院

内容

- 汉字输入系统的分类
- 汉字输入系统的功能
- 汉字键盘输入系统的工作流程

1. 分类

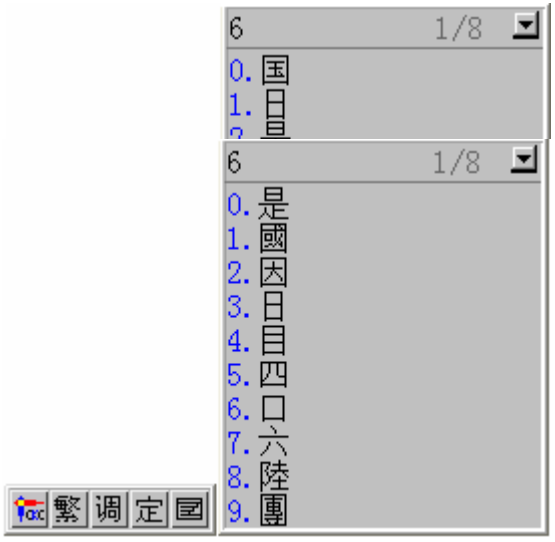
- 按系统在计算机中实现的层次可以分为：
  - 系统层的输入系统
  - 应用层的输入系统

1. 分类

- 按操作系统平台可以分为：
  - DOS下的输入系统
  - Windows下的输入系统
  - Linux下的输入系统
  - Mac OS下的输入系
  - 嵌入系统中的输入系
  - ……

1. 分类

- 按汉字内码的简繁可以分为：
  - 简体输入系统
  - 繁体输入系统



1. 分类

- 按汉字的字符集可以分为：
  - GB2312的输入系统
  - GBK的输入系统
  - GB18030的输入系统
  - BIG-5的输入系统
  - ISO10646的输入系统
  - ……

1. 分类

- 按输入码类别可以分为：
  - 音码输入系统
  - 形码输入系统
  - 音形码输入系统
    - 自然码（其实就是一种双拼的标准双拼方案）
  - 流水码输入系统
    - 区位码、电报码、内码等，一个编码对应一个汉字。

2. 功能

- 汉字/词组输入
  - 单字输入
    - 我 你 的
  - 词组输入
    - 我们 计算机
  - 短语、短句输入
    - 我的 非常多 不明白 科技是第一生产力
  - 句子输入
    - 我到饭店去吃饭。

2. 功能

- 候选字/词提示和选择
  - 重码不可避免
  - 输入系统必须具备候选字/词提示的功能
  - 应该常用用字、词在前
  - 提高容错的功能

2. 功能

- 中英文状态切换
  - 存在中文和英文两种状态
  - 在中文状态下，可以通过输入汉字输入码输出汉字
  - 在英文状态下，整个键盘都是西文符号
  - 常用的中西文切换键有“Caps Lock”和“Shift”等

2. 功能

- 翻页
  - 当输入的汉字输入码对应的候选字/词比较多时，输入系统无法在候选框中显示所有的候选字/词，只能通过分页方式显示。
  - 可使用到翻页键显示上/下一页的候选字/词。
  - 常用的翻页键分为前翻页键和后翻页键，一般输入系统的前/后翻页键为“-”和“+”、“,”和“.”、“<”和“>”、“[”和“]”、“PageUp”和“PageDown”等。

2. 功能

- 输入码清除、删除、结束键
  - 当输入的输入码有误时，可以通过输入码删除键删除刚输入的这个输入码符，也可以通过输入码清除键删除所有输入的输入码。实际上这两个键的作用是提供一种输入码的简单编辑功能。
  - 绝大多数的输入系统不需要输入码结束符，只有输入码键和选择键冲突时才需要输入码结束符来区分两种状态。

2. 功能

- 输入码清除、删除、结束键
  - 在搜狗拼音输入系统中，由于输入码是字母，选择符是数字，所以没有结束键。
  - 在纵横输入系统中，由于输入码和选择符都是数字，所以需要结束键。

2. 功能

- 联想
  - 当用户输入了一个汉字后，系统提示以这个汉字为首字的词组供用户选择（不包括首字），这种技术称为联想技术。
  - 如用户输入了“计”，则系统提示以“计”作为首字的词组剩余部分，如“算机”、“划”和“策”等。

2. 功能

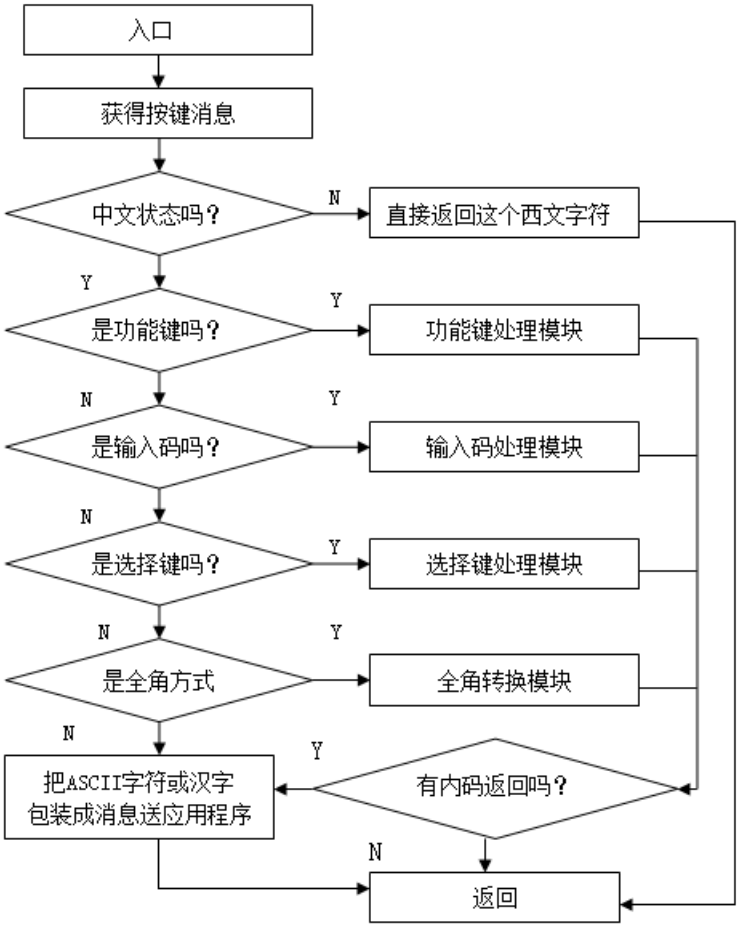
- 全角和半角
  - 键盘上的西文字符在输入计算机时有全角和半角两种方式。
  - 在早期DOS系统下，ASCII字符是中文字符的一半大小，所以称ASCII字符为半角字符。
  - 采用双字节内码、和中文等宽的西文字符为全角字符。
    - 如半角字符“0”的ASCII码是30H，而全角字符“０”的Unicode内码是FF10H。
  - 在Windows等系统中，由于是图形界面，西文字符不是等宽字符，所以一个ASCII字符不再是一个中文内码字符的一半。

