酷音輸入法技術報告

龔律全(lckung@iis.sinica.edu.tw)

April 11, 2002

Contents

1 前言

這份文件的內容在於詳細的分析酷音輸入法的內部結構，主要對象是有意繼續發展及修改酷音的程式，或是對酷音的內部機制有興趣的讀者。對於想知道如何使用酷音輸入法的一般的使用者，請參考附錄中的酷音輸入法使用手冊。

酷音輸入法是由中研院資訊研究所的徐讚昇研究員所指導，台大資訊系龔律全以及台大電機系陳康本所共同發展，另外要感謝XCIN project的謝東翰<thhsieh@linux.org.tw>提供許多幫助，還有許多網路上的朋友，熱心地幫忙測試。

1.1 酷音輸入法是什麼？

簡單來說，酷音輸入法是一個智慧型的中文注音輸入法，它是建構在一個中文輸入系統XCIN[3]之上的一個輸入模組，透過酷音輸入法的智慧猜字，你將可以大大提昇中文輸入的速度，使得在UNIX平台輸入中文，幾乎就像在Windows系統下一樣的方便。

一個所謂智慧型的中文注音輸入法，它是接受使用者所輸入的注音，再根據上下文的關係、使用者以往輸入的習慣，依機率高低列出使用者可能想輸入的字。這樣的系統在Windows平台上已經行之有年，目前最受歡迎包括最早發展的自然注音、倚天中文忘形輸入法，以及微軟的新注音等等，但是這些系統都是在Windows平台上的商業軟體。

反觀在UNIX平台上的中文輸入，從最早工作站上的cxterm，一個獨立的中文終端機，本身支援不少輸入法，但是只能在終端機上輸入中文，對於X下的其他應用程式則是無法作用，相當不方便。後來的XCIN 2.3配合XA+CV，終於能在除了終端機之外的某些X Window程式輸入中文，但是這種hacking的方法相當不穩定，而且缺乏一個統一的標準。後來由X consortium在X11R6.4中訂立了XIM protocol，這是一個全面性的標準，也就是只要應用程式支援XIM，就可以接受來自任何一個XIM server的輸入。

於是國內open source界的幾位前輩的努力之下，第一個由台灣人自行發展的XIM Input Server，XCIN 2.5，終於成功的發展出來，XCIN本身提供了一些傳統的輸入法，例如：注音、倉頡等，但是對於習慣了dos/Win平

台中的智慧型注音輸入法的使用者而言，還剩下一個問題：沒有順手的注音輸入法可用，而這個問題也使得Free Software 如Linux 在台灣遲遲無法順利推廣。

酷音輸入法，就是在這樣的需求下所發展。首先，爲了讓大部分的使用者能迅速適應，酷音的操作介面和在Windows下的輸入法是非常相似的，使用者不需經由任何學習，自然而然就會上手。酷音的另一個特點，就是它的速度：由於程式碼的簡潔明確，資料結構的快速有效率，使得酷音和其他輸入法比較起來，反應時間明顯的短了許多。

另一個重要的特點，酷音輸入法是一個開放原始碼的軟體，它的授權聲明是採用GPL，也就是說，任何人都可以免費的拿到酷音輸入法的原始碼，並且修改它，把成果繼續回饋到開放原碼的社群之中。

2 基本架構

在介紹酷音輸入法之前，首先先介紹幾個名詞：

- XIM (X Input Method)[2]
  X Window 下的輸入法標準，規範了應用程式(application)與輸入法(input method)之間的通訊協定(protocol)。

- XCIN (X Chinese INput method)
  一個中文的XIM Input Server，輸入法模組可以以外掛(plugin)的形式加入XCIN。

酷音輸入法是一個獨立於作業系統的中文注音輸入法，雖然理論上可以在各個平台執行，但目前只有實作XCIN之下的輸入法模組。透過XCIN，酷音輸入法可在X Window下所有支援XIM的應用程式互動、輸入中文，而酷音輸入法本身可再分割爲幾個模組，其名稱及功能如下：

