PANDITHAM: An Optimal Character Oriented Protocol for Thamizh and Multilingual Computing

P. Navaneethan, R. Madheswaran, R. Balasubramaniam, and R.V. Bharathidasan
Department of Computer Applications
PSG College of Technology, Coimbatore, India

INTRODUCTION:

Ever since the advent of Windows Programming, people have succeeded in bringing computer to their respective languages. But the concept of fonts has restricted Multilingual Computing to Multilingual Word Processing only. This can be attributed to the fact that the fonts only map a typical ASCII sequence into one of the characters in a typical language. This paper, looks at some of the key issues involved in the coding of Multilingual characters, and introduces, as a case study, Tamil computing with the help of a new protocol, called PANDITHAM (Protocol for ApplicatioNs Development In THAmizh and Multilingual Computing).

This protocol, which is truly multilingual, helps realize Telephone directories in Thamizh, makes the machine read Thamizh text, facilitates the design of Character Oriented Thamizh Word Processor. This paper also critically looks at the performance of other Encoding schemes like Glyph based encoding, UNICODE, Consonant Vowel scheme, vis-a-vis PANDITHAM. This protocol which is character oriented, is optimal in the sense that it consumes only fewer bytes as warranted by the language.

PHONETIC CHARACTER ENCODING SCHEME

In this coding scheme, apart from consonants and vowels, a composite phonetic letter will also be given a code. We treat Thamizh language as made up of two logical languages, namely, pure Thamizh and Granth (vadamozhi) Thamizh. Refer to Tables I & II for Tamil phonetic character encoding. Since, the number of phonetically differing characters in a language are likely to exceed 128, we may have to use some control characters of ASCII as well. This calls for a Protocol rather than a simple Encoding scheme. Hence, this section introduces a relevant Protocol namely, PANDITHAM.

PANDITHAM

BASIC PRINCIPLES:

- Most of the languages have lexical order associated with their letters. Hence, for each letter we can associate the lexical order number itself as part of coding the letter. Refer to Table I for Tamil letter coding in PANDITHAM. One finds that,

  PANDITHAM (``) = Lexical_order(``)+ 8 = 1 + 8 = 9 (09H)

  PANDITHAM ('[f') = 247 + 8 = 255 (FFH)

Interestingly, PANDITHAM makes use of an 8-bit code; out of the 256 possible codes that can be thought of, 247 have been assigned already starting from 9 (09H) to 255 (FFH). The remaining 9 bytes 0 (00H) to 8 (08H) are to be used either as PANDITHAM Control and/or characters meant for punctuation.
By definition, a Multilingual string may have a combination of letters from more than one language; hence, we first standardize the language aspect, namely, language codes.

We now look at standardization and meaning of PANDITHAM Control and/or Punctuation characters.

- Start of the string: The control character DLC (02H), shall be followed by the code of the language of the ensuing string.
- Change of Language: The change can be in two different ways.

Way 1: The switching of language is such that at least 2 characters are there in the new language.

In this case, once again DLC is used in a similar way.

Way 2: The switching is momentary in nature; i.e., only one character temporarily changes to a new language, and the rest follows the previous language itself. To denote such an occurrence, we shall use the control character MLC (03h) to be followed by the language code.

- Termination of the string: The NULL character (i.e., 00H) shall serve the purpose.
- Frequently used de-limiting characters: Out of the 256 characters, we have already made use of three control characters, namely, DLC, MLC and NULL. The rest can in fact be used to codify letters of various languages. As an instance, the 247 Tamil letters shall be pushed in, and the remaining 6 can be used for codifying frequently used punctuation marks.

PANDITHAM Table I - Pure Thamizh Coding

	0	1	2	3	4	5	6	7
0	NUL	SP	DLC	MLC	ஶ்ரீ		,	-
1	ஏ	ஐ	ஒ	ஓ	ஔ	ஃ	க	கா
2	கோ	கௌ	க்	ங்	ஙா	ஙி	ஙீ	ஙு
3	ச	சா	சி	சீ	சு	சூ	செ	சே
4	ஞீ	ஞு	ஞூ	ஞெ	ஞே	ஞை	ஞொ	ஞோ
5	டெ	டே	டை	டொ	டோ	டௌ	ட்	ண
6	ணொ	ணோ	ணௌ	ண்	த	தா	தி	தீ
7	த்	ந	நா	நி	நீ	நு	நூ	நெ
8	பி	பீ	பு	பூ	பெ	பே	பை	பொ
9	மூ	மெ	மே	மை	மொ	மோ	மௌ	ம்
A	யை	யொ	யோ	யௌ	ய்	ர	ரா	ரி
B	ரௌ	ர்	ல	லா	லி	லீ	லு	லூ
C	வா	வி	வீ	வு	வூ	வெ	வே	வை
D	ழு	ழூ	ழெ	ழே	ழை	ழொ	ழோ	ழௌ
E	ளே	ளை	ளொ	ளோ	ளௌ	ள்	ற	றா
F	றோ	றௌ	ற்	ன	னா	னி	னீ	னு

	8	9	A	B	C	D	E	F
0	.	அ	DLC	MLC	ஶ்ரீ		,	-
1	கி	கீ	கு	கூ	கெ	கோ	கை	கொ
2	ஙூ	ஙெ	ஙே	ஙை	ஙொ	ஙோ	ஙௌ	ங்
3	சை	சொ	சோ	சௌ	ச்	ஞ	ஞா	ஞி
4	ஞௌ	ஞ்	ட	டா	டி	டீ	டு	டூ
5	ணா	ணி	ணீ	ணு	ணூ	ணெ	ணே	ணை
6	து	தூ	தெ	தே	தை	தொ	தோ	தௌ
7	நே	நை	நொ	நோ	நௌ	ந்	ப	பா
8	போ	பௌ	ப்	ம	மா	மி	மீ	மு
9	ய	யா	யி	யீ	யு	யூ	யெ	யே
A	ரீ	ரு	ரூ	ரெ	ரே	ரை	ரொ	ரோ
B	லெ	லே	லை	லொ	லோ	லௌ	ல்	வ
C	வொ	வோ	வௌ	வ்	ழ	ழா	ழி	ழீ
D	ழ்	ள	ளா	ளி	ளீ	ளு	ளூ	ளெ
E	றி	றீ	று	றூ	றெ	றே	றை	றொ
F	னூ	னெ	னே	னை	னொ	னோ	னௌ	ன்

PANDITHAM Table II - Grantha Thamizh Coding

	0	1	2	3	4	5	6	7
0	NUL	SP	DLC	MLC				
1.....								
B								
C	ஹி	ஹீ	ஹு	ஹூ	ஹெ	ஹே	ஹை	ஹொ
D	ஷூ	ஷெ	ஷே	ஷை	ஷொ	ஷோ	ஷௌ	ஷ்
E	ஸை	ஸொ	ஸோ	ஸௌ	ஸ்	ஜ	ஜா	ஜி
F	ஜௌ	ஜ்	கூ	கூா	கூி	கூீ	கூு	கூூ

	8	9	A	B	C	D	E	F
0	NUL	SP	DLC	MLC				
1.....								
B							ஹ	ஹா
C	ஹி	ஹீ	ஹு	ஹூ	ஹெ	ஹே	ஹை	ஹொ
D	ஷூ	ஷெ	ஷே	ஷை	ஷொ	ஷோ	ஷௌ	ஷ்
E	ஸை	ஸொ	ஸோ	ஸௌ	ஸ்	ஜ	ஜா	ஜி
F	ஜௌ	ஜ்	கூ	கூா	கூி	கூீ	கூு	கூூ

DESIGN OF LANGAUAGE AND FONT DATABASE

Any multilingual data processing should be based on the language and not on fonts, which are vulnerable to change. To facilitate this, a database is to be maintained, in the WinSysPath/language directory. The two main tables are the Language database and the Font database. The Language database consists of details like Unique Language Code, Language name, Classification, and Weight. The languages are classified into 3 categories based on the amount of storage requirements. The first category comprises of languages like English, which occupy single byte/character. The best example of second category would be Japanese language, which require 2 bytes/character. The language Thamizh comes under the third category, whose storage requirement lies in between 1 and 2 bytes/character (on the average 1.1 bytes/character). This extra 0.1 is basically due to the presence of infrequently used Thamizh characters (Grantha characters).

The font database stores a unique font code, font name and the language to which it belongs. Apart from these, a default font database is also maintained.

STRUCTURE OF DATABASE

The structure of the above mentioned database is described below:

```
Private Type Lang        /* Language Record Structure */
   LangCode As Byte      /* Unique Language Code */
   LangName As String          /* Language Name */
   Weight As Byte        /* Language Weight for sorting */
Classification As Byte   /*1 English Like, 2 Japanese Like, 3 Thamizh Like */
End Type
Private Type Fonts       /* Font Record Structure */
   FontCode As Byte      /* Unique Font Code */
   FontName As String    /* Name of the Font */
   LangCode As Byte      /* Language it belongs to */
End Type
Private Type Defa        /* Default Font Record Structure */
   LangCode As Byte
   FontCode As Byte
End Type
```

In Tamil, we do have a mix up of pure Tamil (Tamil-1) letters and Infrequently used Grantha Tamil (Tamil-2) letters. Refer to Tables I and II for PANDITHAM Tamil and Infrequently used Tamil tables. Let the language codes of Tamil-1 and Tamil-2, be 08h and 09h, respectively.

Ex. 1: Name: ரங்கராஜன்

Coding Scheme for the Name:

DLC TM1 ர ங் க ரா MLC TM2 ஜ ன் SP DLC ASC R . NULL

Where, SP = Blank Space

ADVANTAGES:

- Uniqueness of Code: The language code in conjunction with the PANDITHAM code uniquely specifies a character. For example, two bytes that account for the Tamil letter k are 08h and 16h, respectively.

- No Kerning Problem: Since there are no glyphs, no chance of kerning problem.
- Speech Synthesis: PANDITHAM defines speech units, each of which symbolizes the sound of a human utterance. These discrete speech units could be joined to regenerate the speech in such a way that the joins are not evident, using Digital Signal Processing.
- Simple and elegant sorting: Since, PANDITHAM tables assign values for various letters in the lexical order, a simple byte by byte comparison of two different strings tell us which should precede the other, in the case of Monolingual strings. It makes no sense to compare simply bytes of letters in the case of Multilingual strings. We can make the comparison of weights of languages as the basis for comparing multilingual strings.

Ex. 2: Name_1 = அருண் and Name_2 = அப்பர்

PANDITHAM representation,

Name_1: DLC TM1 09h A9h 63h Null
| <= Differ here
Name_2: DLC TM1 09h 8Ah 7Eh B1h Null
Name_1 > Name_2 [Since A9h > 8Ah].
Hence, Name_2 precedes Name_1.