2. 功能

- 符号输入

符号	按键	符号	按键	符号	按键
。	.	？	?	“”	“”
，	,	：	:	‘’	‘
、	\	；	;	！	!
《	<	￥	$	×	*
》	>	%	%		

3. 工作流程


入口
获得按键消息
中文状态吗？
N
直接返回这个西文字符
Y
是功能键吗？
Y
功能键处理模块
N
是输入码吗？
Y
输入码处理模块
N
是选择键吗？
Y
选择键处理模块
N
是全角方式
Y
全角转换模块
N
把ASCII字符或汉字
包装成消息送应用程序
Y
有内码返回吗？
N
返回

3. 工作流程

- 中文输入和西文输入的差异
  - 在西文环境下，用户每按下一个按键应用程序都会接收到。
  - 在中文环境下，用户按下一个按键后，应用程序可能接收不到任何的东西，因为输入系统还没有机内码返回，需要更多的按键后才能有机内码返回。

3. 工作流程


①wo
④提示
⑤1
⑥输出汉字到应用程序
用户操作键盘
输入系统
应用程序
②查对照表
③输出汉字串
输入码对照表
我wo
握wo
窝wo
卧wo
挝wo
沃wo
蜗wo
涡wo
斡wo
倭wo
幄wo

汉字键盘输入系统实例分析 之

Windows汉字输入技术

内容

- Windows汉字输入技术概述
- Windows下汉字输入原理
- 在SDK和DDK环境下开发输入系统

1. 概述

- 界面

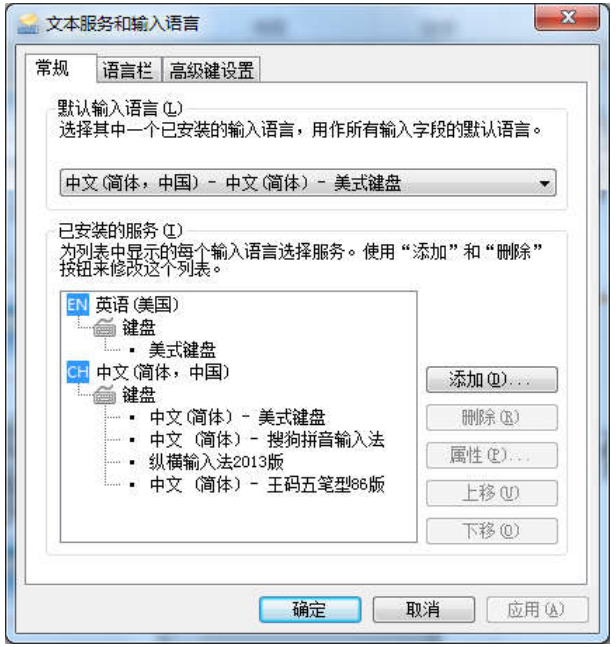
全拼输入法的用户界面

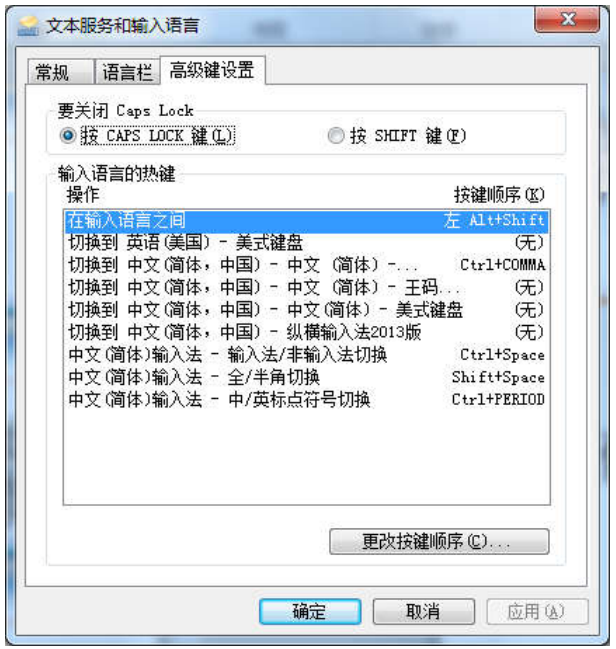
1. 概述

- 有关概念
  - 输入法IME（Input Method Editor）
    - 根据编码规则，实现输入码到机内码的转换，并实现各种输入法内置的功能。
  - 输入法管理器IMM（Input Method Manager）
    - 用于管理在Windows中的输入法，并在输入法和应用程序间建立联系。

