Iterm - Indian terminal emulator for X

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by
Nitu Choudhary

to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
Jan 1997

Certificate

Certified that the work contained in the thesis entitled "*Iterm - Indian terminal emulator for X*", by Ms. *Nitu Choudhary*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr. Rajat Moona)
Associate Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

Jan 1997

Dedicated To

My Parents

Abstract

In this thesis, `iterm` - an X Window based multilingual terminal software, is implemented. This software allows the entry, and simultaneous display of text written in Brahmi-based Indian scripts and English. Keyboard and Display driver are the two basic components of `iterm`. Keyboard driver deals with the entry of text in various scripts, while the display driver is responsible for displaying the text in chosen script. The software has been tested against various applications such as editors, filters, compilers, etc, on Digital Unix and Linux operating systems. Input/Output of only Devanagari and English text has been tested, as fonts for other Indian scripts were not accessible.

Acknowledgements

I express my sincere gratitude to my supervisor **Dr. Rajat Moona**, for his able guidance and valuable suggestions. I thank **Dr. Rajeev Sangal** and **Dr. Vineet Chaitanya**, professor in computer science and engineering and currently involved in research work at University of Hyderabad, for their help and encouragement. I also thank my family members and friends for their help and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Terminals provide an interface via which the computer and users communicate with each other. There are many applications such as word processing, natural language processing, computer aided learning, etc, which needs a multilingual terminal allowing input/output in various languages of the world.

X Window system [9] is a network transparent window system developed at MIT (Massachusetts Institute of Technology). It runs on a wide range of computing and graphics machine. X has a widespread support, and is one of the most extensively used windowing system. One of the major advantages of X Window is that all the X application programs can run without modification on a wide variety of architecture. There are various terminal emulators under X, which provides support for Japanese, Chinese, Korean, and English texts [6].

India is a multilingual country, with 15 official languages spoken throughout India [1], written in several different scripts. The common phonetic structure of Indian

scripts allow easy transliteration between one script and another. Due to this common structure, same terminal can easily support several Indian scripts. Terminals supporting I/O of Indian scripts are available under a wide variety of platforms [4], however, there is no such support for X Window. The motivation for this thesis was to develop a multilingual terminal software running under X Window, which could allow input and output of Indian script characters.

1.2 Terminal emulators for X

Most commonly used terminal emulator for X, which supports input/output of English text is `xterm` [7]. The `xterm` program emulates DEC VT102 and Tektronix 4014 terminals. It allows scrolling of displayed text, and also supports cut and paste feature. Text is coded according to ASCII (American Standard Code for Information Interchange) standards. The `xterm` terminal emulator, however, supports only fixed width fonts and does not provide smooth scrolling, VT52 mode, the blinking character attribute, or the double-wide and double-size character set. Besides `xterm`, several variations of `xterm` exists, with provision for input/output of English text [6].

The `kterm` program [6] is an X11R4-based VT100/VT102 and Tektronix 4014 terminal emulator that supports the display of Chinese, Japanese, and Korean text (in VT mode). It has capabilities of displaying Kanji strings and inputing them with kinput [8] program. Multi-byte coding is used for storing the text.

The `cxterm` terminal emulator [6] is a Chinese `xterm`, which supports both GB312-1980 and the so-called BIG-5 encoding. Hanzi input conversion mechanism is inbuilt in `cxterm`. Most input methods are stored in external files that are loaded at run time. Users can redefine any existing input methods or create their new ones.

Another terminal emulator `hanterm` [6], which is a modified xterm, supports Hangul (Korean writing system) input/output.

1.3 Indian terminals

Hardware support is in form of GIST (Graphics and Intelligence based Script Technology) cards and GIST multilingual video display terminals [4]. GIST card can be used with all IBM-PC compatibles running under MS-DOS/PC-DOS, and Xenix operating systems. GIST card device driver is to be installed along with GIST card. The GIST terminal is compatible with VT52/ANSI/VT100/VT220/VT320 standards. This terminal can be used under multi-user operating systems like Xenix, Unix, VAX VMS or any other system supporting DEC VT100/VT220/VT320/VT52 terminals. On the other hand, software support is provided in form of GIST shell running under MS-Windows.

GIST supports I/O of all major Indian scripts and a number of foreign scripts. This includes Devanagari (used for Hindi, Marathi, Nepali and Sanskrit languages), Bengali, Gujarati, Punjabi, Tamil, Telugu, Malayalam, Kannada, Oriya, and Assamese. Even the "right to left" scripts like Urdu, Sindhi, Kashmiri, Arabic and Persian are supported. It also accommodates some foreign scripts like English, Russian, Arabic, Thai, and Druk (Bhutanese and Tibetan). GIST has provision for automatic transliteration between all Indian languages. All popular database packages, word processors, spreadsheets, compilers and interpreters can be used in any of the above languages. It also allows printing of documents in graphics mode on a variety of printers.

These terminals are designed to support Inscript (Indian Script) keyboard overlay and 7/8 bit ISCII (Indian Script Code for Information Interchange) coding standards [3]. They also require that the fonts used for display of Indian script characters should follow the ISFOC (Indian Standard FOnt Code) [4] standard.

1.4 Features of `iterm`

The `iterm` program is an X Window based multilingual terminal software, similar to `xterm`, providing an interface for I/O of ten Brahmi-based Indian scripts and English. The user can set the keyboard and display mode to communicate with computer in script of their own choice.

The salient features of `iterm` are as follows :

- The entry and simultaneous display of text written in Indian languages and English is supported by `iterm`.
- The `iterm` has been designed to support all the Brahmi-based Indian scripts - Devanagari, Bengali, Gujarati, Punjabi, Tamil, Telugu, Malayalam, Kannada, Oriya, Gujarati and Assamese. But, at present it is configured to support I/O of only Devanagari scripts. A user can, however, easily configure `iterm` to support all the above mentioned Indian scripts.
- The `iterm` program emulates DEC VT102 and Tektronix 4014 terminals, however, only DEC VT102 window supports the display of text written in Indian scripts.
- The `iterm` program is also portable across various architectures which can run X Window.
- A wide range of application programs can run on `iterm`, without any modification.
- The `iterm` also supports scrolling and cut and paste feature.
- Both fixed and variable size fonts can be used to display the text in English or any Indian scripts.
- Keyboard and display are independent of each other. While English script characters may be entered via keyboard, display may be set to show the Indian

script characters and vice versa. This feature allows the existing applications to run under `iterm`.

- English characters are coded according to ASCII standards, while ISSCII-8 and EA-ISCII standards have been used for encoding Indian script characters. However the user can specify his own 7 and 8 bit standards for Indian script.
- Inscript keyboard overlay is supported by `iterm`, but this mapping can be redefined through configuration file as per the choice.
- Any font which may or may not follow the ISFOC standard is supported by `iterm`.
- Composition rules for Indian scripts can be redefined by the user.
- Default values of certain parameters, used by `iterm`, can be changed through its resource database.

Keyboard and display driver are the primary components of `iterm`. Keyboard driver receives input from the keyboard, encodes them and sends these codes to the application program executing on the host computer. The keyboard driver also handles keyboard mapping thus allowing entry of characters in chosen script. Display driver receives the text and displays it in the script selected by the user. The display driver also takes care of text manipulation, scrolling, and cut and paste.

The software has been developed in `C` using `Xlib` and `X toolkit` libraries [10, 11, 12, 13]. It runs under `X11 Release 5` and above. The software has been developed and tested on Digital Unix and Linux operating systems.

1.5 Organization of thesis

Rest of the thesis is organized as follows. Chapter 2 introduces the terms and concepts relevant to discussions held in later chapters. Chapter 3 discusses the design

and implementation details of keyboard and display driver - primary components of `iterm`. Results of various tests are presented in chapter 4. Finally chapter 5 concludes the thesis and makes suggestions for possible extensions. Appendix A contains the summary of VT100 terminal features and `iterm` control sequences. Appendix B and C enumerates the character sets, and Inscript keyboard layout respectively. Appendix D is a user manual which provides guidelines on how to run and customize `iterm`.

Chapter 2

Background

In this thesis, an attempt has been made, to develop an X Window based multilingual terminal, which provides I/O support for Indian languages and English. In order to fully comprehend the capabilities of such a terminal, it is essential, to understand the basic capabilities of a terminal, and computer representation of Indian languages. This chapter provides the background which will be useful later when we discuss the design of `iterm`.

2.1 Terminal

A terminal provides user with the mechanism to communicate with application programs executing on host computer. Terminals consists of transmit and receive blocks. The transmit block interfaces with the keyboard, and sends the characters typed in to the computer. The receive block receives characters from the host and interfaces with the monitor for displaying them. Terminals support the standard I/O operations as well as terminal specific operations to control input/output behaviour and cursor editing. Figure 1 shows the basic building block of a terminal.