I.e., அப்பர் precedes அருண்

This comparison doesn't call for parsing for identification of letters as is there in Glyph Based Coding schemes.

Ex. 3: Name_1 = ரஜினி and Name_2 = ரம்பா

In this example, in Name_1, there is infrequently used Tamil mix, while Name_2 has only pure Tamil. If we decide to have a sorting scheme which gives first preference to Tamil and second preference to Infrequently used Tamil, and last preference to English (ASCII) we need to assign weights as follows.

Weight [TM1] = 1; Weight [TM2] = 2; Weight [ASC] = 3

PANDITHAM representation of Names:

Name_1: DLC TM1 A5h MLC TM2 E7h F5h Null
| <= (Language changes)
Name_2: DLC TM1 A5h 97h 7Fh Null

Weight [Common Lang.] < Weight [New Lang.]
Weight [TM1] < Weight [TM2] (i.e. 1 < 2)
Therefore, Name_2 < Name_1.

Hence, Name_2 precedes Name_1.

I.e., ரம்பா precedes ரஜினி

Ex. 4: Name_1 = Bharathi and Name_2 = pala
Name_1 = DLC ASC 42h 68h 61h 72h 74h 68h 69h NULL
|<= (Languages differ)
Name_2 = DLC TM1 7Fh B3h NULL
Weight [ASC] > Weight [TM1] (i.e. 3 > 1)

Hence, Name_1 succeeds Name_2.

- Flexibility in sorting: The flexibility in the sorting can be visualized by changing the weights for different languages i.e., by setting weight[ASC]=1 , weight[TM1]=3, weight[TM2]=2,

one sees that English string goes first.

- Truly Multilingual: This can support as many as 253 logically different languages.

This includes languages like English, Kannada, Thamizh etc., whose storage requirements are 1, 2,1.1 Bytes/Character respectively.

Considering for example, the Kannada name .v{^wಙ U_Kಢ which
stands for Krishna Reddy in English, and kiRxf]a erdfF in Tamil.

.v{^wಙ U_Kಢ: in Kannada 14 Bytes

DLC KAN 10h A7h 10h 1Bh 10h 05h 10h B3h 10h 55h Null-2

kiRxf]a erdfF: in Tamil 13 Bytes

DLC TM1 18h A9h MLC TM2 D7h 58h 01h ABh 56h 4Ch Null-1

Krishna Reddy: in English 16 Bytes

DLC ASC 4Bh 72h 69h 73h 68h 6Eh 61h 20h 25h 56h 46h 46h 97h Null-1

Where Null-1 represents NULL character in 1 Byte and Null-2 represents the same using 2 Bytes.

It is observed that for the same information, namely, the name, its respective representation in three different languages differs in length.

Table III provides comparison of Performance/Features of the various encoding schemes, including PANDITHAM.

Table III: Comparison of Performance / Features of various schemes

Features ---------- Schemes		Lingual	Bytes per character	Network Congestion	Lexical Order Sorting	Flexibility in language Ordering	Speech Synthesis	Random Processing of letters
7 Bit ASCII		Mono (English)	1	Very Low	Simple	No	Difficult	Yes
8 Bit ASCII (Glyph based)		Bi	1-3	Very High	Complex & Parsing required	No	Complex & Parsing reqd. & discontinous	No
8 Bit ASCII (CV Scheme)		Bi	1-3	High	Simple	No	Parsing reqd. & discontinous	No
Unicode	CV Based	Multi	3.4 (for Tamil)	Extremely High	Simple	No	Parsing reqd	No
	Unique	Multi	2	Very High	Simple	No	Simple	Yes
PANDITHAM		Multi	Best : 1 Worst: 2 Ave.: 1.1 (for Tamil)	Low	Simple	Yes	Simple	Possible only for Mono-Lingual strings

Features ---------- Schemes		Character Rendering	Rendering of Arbirary Strings	Kerning Problem	Mis-Script Problem
7 Bit ASCII		Simple	Yes	No	No
8 Bit ASCII (Glyph based)		Simple, But Time consuming	Yes	Yes e.g.ண்	Yes e.g. eci
8 Bit ASCII (CV Scheme)		Parsing required and Time consuming	No e.g. குர்ஆன்	No	No
Unic ode	CV based	Parsing required and Time consuming	No e.g. குர்ஆன்	No	No
	Unique	Simple	Yes	No	No
PANDITHAM		Simple	Yes	No	No

LIMITATIONS:

- When the rate of switching between languages is very high, then the overhead will be more. In the worst case, when alternate letters of string of n characters, switch between different languages; then the length will be 3n bytes. This case is highly unlikely in practice. For example, the name "Hariharan", if written as, "bvriharn", which has a mix of Kannada, Thamizh, English letters.

Then the length of this string will be 15 bytes; but such a multilingual string can only be hypothetical.

- Random processing of letters: Randomly looking at a byte in a PANDITHAM string will not tell us as to what that character is until we scan back to find the language, which could be arbitrarily be at a larger distance.
- Conversion Issues: Any PANDITHAM string can easily be converted to Unicode string, as PANDITHAM string tells the language(s) of the string. But to do the reverse, one will have to check the language of the code in Unicode.
- Standardization: In order to use PANDITHAM, we need only to standardize the codes for the languages and the respective PANDITHAM tables based on lexical ordering of phonetic characters.

Out of our experience on lexical sorting of Tamil names, we conclude that the codes,
00 (NULL), 02 (DLC) and 03 (MLC) should not be assigned as language codes.

CONCLUSION:

This paper has proposed a new protocol called PANDITHAM, for Multilingual computing, representing multilingual strings made up of phonetic characters. In particular, Thamizh strings are thought of as being made up off two logical Thamizh languages, namely, Pure Thamizh and Grantha Thamizh. This protocol switches to Double Byte Coding Scheme

(DBCS) for those languages having more than 256 characters, like Japanese, Kannada etc. If a Thamizh text is made up of pure Thamizh, then the storage requirement is only 1 byte per character; in case, there is infrequently used Grantha Thamizh in it, the storage requirement is on the average is 1.1 bytes per character.

This paper has looked at the merits and demerits of the various other techniques like Glyph based coding, Character based encoding scheme, like UNICODE. It will be no exaggeration if some one says that UNICODE for Indian Languages, in its current form (i.e. CV Based, 16-bit version of ISCII of CDAC), will be the worst for Indians.

As a case study, Tamil names, represented in PANDITHAM protocol, have successfully been sorted in lexical order. A multilingual text to speech synthesis engine has been developed. The protocol has also been used for developing a Thamizh Database Management System and a Multilingual word processor, in its true sense, by associating languages with fonts.

References:

1. Tamilnet 99 Conference Papers
2. Anbarasan N, 'A Perspective on Evolving standard for Tamil', Appletsoft Bangalore.
3. The Unicode Standard (Version 2) from the Internet
4. Dr. P. Navaneethan, R. Madheswaran, R.Balasubramaniam, N. Rajasekaran, PANDITHAM ' A Protocol for Applications Development in Thamizh and Multilingual Computing', ADCOM-99 Conference Paper.

Acknowledgement:

The authors of this paper acknowledge the help, support and encouragement provided by their Managing Trustee, Mr. G.R. Karthikeyan, Mr. C.R. Swaminathan, Chief Executive, PSG Institutions, Dr. P. Radhakrishnan, Principal, Dr. R. Nadarajan, Head, Dept. of Computer Applications, and Faculty Members of PSG Tech.

Tamil Support Inside the LINUX Kernel

R Vinodh Kumar
Department of Computer Science and Automation
Indian Institute of Science, Bangalore - 560 012, India
<Email: rvinod@csa.iisc.ernet.in>

Abstract

We consider the problem of providing the Operating System (OS) support for Tamil input/output. This paper analyses the issues involved and discusses the details of the implementation for achieving the above-mentioned OS support. We have modified the LINUX kernel to perform Tamil transliteration at the terminal line driver (tty driver). This thereby provides the user with the flexibility of using both English and Tamil interchangeably within the terminal-based applications of LINUX. Performing transliteration at the terminal line discipline module implies device independence. This work is first of its kind and we have not known of any previous implementations of terminal level transliteration engines for any language. We hope that our work will lend a new dimension for providing native language support, particularly in UNIX like operating systems.

1. Introduction

The native language software development in computer systems has been the object of research and work for the past many years. Most of the efforts are towards developing high level software for native language applications. Some of these software applications even require specialized hardware support such as the GIST (Graphics-based Information Systems Technology) add-on cards [6]. But there has not been much progress as far as native language support at the OS level is concerned. This is because most of the currently popular operating systems lack the necessary features to directly support native language I/O. Our work aims at providing the much needed OS support for Tamil I/O. For the very first time the terminal I/O framework of UNIX like operating systems has been exploited to support native language I/O. In this paper, we discuss the various issues involved and provide the details of implementation for the same.

2. Issues involved

Currently the "native" language of LINUX (and many other OS) appears to be English. To make the OS speak in the native language of the user, the OS must provide support for getting input of the native language characters and also displaying the same. So we identify three main issues involved in adopting a natural language for computers. They are:

1. Encoding the script or internal representation.
2. Outputting the script on display.
3. Input.

Above issues are very crucial and can have an everlasting impact on the development of the technology.

Various solutions have been proposed to solve the problem of encoding native language symbols along with the existing ASCII character set. Some solutions try to tamper with the first 128 ASCII characters to encode the native language symbols. But these could wreck havoc with the existing systems built for a predominantly ASCII based environment. As a typical example, file systems use some of the first 128 ASCII characters for special purposes (e.g. slash or backslash characters are typically file path-name delimiters) and hence may run into problems.

Presently native language input is done either using a native language keyboard or by performing transliteration of English alphabet into native language in the case of an English keyboard which is widely in use. To display native language symbols, proper fonts must be loaded.

3. Our Work

For the encoding scheme, we prefer the Extended ASCII coding to 16-bit solutions such as UNICODE [5] because of the advantages enlisted below.

- Size of encoding is 8 bits.
- Only the languages relevant to the user are displayed i.e. two languages English and the native language of the user can be supported. This constitutes our bilingual policy within the OS.
- Existing 8-bit clean applications will support the native language.

We employ the TSCII encoding scheme that makes use of the Extended ASCII codes to encode the Tamil alphabet for our purpose [4]. But it must be emphasized here that our work is encoding-scheme independent, as long as the scheme does not meddle with the ASCII codes, though we have used TSCII for illustration.