1. 概述

- 有关概念（续）





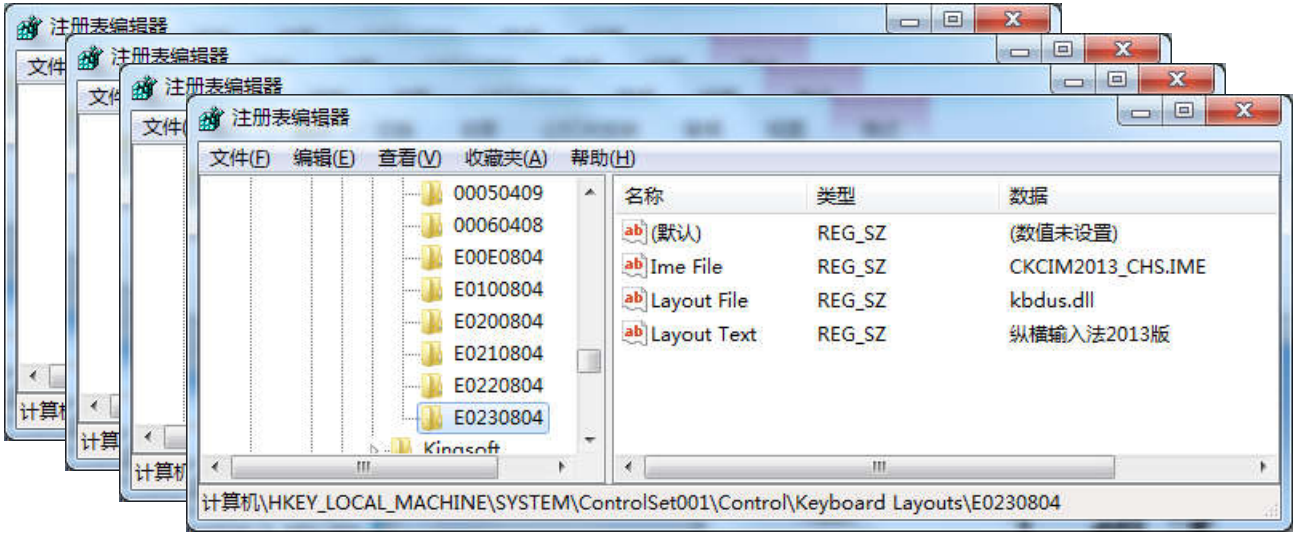
输入法管理器

1. 概述

- 有关概念（续）
  - 多文种环境（Multi-lingual Environment）
    - 所谓多文种环境是指系统可以为使用各种不同语言文字体系的用户提供了支撑他们母语的操作环境。
    - Windows的输入体系支持多文种环境，包括中文、日文、韩文、法语和德语等上百种不同的语言体系。汉语言文字仅是其中的一个环境而言。
  - 键盘布局（Keyboard Layout）
    - 键盘布局规定了键盘上每个按键的物理位置以及每个按键对应的字符。每个键盘布局对应于一种语言，从而根据这种语言的定义和当前的输入法把按键转换为该语言的文字信息。
    - 在Windows中，提供了多种不同的键盘布局，其中就包括各种中文的键盘布局。每个汉字输入法就是一个键盘布局。

1. 概述

- 有关概念（续）



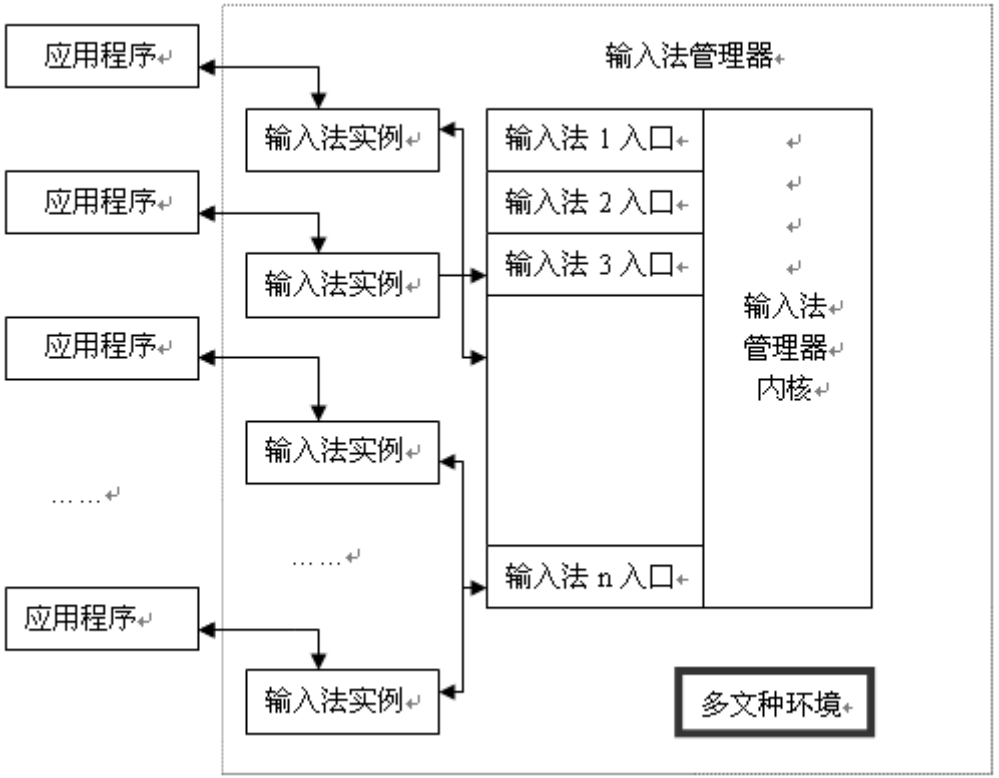
各种输入法的键盘布局

1. 概述

- 特点
  - 每个输入法只是整个多文种环境的一个部件而已，一个多文种环境包含多个输入法；
  - 每个应用程序都拥有自己的一个活动输入法，也就是说，输入法不再为所有的应用程序所共有，同样的一个输入法，可能运行多个实例；
  - 应用程序和输入法之间可以进行消息的传递，应用程序可以控制输入法的工作；
  - 支持从汉字词组到编码的反向转换。