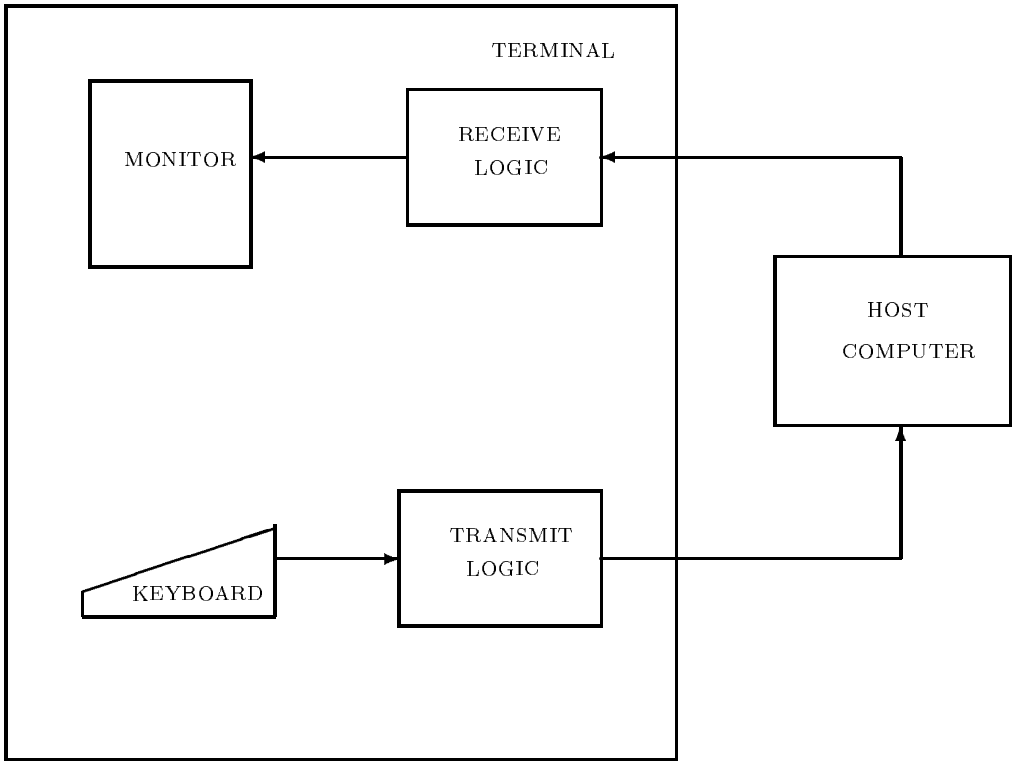


Figure 1: Block diagram of a terminal

The most popular and standard code used worldwide for data exchange between the terminal and computer is ASCII (American Standard Code for Information Interchange). It is a 7-bit code which defines 32 "control characters" and 96 "graphics characters". Refer to Appendix B for ASCII code chart.

The terminal provides capabilities, for displaying a stream of characters received from the computer. However, certain programs like screen editors, requires to manipulate the text that was sent before. They need to scroll the page, insert character, move the cursor, delete lines, etc. So the terminals provide control sequences, which allows the application program to modify the text that has already been displayed.

There are wide variety of terminals available, each of which includes a particular set of features. In the following discussion, the features of a character based terminal are reviewed.

2.1.1 Keyboard

Keyboard, with each terminal, contains standard typewriter keys and some additional keys to generate control sequences, cursor control sequences, cursor control commands, and status indicators. The keys when pressed transmits one or more character codes to the host. Some other keys such as `control` and `shift` do not transmit codes when typed, but modify the codes transmitted by other keys.

The DEC VT style keyboard consists of the following parts :

- **QWERTY keypad:** These keys generate standard ASCII codes. When `caps lock` is selected, the alphabetic keys transmits the uppercase codes. With `shift` selected, the alphabetic keys and numeric keys transmits uppercase and shifted codes respectively.
- **Special keypad:** These keys have some special significance, and consists of the `tab, lock, ctrl, shift, return` and `delete` keys. The tab key sends a horizontal tab character, which moves the cursor to next tab stop. The `ctrl` key used in conjunction with other keys generates control codes, usually in the range of `00H-1FH`. The `caps lock` key has a toggle function, and when selected converts codes generated by QWERTY keypad to uppercase. The `shift` key converts codes generated by the QWERTY keypad to shifted codes. The `return` key sends a `carriage return`. Pressing the `delete` key sends the code for `CAN` (cancel) character. There is also a `compose` character which is used to generate characters not present on the keyboard.
- **Editing keypad and cursor control keys:** These keys, when pressed, generate a set of control sequences for cursor movement and editing.
- **Numeric keypad:** It is used to enter numeric data. Control sequences are generated when in application mode.
- **Function keys:** These keys have functions assigned to them by the application software in use. Keyboard will usually send a pre-defined character

sequence on pressing these keys.

2.1.2 Display

Generally the screen is divided into rows and columns of characters. Codes received by terminals are rendered on the screen, in form of characters. Apart from displaying the normal characters, terminal also receives control sequences, specifying some special action to be taken. Each terminal provides different set of control sequences. Appendix A lists the control sequences provided by VT100 terminal.

The control sequences can be grouped as:

- **Character attributes:** Application program can specify whether the character is to be rendered in bold or reverse video. It can also specify if the character is blinking, or if it is to be underlined.
- **Cursor commands:** The application program can control the movements of cursor, using this feature. Also it can ask the terminal to save and restore the state of a cursor.
- **Line size:** It allows the application program to specify the height and width of the line.
- **Erasing:** The application program specifies the portion of the screen to be erased.
- **Character set:** The terminal can provide for many character sets, one of which may be chosen as the active font. If the received character value is less than or equal to 127 then the character displayed is selected from the GL group. If the received character is greater than or equal to 128 then the character is displayed from GR group. At any time GL and GR group can have one of the four sets defined to them namely G0, G1, G2 and G3. The G0,

G1, G2, G3 character sets are designated to represent one of the character sets, namely US ASCII, UK ASCII, Dec Graphics, etc. Any of these character sets can be invoked by a series of control sequences, as specified in Appendix A.

- **Scrolling region:** Some control sequences are used to set the scrolling region within which the text is to appear.
- **Tab:** Some control sequences are used to set or clear the tabs.
- **Modes:** Control sequences are also provided to set the number of column between 80 and 132, screen mode as reverse or normal, etc. For a complete list of all the modes refer to Appendix A.
- **Editing:** There are several control sequences for insertion and deletion of lines and characters.
- **Reports:** These control sequences are used by application program to get various status reports.
- **Reset:** The terminal can be reset to initial state by this option.
- **Test:** The application program can test for screen alignment.

2.2 Overview of Indian languages

India is a multilingual country having about 15 officially recognized languages, written in various scripts. These existing scripts are derivative of ancient Brahmi and Perso-Arabic scripts. Urdu, Sindhi, Kashmiri are primarily written in Perso-Arabic scripts. All the other Indian languages have evolved from the ancient Brahmi script. The Northern scripts are Devanagari, Punjabi, Gujarati, Oriya, Bengali and Assamese, while the Southern scripts are Telugu, Kannada, Malayalam and Tamil [2].

Different standards have been envisaged for languages which originate from Perso-Arabic scripts, and for languages which originate from Brahmi scripts. The standards for Brahmi-based Indian scripts are reviewed below.

For the following discussion Devanagari, which is the official script of India, is chosen. Devanagari script is used for Hindi, Marathi, Nepali and Sanskrit languages. All the Indian scripts originating from Brahmi have a common structure, and hence all arguments for Devanagari are also applicable to other Brahmi-based Indian scripts. Also for simplicity, elsewhere, the term Indian scripts implies Brahmi-based Indian scripts.

2.2.1 Indian scripts structure

All Brahmi-based Indian scripts are phonetic in nature. The alphabet in each may vary somewhat, but they all share a common phonetic structure. The differences between scripts primarily are in their written forms, where different combination rules get used [3].

Devanagari character set can be categorized into *vowels, consonants, matras, modifiers, numerals, punctuation* and some special symbols like *halant* and *nukta*. Figure 2 shows the set of basic symbols used in Devanagari script.

In Devanagari scripts *consonant* have an implicit *vowel* अ attached to it. A *pure consonant* is obtained by attaching a special symbol called *halant* to the *consonant*. Most of these *pure consonants* have a different graphic form.

Each *vowel* except अ has a corresponding *matra* which can be attached to a *consonant* to form composite characters. The modifiers are *ansuswar* (causing nasalization), *visarg* (introducing aspiration), and *chandrabindu* (causing prolongation). The diacritic mark *nukta* is used along with some *consonants*, and is mostly used to represent some foreign sounds. All *punctuation* marks used in Indian scripts are

Figure 2: Basic Devanagari symbols

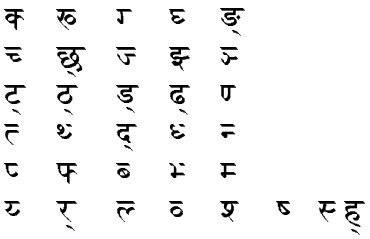


Figure 3: Graphical representation of *pure consonants*

similar to the ones used in English, except for the full-stop, instead of which *viram* is used.

Devanagari script is a logical composition of its constituent symbols in two dimensions. The *matras*, *modifier*, *halant*, and *nukta* can be attached to a *vowel* or a *consonant* to the right, left, top or bottom.

ख खा खि खी खु खू खृ खे खै खो खौ
खँ खं खः
ख्
ख़

Two or more *pure consonants* combine to form a *conjunct*. *Conjuncts* can form composite characters by the addition of *matras*, and *modifier* in the same way as *consonants*. Shape of these *conjuncts* usually differ from those of the constituting *consonants*.

क ् ष = क्ष
प ् र = प्र

2.2.2 Syntax

A word is composed of syllables, which are formed from the alphabets of character set discussed above. There are certain rules by which these characters can be combined. The syntax for formation of a word is given in the following Backus-Naur Formation (BNF) [3].

```
Word                ::= {Syllable}[Cons-Syllable]
Syllable            ::= Cons-Vowel-Syllable | Vowel-Syllable
Vowel-Syllable      ::= Vowel[Modifiers]
Cons-Vowel-Syllable ::= [Cons-Syllable]Full-Cons[Matra][Modifiers]
```

```
Cons-Syllable       ::= [Pure-Cons][Pure-Cons]Pure-Cons
Pure-Cons           ::= Full-Cons Halant
Full-Cons           ::= Consonant [Nukta]

Following conventions are used in the syntax given above :
::=  defines a relation.
{}   encloses items which may be repeated one or more times.
[]   encloses items which may or may not be present.
|    separates items, out of which only one can be present.
```

A syllable can at the maximum have four *consonants*. In the above syntax *nukta* can come only come after certain *consonants* with which it can combine. The above discussion also ignores some *vowels* derived through *nukta*.