- 使用者介面模組(User Interface module)
  提供回應使用者按鍵所呼叫的函式，以及操作edit buffer、選字選詞的功能。

- 字詞庫模組(Dictionary module)
  提供讀取詞庫及字庫，以及動態加入新詞的介面。

- 斷詞模組(Segmentation module)
  從使用者的輸入，根據字詞庫以及猜字的法則，找出最有可能的斷詞方法。

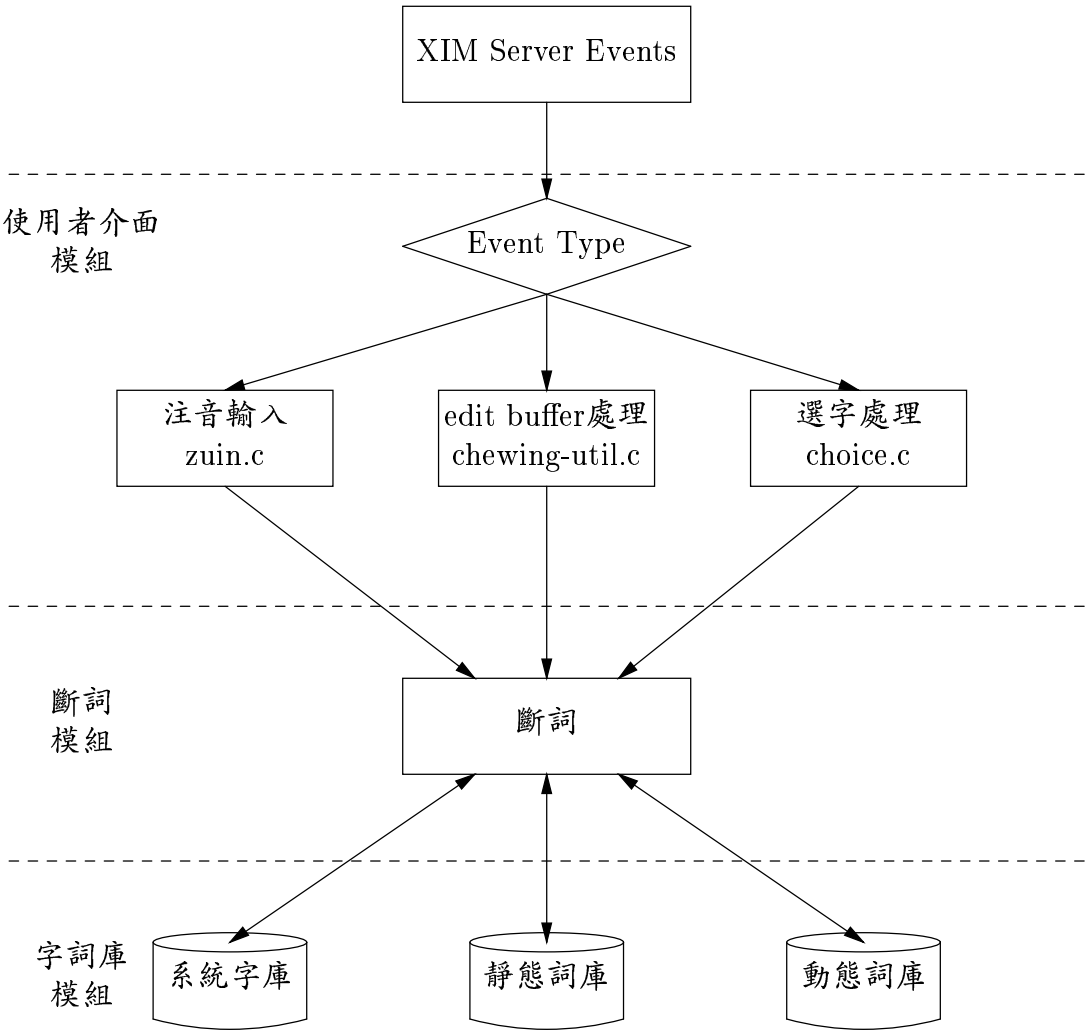

Figure 1: 系統架構圖

Fig1是系統架構圖，系統大致的流程是由XIM server送events 到使用者模組中，再由使用者模組呼叫斷詞模組，而斷詞模組又再呼叫了字詞庫模組，在斷詞完成之後，再度呼叫使用者模組把結果傳回application。

底下我們詳細介紹各模組的功能及使用方法。

3 字詞庫模組

酷音輸入法中最重要的資料，就是字庫與詞庫。詞庫的主要功能是由音節序列(syllable list)，找到所有和它對應的中文詞(phrase)。一個所謂的音

節(syllable)是由一到多個注音符號所組成，例如在「ㄔㄥˊ，ㄕˋ」中，「ㄔㄥˊ」和「ㄕˋ」兩者都是一個syllable，而「ㄔㄥˊㄕˋ」則是一個長度爲2 的syllable list，「程式」及「城市」則是這個注音序列所對應到的phrase，其關係架構如Fig2。

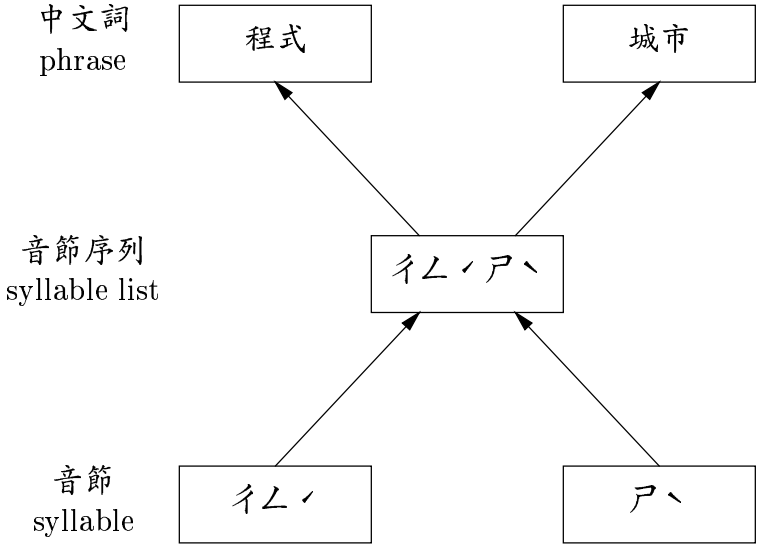

Figure 2: 音節、音節序列、中文詞

另外每個syllable在系統裡，被編碼成一個unsigned short(16 bits)，這使得儲存的空間得以減少，搜尋的速度得以增加。而其編碼的方式就是把注音符號分爲四類：聲母、韻母、「ㄧㄨㄩ」、以及聲調符號「˙ˊˇˋ」。根據每類符號的總數，分配足夠的bits數給該類。結果聲母共需5 bits、韻母4 bits、「ㄧㄨㄩ」2 bits、以及聲調符號3 bits，總共是14 bits就足以表示所有的syllable。這部分編碼的程式位於 key2pho.c 的PhoneBg2Uint函式之中。

每個syllable list可能會對應到0~N個中文詞，我們只注意那些對應到1個以上的詞的syllable list，這種syllable list我們又把他稱爲是一個同音詞。一個同音詞實際上又對應到一個以上的中文詞，從這些同音的中文詞之中挑出使用眞正想輸入的詞，就是斷詞程式最主要的工作。

另外字詞庫模組再根據詞庫的形式，可再區分爲靜態詞庫與動態詞庫兩種。其中靜態詞庫又稱系統詞庫，其中儲存的詞，是整個系統上的使用者所共用的，在系統開始時就已經存在，一般使用者沒辦法直接去修改它。而動態詞庫是使用者在使用酷音輸入法的過程之中，因爲選字或是加詞所學習而來的詞，這些詞通常是屬於比較個人化的詞。這兩種詞庫在使用時機、詞目數量上來說，都有很大的不同，底下我們就分別對這兩個詞庫來說明。

3.1 靜態詞庫

靜態詞庫又稱爲系統詞庫，是整個系統共用的基本詞庫。酷音輸入法目前所使用的詞庫，是來自**libtabe** [1]中的 tsi.src ，目前總詞數約11萬詞。

3.1.1 詞庫原始檔–tsi.src

tsi.src 是一個文字檔，每行是一個詞的記錄(record)，每筆詞又包括了中文詞(phrase)，詞頻(frequency)，注音序列(phone 1..n)等等資料。其格式如下：