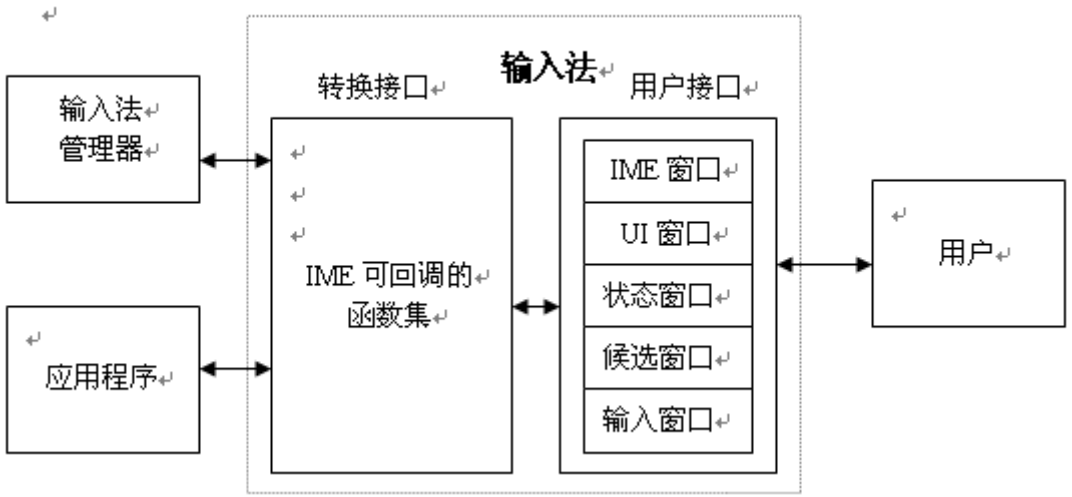
2. 原理

- 汉字输入模型


应用程序
应用程序
应用程序
……
应用程序
输入法管理器
输入法实例
输入法实例
输入法实例
……
输入法实例
输入法 1 入口
输入法 2 入口
输入法 3 入口
输入法 n 入口
输入法
管理器
内核
多文种环境

2. 原理

- IME基本结构


输入法
管理器
应用程序
输入法
转换接口
用户接口
IME 可回调的
函数集
IME 窗口
UI 窗口
状态窗口
候选窗口
输入窗口
用户

2. 原理

- 转换接口主要完成以下的工作：
  - 输入法的初始化和关闭；
  - 代码转换；
  - 输入法的设置以及输入法的界面、状态的修改；
  - 输入字符的处理；
  - 定义新的词组。
- 这些函数的名称、参数等是由输入法管理器确定，对输入法的设计者来说，就是完成这些函数的代码填充问题。

2. 原理

- 用户接口由一些可见或不可见的窗口组成，主要的作用是接收各种消息以及为输入法提供用户接口。
  - IME窗口
  - UI窗口
    - 状态窗口
    - 预编辑窗口
    - 候选窗口

2. 原理

- 处理过程：

1. 当用户在键盘上按下一个键后，在应用程序调用GetMessage/PeekMessage前，系统首先调用输入法的接口函数ImeProcessKey，通过这个函数判别输入法是否需要处理这个键；
2. 如果返回TRUE，表示输入法需要处理这个键，那么系统设置虚键的值为VK_PROCESSKEY，然后应用程序接收到消息WM_KEYDOWN，其中的值为VK_PROCESSKEY；如果返回FALSE，则表示输入法不需要处理这个键，直接返回此键给应用程序，结束；
3. 当应用程序接收到消息WM_KEYDOWN后，在调用函数TranslateMessage时，系统把用户按下键的扫描码和虚键传递给输入法的接口函数ImeToAsciiEx。
4. 输入法在处理了按键后，根据处理方法的不同把各种消息存放在消息缓冲区中，这个缓冲区是作为函数ImeToAsciiEx的一个参数传递过来的。
5. 应用程序在消息队列中得到由输入法产生的消息，进行相应的处理。如果应用程序不处理这些消息，则把它传递给默认窗口（Default Window），然后由系统就把消息传递给默认的IME窗口，由它来处理这些消息。

3. 开发输入系统

- 编程人员利用DDK工具包来开发输入法
- DDK中包含：
  - 输入法的开发环境
  - 相关文档
  - 一个区位输入法的实例代码
- 以DDK中的区位输入法的代码为基础，设计新的输入法，可以大大节省开发时间。

3. 开发输入系统

- 开发输入系统需要以下环境：
  - DDK工具包
  - Win32 SDK 工具包
  - Visual C++ 2.0以上

汉字键盘输入系统实例分析 之

Linux汉字输入技术

内容

- **Linux汉字输入技术概述**
- **XIM协议**
- 输入法服务器和输入法
- **SCIM和IBus**

1. 概述

- 国际化与本地化
  - 国际化（Internationalization, I18n）：在设计软件，将软件与特定语言及地区脱钩的过程。当软件被移植到不同的语言及地区时，软件本身不用做内部工程上的改变或修正。
  - 本地化（Localization, L10n）：本地化则是指当移植软件时，加上与特定区域设置有关的信息和翻译文件的过程。

1. 概述

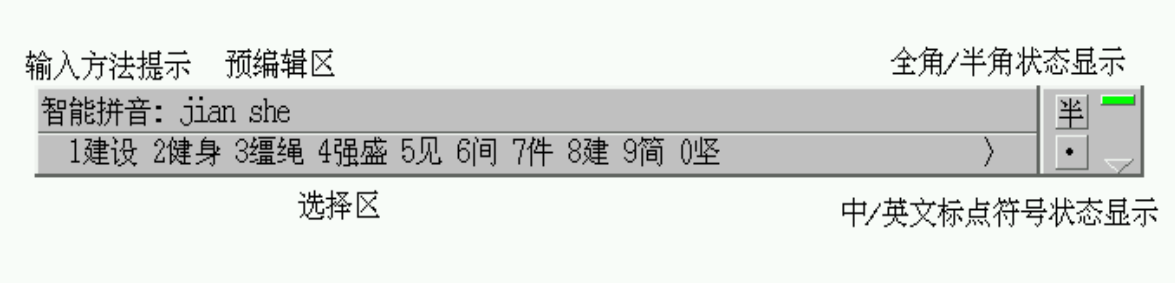
- NLS（National Language Support）
  - 建筑在基于ASCII字符集的Linux核心上，为世界上不同地域、不同语言环境的应用提供国际化和本地化支持。
  - 系统中所有支持多国语言的应用程序，包括X Window都是建立在这个基础上的。
  - 在此基础上，我们可以建立支持各种不同语言文化的民族特征数据库（Locale）、输入方法（IM）、字体（Font）和消息机制（Message）等。