2.3 Coding and keyboard standards for Indian scripts

2.3.1 ISCII standard

Since the 70s, different committees of the Department of Official Languages and the DOE (Department of Electronics) have been evolving different codes and keyboard, which could cater to all the Indian scripts due to their common phonetic structure. In 1980s the ISCII code (Indian Script Code for Information Interchange) was recommended, and it is widely used for internal representation of Indian scripts. Also the keyboard standard for ISCII character set was proposed around the same time, and has become the de facto standard.

ISCII character set

ISCII character set [3] is a super-set of all the characters required in ten Brahmi-based Indian scripts. For convenience, the alphabet of the official script Devanagari (with diacritic marks for non-Devanagari alphabets) has been used in the standard. The ISCII code contains only the basic alphabet required by Indian scripts, and all the composite characters are formed by the combination of these basic characters. Refer to Appendix B for ISCII character set.

ISCII code has the advantage that there is only one unique way of typing a word. The spelling of a word is the phonetic order of the constituent basic characters. This provides a unique spelling for each word, which is not affected by the display rendition.

म च ् छ र = मच्छर
क ि श ् त = किश्त

As shown, display order may be different from the phonetic order. Having a spelling according to the phonetic order allows a name to be typed in the same way, regardless of the script it has to be displayed in, thus simplifying the transliteration procedure.

A word in an Indian script can be displayed in a variety of styles depending on the *conjunct* repertoire used. ISCII codes however allow a complete delinking of the codes from the displayed fonts. An ISCII syllable can be displayed using combination of basic shapes. Different implementations can choose variant techniques in combination of these basic shapes. The same text can thus be seen in different font styles by using a different font composition routine.

आ → अ ा
प ् र → प्र or प्र
ल ् ह → ल्ह or ल्ह

Also use of INV (Invisible) character, explicit *halant* and soft *halant* allow the display to be rendered differently.

- The INV character present in ISCII character set is used for formation of composite characters which require a consonantal base, but where the *consonant* itself should be invisible.

  क ् Inv = क
  Inv ् र = ◌्र

- Many a times it is essential to show an explicit *halant* on the *consonant*. Two consecutive *halants* allows the formation of the explicit *halant*.

  श क ् त ि = शक्ति
  श क ् ् त ि = श्क्ति

- A soft *halant* is formed by typing a *nukta* character after a *halant*. This prevents the preceding *half consonant* to combine with the following *consonant*.

  प त ् त ि य ा ़ = पत्तियां
  प त ् ़ त ि य ा ़ = पत्तियां

ISCII character set has two additional characters: ATR and EXT. The ATR code, followed by a valid ASCII character, defines a font attribute applicable for the following characters. The details are given in Appendix B. The EXT code defines a new character which can combine with the previous ISCII character. This provision has been primarily used for supplementing Vedic signs along with Devanagari text.

ISCII codes are rendered on the display device according to the display composition methodology of the selected script. Transliteration to another script can be obtained by merely redisplaying the same text in a different script.

Eight-bit ISCII code

In this section ISSCII-8 (Indian Script Standard Code for Information Interchange) [3], as standardized by DOE in 1986, is reviewed. The lower 128 characters of the 8-bit table contain the ASCII character set, while upper half of the table is used for Indian script code. The first two columns in upper half of the table is reserved for control characters as per the recommendation of ISO (International Standard Organization). Refer to Appendix B for the table. This coding scheme allows Roman characters to be freely mixed with Indian scripts.

Seven-bit ISCII code

Seven-bit coding is recommended for those computers and packages which do not allow the use of 8-bit codes. In 7-bit coding 128 positions are available for representing all the characters of the script.

In ISSCII-7 [3] coding, control codes of ASCII are retained and all other symbols are used for representing the Indian script alphabets. Refer to Appendix B for ISSCII-7 table. This coding however has the disadvantage that Roman scripts cannot be mixed with Indian scripts.

Another 7-bit coding EA-ISCII (English Alphabet ISCII code) [3] allows Roman scripts to be mixed with Indian scripts. Refer to Appendix B for the table. The English upper and lowercase alphabet are interpreted as the corresponding Indian script character shown in the middle of the column, when an 'x' is present at the beginning of the word. The characters shown towards the right of a column are obtained by appending the *nukta* code, to the corresponding Indian script character shown in the middle of a column. All the vowels, except अ, are obtained by appending the corresponding *matra* to अ.

- *Inscript keyboard*

The Inscript (Indian Script) overlay [3] can be used on any QWERTY Keyboard. The Inscript Overlay contains characters required for all the Indian scripts, as defined by the ISCII character set. It is optimized from phonetic/frequency considerations which allows ease in typing the Indian scripts. Appendix C contains the Inscript overlay for the ISCII character set as well as for individual Indian scripts.

2.3.2 Other standards

Another popular representation, published by NCST (National Centre for Software Technology) [5], is *pure consonant* based coding. In this representation the *consonants* are always in their pure form i.e with *halant*. *Vowels* when added to *consonants* results in the corresponding *matra* symbol on the *consonant*. The coding table is a 7-bit table where some of the ASCII codes are replaced by the Indian script characters. This coding facilitates automatic alphabetization in perfect order of Devanagari. Also it does not disturb basic ASCII codes of most of the signs which are common in Devanagari and Latin.

Chapter 3

Design and implementation

Instead of developing afresh, `iterm` has been a result of modification of `xterm` - a terminal emulator for X [7]. Thus `iterm` inherits all features of `xterm`, and in addition provides entry and display capabilities for Indian scripts and variable width English fonts. This chapter discusses the design and implementation of `iterm`. Instead of examining in detail all the features supported by `iterm`, main stress is laid on how `xterm` was modified to provide support for Indian scripts.

3.1 Design issues

A multilingual terminal supports a group of languages/scripts, each of which has some special requirements. The terminal should be designed to efficiently handle all the common requirements, while at the same time it should also be able to deal with the additional requirments for each language. Hence `iterm` should support the distinct features of Indian scripts in addition to the common features of Indian and English scripts. English has the advantage of linearity, that is, it is typed and displayed in same sequence as it is written. However, Indian languages are non linear in nature. Several Indian language specific issues are:

- Indian script characters are of variable width. The symbols of the script can be attached either to the left, right, top or bottom of the previous symbol.
- As there are a large number of distinct display shapes in Indian scripts, a character code usually do not correspond to a single display shape. Several character codes may combine to form one display shape, or one character code may be represented by several display shapes, available in the font.
- The order in which the characters are typed is not necessarily the order in which they are displayed.
- As the characters are typed, they may combine with earlier typed characters to form an entirely new display shape.
- The character codes can only combine according to certain rules, to generate appropriate display shapes.
- The generation of font codes (display shapes) are context dependent.

A truly usable multilingual terminal, which can handle Indian scripts, should also have flexibility to support several standards in use. Some of the issues related to this are:

- Codes are used for internal representation of characters. Unlike in English scripts where ASCII is the de facto coding scheme, an Indian language has several coding schemes in use. So the terminal should be flexible to support any of the current and future coding standards.
- Keyboard mapping is essential to input characters in variety of scripts. There should be some provision to handle keyboard mapping according to users choice.
- Fonts are required for display of characters. There might be many fonts which may not follow the standard coding. There should be flexibility to support any font.

- The existing software packages should be able to run without any modification.

3.2 General design of `iterm`

As `iterm` is derived from `xterm`, it provides DEC VT102 and Tektronix 4014 compatible terminals, however, only VT102 terminal supports the display of Indian scripts. The VT102 terminal support is fairly complete, but does not emulate smooth scrolling, VT52 mode, the blinking character attribute, or the double-wide and double-size character sets. Appendix A contains the list of control sequences supported by `iterm`.

In addition to emulating a terminal, `iterm` also provides cut and paste features and support scrolling, whereby the number of lines in the scrolling region can be specified. A status line is provided at the bottom, which contains the current terminal mode for keyboard and display. Menus are present which allows the user to change the terminal settings, fonts, and send various signals to `iterm`. Being an X application program `iterm` also provides the user with screen resizing and refreshing features. Resource files allow the overriding of initial values of parameters, used by `iterm`.

The `iterm` allows existing text based applications to be run using Brahmi-based Indian languages along with English, where English letters can be freely mixed with any Indian script text. The same keyboard can be used to switch between English or Indian script inputs, by pressing some special keys. Similarly the display can be set to show the characters in English or Indian scripts, either by pressing some special keys or by an escape sequence. One of the Indian script out of those supported by `iterm` may be chosen from a menu. The status line shows the current Indian script. The keyboard mapping, character coding scheme, letter composition rules and font tables specific to the particular Indian script are reloaded, whenever a script is selected.

In Indian scripts the width of font characters are variable, and the characters may

be glued horizontally or vertically. Also there is no one to one correspondence between the character codes and the shapes to be displayed. A single character code may cause a combination of shapes to be displayed, while more than one character code may lead to only a single display shape. Since the terminal has to support the variable width of shapes, it does not assume the applications restriction on the number of characters that can be displayed in a row. The characters can be displayed till the total width of the characters in that row becomes equal to the width of the screen.

However many applications, such as editors, require to determine the number of characters that can be displayed per row. Depending on this information the application sends only the specified number of character to be displayed in a row. Due to this, a sentence which could have been displayed in a single row, maybe split over two rows by the application program. On the other hand, sentence may be wrapped to the next line by `iterm`, if the total width of the characters exceeds the width of the screen and autowrap feature is enabled. Since there will be discrepancy in lines displayed and the number of lines known to the editor in its data structure, editing problems will occur. There is no workaround for this problem except choosing judiciously the number of characters that can be displayed per row. For this a display shape is chosen as the base character, and the minimum number of characters that can be displayed per row is calculated by dividing the width of the window by the display width of this character. This font code should be such that it represents the average width of the most frequently used characters in the font. In `iterm` the base characters may be specified through its configuration files.