[phrase] [frequency] [phone 1] [phone 2] .. [phone n]

phrase 中文詞

frequency 詞頻

phone 1..n 此詞所對應的注音序列

以下是三個範例：

一病不起　4 ㄧ ㄅㄧㄥˋ ㄅㄨˋ ㄑㄧˇ
一病不起　4 ㄧˊ ㄅㄧㄥˋ ㄅㄨˋ ㄑㄧˇ
八國聯軍　28 ㄅㄚ ㄍㄨㄛˊ ㄌㄧㄢˊ ㄐㄩㄣ

從這個例子中，“一病不起”是詞目，而4是這個詞的頻率。頻率的大小和這個詞在統計上的機率成正比，也就是說，“八國聯軍”這個詞在系統中的機率比“一病不起”要高了7倍，詞頻愈高的詞愈容易被斷詞程式所選用。由於這個頻率是統計一般性的中文文章中的字詞而得，所以在使用過程中，這個數字並不會改變，這也就是靜態詞庫爲何被稱爲靜態詞庫的原因。

而在上面例子中，注意到如果同一個中文詞有兩個以上的讀音(破音字)，則在 tsi.src 中需以兩行分別列出。例如“一病不起”中的“一”，有人讀一聲，也有人讀二聲，則在 tsi.src 中應有如上的兩行，以表示兩種讀法都是合法的。

有了 tsi.src 這個原始檔之後，實際上酷音輸入法在執行的時候，並不能直接讀入 tsi.src ，原始檔必需先經過 sortdict ， maketree 這兩個程式處理，產生出最後執行時所讀取的三個資料檔： fonetree.dat , ph_index.dat , dict.dat ，這三個資料檔的名稱及內容如Table1。

底下我們詳細介紹 sortdict 與 maketree 做了哪些事：

名稱	內容	敘述
fonetree.dat	詞樹結構檔	儲存同音詞所構成的詞樹
ph_index.dat	詞庫索引檔	指出某個同音詞在詞庫內容檔中的起始位置
dict.dat	詞庫內容檔	實際存放詞的檔案

Table 1: 詞庫資料檔名稱及內容

3.1.2 Sortdict

sortdict 這隻程式的功能產生詞庫索引檔和詞庫內容檔，其流程如下：把 tsi.src 中的詞全部先讀入記憶體中，再依照其注音序列來排序，排序之後一邊把詞的內容寫入內容檔之中，一邊把同音詞之中的第一個詞的位置當做索引，寫入索引檔之中。這個程式同時還有一個功能，就是產生 maketree 所需的詞樹中介檔，這個中介檔的內容就是所有同音詞的注音序列，按照排序之後的結果來儲存。

3.1.3 Maketree

maketree 這隻程式負責產生辭樹的結構檔。辭樹的資料結構是一個trie，也就是在辭樹上的每個除了根節點(root)之外的節點都代表了一個音節(syllable)，每個可以從根節點走到的節點，其路徑上的syllable合起來，就代表了這個節點的syllable list(參見Fig 3) 而並不是所有的節點都代表一個詞，有些是被其他更長的詞的所包含，例如"國民大會"中，"國民"是一個詞，"國民大會"也是一個詞，但是"國民大"則不是一個詞。對於這些不是詞的節點，我們令其phone_phr_id欄位之值爲-1，而對於其他的節點則依照BFS編號。

maketree 程式演算法簡略的來說，就是先把同音詞依序讀入，一個一個插入辭樹中，接著再用BFS(Breadth First Search)把每個節點編號，第三步則是把每個節點的注音，同音詞編號，以及子節點的起始編號與截止編號，寫入詞樹結構之中。由於輸入的同音詞已經事先排序過了，所以我們得到了兩件好處。第一，是有相同prefix的詞，短的一定比長的先被插入，第二，某個節點的所有子節點，是依照其注音的大小的順序被插入的。

因此，若我們把這個詞樹用BFS的方式編號，把這個編號叫做node_id，則可以得到父節點的node_id一定比子節點小，且同一個父節點的子節點們，它們彼此之間的node_id會是只差一號的，例如Fig 3中的「國民」、「國家」兩個詞的node_id就只差一號，「國民大會」和「國民小學」的node_id也只差一號。因此要紀錄一個父節點的所有子節點，只需要記錄

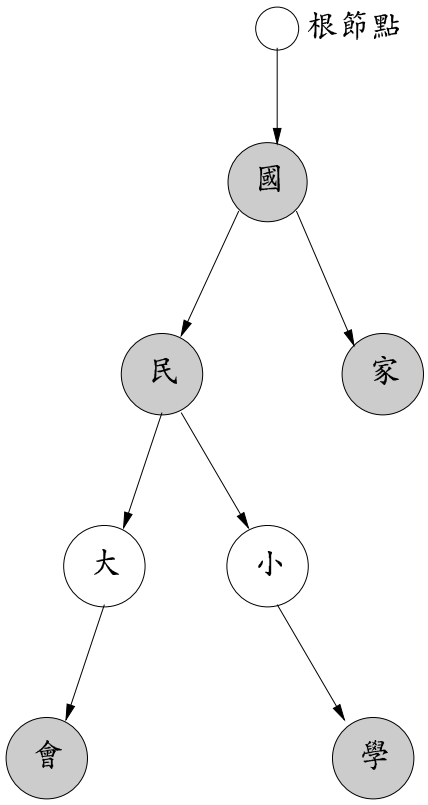

Figure 3: 詞樹-國民大會(灰色節點爲中文詞，白色則不是)

其子節點中，最小的node_id與最大的node_id即可，其他子節點的node_id，一定會介在這兩個最大與最小值之間，這個性質在之後搜尋詞樹的時候將會被利用來加速搜尋所需的時間。