1. 概述

- XIM(X Input Method)协议
  - X11R6工作组定义，用来规范X Window系统中的文本输入问题。
  - 根据XIM标准开发的中文输入法服务器：
    - Rfinput
    - Chinput、miniChinput
    - Xcin等输入法服务器
  - 日文、韩文的输入法服务器：
    - ami和hanIM都是支持韩文的XIM输入法服务器。
    - kinput2是支持日文的XIM输入法服务器。

1. 概述

- XIM输入法服务器的风格
  - Root风格



1. 概述

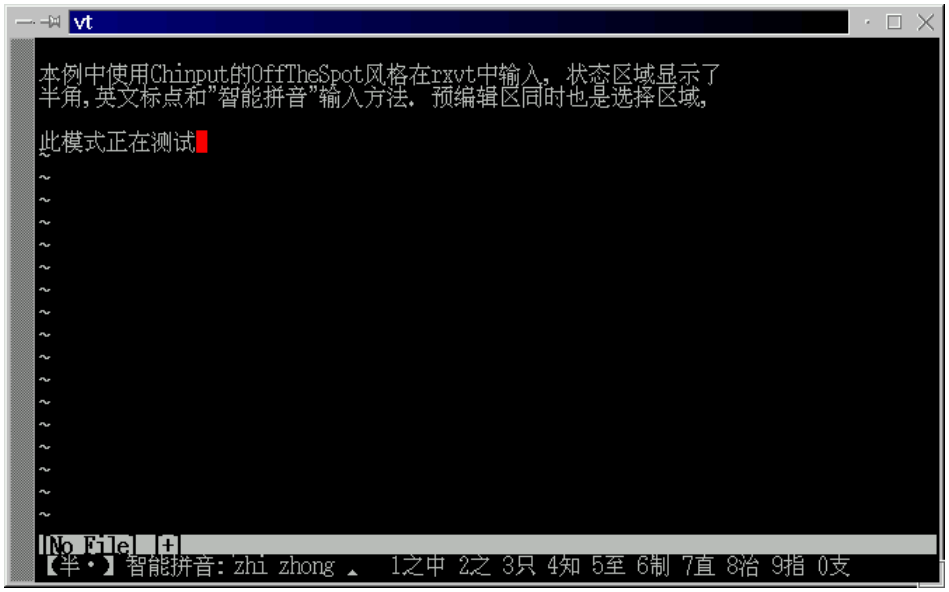
- XIM输入法服务器的风格（续）
  - Over-The-Spot风格

1. 概述

- XIM输入法服务器的风格（续）
  - On-The-Spot风格

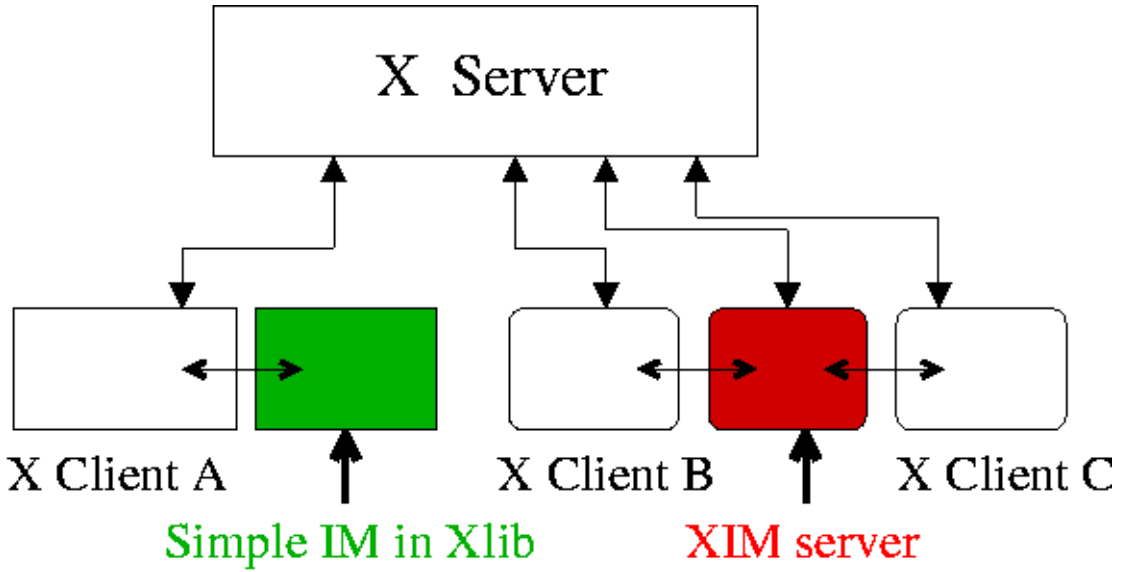
1. 概述

- XIM输入法服务器的风格（续）
  - Off-The-Spot风格



2. XIM协议

- XIM协议框架

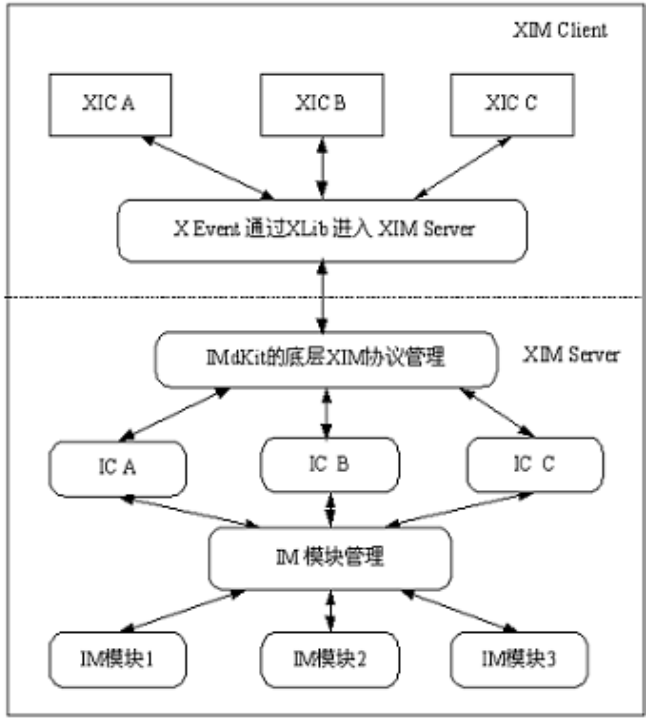


2. XIM协议

- XIC与IC
  - 我们把每个准备接收来自XIM Server输入的窗口称为一个XIC（X Input Context），它是存在于XIM Client端。
  - 而在XIM Server端，对于每一个请求输入的XIC来说，XIM Server都为它准备一个IC（Input Context）结构来提供服务。

2. XIM协议

- XIC与IC在XIM结构中的关系



2. XIM协议

- XIC结构

```
typedef  struct{
    char            IMName[20];       //输入法的名称
    char            IMLocale[10];     //用户的Locale设置
    Ximage          *IMImage;         //状态栏中的输入法图标
    Window          IMWindow;         //当前输入法窗口值
    (int *)         Init(void);       //输入法的初始化函数
    (int *)         Filter(int,int);  //输入法的字符处理函数
    (int*)          ClearUp(void);    //输入法结束时调用的函数