The `iterm` allows the user and applications to communicate with each other in different languages, translating keystrokes to codes and codes to display shapes. The English characters are coded according to ASCII, while Indian scripts are coded separately according to ISCII standards. There are some application programs which permits the usage of 8-bit codes, while other software packages allow only 7-bit codes to be used. To allow all kinds of application to run, iterm supports character codes for Indian scripts in two modes: seven or eight bit, and can switch to either coding

scheme dynamically. To provide further flexibility, codes may be defined through the configuration files. The current files, however, support ISCII coding standards: EA-ISCII (7 bit), ISSCII-8 (8 bit).

Keyboard and display driver are the important components of `iterm`. Their functioning, however, is independent of each other. Thus while the user can type in English characters, display may be set to show the text in one of the Indian scripts. This is necessary to support all the existing applications. Keystrokes from user are received by the keyboard driver, which converts them into appropriate code and transmits them to the application program. Similarly the codes received from the application programs are processed to check for control sequences. Special action is taken upon receiving a control sequence (as given in Appendix A), while other character codes are displayed. Display driver interprets these codes according to the chosen coding standard, converts them to display shapes and passes to X for display.

3.3 Configuration files

There are several configuration files used by the `iterm`. There is a specification file, which lists the scripts to be supported by `iterm`. Corresponding to each script, the user can specify his own coding schemes, keyboard mapping, character composition rules and font map. All these are provided in form of several files listed in the specification file. The details of each configuration file is discussed in this section.

3.3.1 Specification file

The main configuartion file, called the specification file, contains the list of Indian scripts to be supported by `iterm`. It also contains the default Indian script to be used initially. Other scripts can be selected from the menu provided in `iterm`.

The information presented in this file for each specified Indian script contains the following:

- Name of the font to be used for normal display.
- Name of the font to be used for bold display.
- Coding scheme file for specifying codes corresponding to each character.
- Keyboard map file for providing keyboard overlay.
- Font map file which contains mapping between characters and display shapes.
- Type map file which groups the character set into several user defined categories.
- Rules file which contains the display shape formation rule.

A summary of information presented in the specification file is shown in table 1.

Default Indian script							
Devanagari							
Indian script	Font		Files				
	Normal	Bold	Coding scheme	Keyboard map	Font map	Type map	Rules
Devanagari	dvng10	dvng10	iscii	keybd	font1	type	rule1
Gujarati	gujr10	gujr10	iscii	keybd	font2	type	rule2
Tamil	tam10	tam10	iscii	keybd	font3	type	rule3

Table 1: Example - specification file

3.3.2 Coding scheme file

This file contains the coding scheme for each Indian script character. As the `iterm` supports two modes of display, seven and eight bit, both codes are provided in this

file. For each Indian script character only one character code is generated in eight bit mode, while several codes may be generated in seven bit mode. Each Indian script character is represented by a user defined name such as *visarg, chandrabindu*, etc. These names are later used in other files. A name "Inv" is, however, reserved and a character code must be defined for this. It is used to complete a character which does not form a valid syllable, as per the rules specified in the rule file.

Character	String description	8 bit code	7 bit codes
ँ	Chandrabindu	161	A
ः	Visarg	163	B x
आ	Aa	165	C k
इ	I	166	C l
क	Ka	179	D
ख	Kha	180	E

Table 2: Example - coding scheme file

Codes used in seven bit mode are printable characters only, and are represented by ASCII characters whose ASCII code is above 32. Table 2 shows a part of the code file.

3.3.3 Keyboard map file

The mapping between the keyboard characters and the Indian script characters are enumerated in the keyboard map file. Only the keys of QWERTY keypad can be mapped to Indian script characters. A part of the keyboard map is shown in table 3. The first entry in the file, for example, denotes that by pressing '&' character on the keyboard, three Indian script characters (क, ्, ष) are generated whose codes are specified in the coding scheme file.

Keyboard characters	Indian script characters
&	क ष
#	◌ृ र
H	फ ◌ॅ
I	घ
X	◌ँ
D	अ

Table 3: Example - keyboard map file

3.3.4 Font map file

The font map file contains the following information:

- Font display code for determining the number of characters per row is specified separately for seven and eight bit modes.
- Sequence of font display code (for example: ि, ◌ै) to be moved to the beginning or end, for displaying the font string according to the given specifications, is also mentioned.
- Font table specifies the mapping between Indian script characters and font codes (display shapes). The font table is divided into several user defined categories (see table 4), and for each category the font mapping is defined. These category names are used later for specifying the combination rules. Font table is searched to generate the equivalent display shape for given string of character codes. "Conjunct" is a reserved category, and the mapping provided under this group is first searched.

3.3.5 Type map file

As discussed earlier, a word in Indian script is composed of syllables. A syllable is formed by combination of several characters, according to some specified rules.

Font Type	Characters	Font display symbols
Conjunct	क ् ष ज ् ञ प ् र	क्ष ज्ञ प्र
Vowel	अ आ ि ई उ	अ अ ा इ ई उ
Consonant	क ख ग घ ङ	क ख ग घ ङ
Half-Consonant	क ् ख ् ग ् घ ् ङ ्	क ख ग घ ङ ्
Matra	ा ि ी ु ू	ा ि ी ु ू

Table 4: Example - font table

The rules specification requires categorization of the character set. This character categorization is specified in the type map file. The category names are user defined and are used in defining syllable and combination rules. A character not listed in the type map file is assigned the default type provided by the user. Table 5 shows some user defined categories.

3.3.6 Rules file

The rules file lists the rules for character combination and syllable formation.

Default type	
Invalid	
Character	Type
अ	Type_Vowel
आ	Type_Vowel
इ	Type_Vowel
ई	Type_Vowel
क	Type_Consonant
ख	Type_Consonant
ग	Type_Consonant
घ	Type_Consonant
ं	Type_Modifier
ँ	Type_Modifier
ः	Type_Modifier
ा	Type_Matra
ि	Type_Matra
ु	Type_Matra
्	Type_Halant

Table 5: Example - type map file

Syllable rules

The syllable rules uses the character categories specified in the type map file. It lists all the valid categories, characters from which can be combined to form valid syllables. Table 6 lists some of the valid syllables.

Valid syllables			
1	Type_Vowel	Type_Modifier	
2	Type_Vowel		
3	Type_Consonant	Type_Matra	Type_Modifier
4	Type_Consonant	Type_Matra	
5	Type_Consonant	Type_Modifier	

Table 6: Syllable rules

- *Combination rules*

Characters in a syllable are converted into font display codes. The generation of these font display codes are context sensitive. For example, *consonants* when followed by a *halant* at the end of the word, is depicted as the consonant with halant attached at the bottom, while if the same combination occurs in the middle of the word, it is shown in it's pure form (refer figure 3).

The combination rules allow the user to specify the mapping between the input characters and the output font display shapes. This mapping is listed in form of rules, and it specifies the combination of character in various categories (as specified in type map file) which generates the font display code in several font categories (as specified in font table). Table 7 lists some combination rules.

Input character type			Display symbols type	
Type_Consonant	Type_Halant	End	Consonant	Halant
Type_Consonant	Type_Halant		Half-Consonant	
Begin	Type_Cons-r	Type_Halant	Reph	
Type_Vowel			Vowel	
Type_Matra			Matra	
Type_Modifier			Modifier	

Table 7: Combination rules

Begin and End are reserved keywords which can be used in the rules to denote the beginning and end of the syllable.

3.4 Keyboard

To input text in Indian and English scripts, the keyboard has provision for entry of Indian script and English characters. The same key represents characters from different languages, and depending on the mode of the keyboard, appropriate characters are generated. Thus `iterm` is designed to provide support for a keyboard containing English characters with an overlay provided for characters of Indian scripts.

The standard Inscript keyboard overlay, discussed in previous chapter, is supported. However the mapping can be modified according to ones own requirement.

The keyboard consists of the standard typewriter keys along with some additional keys, as reviewed in the previous chapter. The keyboard driver checks for the type of key pressed and if any other key apart from the keys from QWERTY keypad is chosen, standard escape sequences are sent to the application program. Appendix A lists these escape sequences. Also, there may be some special keys mapped to perform some specific functions. In that case, the corresponding action is carried out and no sequence is sent to the application program.

However if any of the keys from the QWERTY keypad is pressed, keyboard mapping is performed to generate appropriate characters. Also these characters are converted into codes which is then sent to the application program.

3.4.1 Keyboard mapping

In order to generate the relevant characters, keyboard driver keeps track of the keyboard mode. Generally, on pressing a key, the English characters are selected. However, if the keyboard mode is set to generate characters from other Indian scripts, English characters are mapped to characters in Indian script. This map is loaded at the initialization.

A function is provided to switch between the two modes. To select between the modes a key is mapped with this function, which is automatically invoked whenever the key is pressed. The key to be mapped to the function can be selected through resource database.

3.4.2 Encoding of characters

English characters are encoded according to the ASCII standard. However, the Indian script characters can be encoded according to the ISSCII-8 or EA-ISCII coding. The user chooses between one of these codings. Appropriate character codes are generated for each character and sent to the application program. Mapping between characters and codes are read from the file containing the coding scheme.

There are special functions to switch between eight bit and seven bit coding. These functions can be registered with X and are called whenever a special user defined key is pressed.

In seven bit character set (EA-ISCII), same code is used to represent English and Indian script alphabets. There is an escape character 'x' which when present at the beginning of the word indicates that the word is written in Indian script. Hence, whenever keyboard is set to generate characters of Indian script, and these characters are to be encoded according to the seven bit (EA-ISCII) standard, an escape character is inserted at the beginning of the word. This automatic insertion of escape character by keyboard driver prevents the user from explicitly typing it.

3.5 Display

Display driver displays the codes received form the application program at the current cursor location. It also provides some functionalities for manipulating the text. Some other features like cut and paste, and scrolling are also supported by the display driver. In this section all the above functionalities are discussed.