總結來說，一個詞樹節點(node)中所需記錄的資料有下列幾項：

1. 音節(syllable)
   該節點所代表的注音

2. 同音詞編號(phone_phr_id)
   該節點所代表的同音詞，若無對應詞則爲-1

3. 子節點起始編號(first_child)
   連結到第一個子節點

4. 子節點結束編號(last_child)
   連結到最後一個子節點

而節點本身的編號並不需要另外儲存，因爲全部的節點可視爲一個大的陣列，每個節點所在位置的流水號就是該節點的編號，把全部的節點資料輸出到 fonetree.dat ，就完成了 maketree 的所有工作。

maketree 的複雜度分析如下：首先需$O(n)$ 的時間把同音詞讀入並建立詞樹，其中n 是同音詞的個數。詞樹的建立過程中，每次插入需要最多k次binary search，k是一個小於詞數高度的數字，由於詞樹高度決定於詞庫中最長詞的長度，所以k可視爲被一個constant所限制住，因此建立詞樹的複雜度是$O(kn)$。詞樹建立完成之後，又需要以BFS 方式travel 詞樹2次，因此總複雜度是$O(n)$ 。

3.1.4 搜尋詞樹

經過了 sortdict 以及 maketree 兩個步驟之後，我們已經得到了一個完整的詞樹結構，在辭樹上尋找一個同音詞，可根據演算法Listing 1。

3-5 從根節點開始，找根節點之下的子節點。

7-9 以binary search搜尋是否有子節點的鍵值(注音)和目標注音序列的第i個(phone_seq[i])相同。

10-11 如果沒有，則表示這個詞不在詞樹之中。

13-14 如果有的話，就把目前的節點設定爲找到的這個節點，再依序找phone_seq[1] 、phone_seq[2] ...。

Listing 1: 搜尋詞樹之演算法

```
1  FindPhrase(root,phoneseq[])
2  begin
3      parent := root;
4      i := 0;
5      while i <= length(phoneseq) do
6      begin
7          p := BinarySearch( phoneseq[i]
8                           , FirstChd(parent)
9                           , LastChd(parent) );
10         if p = nil then
11             return NOT FOUND;
12         else
13             parent := p;
14         i := i + 1;
15     end;
16     return p;
17 end;
```

16 如果可以一路找到最後一個，就代表存在這個同音詞。

以上搜尋詞樹的程式碼是在tree.c :TreeFindPhrase()函式中被實作。

在得到同音詞編號phone_phr_id之後，我們可以從詞庫索引檔(ph_index.dat)中得知這個同音詞在詞庫內容檔(dict.dat)中的第一筆資料起始位置，再到該檔中去把該同音詞的資料讀出，就完成了讀取詞的動作。

以上搜尋演算法的複雜度分析如下：由於長度是n 的輸入序列必須做n次binary search，但是每次binary search 最多不會超過$log_2(c)$ 次，c是詞樹的最大分枝度(degree)，是一個隨詞樹大小增加非常緩慢的函數，所以在詞樹數目變動不會太大的情形之下，可以當做是一個常數，是故總共的複雜度爲$O(log_2(c) * n)$。在目前的詞庫中，$log_2(c)$ 的實際值是11，也就是一個node 最多同時有1037 個children(root node有最多的children)。

3.1.5 靜態詞庫API

本節介紹詞庫模組中最重要的幾個function：

`int InitDict(const char *prefix)`

`prefix` 指向資料檔所在的位置

初始化詞庫模組

```
int GetPhraseFirst(Phrase *phr_ptr, int phone_phr_id)
```

phr_ptr 要填入的Phrase結構

phone_phr_id 要搜尋的詞之同音詞id

取得同音詞id的第一個中文詞

```
int GetPhraseNext(Phrase *phr_ptr)
```

phr_ptr 要填入的Phrase結構

取得同音詞id的其他中文詞，若已取得所有的詞，則傳回NULL

3.2 動態詞庫

動態詞庫又稱爲使用者詞庫，是使用者在輸入的過程中自行加入或是系統自動學習而來的詞。和系統詞庫不同的地方，在於動態詞庫是依照每個人的輸入習慣、詞彙的使用不同而有所不同，所以動態詞庫通常不具一般性，因此系統中的每個使用者都有屬於自己的動態詞庫。

3.2.1 檔案格式

這些詞是以hash table的格式儲存在檔案之中。其hash function是以詞的syllable list爲key，方法是把syllalbe list中的每一個syllable_id做XOR，也因此相同發音的詞會被hash到同一個linked list之中，而在這個linked list之中再採用linear probe 的方式來找到相同的詞。例如「程式、城市」這兩個詞因爲syllable list相同，所以經過hash function得到的slot number一定是一樣的，如果要進一步找出「程式」這個詞，就必須在該slot number的linked list之中一一比對，找出「程式」這個詞。

動態詞庫的資料檔案通常是位於 $HOME/.XCIN/hash.dat ，而檔案的格式是random access file，也就是每個record的檔案長度都是相同的。每個record 所紀錄的欄位對應到程式中的User_Phrase_Data 結構，其中包括了下面幾個欄位，並以「酷音」爲例：

1. wordSeq
   詞的本身，eg:“酷音”

2. phoneSeq
   詞的音節序列，eg:'ㄎㄨ4'的編碼是5380，'ㄧㄣ'的編碼是208

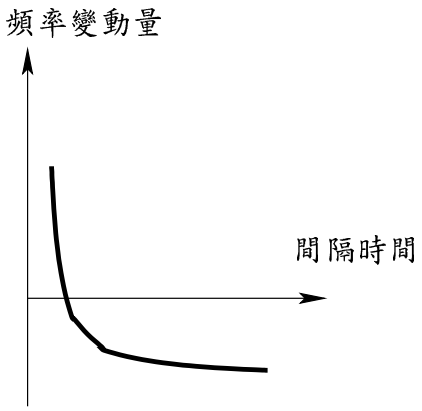

Figure 4: 頻率變動量與間隔時間關係示意圖

3. `userFreq`
   使用頻率，會隨著使用次數及相隔時間而增加或減少，第一次進入系統時會有預設值，之後每隔一段時間或是再次被挑選，則系統會更新它的頻率。此頻率和靜態詞庫的頻率並不能直接比較，只能反應出這個詞在系統中常被使用的程度