For the input methods, instead of performing the transliteration of the English characters into the Tamil alphabet at the application level, we have moved it to the kernel level so that all applications can utilize the transliteration to provide native language support. In UNIX environment, terminals (also referred to as tty தூ terminal teletype) act as generic input/output interface for an application [1]. An application reads characters from the terminal as input and sends its output to the terminal. Terminals are abstractions of the I/O devices that hide the details of these devices and thereby ensure device independence [2]. We have placed a transliteration engine for Tamil at the tty I/O level so that applications can read the ASCII as well as TSCII characters. The alternative to this approach would be to carry out the transliteration at the keyboard driver. We have not pursued this option since the user would then be restricted to working only at the system console to avail of the transliteration facility. But with our scheme this facility is available to users across the network who execute remote logins to the system since the transliteration is accomplished at device independent terminal level.

For the display of Tamil, we rely upon the features of X terminals in the X environment that allow us to load the appropriate fonts [8]. The finer points of implementation are discussed at length in the ensuing section.

4. Implementation Details

4.1 Platform of Implementation

We have chosen LINUX as the platform of our implementation for the following reasons.

LINUX is a freely available, open source UNIX-type OS [7]. This fast growing platform allows programmers to modify the OS kernel to suit their own needs. This not only liberates us from the clutches of 'proprietary' operating systems that would restrict our efforts in native language software to the application level, but is also a boon to us on the economic front.

4.2 Terminal I/O

In UNIX environment, physical devices are represented by files that allows portable programs that can access both the various devices and the files with the same system calls, for example, read and write [3]. Further, terminals allow different character devices to provide I/O to the high level applications without any modifications to them. Within this terminal device framework,

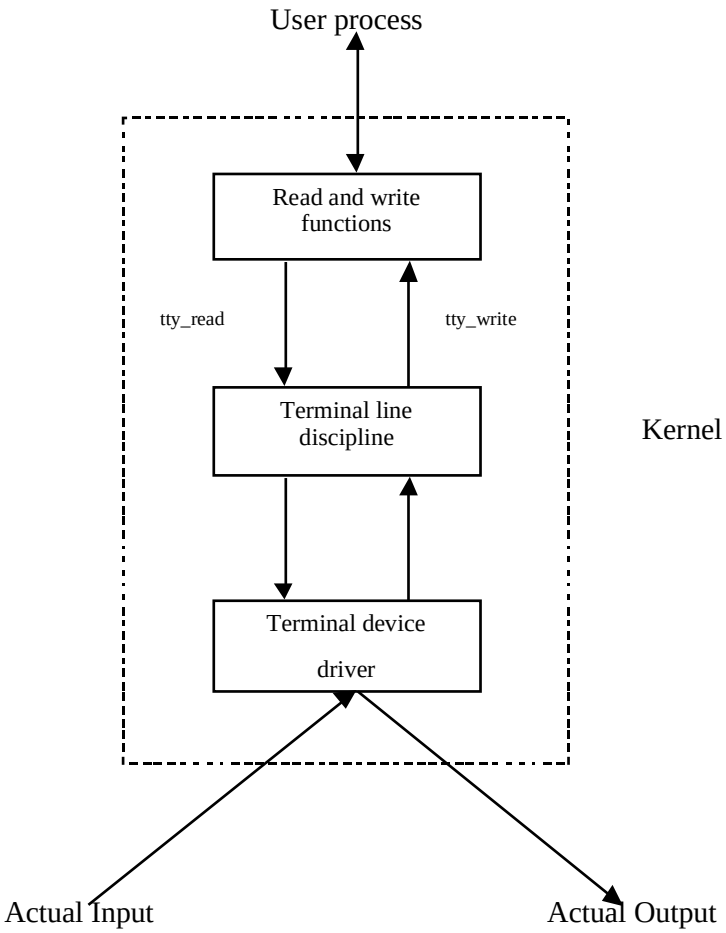


Fig 1. *The Standard Terminal I/O Model within the Kernel*

there exists a module called terminal line discipline that sits between the kernel's read/write functions and the actual device driver and implements all the necessary I/O processing (e.g. handling canonical input, echo on/off etc). We show this in Fig. 1. [1]

In LINUX-2.0 version, we have modified the tty I/O routines (tty_read and tty_write) of the terminal line discipline so that these routines transliterate characters if the terminal is in transliteration mode. We have introduced a special control character called "Tamil Lock" (Ctrl-T) similar to the POSIX.1 special characters (e.g. SUSP, WERASE, DISCARD etc) [1] which will toggle the transliteration mode ON/OFF. With the induction of the transliteration engine into the terminal line discipline, the altered terminal I/O scenario within the kernel appears as shown in Fig. 2. The display of Tamil characters is done by setting the TSCII fonts (e.g. Mylai-TSC, TSC-Akaram, Sri-TSC etc) for the X terminal in the X environment [4].

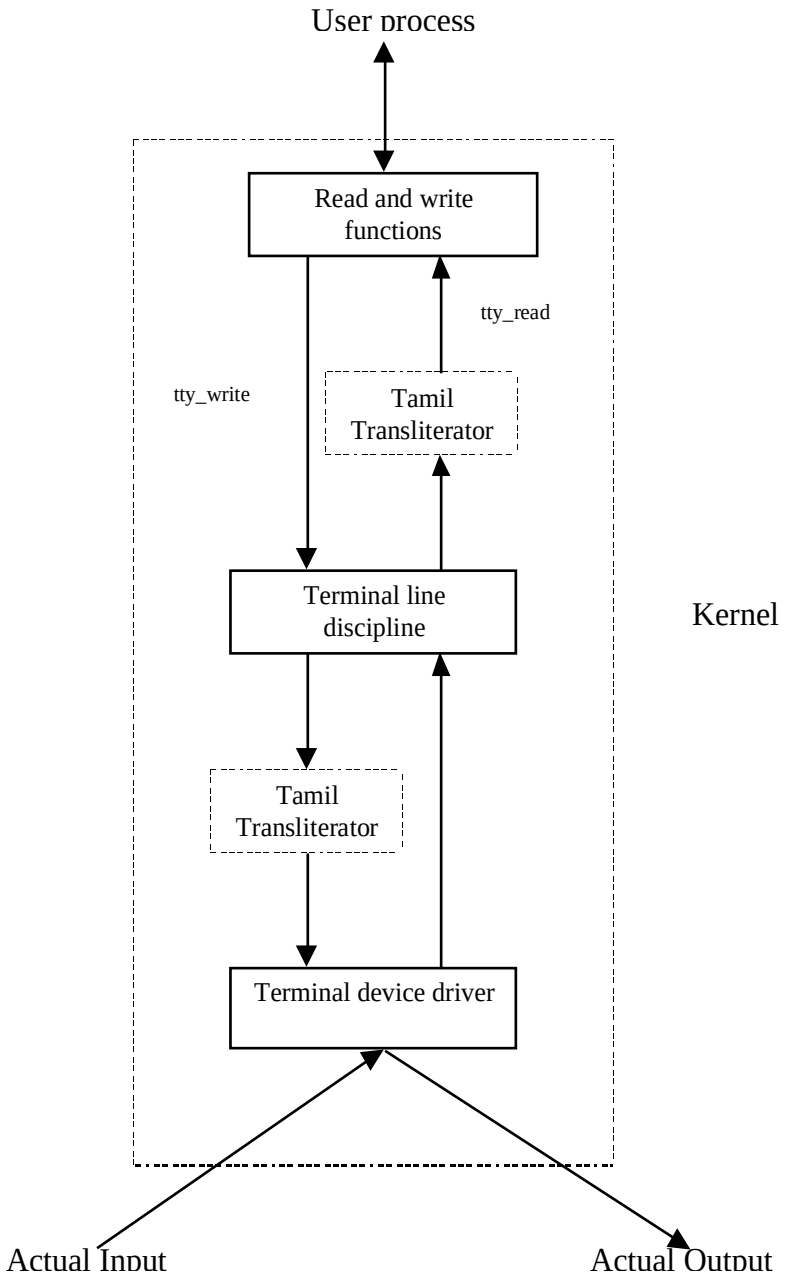


Fig 2. *The Altered Terminal I/O Scenario within the Kernel*

4.3 Tamil Transliterator

The Tamil transliteration engine is a Mealy-type deterministic finite-state automaton (DFA) [8]. We have implemented the state transition table for this DFA by maintaining different tables for different state modifier inputs. Each entry in these tables consists of the present state, the next state and the TSCII string to be output on this transition and the number of backspaces to be sent to erase the previous Tamil alphabet. State modifier inputs are characters, usually with vowel sounds, (typically the Tamil Uyir Ezhuthu) that could possibly modify the previous Tamil alphabet typed. This at times might necessitate sending backspaces to delete the previous alphabet sent to the application. Apart from these tables we also need a direct transliteration table for non-modifier inputs. This is illustrated clearly with the following example.

Example:
In transliteration mode, when the user types 'k', 'k' is transliterated as Mey Ezhuthu '‟' (pronounced ik) directly since it is a non-modifier. If the next character typed is 'a', then this Tamil alphabet must become Uyir Mey Ezhuthu 'k' (pronounced ka) due to 'a' being a state modifier. So the output TSCII string consists of a backspace followed by the TSCII code of 'k'. The backspace is sent to delete the letter '‟'.

Thus the Tamil transliteration engine maintains state information about the previous Tamil character and also contains a buffer that acts as a backlog for characters (implemented as FIFO queue). The necessity for a backlog buffer arises because the transliterator may output TSCII strings of size greater than one for single character inputs. The characters read from the input device are fed into the Tamil transliteration engine and its output is sent to the application/display. The schematic diagram for the transliteration engine is shown in Fig. 3.

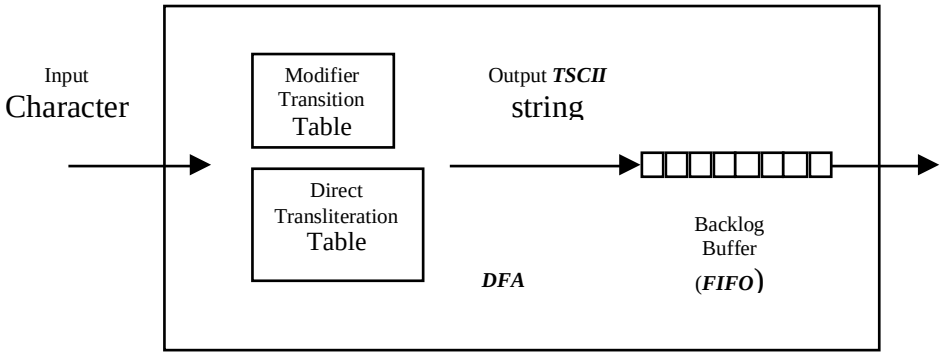


Fig 3. *The Transliteration Engine*

5. Analysis of our work

The solution is not restricted to the Tamil language and can be extended to support other languages by replacing the transition tables of the DFA. In this modified kernel, existing applications that are 8-bit clean can support Tamil. For example, existing editors can directly be used for editing Tamil text also. Tamil application programmers are relieved of the burden of re-inventing I/O methods for Tamil since now the OS takes care of them. English and Tamil can be

used interchangeably anywhere within the operating system after loading the proper fonts. So this solution best suits the scenario where the users are bilingual in nature.