The received character codes are converted into font display codes, generation of which depends on the active character set. Display can be set to any one of the character set: US ASCII, UK ASCII, DEC Graphics, ISSCII-8 and EA-ISCII. The UK ASCII character set is the same as the US ASCII character, apart from the minor

difference that dollar sign in US ASCII is replaced by the pound sign in UK ASCII. The DEC special graphics character set is the same as ASCII character set except for the characters between 0x5f and 0x7e which are special line drawing characters. To refer to various character sets see Appendix B. The list of escape sequences sent to choose between any of these character sets are given in Appendix A.

After the generation of font display codes, exact position at which they are to be displayed, is determined. This is done by adding width of all previously displayed characters in the current row before the cursor position. All the characters may not fit in the same row, and if autowrap is enabled, the extra characters are displayed in the next row. Depending on the attributes the characters may be displayed as normal characters or may be printed in bold or reverse video.

Display of English characters involves placing the characters one after another in a linear sequence. However, Indian scripts, as reviewed, are very complex and there is a dependency between characters to be displayed and characters which are to the left of cursor. The new display symbols may be added to left, right, top or bottom of the previous symbol. Also new character codes may cause the character to the left of the cursor to be redisplayed. One such example is illustrated in figure 4.

Typing sequence	Display
प	प
प ्	ᳲ
प ् र	प्र
प ् र ि	प्रि

Figure 4: Composition of characters in Indian script

Hence, for proper display of characters, whenever characters are to be displayed, font display codes are generated for the whole word. If required, the previous characters are erased from the screen, and the new symbols are displayed. Display of characters in Indian script incurs some overhead, which is necessary for proper display of text.

3.5.1 Screen buffer

Codes received by the program are stored in the buffer along with its attributes. This is essential, as `iterm` allows scrolling, and also provide features for manipulating the text. The attributes of the character can be set by various control sequences, as discussed in previous chapter.

The screen buffer is big enough to hold the character rows currently displayed on the screen and the rows that are to be saved for scrolling. Each element of screen buffer points to a fixed size array, of characters. This array can store twice the maximum number of characters that can be displayed per row with their attributes. The maximum number of characters that can be displayed is equal to the pixel width of the screen. This maximum would be reached when each character on the screen is only of one pixel width (Example: *Viram*).

In case of English characters, the codes stored in the buffer has direct correspondence with the font display code. This prevents unnecessary translation between the codes and the display shapes in the font every time they are to be displayed. However, if the display is to be in any of the Indian languages, it is not possible to store the font display codes in the buffer. This is because one character code can generate a combination of font display codes. Also the new character codes may combine with previous character codes to form a new font display code, for which we require to store the previous codes. Hence to distinguish between the English and Indian characters, some information is stored along with the attributes which depicts the presence of Indian script characters.

3.5.2 Generation of display symbols

For English text, the character codes and font display codes have one to one correspondence. Also the characters are simply juxtaposed and each character is displayed independent of other characters present in that row. So the font display code

generation just involves mapping of character codes to font display codes.

However, to generate characters for several Indian scripts, several aspects are to be considered. The characters can be only combined according to some rules, and depending upon the context a character may be completely modified. To generate font codes for Indian scripts, steps involved are:

- **Step 1:** The word is checked for valid syllables. If there is an invalid symbol an "INV" character is inserted to make it a valid syllable.
- **Step 2:** The syllable is first searched for the presence of *conjuncts* in the font table. If *conjuncts* are found then the set of input character codes which matches the *conjunct* is replaced by corresponding font display codes.
- **Step 3:** For the rest of the character codes in the syllable, combination rules are checked and font table is searched, replacing the input character codes by the corresponding font display codes.
- **Step 4:** After the input string has been converted to a string of font display codes, some of them are moved for proper display.

An example demonstrating the various steps is shown in figure 5.

3.5.3 Cursor movements

The cursor is displayed as a block cursor in inverse mode, the width of which depends on the character on which it is placed. There is a horizontal cursor which shows the logical positioning of the character. Whenever the current window is unselected, the block cursor is changed to outline cursor.

In English there is one to one correspondence between the input character and the symbol displayed, so the cursor shows the actual character. However, in Indian

	Input (keyboard)	Font code generation			Display
		syllable	syllable	syllable	
Step 1	प ् र ा र ् थ न ा	प ् र ा	र ् थ	न ा	प्रार्थना
Step 2		प्र ा	र ् थ	न ा	
Step 3		प्र ा	र्थ	न ा	
Step 4		प्र ा	थ र्	न ा	
Step 1	श क ् त ि	श	क ् त ि		शक्ति
Step 2		श	क ् त ि		
Step 3		श	क त ि		
Step 4		श	ि क त		

Figure 5: Generation of display symbols

scripts many input characters may combine to form one display symbol. For example, क्ष is a combination of क, ् and ष. To make it easier to determine as to which of the characters cursor is positioned on, actual character is displayed on the status line.

To move the cursor from one position to another, first the cursor at the current location is hidden, and then a new cursor is drawn at the requested position. To draw the cursor, the character to be represented by the cursor is determined. Then the position and width of that character is found and a rectangular block is drawn surrounding that character.

There are various control sequences to manipulate the cursor. The cursor may be moved in any direction left, right, up or down till the screen boundaries are reached. Insertion of carriage return causes the cursor to be moved to the next line, while insertion of tabs causes the cursor to move horizontally. The detailed list of cursor movements is given in Appendix A.

3.5.4 Text manipulation

The terminal emulator allows application program to edit the stored text. Characters can be inserted at the current cursor position or they may overwrite the existing characters. Various control sequences are provided for deletion of characters and lines, insertion of blank characters and lines, erasing of certain portions of screen, and scrolling of text.

Insertion, replacement and deletion of characters requires the updation of screen buffer. To insert characters at the current cursor location, all characters between the cursor and the rightmost character of that row are moved to the right, by the number of characters that are to be inserted. The characters to be inserted are then copied at the current location. Overwriting of characters simply involves copying the characters at the current cursor location. Deletion of characters require that all characters to the right of the characters to be deleted should be moved to the left, by the number of characters that are to be deleted.

After the screen buffer is updated, these results are to be shown on the display. The insertion, deletion and replacement of English characters are very simple, as there are no relationships between the constituent symbols. Insertion and deletion of characters are reflected by first clearing the screen from the current cursor position to the end of the screen, and then displaying all the characters stored in the screen buffer from the current cursor location. Overwriting of characters simply involves erasing only a portion of screen, mainly the characters which are to be overwritten, and the new characters are displayed at that location. If the font being used has variable width, then the characters to the right may be required to be moved to the left or right.

However when Indian characters are edited some additional processing is required. This is because insertion, overwriting and deletion of characters may affect the characters to the left and right of the cursor location as shown in figure 5.

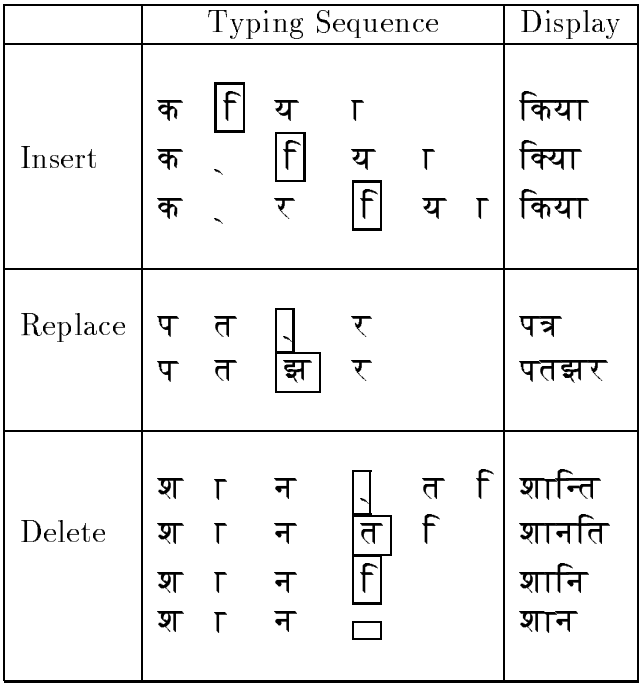


Figure 6: Insertion, replacement and deletion of characters

So the word boundary is determined, and instead of redisplaying the characters from current cursor location characters are redisplayed from starting of word.

To provide scrolling all characters in the scrolling region are stored in a buffer. Pointers denote the region which is currently displayed. This pointer is moved up or down whenever scrolling is requested. To reflect the affect of scrolling on the screen, screen is cleared and new text is displayed.

3.5.5 Cut and paste

The `iterm` allows already displaed text to be selected and copied within the same or other window. The selection functions are invoked when the mouse buttons are used. Pointer button one is used to save text into the cut buffer. The cursor is moved to the beginning of the text, and then the button is held down, while the cursor is moved to the end of the region and button is released. Selected text is

highlighted and is saved in the global cut buffer. Double clicking selects by words while triple-clicking selects by lines. Pointer button two pastes the text from the buffer. Pointer button three is used to extend the current selection. The assignment of the functions described to keys and buttons may be changed through the resource database.

X sends pixel position of the mouse when buttons are pressed or released. To mark the text to be cut, character on which the mouse button is pressed or released is determined. Pointers are used to mark the selected area. When the selection area is extended new character position is determined, and pointer values are changed. Once the selection is completed these characters are copied into the global cut buffer. To paste the text, these characters are inserted as keyboard input.