4. `recentTime`
   上次使用這個詞的系統時間，系統時間的計算是依照使用者按鍵的個數

3.2.2 加入或修改動態詞庫

有兩種情形會使一個詞被加入動態詞庫。第一種情形是一個原本就存在的詞，當它在選字的過程中被使用者挑選，則它的頻率會改變，但是這個改變不應該影響靜態詞庫，於是乎這個詞會被加入動態詞庫之中。第二種情形是使用者以手動加詞的方式加入一個原本不在系統中的詞。

動態詞庫中一個特別的地方，就在於詞頻的更新，在酷音輸入法中，我們假設詞頻的變動量是與上次使用之間的一個函式：距離上次使用時間愈近，詞頻增加的量就愈大，相反地，如果上次使用是很久很久以前的事，詞頻不僅不會增加，反而會隨之減少。因此以間隔時間爲橫軸、頻率改變爲縱軸，得到一個如Figure 4 所表示的頻率變動圖。這裡抽象的時間該如何定義呢？如果是用眞實世界的時間，那假使隔了一段時間沒使用酷音，那幾乎所有的詞頻都會被調降，於是我們採用輸入按鍵的個數來當做系統的時間，也就是使用者每敲入一個鍵，就等於酷音時間過去一秒鐘，如果某個詞的間隔時間爲100，就表示上次使用這個詞是100次按鍵之前。

這個方法看起來很合理，卻沒有考慮到同音詞之間頻率的關係。我們以動態詞庫改變靜態詞庫的頻率，爲的是希望斷詞程式能夠自動選出我們常打的詞。例如「程式」和「城市」，如果我們連續幾次輸入都是「ㄔㄥˊ2ㄕˋ4」，斷詞程式都是選出「程式」，但是由於我們現在輸入的文章當中比較多「城市」而不是「程式」，所以我們希望在幾次選詞之後，酷音能夠知道我們比較想輸入的是「城市」而不需再選字。要達到這樣的效果，頻率的變動量不應該只是時間的函式，也應該和其他同音詞的頻率相關。同音詞之間的頻率如果原本就相差很大，則每次頻率變動的量也比較大，這樣後面的詞才有可能超越前頭的詞。若同音詞之間的頻率相差小，頻率變動量小，但是只需要比較少的選字次數後面的詞就能超過前面的詞。

3.2.3 動態詞庫API

```
int UserUpdatePhrase(const uint16 phoneSeq[],const char wordSeq[])
```

phoneseq[] 音節序列

wordSeq[] 對應的中文詞

更新或新增一個動態詞

```
int UpdateFreq(int freq,int maxfreq,int origfreq,int deltatime)
```

freq 目前的詞頻

maxfreq 同音詞中最大的頻率

origfreq 原本的詞頻

deltatime 自上次使用經過的時間

更新一個詞的頻率

```
UserPhraseData *UserGetPhraseFirst(const uint16 phoneSeq[])
```

phoneSeq[] 欲讀取的詞之音節序列

讀取動態詞庫中第一個詞

```
UserPhraseData *UserGetPhraseNext(const uint16 phoneSeq[])
```

phoneSeq[] 欲讀取的詞之音節序列

讀取動態詞庫中的其他詞，若已沒有下一個，傳回NULL

3.2.4 動態詞與靜態詞的配合

在詞庫被斷詞模組或是選字模組使用的過程之中，動態詞庫和靜態詞庫會一起被搜尋，實際上是動態詞庫的資料優先於靜態詞庫的資料，也因此在斷詞的時候，使用者最常使用的詞會被放在頭幾個。也因此每位使用者常使用的詞，必須分隔在各自的動態詞庫中提高頻率，不會因爲某個使用者比較常使用某些詞，而影響到系統詞庫。

4 使用者介面模組

在酷音輸入法中其實並不會直接呼叫Xlib或是其他toolkit去顯示GUI，這些工作已經統一交由XCIN去執行，酷音只要把一些資訊：edit buffer內容、游標位置、選字資訊都傳給XCIN，所以使用者介面模組的功能，包括與(1)XCIN之溝通介面，(2)處理使用者的輸入動作，例如注音的輸入、游標的移動、選字、加詞等等動作，再呼叫實際執行的函式，(3)處理注音的輸入，根據目前的鍵盤模式，轉換注音輸入，(4)處理使用者的選字動作等等。

4.1 與XCIN之溝通介面

在XCIN及酷音執行之後，使用者每按下一個按鍵，ChewingKeystroke() 會被呼叫，這個函式最重要的功能之一就是對照XWindow中所定義的按鍵代碼，呼叫位於 chewingio.c 中的 OnKey<KeyName> 等等函式，

酷音輸入法對外溝通的介面函式都在 chewingoio.c 中，裡頭的函式都是以 OnKey<KeyName> 的名稱命名。這些函式都有兩個共通的引數，一個是型態爲指向 ChewingData 結構的指標，另一個是型態爲指向 ChewingOutput* 結構的指標。而前者是儲存著目前輸入法的狀態，是屬於酷音輸入法內部使用的資料，而後者用來與外界溝通的資料，其欄位內容及意義如Table2。

透過 ChewingOutput 資料結構，酷音輸入法得以與外層的系統做溝通。按鍵透過 chewingio.c 裡面的 OnKey<Keyname> 等函式進入酷音，然後依照按鍵來分配功能，每個按鍵相對應的工作各不相同，而其中共同的複雜工作就可以呼叫 chewingutil.c 裡面的函式來支援，這些工作或著改變了 ChewingData 的內容，或是把輸出結果放進 ChewingOutput 。其中有更改到edit buffer的event必須呼叫 CallPhrasing 來進行斷詞，之後程式流程又回到 ChewingKeystroke() 之中，這時候 MakeInpInfo() 被呼叫，所有 ChewingOutput 的資料被轉換成XCIN所需要的格式。這個流程可以參見圖Fig5。