6. Future Work/Extensions/Recommendations

- The engine can be developed as a separate LINUX module that allows loading of different modules dynamically for different languages.
- The engine can also be ported to the X-server so that all X applications can use the transliteration facility.

(Note: X applications read directly from the keyboard in raw mode and do not read from the terminal - so it is necessary to modify X-server.)

- The transliteration lock must be standardized as a POSIX special control character.
- Application writers must make their software 8-bit clean to ensure that their programs are native language supportive.

The above mentioned extensions will be the next phase of our work in making our own Tamil language a part and parcel of LINUX just as Russian, Japanese or Chinese is.

7. Conclusion

Thus, for the very first time, the terminal I/O framework of UNIX like operating systems has been exploited to support native language I/O. The transliteration engine in the terminal line drivers has been implemented successfully. This allows the users, including remote users, to interact with the system in both English and Tamil interchangeably. Most of the existing applications can now support Tamil I/O without any modifications. Tamil application programmers are relieved of the burden of devising I/O methods for Tamil since now the OS takes care of them. Because of the device independent nature of the terminals, no changes to the I/O device drivers are required.

Acknowledgement

I express my sincere thanks to Professor Dr. K.Gopinath for his able guidance, invaluable support and encouragement for this work.

References

1. Advanced Programming in the UNIX environment - W.Richard Stevens.
2. Modern Operating Systems - Andrew S. Tanenbaum
3. LINUX Kernel Internals - M Beck, H Bohme, M Dziadzka, U Kunitz, R Magnus, D Verworner
4. http://www.tamil.net/tscii/
4. http://www.unicode.org
5. GIST for Vernacular language computing ‡ Alok Jain
6. http://www.linux.org
7. http://www.X.org
8. Introduction to Automata Theory, Languages and Computation - Hopcroft, Ullman (1979)

A Bilingual Search Engine for Tamil & English Sites

Dr. M. Ponnavaikko, Prof. & H.O.D., CSE & A. &
K. Karthik, Final Year M.C.A. Student
Crescent Engineering College, Vandalur, Chennai-600 048.

Introduction

Word Wide Web - A gift to the Information Technology that connects people of different occupation like Business, Engineering, Medicine, Law etc. Day by day the number of documents available on the Internet are increasing in an exponential manner. Each and every document is uniquely identified by a name, which we call as "UNIFORM RESOURCE LOCATOR" (URL).

The huge nature of Internet suggests that we need a way to find out the relevant documents required by a user by only giving keywords without knowing the exact URL's. This task of identifying the document collection for the specified keywords is done by a special program, which we call a "Search Engine".

Search Engines help in pooling information from the various Web sites. Current Search Engines like YAHOO!, ALTAVISTA and others are capable of finding documents that are in English language only. At present Tamil Web sites are becoming popular and there are more than 800 sites available in Tamil. To visit these sites, one has to remember the exact URL of these sites. At present there is no Search Engine to cover these sites.

This paper presents a Search Engine that can search documents in TAMIL & ENGLISH from the web sites. A Search Engine in the name of Thiruvalluvar presented in this paper is the outcome of an MCA project work carried out at the Department of Computer Science, Engineering and Application by Mr.Karthik under the guidance of Dr.M.Ponnavaikko.

How does a Search Engine Work ?

A Search Engine continuously spends out so-called 'Spiders', a special kind of program, which starts in a home page of a Server and pursues all links stepwise. Word indices are created from individual pages and the database is updated (fig.1).

1. When a site has to be covered by a SEARCH ENGINE then it should be registered with that Search Engine.
2. The user has to give the data to be searched, in the space provided by the SEARCH ENGINE's current page and the query is subsequently forwarded to the database.
3. The result will be displayed with the information that correspond to that search with clickable URLs which may take you to the pages that are related to your search.

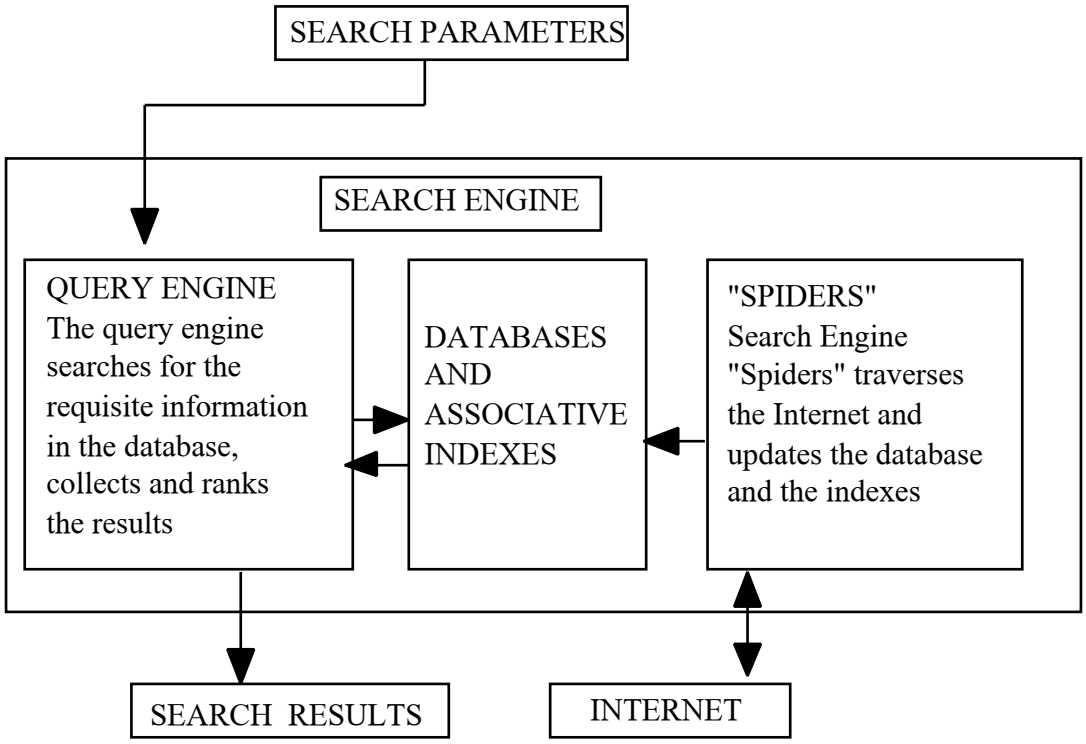


Fig. 1: Concept of a Search Engine

4. The results are Rank ordered. Here, Rank may be Confidence Ranking (based on no. of occurrences of keywords in the document) or Relevancy Ranking (based on no. of links to that site or keywords related to the content of that site).

What Is a Bilingual Search Engine?

A Bilingual Search Engine is capable of finding documents that are in two languages. The two languages considered here are TAMIL & ENGLISH. This Search Engine will take keywords both in TAMIL & ENGLISH and retrieve documents whose content will have the given keywords in some place.

Need for a Bilingual Search Engine

At present TAMIL Web sites are available in the Internet, which are huge in number. If one has to visit to these sites, then he/she should know the exact addresses. This is impractical, as there are enormous number of sites available.

To overcome this difficulty we can create a Search Engine, which can cover these sites. On accepting the keywords from the user, this Search Engine will return back the sites that will contain the given keywords.

The font family TABxxx, standardized by the GOVERNMENT OF TAMIL NADU, based on the recommendations of TamilNet99, includes two languages namely Tamil and

English. The Search Engine developed is for the sites that use TABxxx fonts. The design issues of such a Search Engine includes the following :

Development of 'Thiruvalluvar', a Bilingual Search Engine

The Bilingual Search Engine 'Thiruvalluvar' has the following features :

1. It can search a Tamil web site coded with TABxxx fonts as well as English web sites.

2. It can search only the sites which are registered. Registration of a site is done through the registration module of the Search Engine. While registering a site 25 keywords for each HTML file is obtained and stored in the Server's database.

3. The concept of Set Theoretic Model is used for the development of the Search Engine.

4. The Search Engine works in two modes, namely Static Search and Dynamic Search. Static Search finds documents that are having keywords registered by the site owner in the search string which are maintaining in the Server's database. Dynamic Search finds documents that are having the keywords inside the HTML file content as a real time process.

5. The search results are presented as an extract of all the URLs which contains the search string.

6. Complex search strings having AND, OR, NOT functions are used for searching the URLs containing the search strings under both static and dynamic modes.

7. The Search Engine has classified categories and a search directory. This gives an option to search on a desired category, which reduces the search time.

8. The Search Engine is provided with a keyboard layout (TamilNet99 keyboard) with the help of which Tamil as well as English characters can be input through mouse.

Implementation and Testing

The Search Engine 'Thiruvalluvar' is implemented using JDK1.2, JSDK, JDBC, Java Script & HTML. Oracle is used to develop the database for the Server of the Search Engine. This has been tested successfully in an Intranet environment in the college. The Home Page (Fig.2), the Home Page with keyboard layout (Fig.3), the Page for Site Registration (Fig.4), Advanced Search Page (Fig.5) and a Page showing the search results when the word இந்தியா was searched (Fig.6) are presented.

The main constraint of this Search Engine is that the pages can be browsed only through Netscape Navigator 4.0 and above.