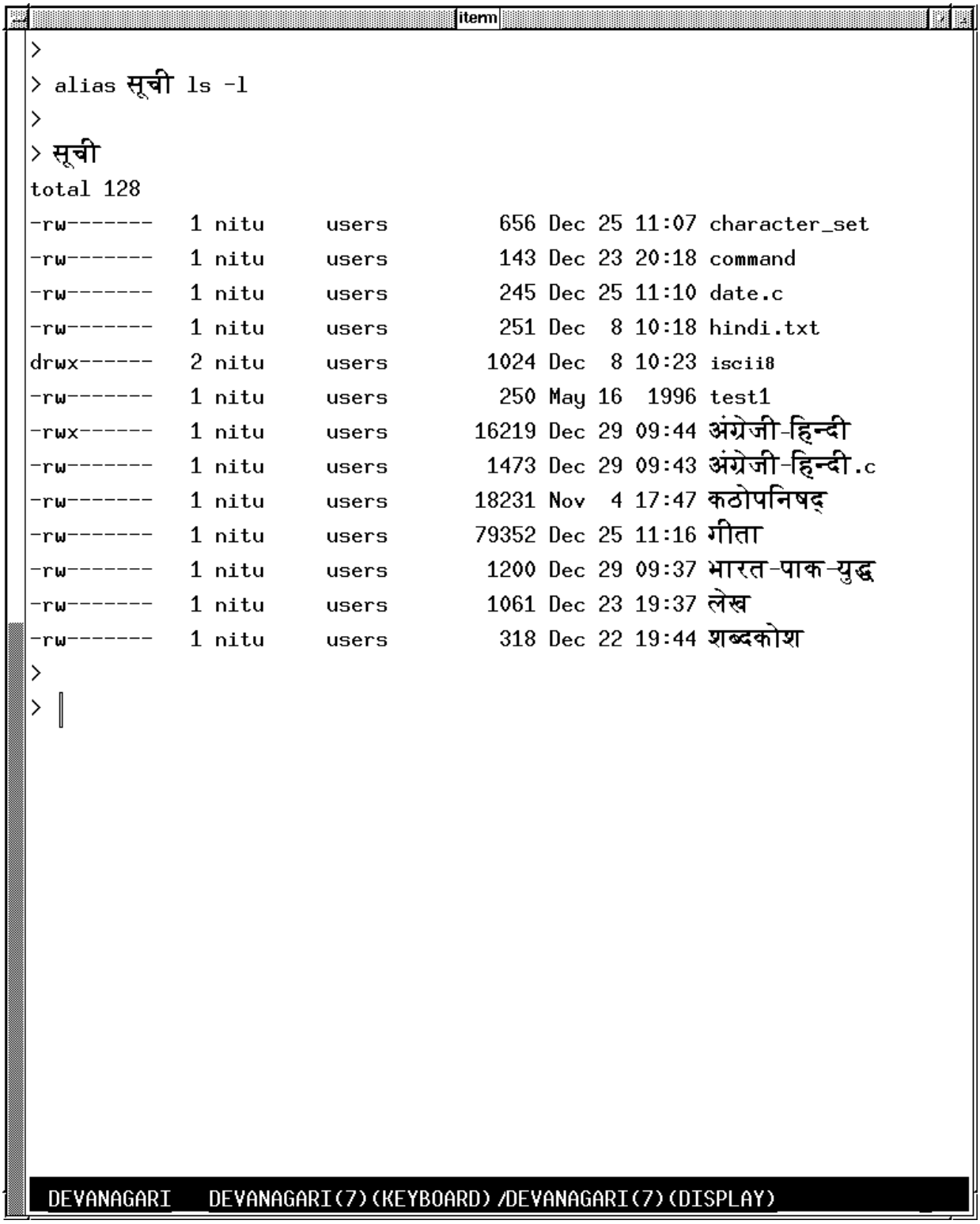
Chapter 4

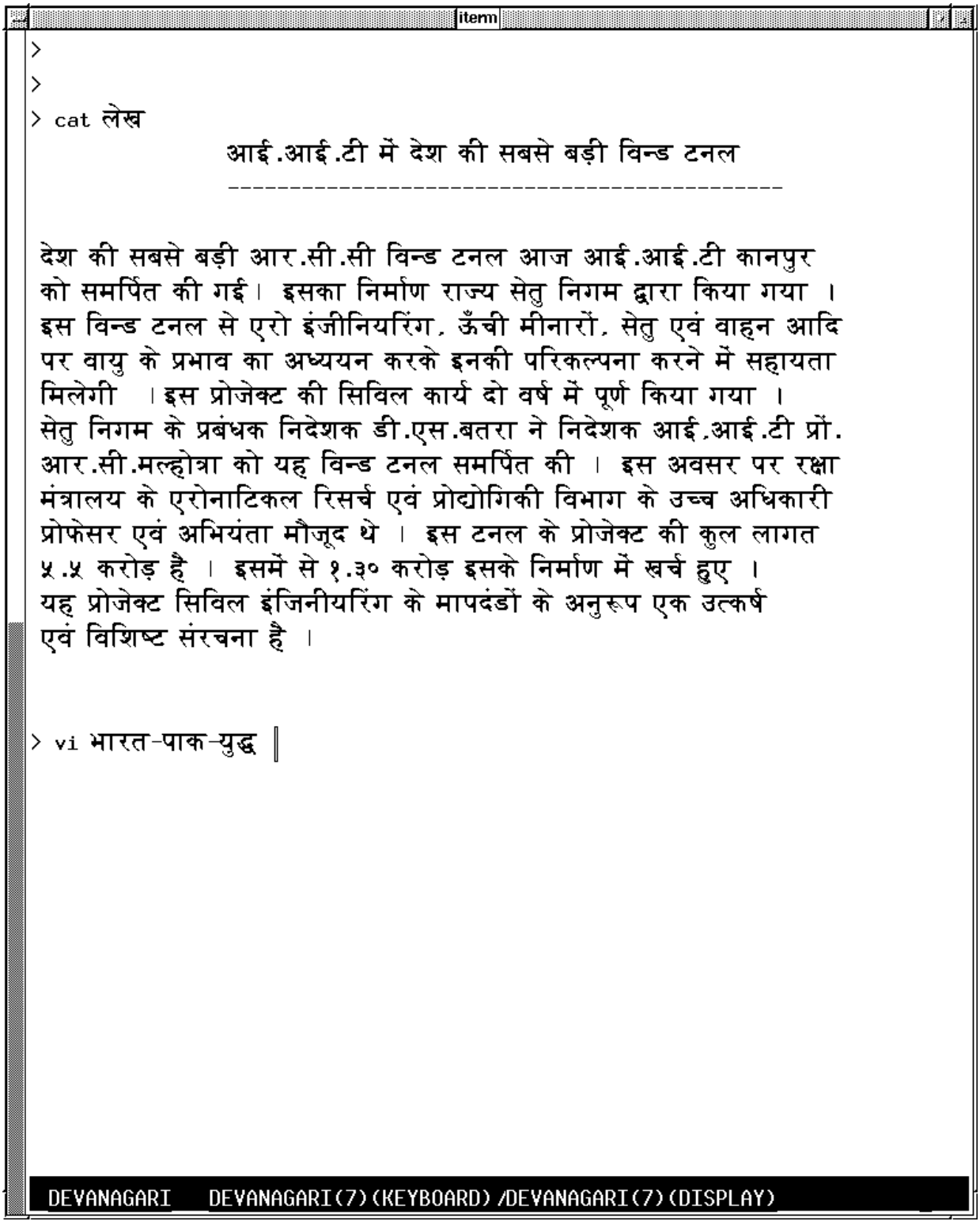
Results

The `iterm` supports all application programs that can run on xterm. The software has been tested against some application programs like `vi, more, cat, ls` and `C` compiler; and the results are presented in this chapter. The software was tested under Digital Unix and Linux operating system. As fonts for all Indian scripts were not available, tests were carried out with only Devanagari and English scripts.

Various snapshots of the screen showing interaction between a user and machine were taken. EA-ISCII was used for input and output of devanagari script while ASCII code was used for entry and display of English text. As EA-ISCII code allows English characters to be mixed with Devanagari characters, any text displayed using this coding scheme contains both English and Devanagari words.

With `alias` command user can create Hindi equivalent of English commands. One such example is shown in figure 7. Figure 7 also shows the contents of present directory. Figure 8 shows the contents contents of a file viewed by `cat` command. Figure 9 shows the file being edited using `vi` editor. Figure 10 and 11 displays the program written in `C`. Output of the program is presented in figure 12.


iterm
> alias सूची ls -l
> सूची
total 128
-rw------- 1 nitu users 656 Dec 25 11:07 character_set
-rw------- 1 nitu users 143 Dec 23 20:18 command
-rw------- 1 nitu users 245 Dec 25 11:10 date.c
-rw------- 1 nitu users 251 Dec 8 10:18 hindi.txt
drwx------ 2 nitu users 1024 Dec 8 10:23 iscii8
-rw------- 1 nitu users 250 May 16 1996 test1
-rwx------ 1 nitu users 16219 Dec 29 09:44 अंग्रेजी-हिन्दी
-rw------- 1 nitu users 1473 Dec 29 09:43 अंग्रेजी-हिन्दी.c
-rw------- 1 nitu users 18231 Nov 4 17:47 कठोपनिषद्
-rw------- 1 nitu users 79352 Dec 25 11:16 गीता
-rw------- 1 nitu users 1200 Dec 29 09:37 भारत-पाक-युद्ध
-rw------- 1 nitu users 1061 Dec 23 19:37 लेख
-rw------- 1 nitu users 318 Dec 22 19:44 शब्दकोश
DEVANAGARI DEVANAGARI(7)(KEYBOARD)/DEVANAGARI(7)(DISPLAY)


iterm
>
>
> cat लेख
आई.आई.टी में देश की सबसे बड़ी विन्ड टनल
देश की सबसे बड़ी आर.सी.सी विन्ड टनल आज आई.आई.टी कानपुर
को समर्पित की गई। इसका निर्माण राज्य सेतु निगम द्वारा किया गया ।
इस विन्ड टनल से एरो इंजीनियरिंग, ऊँची मीनारों, सेतु एवं वाहन आदि
पर वायु के प्रभाव का अध्ययन करके इनकी परिकल्पना करने में सहायता
मिलेगी ।इस प्रोजेक्ट की सिविल कार्य दो वर्ष में पूर्ण किया गया ।
सेतु निगम के प्रबंधक निदेशक डी.एस.बतरा ने निदेशक आई.आई.टी प्रों.
आर.सी.मल्होत्रा को यह विन्ड टनल समर्पित की । इस अवसर पर रक्षा
मंत्रालय के एरोनाटिकल रिसर्च एवं प्रोद्योगिकी विभाग के उच्च अधिकारी
प्रोफेसर एवं अभियंता मौजूद थे । इस टनल के प्रोजेक्ट की कुल लागत
५.५ करोड़ है । इसमें से १.३० करोड़ इसके निर्माण में खर्च हुए ।
यह प्रोजेक्ट सिविल इंजिनीयरिंग के मापदंडों के अनुरूप एक उत्कर्ष
एवं विशिष्ट संरचना है ।
> vi भारत-पाक-युद्ध
DEVANAGARI DEVANAGARI(7)(KEYBOARD)/DEVANAGARI(7)(DISPLAY)