} InputMethod;
```

2. XIM协议

- IC结构

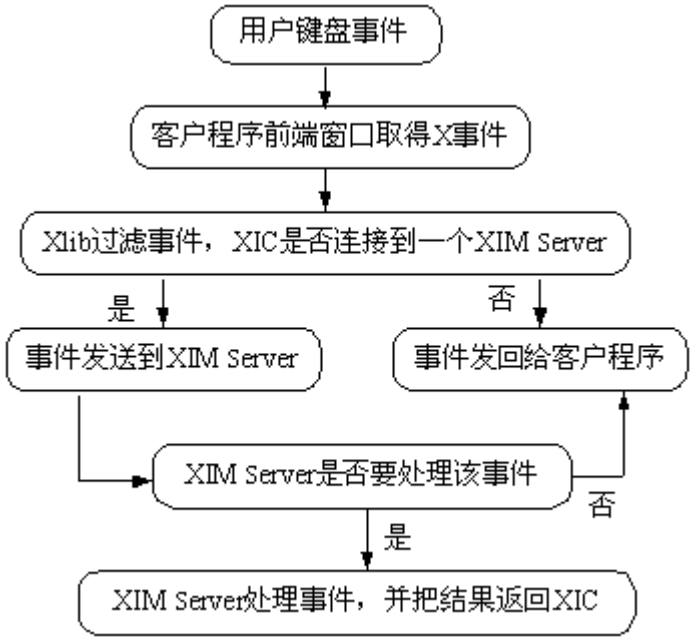
```
typedef struct _IC {
    CARD16              id;                 //IC的唯一ID号
    INT32               input_style;        //输入风格
    Window              client_win;         //客户程序窗口
    Window              focus_win;          //当前焦点窗口
    char *              resource_name;      //资源名称
    char *              resource_class;     //资源类型
    PreeditAttributes   pre_attr;           //预编辑区域属性
    StatusAttributes    sts_attr;           //状态区域属性
    int                 indexIM;            //输入法的类型
    Boolean             bCorner;            //全/半角状态，
    Boolean             bPunc;              //中文/英文标点符号状态
    Boolean             bMainShow;          //输入法状态栏是否可见
    struct _IC *        next;
} IC;
```

2. XIM协议

- IC结构链表
  - indexIM字段：该字段用来说明输入法的类型。
  - bCorner字段：该字段用来说明输入法的全/半角状态。
  - bPunc字段：该字段用来说明输入法的中文/英文标点符号状态。
  - bMainShow字段：该字段用来说明客户程序中输入法的状态栏是否可见。

2. XIM协议

- XIM协议的处理流程



2. XIM协议

- 支持XIM协议的客户软件介绍

软件/库	输入风格
GTK+	Root,OverTheSpot
Qt	Root,OverTheSpot,OffTheSpot
Motif	Root,OverTheSpot,OnTheSpot,OffTheSpot