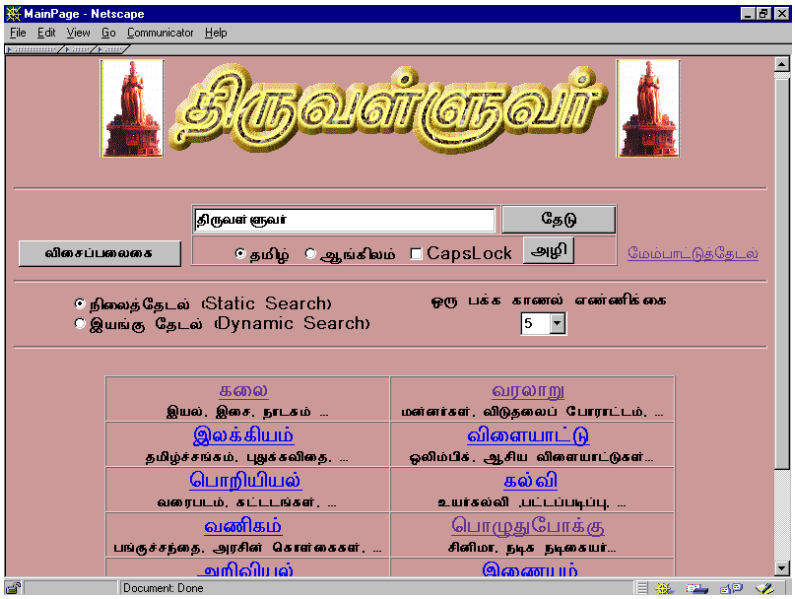


Fig. 2 Home Page

Fig.3 Home page with Bilingual Keyboard

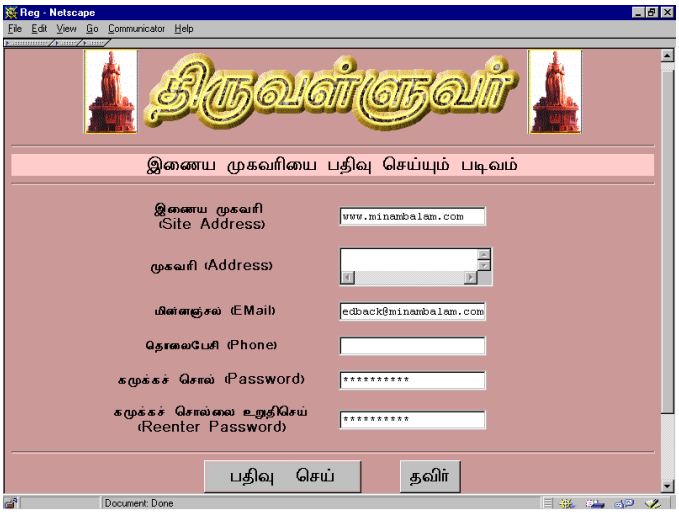


Fig.4 Page for Site Registration

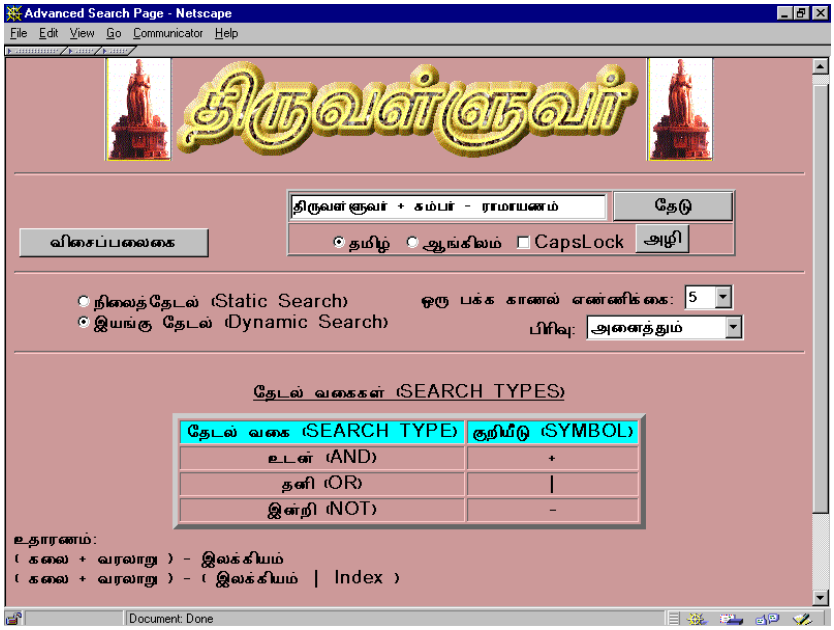


Fig. 5 Advanced Search Page

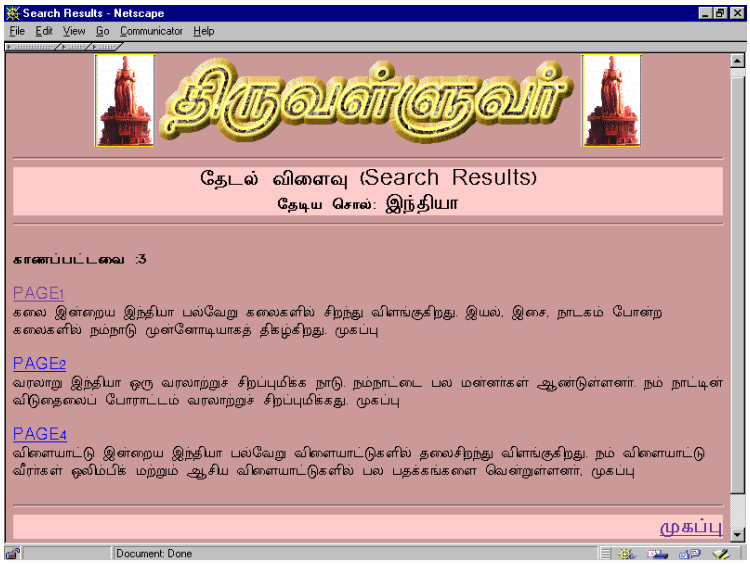


Fig.6 Page for Search Results

Conclusion

The main problem of net users, searching for Tamil sites is the non-availability of a tool for searching Tamil web sites. In this effort an effective search engine in the name of Thiruvalluvar has been developed as a Bilingual Search Engine for searching Tamil as well as English web sites. This search engine incorporates both static and dynamic searches. The only limitation of this tool is that it can be used only through Netscape Navigator 4.0 and above.

"Tamil Java" : A Tamil Preprocessor for Java

Dr.M.Ponnavaikko, Prof. & HOD,CSE&A,
S.S.Sriram & S.Syed Shajahan, Final year students of B.E.(CSE)
Crescent Engineering College,Vandalur, Chennai-600 048

Introduction

The invention of the computer has been the greatest achievement of this century; English has a great impact on all the applications. But the people who don்t know English were not able to use the computer effectively, There is a need for developing software in Tamil so that Tamil speaking people can use computers effectively. It is very difficult to design and develop basic Tamil Software for every application, particularly with reference to the exploding rate of growth of computer software. But it is possible to develop preprocessors for every new English based software so that people can use it easily. Based on this fact "தமிழ் ஜாவா" has been created through a student project.

In Java there are two kinds of programs, namely application programs and applet programs. This preprocessor for Java is developed for writing Java application programs. The preprocessor developed has two parts. One is to develop Tamil Java syntax for developing Java application programs and the other is to generate error messages in Tamil while compiling the Java programs. This paper presents these aspects with examples.

Tamil Java Programming

The Tamil Java Programs are developed using the Tamil syntax designed for this purpose. The Java syntax is given in the Appendix.

The "தமிழ் ஜாவா" programs written in Tamil are converted into an intermediate English Java program by the Java preprocessor designed and developed in this work when the Tamil Java program is compiled. The execution sequence of Tamil Java program is given in Fig.1.

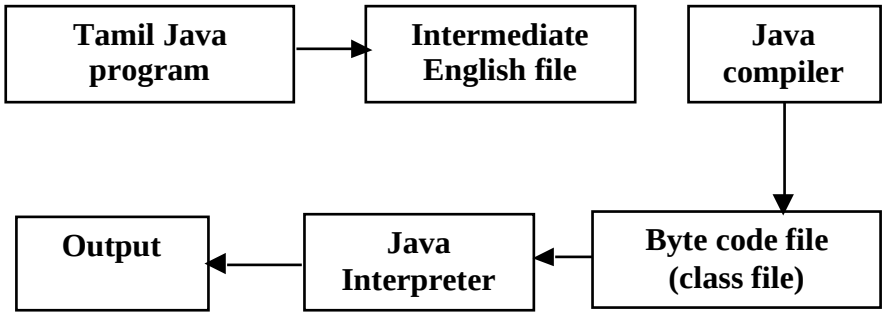


Fig 1 Execution of a Tamil program

தமிழ் ஜாவா பயன்பாட்டுப்பகுதி தொடரியல்கள்

The different statement that are used in the java programs are

1. Declaration statements

முழு எண்	int i;

2. Assignment statements

க = ம * வ - த;	a=b*c-d;

3. Condidtional statements

a. எனில் கூற்று (IF statement)

எனில் (நிபந்தனை)
{
----------;
----------;
}
	if(condition)
	{
	----------;
	----------;
	}

4. Iterative statements

a. உண்மையெனில் கூற்று (while statement)

உண்மையெனில் (நிபந்தனை)
{
_____;
_____;
}
	while (condition)
	{
	_____;
	_____;
	}

5.Input / Output Statements in Java

அமைப்பு உள்படி (பணம்);
System.in.read(amount);

The syntaxes for the Tamil java have been framed[refer appendix]

Errors Messages:

It is human nature to commit errors and hence, the program written may contain some errors. The preprocessor developed incorporates error messages in Tamil. The errors committed are found by using hash table entries, during the conversion of Tamil file to intermediate English file. The Hash tables used are given in the Table 1.

Table 1: Hash tables that were used in the project

Hash Table No.	Fields in the Table
1	Tamil Java Keyword: Index
2	Class Name Index: Tamil Class Name
3	Method Name Index: Tamil Method Name
4	Object / Variable Name Index: Object / Variable Name
5	Method Name Index : Class Name Index
6	Object / Variable Name Index : Class / Method Name Index
7	Tamil letter : English Letter
8	Tamil word : Assigned English word
9	Derived Class Name Index : Base Class Name Index

This software has implemented 9 error messages as given in Table 2.

Table 2: Errors that were handled in the project

ERROR NUMBER	ERROR MESSAGE
1	'{' அடைப்புக்குறி விடப்பட்டுள்ளது
2	'}' அடைப்புக்குறி விடப்பட்டுள்ளது
3	'(' அடைப்புக்குறி விடப்பட்டுள்ளது
4	')' அடைப்புக்குறி விடப்பட்டுள்ளது
5	" விடப்பட்டுள்ளது
6	வறையறுக்கப்படாத மாறி
7	வறையறுக்கப்படாத உறுப்பு
8	இந்த வகுப்பின் செய்முறை காணவில்லை
9	இந்த வகுப்பின் மாறி காணவில்லை

The situations where the errors will be displayed are given below.

'{' '}' அடைப்புக்குறி விடப்பட்டுள்ளது

This error message is displayed when the matching flower brackets are missing in the program..