भारत-पाक युद्ध

भारत-पाक के १९७१ के युद्ध में पाकिस्तान के तत्कालीन सेनाध्यक्ष जनरल याहिया खां युद्ध नहीं चाहते थे। वह जानते थे कि भारतीय सेना युद्ध के लिए पूर्णयता सुसज्ज है। पाकिस्तानी सेना इसका मुकाबला नहीं कर सकेगी। याहिया ने इन्दिरा गांधी के मामा राजा जे.के.अटल को, जो तब पाकिस्तान की कूटनीतिक यात्रा पर थे, बातचीत में अपने अहसास से अवगत कराया। याहिया बराबर कहते थे - राजा साहब मैडम से कहिए पाकिस्तान लड़ाई नहीं चाहता ।

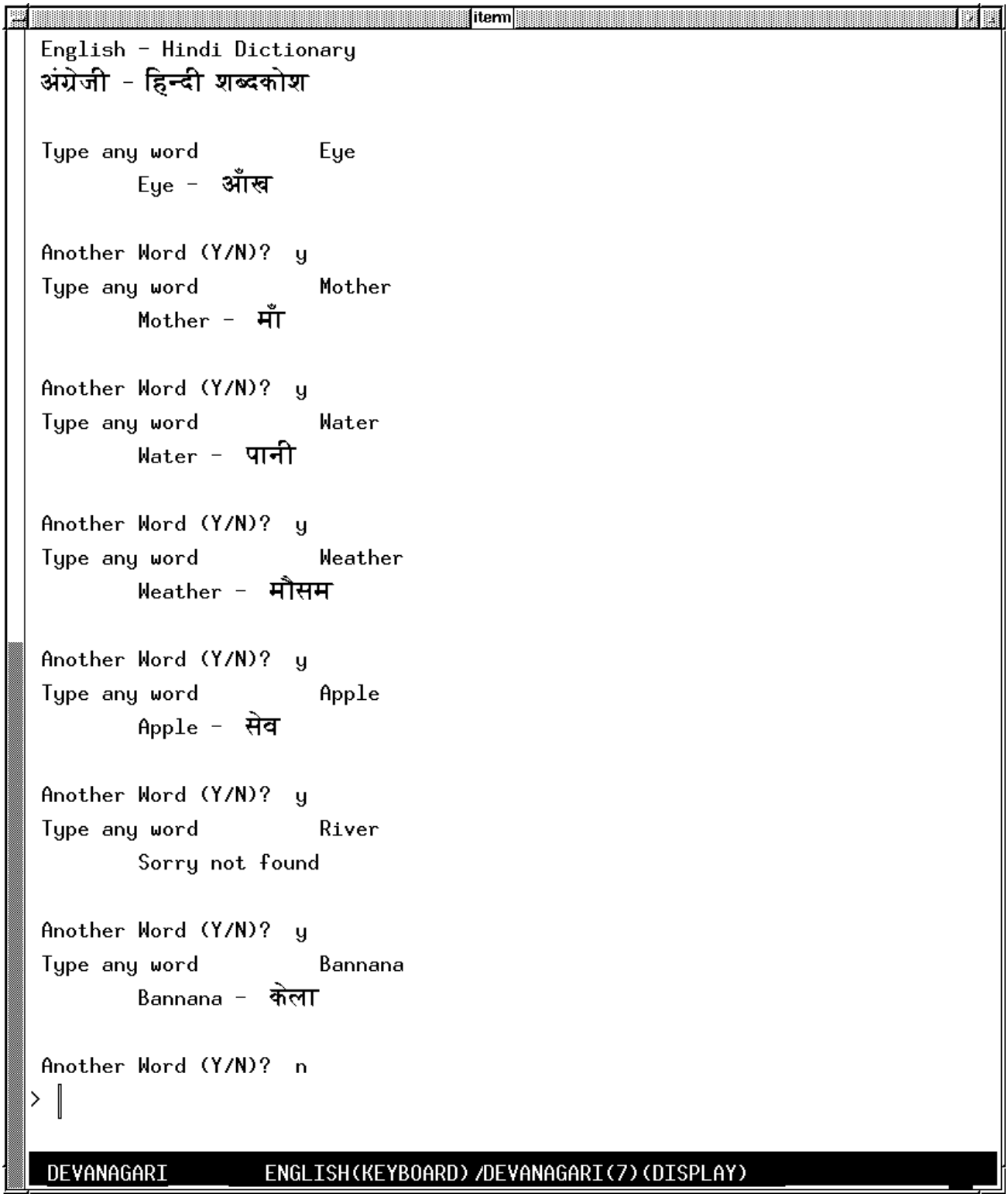
एक और पाकिस्तान का सेनाध्यक्ष लड़ाई न लड़ने का अहसास कर रहा था । और दूसरी तरफ वहां के टेलिविजन पर दिखाया जा रहा था कि पाकिस्तानी सेना जोधपुर, बीकानेर और मैरठ तक में घुस चुकी है। दिल्ली फतह करीब है । दो देशों के मध्य युद्ध और तनाव के माहौल में एक पक्ष जनता भी होती है । इस पक्ष का भी नजरिया दूसरा था । पाकिस्तानी जनता का सोच था कि भारत-पाक के प्रत्यक्ष युद्ध में सोवियत संघ और अमेरिका परोक्ष लड़ाई के मैदान मैं है । तब की दोनों महाशक्तियां भारत-पाक को मोहरा बनाए हुए थे ।

```
/*  अंग्रेजी-हिन्दी.c
    यह प्रोग्राम English और  हिन्दी  मे लिखा गया है
    It generates the hindi equivalent word of English.
    The database is read in from "शब्दकोश"
*/
#include<stdio.h>
#define No_Char 20
struct _table{
  char eng[No_Char];
  char hindi[No_Char];
};
main()
{
  FILE *fp;
  int j,len;
  char ch,str[No_Char];
  struct _table *table;

  fp = fopen("शब्दकोश" , "r");
  if (fp == NULL) {
  printf("Unable to open file शब्दकोश");
  exit(-1);
  }
  fscanf(fp,"%d",&len);
  if (len > 0) {
          table = (struct _table *) malloc( sizeof(struct _table) * len);
          if (table == NULL) {
                  printf("unable  to allocate memory - exiting \n");
                  exit(-1);
          }
  }
  else {
```

```
                printf("unable  to allocate memory - exiting \n");
                exit(-1);
          }
  }
  else {
       printf("Sorry, no words in the databse - exiting");
       exit(-1);
      }
 system("clear");
  for(j = 0 ; j < len ; j++)
    fscanf(fp,"%s%s",table[j].eng, table[j].hindi);
 printf(" English - Hindi Dictionary \n");
 printf(" अंग्रेजी - हिन्दी शब्दकोश \n\n");
while(1){
 fflush(stdin);
 printf(" Type any word \t \t"); scanf("%s",str);
 for(j = 0 ; j < len ; j++) {
  if (!strcmp(table[j].eng,str)) {
               printf("\t %s -  %s \n",table[j].eng,table[j].hindi);
               break;
   }
 }
 if (j == len) printf("\t Sorry not found \n");
 printf("\n Another Word (Y/N)?  ");
while(1){
 fflush(stdin); scanf("%c",&ch);
 if (ch == 'n' || ch == 'N') exit(1);
 if ( ch =='y' || ch =='Y') break;
}
 }
}

> cc -o अंग्रेजी-हिन्दी अंग्रेजी-हिन्दी.c
 DEVANAGARI       ENGLISH(KEYBOARD)/DEVANAGARI(7)(DISPLAY)
```


iterm
English - Hindi Dictionary
अंग्रेजी - हिन्दी शब्दकोश
Type any word Eye
Eye - आँख
Another Word (Y/N)? y
Type any word Mother
Mother - माँ
Another Word (Y/N)? y
Type any word Water
Water - पानी
Another Word (Y/N)? y
Type any word Weather
Weather - मौसम
Another Word (Y/N)? y
Type any word Apple
Apple - सेव
Another Word (Y/N)? y
Type any word River
Sorry not found
Another Word (Y/N)? y
Type any word Bannana
Bannana - केला
Another Word (Y/N)? n
DEVANAGARI ENGLISH(KEYBOARD)/DEVANAGARI(7)(DISPLAY)

Chapter 5

Conclusion

The `iterm` is a modified `xterm`, which in addition provides I/O support for text composed in various Brahmi-based Indian scripts. Also `iterm` supports both fixed and variable size fonts. Hence the text can be displayed using any font style, which was not possible in `xterm`, as it had support for only fixed width fonts.

The `iterm` has been designed to support all the Brahmi-based Indian languages, but, due to fonts being inaccessible, it has only been configured to support Devanagari scripts. However, the user can configure `iterm` to support any other Brahmi-based Indian languages. The `iterm` provides the flexibility to define ones own coding scheme and keyboard mapping. The default coding scheme used by `iterm` are ISSCII-8 and EA-ISCII. Inscript keyboard overlay has been supported. Any font can be used for display of Indian script characters.

The default coding scheme and keyboard mapping can be overridden by modifying the configuration files. One of the configuration files contains the font table, which is to be supplied with every font. The configuration files also contains the rules for composition of characters, which can be redefined. These values are read in at the run time. These configuration files makes the implementation of `iterm` independent of any coding scheme, keyboard mapping, fonts and character composition rules.