3. 输入法服务器和输入法

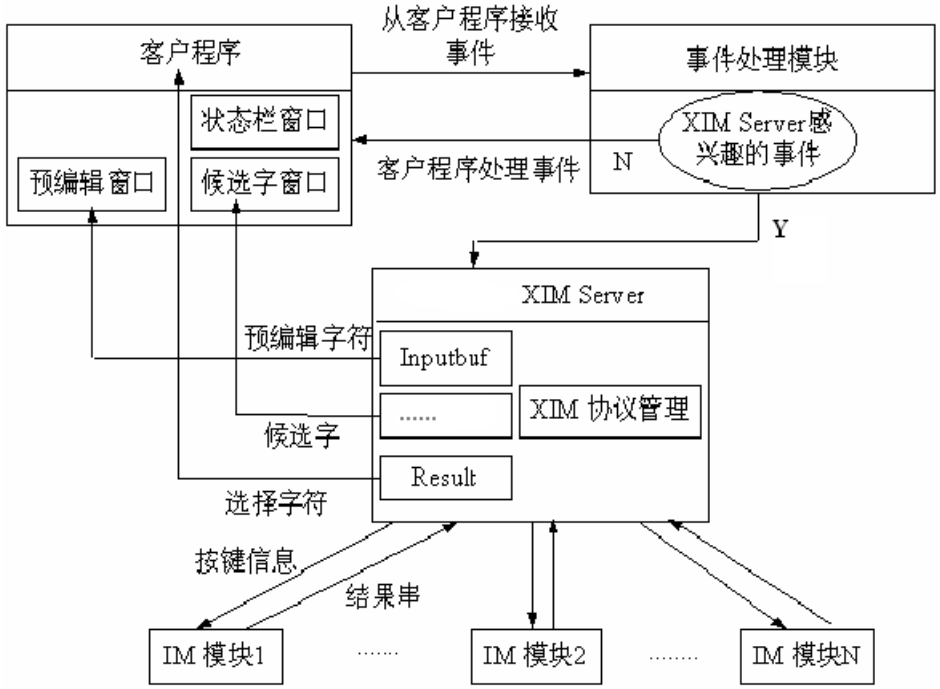
- 输入法服务器
  - 用户界面
    - 预编辑窗口
    - 候选窗口
    - Rfinput还实现了输入法状态栏窗口和嵌入图标
  - 事件处理
    - Expose事件：预编辑窗口、候选窗口以及状态栏窗口或窗口的一部分可能被其他窗口遮挡。当遮挡窗口被关闭后，上述三个窗口或窗口的一部分变成可见了，系统需要重新绘制这些窗口。
    - ButtonPress事件：当用户需要改变输入法状态时，用户可以通过单击输入法状态栏中的全/半角、中文/英文标点图标。同时用户可以通过按住鼠标左键拖动输入法状态栏到屏幕的任何位置。当用户进行上述动作时，将发生该事件。
    - MotionNotify事件：当我们在输入法状态栏上移动鼠标时，鼠标的形状将根据鼠标的位置作相应的改变。比如，如果鼠标处于输入法的全/半角图标时，鼠标形状将变为手形图标。当鼠标处于状态栏的边缘时，鼠标形状将变为拖拉图标。当用户移动鼠标时，将发生该事件。

3. 输入法服务器和输入法

- 输入法服务器（续）
  - XIM协议处理
    - XIM Client与XIM Server通过XIM协议进行通讯。因此输入法服务器必须实现XIM协议。miniChinput实现了XIM协议的四种输入风格，而Rfinput仅实现了常用的光标跟随模式。
  - 输入法与输入法服务器通信
    - 第一种为miniChinput采用的方法，它为用户提供了多个编程接口，用户只需在相应的编程接口中实现自己的函数，然后编译成动态链接库，最后将动态链接库保存在输入法服务器指定的位置。
    - 第二种方案为Rfinput采用的方法，输入法与服务器之间通过预编辑缓冲区、候选字缓冲区以及选择结果缓冲区进行交互。

3. 输入法服务器和输入法

- 输入法服务器总体架构



3. 输入法服务器和输入法

- 输入法服务器与输入法通信的实现
  1. XIM服务器得到用户的ASCII码和扫描码，通过函数Filter ()将ASCII码和扫描码传递给输入法程序；
  2. XIM服务器判断Inputbuf缓冲区是否不为空以及result缓冲区是否为空，如果是，则直接跳转至下一步；否则隐藏预编辑窗口，设置bDisInput的值为False，并跳转至6；
  3. XIM服务器判断bDisInput的值是否为True，如果是，则刷新预编辑窗口中的内容，并跳转至下一步；否则，在ic_list链表中查找到相应的IC结构，在客户程序中为预编辑窗口定位，画出预编辑窗口，显示用户输入的字符；
  4. XIM服务器判断Selhzbuf缓冲区是否不为空以及result缓冲区是否为空，如果是，则跳转至下一步；否则隐藏候选字窗口，设置bDisInput的值为False，并跳转至6。
  5. XIM服务器判断bDisInput的值是否为True，如果是，则刷新候选窗口中的内容，函数返回；否则，在ic_list链表中查找到相应的IC结构，在客户程序中为候选窗口定位，画出候选窗口，显示候选字列表，函数返回；
  6. 判断用户的按键信息是否为标点符号，如果是，则跳转至下一步。否则，跳转至8；
  7. 判断用户是否处于半角状态，如果是，直接跳转至下一步；否则将相应的英文标点转换为中文标点，保存在result缓冲区中；
  8. XIM服务器调用函数CommitResult将result缓冲区中的字符传递给客户程序，结束；

4. SCIM和IBus

- SCIM（Smart Common Input Method）是一个支持多国语言的输入法平台，主要用在UNIX / 类UNIX环境中。它曾经作为多数Linux发行版中的默认输入法。

4. SCIM和IBus

- SCIM的特点
  - 使用C++编程语言写成，完全支持面向对象。
  - 高度模块化。
  - 可塑性极高的结构。
  - 简单的使用接口。
  - 完全国际化及支持UCS-4／UTF-8编码。
  - 图形用户界面具有丰富的功能。
  - 统一的设置架构。

4. SCIM和IBus

- SCIM的组成部件
  - 配置模块（Config）
    - 多个应用程序，如Panel、输入法服务进程、手写、输入设置界面等，往往要共享一些配置信息。所有的配置信息由一个应用程序负责统一管理。只有它能够直接存取配置文件，其它应用程序，都需要通过向它发送请求来存取配置信息。
  - 输入法前端模块（FrontEnd）
    - 前端是指输入法服务进程中接收请求的模块。它负责接收来自客户端应用程序的请求，把请求转发给具体的输入法引擎或者配置模块，最后把处理的结果返回给客户端应用程序。

4. SCIM和IBus

- SCIM的组成部件（续）
  - 输入法引擎模块（IMEngine）
    - 输入法引擎就是输入法的具体实现。SCIM只是一个输入法框架，其它输入法要挂到SCIM里来，就需要实现一个输入法引擎，它需要实现IMEngineInstanceBase接口，并编译成一个动态库，放在指定的目录之中。
  - 进程间通信模块（IPC）
    - SCIM采用本地的socket方式，应用层协议封装在Transaction中，它负责把特定的请求或事件等打包成数据包，也负责从数据包中取出请求或事件等。至于数据的实际传输，由Socket实现。