'(' ')' அடைப்புக்குறி விடப்பட்டுள்ளது

This error message is displayed when the matching brackets are missing in line.

" விடப்பட்டுள்ளது

This error message is displayed when the matching பு are not found in the line.

வறையறுக்கப்படாத மாறி

This error message is displayed when any variable, used is not declared. This error is mainly for the local variable.

இந்த வகுப்பின் செய்முறை காணவில்லை

This error message is displayed when the function called does not belong to this class or to the class that the object belongs to.

இந்த வகுப்பின் மாறி காணவில்லை

This error message is displayed when the function called does not belong to the class that tha object belongs to.

வறையறுக்கப்படாத உறுப்பு

This error message is displayed when the object that is used is not declared.

A Java application program for Matrix multiplication

A Tamil Java program written, using the syntax developed for Matrix multiplication is given below :

```
வகுப்பு பெருக்கல்
{
        பொது மாறா வெறுமை முதல் ( சரம்    )
        {
                முழுஎண் ம1  [][]={
                                {2,2,2},
                                {1,1,1},
                                {1,1,1}
                              };
                முழுஎண் ம2[][]={
                                {2,1,1},
                                {2,1,1},
                                {2,1,1}
                              };
                முழுஎண் ம3[][]= பது முழுஎண்  [3][3];
                முழு எண்,இ.க.உ;
                இ = க * உ = 0
                 அமைப்பு.வெளி.அச்சிடு ("  இரண்டு அணிகளை பெருக்கும் நிரல் ");
        அமைப்பு.வெளி.அச்சிடு ("        ---------------------------------------");
        உண்மையெனில்  (_<3)
                {
                        க=0;
                        உண்மையெனில் (க<3)
```

```
                    {
                      ம3[_][க]=0;
                      உ=0;
            உண்மையெனில்  (உ<3)
                      {
                              ம3[_][க]+=ம1[_][இ]*ம2[உ][ல];
                              உ++;
                       }
                      க++;
                    }
                    இ++;
              }
              அமைப்பு.வெளி.அச்சிடு ("   முதலாம் அணி  ம1 :: ");
              ஆக (இ=0;இ<3;இ++)
              {
                ஆக (க=0;க<3;க++)
    {
                  அமைப்பு.வெளி.அச்சிடுக  (ம1[இ][க]+"  ");
    }
                அமைப்பு.வெளி.அச்சிடு ();
              }
              அமைப்பு.வெளி.அச்சிடு ();
        அமைப்பு.வெளி.அச்சிடு ("     இரண்டாம் அணி ம2 :: ");

              ஆக  (இ=0;இ<3;இ++)
              {
        ஆக (இ=0;இ<3;இ++)
              {
                      அமைப்பு.வெளி.அச்சிடு (ம2[இ][க]+"  ");
              }
                  அமைப்பு.வெளி.அச்சிடு ();
              }
              அமைப்பு.வெளி.அச்சிடு ();
        அமைப்பு.வெளி.அச்சிடு ("   இரண்டாம் அணிகளை பெருக்கியதற்கு பிறகு.....");
        அமைப்பு.வெளி.அச்சிடு ("  மூன்றாம் அணி ம3 :: ");
              ஆக (இ=0;இ<3;இ++)
              {
                ஆக (க=0;க<இ;க++)
     {
                    அமைப்பு.வெளி.அச்சிடு  (ம3[இ][க]+"  ");
                  }
      அமைப்பு.வெளி.அச்சிடு ();
              }
        }
}
```

This program was compiled and executed. The intermediate English Java program generated by the preprocessor is given below. It may be noted that this English Java program will not be seen by the user when he is compiling and executing the Tamil Java program.

```
import java.io.*;
class YeHparukaIkalaI
{
	public static void main(String cha[])
	{
		int ma1[][]={
				{2,2,2},
				{1,1,1},
				{1,1,1}
			};
		int ma2[][]={
				{2,1,1},
				{2,1,1},
				{2,1,1}
			};
		int ma3[][]=new int[3][3];
		int ye,ka,ooh;
		ye=ka=ooh=0;
		System.out.println("   இரண்டு அணிகளை பெருக்கும் நிரல் ");
	System.out.println("        ---------------------------------------");
	while(ye<3)
		{
			ka=0;
			while(ka<3)
			{
			  ma3[ye][ka]=0;
			  ooh=0;
		while(ooh<3)
				{
					ma3[ye][ka]+=ma1[ye][ooh]*ma2[ooh][ka];
					ooh++;
				}
			  ka++;
			}
			ye++;
		}
		System.out.println("  முதலாம் அணி ம1 :: ");
		for(ye=0;ye<3;ye++)
		{
		  for(ka=0;ka<3;ka++)
	  {
			System.out.print(ma1[ye][ka]+"  ");
```

```
        }
                System.out.println();
              }
              System.out.println();
        System.out.println("  இரண்டாம் அணி  ம2 :: ");

        for(ye=0;ye<3;ye++)
              {
                 for(ka=0;ka<3;ka++)
         {
                    System.out.print(ma2[ye][ka]+"  ");
        }
                 System.out.println();
              }
              System.out.println();
        System.out.println("    இரண்டு அணிகளை பெருக்கியதற்கு பிறகு.....");
        System.out.println("   மூன்றாம் அணி  ம3 :: ");
              for(ye=0;ye<3;ye++)
              {
                 for(ka=0;ka<3;ka++)
         {
                    System.out.print(ma3[ye][ka]+"  ");
                 }
        System.out.println();
              }
        }
}
```

The results computed for the Tamil Java program is given below :

```
ச:\> ஜாவாசி பெருக்கல்.  ஜாவா
    வெற்றிகரமாக தொகுக்கப்பட்டது

ச:\> ஜாவா பெருக்கம்

இரண்டு அணிகளை பெருக்கும் நிரல்
---------------------------------------
    முதலாம் அணி   ம1 ::

2 2 2
1 1 1
1 1 1

   இரண்டாம் அணி ம2 ::
2 1 1
2 1 1
```

```
2 1 1

    இரண்டு அணிகளை பெருக்கியதற்கு பிறகு.....
    மூன்றாம் அணி ம3 ::

12 6 6
 6  3 3
 6  3 3

ச:\>
```

Conclusion

The Tamil preprocessor for Java was found very effective and user friendly. This preprocessor does not exhaustively cover the entire of Java syntax for want of time and effort, since this was developed as a part of the B.E. curriculum along with the regular theory papers. However, it would not be very difficult to develop a preprocessor for complete Java. At this juncture it is appropriate to state that the students of 1999 batch of this department had developed a Tamil preprocessor for "C" language. In a similar way Tamil preprocessors for all the compilers and Tamil Window Manager for all Operating Systems would help greatly the Tamil speaking world for using computers effectively.

Idham-2000:
Advanced Tamil Interface for Microsoft Windows

Manoj Annadurai & Benjamin Martyn
Chennai Kavigal, Chennai, Tamilnadu

"It was not my opinion; I think there is no sense in forming an opinion when there is no evidence to form it on. If you build a person without any bones in him he may look fair enough to the eye, but he will be limber and cannot stand up; and I consider that evidence is the bones of an opinion" - Personal Recollections of Joan of Arc

"I was seldom able to see an opportunity until it had ceased to be one" - Mark Twain's Autobiography

i2000 - Its implication for Tamil:

"In a central Asian country, where western tongues are rarely spoken", an eight year old, tells his father that he has to learn English. The father asks why. "Because, father, the computer speaks English." That story, notes Asiaweek, "illustrates what many consider to be a insidious side effect of the Information superhighway..."

Use of other languages, is slowed in some cases because of the difficult of adapting them to the English-based keyboard. "There will be a price to pay", says Asiaweek. "Linguists predict that half of some 6,000 languages spoken today, will fall into disuse by the end of the next century, possible within the next 20 years".

Newsweek magazine suggests that languages are being "pushed into oblivion by other 'big' languages." Professor Stephen Wurm, editor of Atlas of the World's Languages in Danger of Disappearing, published by the United Nations Educational. Scientific, and Cultural Organization, adds ; "There's often a belief that you should forget the 'small' languages of the minorities, because they have no value".

"It is said that in these days of electronics and computers English is essential. Japan is a leading country in scientific development and the students there learn in their mother tongue. Chine, soon after the British handed over Hong Kong, did away with the English based education.

Even in India English is the lingua franca of only a microscopic section of the ruling gentry. English may be useful to maintain colonial and semi-colonial rule but it may never be the fulcrum of our cultural and emotional integration. In short, English is not the language of emotion. And as long as the language of emotion does not become the language of education knowledge cannot be born" - A.K.ROY - The Hindu

In a letter dated May 20, 1999 addressed to the Secretary - Information Technology, Govt. of Tamil Nadu, Microsoft Corporation states that" ... our plan is to ship only the INSCRIPT

keyboard in Windows 2000...". This in effect would nullify, all the path-breaking, historic efforts taken by the Tamil Nadu Government, in one, swift, complete stroke. Given the growth and usage of the Windows computing platform, it is an obvious prediction to state that Windows 2000, (to be released in late 1999) would be on all desktops by 2000. Thus, Microsoft would be introducing a keyboard layout and a font encoding scheme, which stands in stark contradiction to what the Govt. has painstakingly built and recommended.

What i2000 is:

i2000 is envisioned to be a Tamil Veneer for Microsoft Windows 32-bit. As a module i2000 would add the following functionality:

Amplify Windows support for Tamil programs
Bring about a radical change for "all-Tamil" Computing
Enhance the User interface, making computing many times easier.
Build up a 100% Tamil computer from the ground up.
A comprehensive help system ensures computing is a lot more fun and productive.
iSteersman will assist you with many common computer problems.

Amplifying Windows support for Tamil programs:

Tamil as a language has many a difference in structure when compared to English. A Programmer dedicated to Tamil computing, would be aware of such limitations faced when using an Application Programming Interface, which does not consider these problems, such as what is provided by Microsoft Windows. i2000, would thus make a lot of difference from the Computing providers viewpoint.

Bring about a radical change for "all-Tamil" Computing:

In every page of Computing history, the Interface has been only designed for "big" languages. Hence rich languages, such as Tamil would fail to enter computing "big-time". i2000, brings about a total Tamil Environment. This ensures that users with Zero knowledge of the English language, can not just operate, but effectively use computing power. i2000 supports connectivity with all major network technologies, such as Windows NT and Novell Netware. This also stretches into internet computing, as a complete Tamil Browser is included as part of the package. Hence the Tamil user, will no longer be limited in growth of his knowledge, and will have no less than the very best.