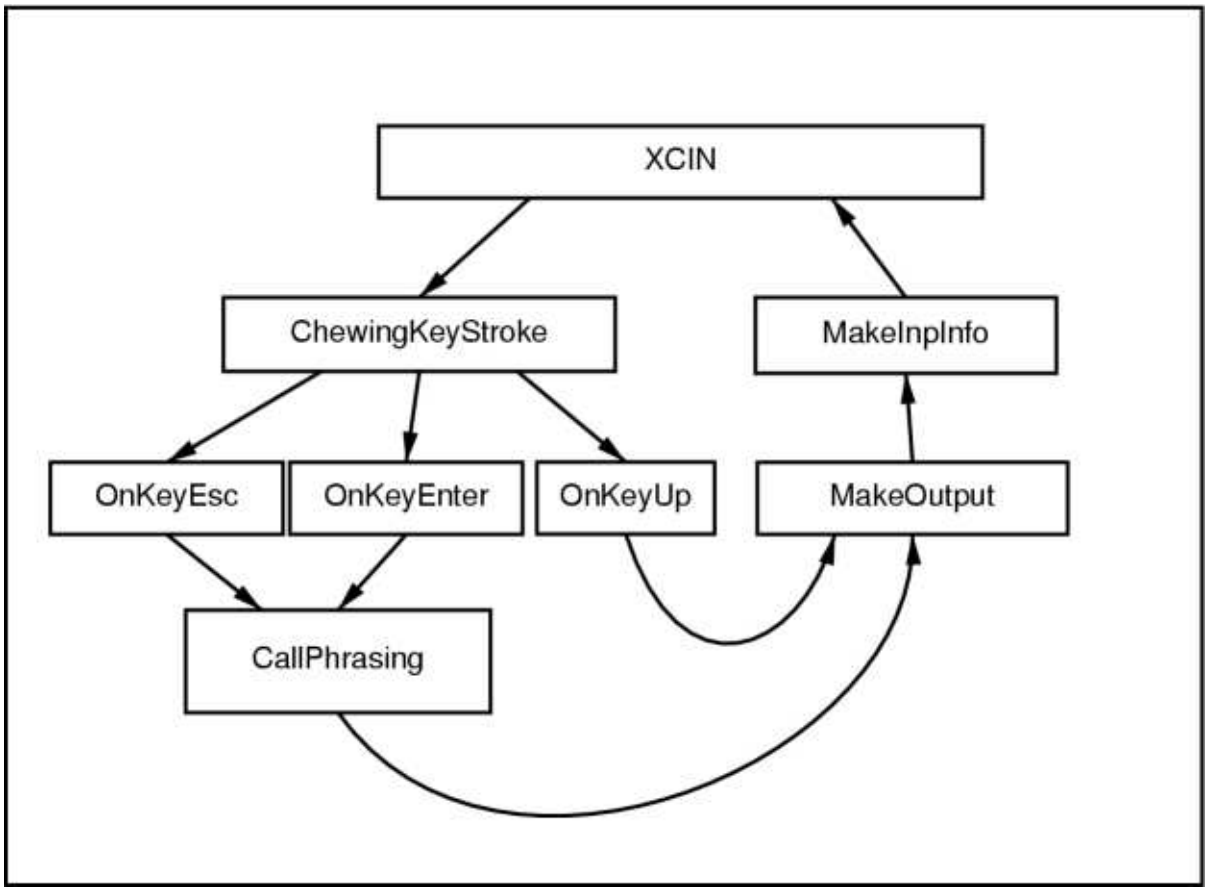

Figure 5: 使用者模組流程圖

欄位名稱	型態	意義
chiSymbolBuf[]	wch_t	Edit buffer內容
chiSymbolBufLen	int	Edit buffer長度
chiSymbolCursor	int	目前cursor所在位置
zuinBuf[]	wch_t	已輸入的注音符號
dispInterval[]	IntervalType	要顯示的斷詞方式
nDisplayInterval	int	
dispBrkpt[]	int	要顯示的中斷點
commitStr[]	wch_t	使用者完成輸入，要送出的字串
pci	ChoiceInfo*	選字資訊
bChiSym	int	中文或英文模式
keystrokeRtn	int	傳回值
bShowMsg	int	
showMsg	wch_t	要告知使用者的訊息
showMsgLen	int	

Table 2: `ChewingOutput` -結構定義

4.2 注音輸入與鍵盤配置

在酷音輸入法中，注音的輸入方式被獨立成爲一個模組，只要按照既定的函式格式，程式開發人員可以自行加入，除了目前鍵盤對應外的其他的輸入方式。而目前酷音0.0.5版支援的輸入方法，包括一般鍵盤與許氏鍵盤，其位在 zuin.c 之中的處理函式如下：

- `HsuPhoInput()` 許氏鍵盤輸入
- `DefPhoInput()` 一般鍵盤輸入(注音與按鍵一對一對應)

其中一般鍵盤輸入可以支援所有一對一的鍵盤對應，目前支援有一般、IBM、精業、倚天等四種注音配置方式。開發人員若想加入新的對應，只需在 key2pho.c 的 `key_str[]` 陣列中加入對應按鍵，另外在 zuin.h 中更改 `KB_TYPE_NUM` 的定義敘述即可。 `key_str[]` 的格式相當簡單，它是一個陣列，對應著是ㄅㄆㄇㄈ依順序對應到那個按鍵(ASCII Code)，例如標準鍵盤的ㄅ、ㄆ、ㄇ、ㄈ分別對應到'1','q','a','z'，所以底下Fig 6 的第1 行資料便是"1qaz..."。

整個注音模組的處理流程如下：在 chewingio.c 中的 `OnKeyDefault()` 會把注音按鍵輸入送至 zuin.c 中的 `ZuinPhoInput()` 函式，這個函式則會根據目前所選擇的輸入法，把按鍵分送至對應函式，如 `DefPhoInput()` 或 `HsuPhoInput()` ，

```
char *key_str[MAX_KBTYPE] = {
  "1qaz2wsxedcrfv5tgbyhnujm8ik,9ol.0p;/-7634",      /* standard kb */
  "bpmfdtnlgkhjvcjvcrzasexuyhgeiawomnkllsdfj",      /* hsu */
  "1234567890-qwertyuiopasdfghjkl;zxcvbn/m,.",      /* IBM */
  "2wsx3edcrfvtgb6yhnujm8ik,9ol.0p;/-['=1qaz",      /* Gin-yieh */
  "bpmfdtnlvkhg7c,./j;'sexuaorwiqzy890-=1234"       /* ET  */
} ;
```