4. SCIM和IBus

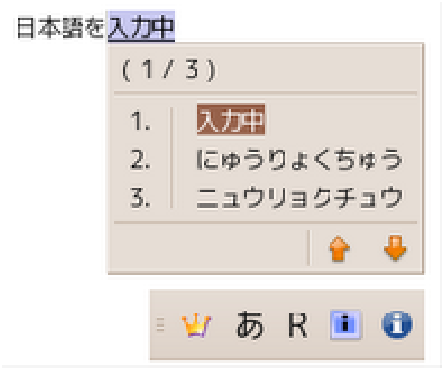
- SCIM的组成部件（续）
  - 输入法Panel模块
    - Panel不但给用户一种直观的反馈，如候选字，联想词组等，也提供了一些辅助功能，如中英文切换、全角半角切换，查看帮助信息等。
    - Panel是有GUI界面的，要与特定的GUI绑定起来。SCIM是一个输入法框架，它并不依赖于特定的GUI。尽管SCIM实现了一个基于GTK的Panel，但它并不属于核心之列，它是一个完全独立的工具。
  - 输入法Helper模块
    - 一些新的输入法方式，如手写输入法等，当作传统的IMEngine来实现，比较麻烦也不太优雅。SCIM把这些输入法作为特殊处理，通过Helper集成进来。

4. SCIM和IBus

- SCIM的组成部件（续）
  - 模块动态加载机制（Module）
    - SCIM只是一个框架，具体的输入法是通过动态库的方式加载进来的，而不是在编译时静态的绑定的。SCIM实现了一个Module类，封装了操作系统底层函数，提供面向对象的接口。另外，在此基础之上，还实现了其它几个类对Module进行包装，提供更具体的服务，它们是：
      - ConfigModule 用于动态加载配置模块；
      - FilterModule 用于动态加载过滤器模块；
      - FrontEndModule 用于动态加载前端模块；
      - IMEngineModule 用于动态加载引擎模块；
      - HelperModule 用于动态加载辅助功能模块。
  - 其他组件
    - 过滤器（Filter）模块，提供动态的转换或过滤功能，可以实现诸如在中文繁体和简体之间进行转换的功能。
    - 异常处理，SCIM完全采用OOP编程，使用了C++的异常处理机制，大部分模块都有自己的异常处理类。
    - Signal/Slot：为解决软件各层间的耦合，SCIM采用了Signal/Slot机制。

4. SCIM和IBus

- IBus（Intelligent Input Bus）是类Unix操作系统下的多语输入法平台。因为它采用了总线（Bus）式的架构，所以命名为Bus。



4. SCIM和IBus

- IBus的主要目标：
  - 更加易用的全功能输入法用户接口
  - 使用验证方法加强安全性
  - 为输入法开发者提供一个统一的接口及函数库
  - 符合来自不同地域、文化的用户需求

4. SCIM和IBus

- IBus的特点：
  - 输入法引擎可以随时加载与卸载
  - Systray支持
  - XKB支持
  - 配置选项的变动可实时生效
  - 提供 C 与 Python 的接口

4. SCIM和IBus

- IBus体系架构
  - IBus 是用 C 及 Python 开发的，如此可以避免 C++ ABI transition 问题。IBus主要通过下列三种服务(Service)来提供功能：
    - 输入法引擎服务：为输入法本身。
    - 配置服务：管理IBus以及输入法的设置选项，配置文件多以XML的形式存在。
    - 控制面板服务：提供诸如语言条，候选字菜单等用户界面。
  - IBus使用D-Bus作ibus-daemon服务与IM客户端（像是konsole、gedit、firefox等）之间的通信。ibus-daemon通过接收来自服务的登录请求，并发送D-Bus消息来管理服务及IM客户端。
  - IBus采用 C/S 结构，所有输入法引擎都是单独的进程，可以防止引擎之间互相影响，同时可以轻松实现引擎的动态加载和卸载。
  - IBus支持XIM协议，以及GTK IM模块和QT IM模块。

4. SCIM和IBus

- IBus的组成
  - 面板，提供与用户界面直接相关的接口。
    - 实现预编辑缓冲区，辅助文本，语言栏，按钮等操作。
    - 文本属性列管理。
    - 热键及其相关事件设置，实现热键的添加，删除，查找等操作。
    - 候选列表的处理。
    - 输入法引擎属性的用户界面组件设置。
    - 显示相应的输入法引擎的状态。

4. SCIM和IBus

- IBus的组成（续）
  - 输入法引擎模块，提供实现具体输入法必要的对象和功能函数，包括
    - 组件说明，是一个可执行程序，输入法引擎开发者可以提供一个XML文件来生成一个新组件对象。
    - 输入法引擎抽象对象，提供输入引擎的设计框架。
    - 输入法引擎描述，存储输入法引擎的描述数据，可以通过加载一个XML文件生成该对象。
    - 输入上下文代理，管理各种自然语言输入方法的上下文，客户端通过调用IBusInputContext来调用IBusEngine。
    - 键盘映射处理，定义键盘扫描码与键盘字符（数字，阿拉伯字母，标点符号）的映射关系。键盘符号定义。

4. SCIM和IBus

- IBus的组成（续）
  - 配置模块，提供配置IBus和输入法引擎的配置方法。如通过配置文件获取、设置引擎配置。
  - 通信模块，提供各个服务之间的通信功能。
  - 内部接口模块，提供IBus内部使用的定义和功能。

4. SCIM和IBus

- 现有输入法引擎列表
  - ibus-anthy: 日文输入法。
  - ibus-bopomofo: 注音输入法，基于ibus-pinyin引擎开发。
  - ibus-chewing: 新酷音输入法。
  - ibus-hangul: 韩文输入法。
  - ibus-m17n: 使用m17n-db的多语输入法。
  - ibus-pinyin: 拼音输入法，为IBus主要开发者所开发。
  - ibus-table: 码表输入引擎。

4. SCIM和IBus

IBus输入法配置界面