Enhance the User Interface, making computing many times easier:

i2000 improvises time-tested enhancements to the Windows 32-bit computing platform. This would be a first in many ways. Work is already underway, researching into ways in which computing workload of the end user can be downloaded on to the computer.

Building a 100% Tamil Computer-from the ground up:

Right from the moment the OS starts, all processing will be done in the Native language-Tamil. This means constructing a system from the ground up, including updates and patches for basic to advanced services provided by Microsoft's Windowing Environment. i2000 will continue support for English applications and other existing software designed by ISV's.

A Comprehensive help system ensures computing is a lot more fun and productive.

Always available whenever a user requires it, it provides help on a wide variety of tasks-from Installing a Software to i2000 advanced Internet/Intranet options.

Active Help:
i2000 will have Web-based on-line help. Help and additional information on advanced i2000 configuration could be found on Chennai Kavigal or on Elcot's website. This Web location will also contain i2000 FAQ's and related RFC's.

i Steersman will assist you with many common computer problems.
For Instance, if you are having difficulty installing a new printer, the iSteersman will walk you through the process, step by step. iSteersman uses new technology to keep its help information in track with current topics and standards.

End Package:

1. A Software that would:
   Amplify Windows support for Tamil programs
   Bring about a radical change for "all-Tamil" Computing
   Enhance the User interface, making computing many times easier.
   Build up a 100% Tamil computer from the ground up.
   A comprehensive help system ensures computing is a lot more fun and productive.
   Assist with iSteersman to solve many common computer problems.
2. Two thousand copies on CD-ROMs
3. Complete Licensing and Copyrights of the finished product necessary for further reproduction.

Benefits:

Tamil language, The people of Tamil Nadu, and Tamil Nadu Government will be the first to posses such a software, not only in India but perhaps in the World. This revolutionary concept would trigger a avalanche of Tamil software not only in the word processing arena but perhaps even in accounting, educational and research applications. i2000 would push the usage of computing from current 2% to a very large number at an exponential rate.

134 empty

Development of PANDITHAM-Based Applications for Thamizh

P. Navaneethan, R. Madheswaran, R. Balasubramaniam,
and R.V. Bharathidasan
Department of Computer Applications
PSG College of Technology, Coimbatore , India

INTRODUCTION

PANDITHAM (A Protocol for ApplicatioNs Development In THAmizh and Multilingual Computing) has a broad spectrum of applications like Word Processor, Text to Speech Synthesis, and Thamizh Database Management System.

This paper looks at some of the key issues involved in the implementation of a Multilingual word processor and Thamizh Database Management System. The paper also highlights the techniques used for Multilingual Text to Speech synthesis, based on PANDITHAM protocol.

Coding Scheme for the Name: ர ங் க ரா ஜ ன்
DLC TM1 ர ங் க ரா MLC TM2 ஜ ன் SP DLC ASC R . NULL
Where, SP - Blank Space

1) MULTILINGUAL WORD PROCESSOR

The word processors that are currently available support multilingual computing only if the word processor supports the font corresponding for that language. But a true Multilingual word processor must support the concept of a language and a number of fonts for that particular language. This character oriented word processor that is designed, based on PANDITHAM, goes with the language and not on the fonts themselves. All the common features of the word processor are supported, and in addition, this word processor supports the concept of language also. The features of this word processor make it easily extendible to include many other languages also. A particular language has a default font. The language would also support several other fonts for better user interface and to make documents have an aesthetic look. Since the Word processor developed is character oriented, the problems associated with Glyph based system like Kerning and Mis-scripting do not find a place.

DESIGN OF LANGAUAGE AND FONT DATABASE

As mentioned earlier, any multilingual data processing should be based on the language and not on fonts, which are vulnerable to change. To facilitate this, a database is to be maintained, in the WinSysPath/language directory. The two main tables are Language database and the Font database. The Language database consists of details like unique Language code, Language name, Classification, and Weight. The languages are classified into 3 categories based

on the amount of storage requirements. The first category comprises of languages like English, which occupy single byte/character. The best example of second category would be Japanese language, which require 2 bytes/character. The language Thamizh comes under the third category, whose storage requirements lies in between 1 and 2 bytes/character (on the average 1.1 bytes/character). This is basically due to the presence of infrequently used Thamizh characters (Grantha characters).

The font database stores a unique font code, font name and the language to which it belongs. Apart from these, a default font database is also maintained.

STRUCTURE OF DATABASE

The structure of the above mentioned database is described below

```
Private Type Lang          /* Language Record Structure */
   LangCode As Byte        /* Unique Language Code */
   LangName As String      /* Language Name */
   Weight As Byte          /* Language Weight for sorting */
Classification As Byte     /*1 English Like, 2 Japanese Like, 3 Thamizh Like */
End Type
Private Type Fonts         /* Font Record Structure */
   FontCode As Byte        /* Unique Font Code */
   FontName As String      /* Name of the Font */
   LangCode As Byte        /* Language it belongs to */
End Type
Private Type Defa          /* Default Font Record Structure */
   LangCode As Byte
   FontCode As Byte
End Type
```

Multilingual Word Processor (MLWP) is a multiple document interface application. Any number of documents can be open in the main window. A file can be keyed in the document space. Then, the same can be saved in two formats. The first one being the standard Multilingual data format, PANDITHAM. The second one being the formatted standard output, the Rich Text Format. The same can be opened and loaded into MLWP irrespective of the stored format. The opened file can be sent to any installed printer either as the selected portion or as a whole.

Standard editing features like cut, copy, paste, undo is added. The same is visualized in the tool bar. The text can be right aligned, left aligned or center aligned. A status bar is also added to say current status with date and time. Both the toolbar and the status bar can be toggled for visibility as desired by the user. The windows opened in the main window can be easily arranged and manipulated. The provision to incorporate a complete help engine is provided.

As most of the processing in the MLWP are based on the language, a separate database for language and font is maintained. This is taken care by the ЧLanguageFontMaintenanceў module of the MLWP. In the document window, all the possible languages are listed. According to the language selected, fonts belonging to that particular language alone are listed. Also, if a selection in a document consists of multiple language then the font change is not allowed, though MLWP allows for attribute (like bold, italic) change.

For Thamizh, the phonetic keyboard layout standardised in TamilNet ரு99 is used. The features like spell check and grammar can easily be incorporated once sufficient data has been entered.

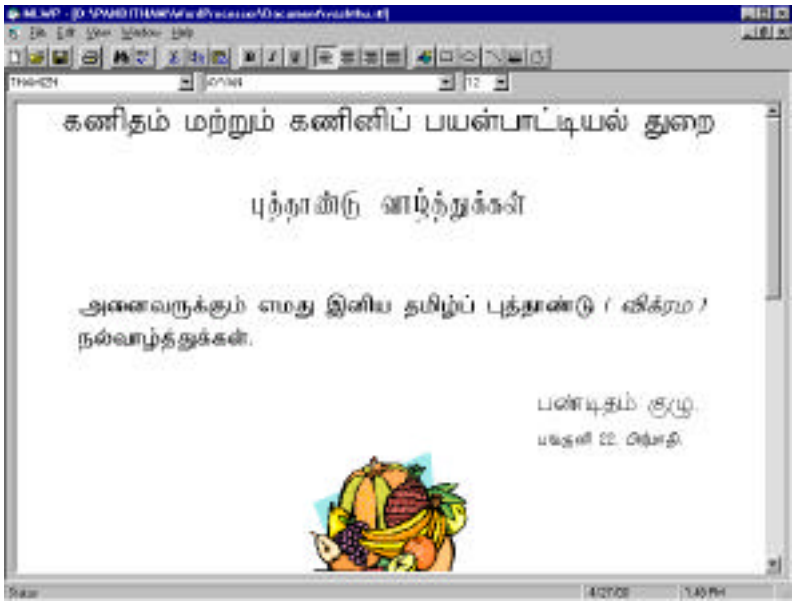


Fig 1 shows our Multilingual Word Processor in action

2) THAMIZH DATABASE MANAGEMENT SYSTEM

One of the important functions of a DBMS is sorting. The process of Sorting the multilingual data, which is based on Glyph, takes much time, since, it requires complex parsing. A Character oriented PANDITHAM protocol helps sort multilingual string very fast. Moreover, a Thamizhan may like to have Thamizh names first, while sorting names. Likewise, people who speak Kannada, Telugu etc., may like to have their language before any language. Hence it is necessary to implement Language based ordering too. But, this is not feasible with the existing DBMSs, since, they do not have the concept of language. Also, they donரு்t have any standard storing formats for multilingual data.

The purpose of this Thamizh Database Management System (TDBMS) is to enable the user, who knows Thamizh and who has an inclination towards database application, to develop and maintain a Thamizh database of his/her need. This Thamizh DBMS would facilitate the definition of database, the field names, the data types, the constraints and the interfaces, specified in Thamizh. To sort the multilingual data, stored in the database, a multilingual string compare function has been developed. Most of the basic DBMS features have been incorporated. The languages, fonts and the default font for a language are used by Thamizh DBMS, are available in the \WINSysPath\Language directory. Any other PANDITHAM based multilingual tool could use this.

Consider a table with n number of fields. It will create n + 3 files, to store the table completely. The meta-data, i.e., the description of the table is stored in one file. The meta-data file will have the following data.

Number of Records
Number of Fields
FieldName FieldType FieldWidth Constraint

Fig.2 to Fig.5 provides snapshots of some of the screens that are there in the PANDITHAM based Thamizh Database Management System

Fig. 2 shows the Main Menu of the application.

Fig 3 shows the input screen for the database name.

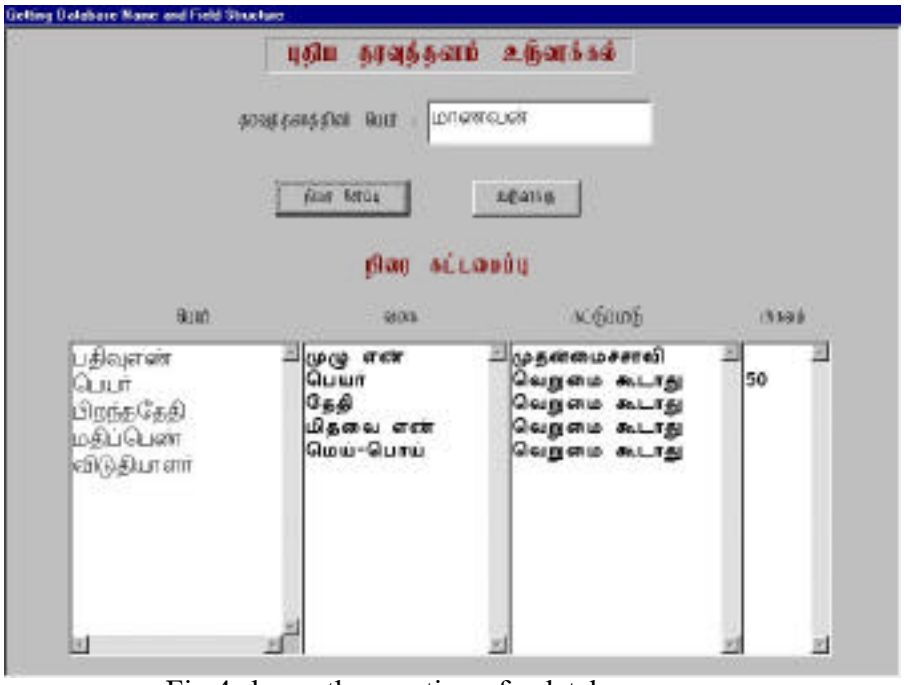


Fig 4 shows the creation of a database

The main menu of the Thamizh DBMS, contains the following items.