Figure 6: key_str[] in key2pho.c

代號	意義
ZUIN_ABSORB	正確的按鍵但是尚未完成組字
ZUIN_COMMIT	完成組字，而且是正確的音
ZUIN_KEY_ERROR	此按鍵並不符合這個輸入法
ZUIN_ERROR	發生內部錯誤
ZUIN_NO_WORD	完成組字，但是沒有這個音的字存在

Table 3: 注音輸入函式傳回值及意義

再根據傳回值來決定注音輸入狀態。詳細的注音輸入狀態共分成六種，參見Table 3。

4.3 選字處理

選字是注音輸入過程中，除了注音輸入之外最重要的events，每當選字鍵被使用者按下時，選字的模組就會被驅動，呼叫位於 choice.c 中的 ChoiceFirstAvail() 進入選字模式。 ChoiceFirstAvail() 這個函式是負責初始化相關結構。接著呼叫 SetAvailInfo() ，這個函式會把目前游標所在的位置之後的所有可能詞，依長度由長至短，都載入一個名叫 AvailInfo 的結構之中，以提供同一次選字過程中的可選詞長度資訊，其結構定義如下：

```
typedef struct {
  struct {
    int len ;
    int id ;                  // phone id
  } avail[MAX_PHRASE_LEN] ;  // all kinds of lengths of available phrases.
  int nAvail ;               // total number of availble lengths
```

```
  int currentAvail ;          // the current choosing available length
} AvailInfo ;
```

因爲游標在同一位置按下選字鍵時，使用者想選的詞長度有許多種可能，例如第一次是選四字詞，第二次是二字詞，第三次才是單字詞等，AvailInfo.avail[] 就是儲存這幾種可能的陣列。SetChoiceInfo() 是針對某個長度的詞，讀入所有的可選詞，包括動態詞庫與靜態詞庫，置入 ChoiceInfo 結構之中，此結構中還包含了選字頁的總數和目前切換到第幾頁的資訊，而其中 totalChoiceStr 存放眞正供選擇的字串，動態詞庫的詞放在靜態詞庫之後，而且是依它們的頻率由高到低來排序。

```
typedef struct {
  int nPage ; // total page number
  int pageNo ; // current page number
  int nChoicePerPage ;  // number of choices per page
  char totalChoiceStr[MAX_CHOICE][MAX_PHRASE_LEN*2 + 1] ;
  int nTotalChoice ; // number of strings to choose
  int oldCursor, oldChiSymbolCursor ;
} ChoiceInfo ;
```

當使用者按下第二次選字鍵(方向鍵-下)時， ChoiceNextAvail() 傳回比上一次選的詞短的下一種選法，在確定選擇之後， ChoiceSelect() 把選到的字送出，而當使用者不想選字，而是按下Esc鍵跳離選字模式，則 ChoiceEndChoice 則是跳離選字模式。

總結選字處理的函式如Table4，檔案位於 choice.c 。

函式	功能
ChoiceFirstAvail	初始化選字相關資訊，呼叫SetAvailInfo()
SetAvailInfo	設定AvailInfo結構，用以提供可選詞有哪幾種長度
SetChoiceInfo	實際讀入某個長度的所有詞，包括靜態與動態詞庫
ChoiceNextAvail	傳回下一種可選詞長度
ChoicePrevAvail	傳回上一種可選詞長度

Table 4: 選字模組-重要函式列表

5 斷詞模組

斷詞模組是整個輸入法程式的核心，它的輸入有三個：

1. 注音序列
   目前edit buffer中的注音

2. 中斷點
   使用者輸入的中斷點，包括用Tab鍵產生，以及因爲中英混合而產生的中斷點

3. 已選字
   使用者已選字的資訊

斷詞的輸出則是最有可能的斷詞方式及同音詞列表。

簡單的來說斷詞演算法的步驟，一開始是找出所有可能的詞，這些詞的起始範圍(interval)構成了一個連接圖(word graph)，接著在這連接圖之內，我們可以先把某些不需要的interval去掉，再找出相容的幾種連接方式，再從這幾種連接方式中找出可能性最大的，也就是累計頻率最高的那一種分割方式。

5.1 搜尋interval

在搜尋可能的intervals時，`FindInterval()`會到詞樹中去尋找可行的interval，詞樹的資料結構以及搜尋詞的方式在本文件詞庫模組的部分已經說明。不過有兩個不同點。首先，在固定的syllalbe list之下，我們對於同一個起始點的intervals之間的搜尋是採用incremental的方式，也就是每次搜尋並不一定是從根節點開始，而是從詞樹中前一個節點再繼續。例如輸入的注音是 ㄍㄨㄛ2 ㄇㄧㄥ2 ㄉㄚ4 ㄏㄨㄟ4(國民大會)四個字，在搜尋到「國民」這個詞之後，我們已經取得了詞樹中第二個字的節點$\mathcal{N}$，於是若要測試「國民大」是否也是一個詞，只需從$\mathcal{N}$ 再往下搜尋一個子節點，以此類推，「國民大會」也只需要再一次子節點的搜尋。

依這個演算法，對於一個長度爲n 的注音序列來說，假設詞樹的平均分枝數爲k ，若限定詞是從第一個音開始，則搜尋第一個音需要$log_2(k)$ ，再接著搜尋第二個音所需的時間一樣爲$log_2(k)$ ，這樣的搜尋總共n 次，再把從第1、第2...、第n個字開始的時間加總起來，得到時間複雜度爲：

$$\begin{aligned}
T_1(n) &= log_2(k) + log_2(k) + \cdots + log_2(k) \\
&= n \cdot log_2(k) \\
T(n) &= T_1(n) + T_2(n) + \cdots + T_n(n) \\
&= log_2(k)[n + (n-1) + (n-2) + \cdots + 1] \\
&= log_2(k) \cdot O(n^2) \\
&= O(n^2)
\end{aligned}$$