கோப்பு (File)
பயன்பாடு (Application)
உதவி (Help)

The கோப்பு menu has following sub-menus.

புதியது (New), திறக்கவும் (Open), நன்றி (Thanks)

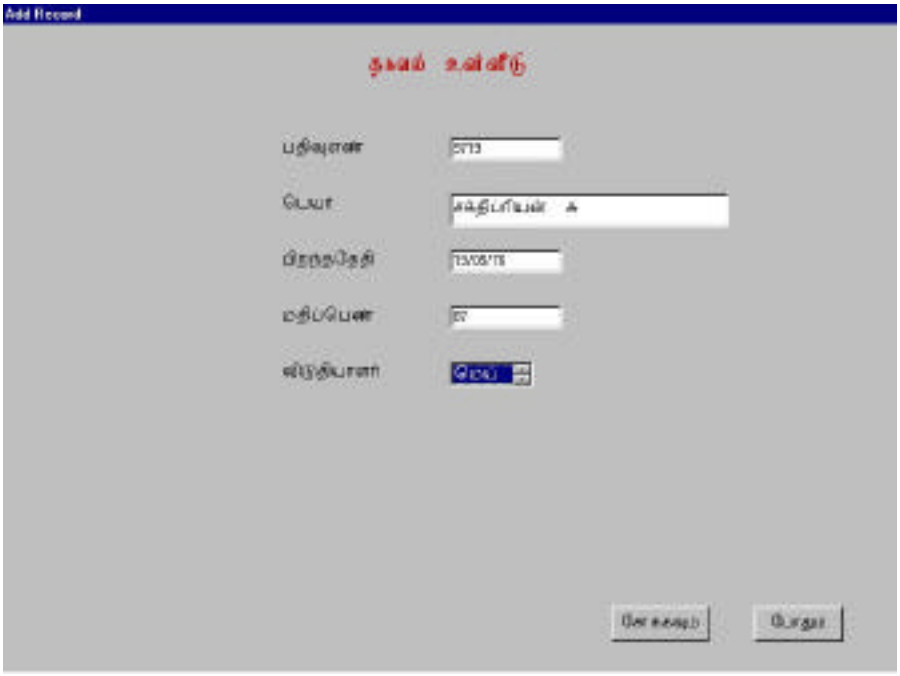


Fig. 5 shows a typical data entry screen

An important feature of this TDBMS is that for every data type, there is an exclusive data entry form, which can be invoked at entry time.

3) THAMIZH TEXT TO SPEECH SYNTHESIS

If we analyze the transition that has underwent in human communicating with computer; it can be realized that, this transition has been smooth and effective. In the beginning, every input-output operation was using toggle switches and bulbs. Then came punched cards, keyboards, pointing devices. But, march to make computers communicate in ways that come naturally to humans continues. In the quest for a perfectly transparent user interface, speech is perhaps the final frontier, short of direct brain-link, and hence speech would contribute to an ideal user interface, especially for na6ve users, because it is natural, flexible, efficient and economical form of communication. The various process associated with text to speech synthesis, in, general is shown in Fig.6

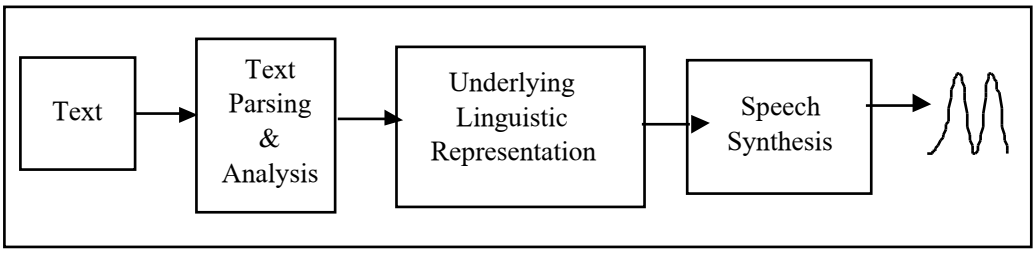


Fig.6

The linguistic representation is done using PANDITHAM, where the coding scheme helps in finding compact descriptions for speech signals or phonemes. PANDITHAM defines speech units, each of which symbolizes the sound of a human utterance. These discrete speech units are joined to regenerate the speech in such a way that the joins are not evident, using digital signal processing techniques.

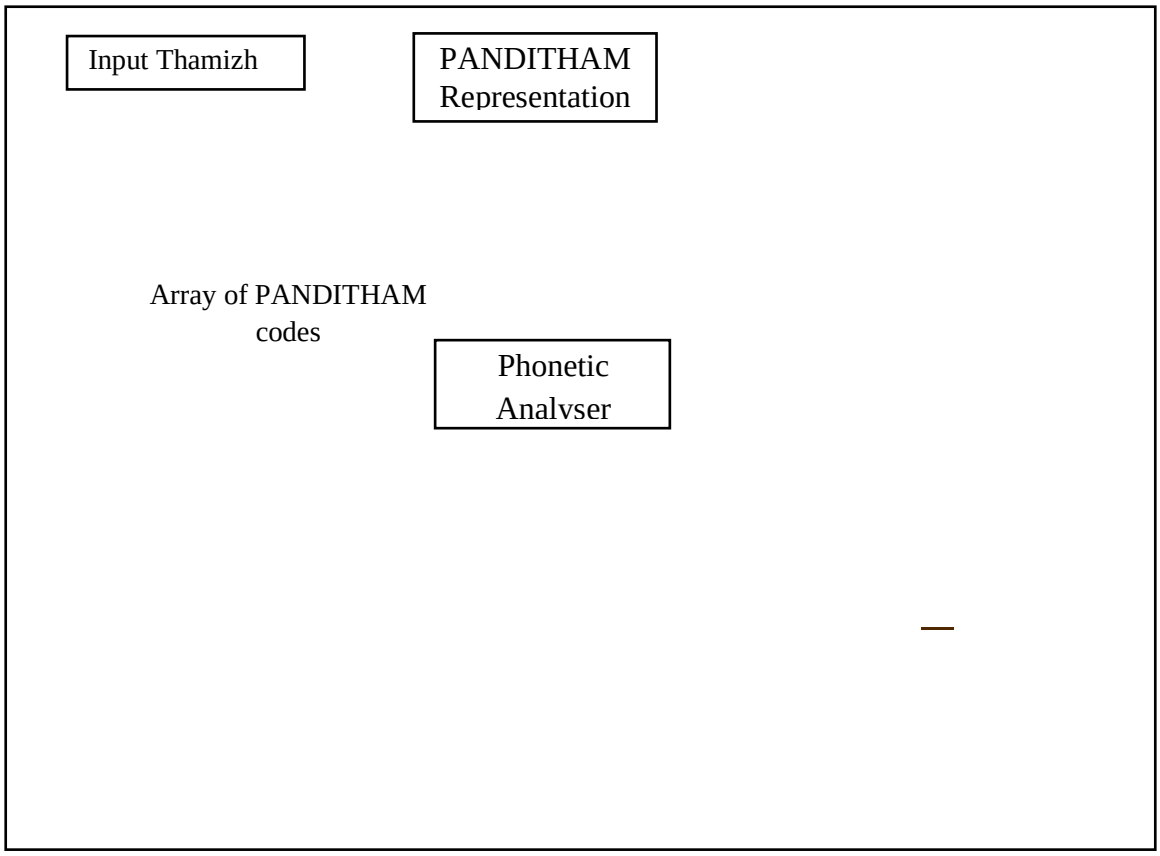
The Fig. 7 shows how PANDITHAM can be used for synthesizing speech from the given Thamizh text.

The synthesizer uses concatenative techniques for speech synthesis. The basis on which this concatenation has to be done forms the phonetic analyser. Phonetic analysis is concerned with processing input text, and producing the current phonetic representation for that text.

The straightforward method would be to have a look-up of a database, which contains the phonetic representations. This database is constructed on the basis of a table, which has been constructed by analyzing various words, and then, identifying various phonemes, which needs to be processed as a single diphone. This table helps in determining optimal combination of wave files for better quality speech output.

The input Thamizh string is translated into PANDITHAM codes, which has to be analysed before deciding on the wave files, which are to be played to give the best quality output. The Diphone is identified in a given Thamizh string by searching a database of Diphone wave files. Care has been taken to pronounce or read the trivial cases in Thamizh like kbfk, kbfBkfekaqf, kdfD, kdfci etc.

There are few other special cases in Thamizh, where three phonemes join as single phoneme. For example in case of `rfcfc[a, if synthesized separately, would not give the correct effect, and hence rfcfc has to be recorded separately and treated as a single phoneme, may be we can call this as triphone.


Input Thamizh
PANDITHAM Representation
Array of PANDITHAM codes
Phonetic Analyser

References:

1. Tamilnet 99 Conference Papers
2. Anbarasan N, 'A Perspective on Evolving standard for Tamil', Appletsoft Bangalore.
3. The Unicode Standard (Version 2) from the Internet
4. Dr. P. Navaneethan, R. Madheswaran, R.Balasubramaniam, N. Rajasekaran, 'PANDITHAM' A Protocol for Applications Development in Thamizh and Multilingual Computing , ADCOM-99 Conference Paper.
5. Alan .V Oppenheim, Roal W. Schafer, Digital Signal Processing, PHI,1975

Acknowledgement:

The authors of this paper acknowledge the help, support and encouragement provided by their Managing Trustee, Mr. G.R. Karthikeyan, Mr. C.R. Swaminathan, Chief Executive, PSG Institutions, Dr. P. Radhakrishnan,Principal, Dr. R. Nadarajan, Head, Dept. of Computer Applications, and Faculty Members of PSG Tech.