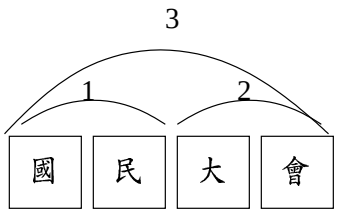

Figure 7: 去除贅詞-1

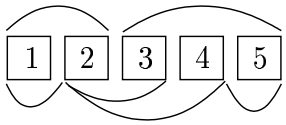

Figure 8: 去除贅詞-2

另外，由於使用者可以選擇中斷點及已選字，這些資訊都提供了斷詞的指示，也就是在可能的interval中，跨過中斷點及不包含已選詞的同音詞是必須被去除的。例如使用者在游標位置1 與2 之間下了一個中斷點，則[1,3]或[1,4]這些intervals 都必須被去除。而例如使用者選了在「中國民眾」中，已經選了「國民」這兩個字，則我們必須檢查所有包含[2,4]的intervals中，是否都存在一個詞在相對應的位置上有"國"及"民"。

5.2 去除贅詞

爲了增進斷詞處理的速度以及正確性，斷詞過程中不可避免的需要去除贅詞。去除贅詞的步驟分成兩個，第一部分Discard1函式是把某些範圍(intervals) 完全包含於其他詞的詞去除。例如在Figure 7之，詞(1)"國民"及詞(2)"大會"就被詞(3)"國民大會"所包含，所以這兩個詞的存在對於斷詞沒有幫助，可以刪去。

第二部分Discard2 則是把沒有辦法被從頭到尾連接的intervals 除去。如Figure 8 ，現在有6 個詞，其interval分別是[1,1], [1,2],[2,3],[2,4],[5,5],[3,5]，而整個輸入的長度是5。我們觀察到有兩種從頭到尾連接的方式，分別是：[1,2],[3,5] 以及[1,1],[2,4],[5,5] 這兩種方式。但是[2,3]這個詞，因爲沒辦法被連接到輸入字串的首尾，所以被刪去。經過這兩個步驟後所剩的intervals 都是我們認爲可能的斷詞方式，而這些斷詞的方式有許多種，我們得找一個可能性最大的傳回給使用者。

5.3 傳回可能的切分法

這部分的程序是在 RecursiveSave 中完成，這個函式把所有可能的連接方式都展開，把結果存到一個linked list之中。舉例而言，輸入"ㄌㄧㄡ2 ㄙㄨ ㄢㄖㄨㄥ2 ㄧ4 ㄒㄧㄤㄉㄤㄌㄧ4 ㄏㄢ4"，可能會有下列兩種斷詞方式：

硫酸 溶液 是 非常 厲害
硫酸 溶液 是非 常例 害

則RecursiveSave會把兩種可能的切分法都傳回。

接下來，SortListByFreq函式，就是從可能的切分法中，把該切分法內所包含的詞，這些詞的頻率和最大的切分法找出來。之後OutputRecordStr會從詞/字庫把頻率最大的詞/字填入intervals之中，所得到的就是使用者看到的字串了。

6 結語

以上我們詳述了酷音系統的內部結構和演算法，這個系統的開發平台是RedHat Linux 6.X 及XFree86 3.X，programming language是C。這個專案約是在1998年秋季開始，由徐讚昇研究員指導兩位台大電機及資訊系學生撰寫，到目前爲止，程式碼的總長度約爲5000 行，詞庫大小約十三萬(來自libtabe)，開發平台是RedHat Linux，測試平台則包括RedHat Linux以及FreeBSD。而這個成果已經在http://chewing.good-man.org/ 開發給所有人download 最新版本以及所有原始碼，在成果發表之後，有許多使用者寫信來提供意見及表示感謝，表示酷音的確是UNIX 上中文使用者等待已久的智慧型中文輸入法。

6.1 貢獻

酷音輸入法是目前UNIX 平台上的智慧型中文輸入法之中，最受使用者好評的一個輸入法。原因在於下列幾點：

1. 親和的使用者介面
   酷音的使用方式和DOS/Win 上的中文輸入法相近，使用者可在最短的時間內上手。例如特殊符號快速輸入，支援各種鍵盤配置，自動學習新詞，手動斷詞等等功能，都是其他輸入法所沒有的，UNIX下的應用程式向來以使用者介面不夠親和被詬病，酷音卻是一個難得的例外。

2. 有效率可移植的斷詞核心
   酷音輸入法的斷詞核心演算法經過嚴格的分析與測試，無論使用者輸入如何複雜的語句，都可以在最短的時間內產生正確的回應。而斷詞核心本身具有完整的API，可獨立出來，供其他project使用。

   例如在目前最流行的PDA、手機或是資訊家電用品上，其實都需要一個佔記憶體小、速度快的智慧型中文輸入法，酷音輸入法在這方面的優勢甚至比市面上任何一個商業的輸入法都強，斷詞模組可抽出單獨運作，套上不同的輸出入介面，開發人員即可把酷音輸入法移植到任何一個支援C語言的平台上。另外詞庫的大小也可以根據需求調整，不必擔心記憶體容量不足。

6.2 未來發展

1. Language Model
   目前酷音只使用了uni-gram的Language Model，若能克服記憶體大小的問題，把bi-gram language model加入程式之中，相信正確率會有相當大的進步。

2. rule-based 可擴充斷詞模組
   斷詞核心在設計上，預留了未來的擴充空間，新加入的parsing rule，可以以filter的形式，不用更改到舊有的程式，只需把新增的模組掛在酷音之上，就可以增加斷詞的功能。

   例如判別數字、判別年份、判別姓名、判別地址等等filter，都可由使用者提供模組，外加在酷音之上，如此酷音的正確率將能夠再進一步提昇，

References

[1] Shawn, *http://libtabe.sourceforge.net*, Libtabe - A Chinese chracter/word handling library

[2] Masahiko Narita and Hideki Hiura, *The Input Method Protocol – Version 1.0*, X Consortium Standard, X Version 11, Release 6.4

[3] Tung-Han Hsieh, *http://xcin.linux.org.tw*, The XCIN Project

[4] 鄭國揚、余方國, 一個沒有混淆現象的中文注意輸入法之探討, 中研院資訊所TR-87-007

[5] 許聞廉, *Chinese parsing in a phoneme-to-character conversion system based on semantic pattern matching*, International Journal on Computer Processing of Chinese and Oriental Languages 40, (1995),227-236

[6] 許聞廉、陳克健，自然智慧型輸入系統的語意分析—脈絡會意法, Proceedings of the 6th International Symposium on Cognitive Aspects of the Chinese Language, (1993), 527-540