Keyboard and display driver of `xterm` has been extended to work with Indian scripts. The keyboard driver handles the input of Indian scripts, and English characters. The input handling requires keyboard mapping of characters to codes. The display driver deals with display of text in any one of the selected scripts. Character composition of the input characters are performed before displaying them. It also allows scrolling and manipulation of entered text. Cut and paste features are also supported by the display driver.

All the application programs which support 7/8 bit coding schemes can run on `iterm`. It allows variable width fonts to be used due to which the number of characters that can be displayed per row actually varies. This causes certain difficulties in vi which allows only a fixed number of characters to be displayed per row.

5.1 Future work

The `iterm` has not incorporated the ATR and EXT characters provided in ISCII coding scheme. It can be extended to support the above characters, thus allowing free mixing of all the Indian scripts and further extension of character set.

The Indian scripts are a derivative of ancient Brahmi and Perso-Arabic scripts. Various Perso-Arabic based Indian scripts Urdu, Sindhi and Kashmiri differ from Brahmi-based Indian scripts as they are written from right to left. The `iterm` only supports the input and output of Brahmi-based Indian scripts. It can further be extended to provide similar support for Perso-Arabic based scripts.

Only screen input and output of Indian scripts and English is provided by `iterm`. Printing of Indian languages is currently not supported by `iterm`. The `iterm` could be extended to print various scripts in graphics mode on a variety of printers.

Existing editors such as vi running under Digital Unix do not support eight bit input and output. Also due to the variable width of the font the text when written in vi

is not justified properly. There is a need for an editor which would allow the entry of eight bit codes and which would not assume fixed number of characters per row.

Appendix A

Control sequences

Appendix A lists the control sequences provided by a VT100 terminal and the control sequences supported by `iterm`. It also enumerates control sequences generated by `iterm` when special keys are pressed.

A.1 DEC VT100 features

A.1.1 ANSI compatible mode

- *Character attributes*

ESC [Ps;Ps;Ps;...,Ps m

Ps = 0 or None	All Attributes Off
1	Bold on
4	Underscore on
5	Blink on
7	Reverse video on

Cursor movement commands

ESC [Pn A	Cursor Up
ESC [Pn B	Cursor Down
ESC [Pn C	Cursor Right
ESC [Pn D	Cursor Left
ESC [Pl;Pc H	Cursor Position
ESC [Pl;Pc f	Cursor Position
ESC [H	Cursor Home
ESC D	Index
ESC M	Reverse Index
ESC E	Next Line
ESC 7	Save Cursor and Attributes
ESC 8	Restore Cursor and Attributes

- Pn = decimal parameter in string of ASCII digits.(default 1)
- Pl = line number (default 0)
- Pc = column number (default 0)

Line size (double-height and double-width) commands

ESC # 1	Change this line to single-width, double-height top half
ESC # 2	Change this line to single-width, double-height bottom half
ESC # 3	Change this line to double-width, double-height top half
ESC # 4	Change this line to double-width, double-height bottom half
ESC # 5	Change this line to single-width, single-height
ESC # 6	Change this line to double-width, single-height

Erasing

ESC [K	From cursor to end of line
ESC [0 K	From cursor to end of line
ESC [1 K	From beginning of line to cursor
ESC [2 K	Entire line containing cursor
ESC [J	From cursor to end of screen
ESC [0 J	From cursor to end of screen
ESC [1 J	From beginning of screen to cursor
ESC [2 J	Entire screen

Character set

G0	G1	G2	G3	Ch. Set
ESC(A	ESC)A	ESC*A	ESC+A	UK ASCII
ESC(B	ESC)B	ESC*B	ESC+B	US ASCII
ESC(0	ESC)0	ESC*0	ESC+0	DEC Special graphics
ESC(1	ESC)1	ESC*1	ESC+1	Alternate character Rom
ESC(2	ESC)2	ESC*2	ESC+2	Alternate character Rom special graphics character

The character set G0, G1, G2 and G3 are invoked by following sequences.

Control name		Coding	Function
LS0	Lock Shift G0	SI	Invokes G0 into GL (default)
LS1	Lock Shift G1	SO	Invokes G1 into GL
LS1R	Lock Shift G1 Right	ESC ~	Invokes G1 into GR
LS2	Lock Shift G2	ESC n	Invokes G2 into GL
LS2R	Lock Shift G2 Right	ESC }	Invokes G2 into GR (default)
LS3	Lock Shift G3	ESC o	Invokes G3 into GL
LS3R	Lock Shift G3 Right	ESC |	Invokes G3 into GR
SS2	Single Shift G2	SS2	Invokes G2 into GL
		ESC N	for the next graphics character
SS3	Single Shift G3	SS3	Invokes G3 into GL
		ESC O	for the next graphics character

Scrolling region

ESC [Pt ; Pb r

Pt is the number of the top line of the scrolling region;

Pb is the number of the bottom line of the scrolling region and must be greater than Pt.

(The default for Pt is line 1, the default for Pb is the end of the screen)

TAB stops

ESC H	Set tab at current column
ESC [g	Clear tab at current column
ESC [0 g	Clear tab at current column
ESC [3 g	Clear all tabs

Modes

	To Set		To Reset	
Mode Name	Mode	Sequence	Mode	Sequence
Insert/Replace	Insert	ESC [4h	Replace	ESC [4l
Line feed/new line	New line	ESC [20h	Line feed	ESC [20l
Cursor key mode	Application	ESC [?1h	Cursor	ESC [?1l
ANSI/VT52 mode	ANSI	ESC <	VT52	ESC [?2l
Column mode	132 Col	ESC [?3h	80 Col	ESC [?3l
Scrolling mode	Smooth	ESC [?4h	Jump	ESC [?4l
Screen mode	Reverse	ESC [?5h	Normal	ESC [?5l
Origin mode	Relative	ESC [?6h	Absolute	ESC [?6l
Wraparound	On	ESC [?7h	Off	ESC [?7l
Auto repeat	On	ESC [?8h	Off	ESC [?8l
Cursor Type	Block	ESC [?20h	Line	ESC [?20l
Cursor Enable	On	ESC [?25h	Off	ESC [?25l
Interlace	On	ESC [?9h	Off	ESC [?9l
Graphic proc. option	On	ESC 1	Off	ESC 2
Keypad mode	Application	ESC =	Numeric	ESC >

Editing functions

ESC [Pn P Delete Characters
ESC [Pn L Insert Lines
ESC [Pn M Delete Lines
ESC [Pn @ Insert Blank Characters

Reports

Cursor position report

Invoked by	ESC [6 n
Response is	ESC [Pl; Pc R

⋆ Pl = line number;

⋆ Pc = column number;

Status report

Invoked by	ESC [5 n
Response is	ESC [0 n (terminal ok)
	ESC [3 n (terminal not ok)

What are you

Invoked by	ESC [c or ESC [O c	
Response is	ESC [?1 ; Ps C	
	Ps =	0 Base VT100, no options
		1 Processor option (STP)
		2 Advanced Video option (AVO)
		3 AVO and STP
		4 Graphics processor option (GO)
		5 GO and STP
		6 GO and AVO
		7 GO, STP, and AVO

⋆ Alternately invoked by ESC Z (not recommended.) Response is the same.

Reset

ESC c

- *Screen alignment*

ESC # 8 Fill Screen with "Es"

A.1.2 VT52 compatible mode

ESC A	Cursor Up
ESC B	Cursor Down
ESC C	Cursor Right
ESC D	Cursor Left
ESC F	Select Special Graphics character set
ESC G	Select ASCII character set
ESC H	Cursor to home
ESC I	Reverse line feed
ESC J	Erase to end of screen
ESC K	Erase to end of line
ESC Y line column	Direct cursor address (see note 1)
ESC Z	Identify (see note 2)
ESC =	Enter alternate keypad mode
ESC >	Exit alternate keypad mode
ESC <	Enter ANSI mode

- NOTE 1: Line and column numbers for direct cursor address are single character codes whose values are the desired number plus 37 (in Octal). Line and column numbers start at 1.
- NOTE 2: Response to ESC Z is ESC / Z.

A.2 `iterm` control sequences

A.2.1 VT102 mode

Most of these control sequences are standard VT102 control sequences, but some sequences from later DEC VT terminals are also present. VT102 features not supported are smooth scrolling, double size characters, blinking characters, and VT52 mode. There are additional control sequences to provide `iterm`-dependent functions, like the scrollbar or window size.

Control sequences for character attributes, cursor movement commands, erasing, scrolling region, tab stops, editing functions, reset, screen alignment are the same as VT100 control sequences. Bold characters are drawn on receiving control sequences for blinking attribute.

- *Character set*

G0	G1	G2	G3	Ch. Set
ESC(A	ESC)A	ESC*A	ESC+A	UK ASCII
ESC(B	ESC)B	ESC*B	ESC+B	US ASCII
ESC(0	ESC)0	ESC*0	ESC+0	DEC Special graphics
ESC(1	ESC)1	ESC*1	ESC+1	EA-ISCII
ESC(2	ESC)2	ESC*2	ESC+2	ISSCII-8

The character set G0, G1, G2 and G3 are invoked by the same sequence as specified for VT100 terminals.

Modes

Control sequences to set various modes - insert/replace, line feed/new line, keypad mode, cursor key mode, column mode, scrolling mode, screen mode, origin mode, wraparound and auto-repeat, are the same as that for VT100 terminal.

Dec private mode

ESC [? Pm h - Set (DECSET)

Ps =	2	Designate US ASCII for character sets G0-G3
	9	Send Mouse X & Y on button press
	3 8	Enter Tektronix mode
	4 0	Allow 80 ↔ 132 Mode
	4 1	more(1) fix (see curses resource)
	4 4	Turn on Margin Bell
	4 5	Reverse-wraparound mode
	4 6	start logging
	4 7	use alternate screen buffer
	1 0 0 0	send mouse x & y on button press and release
	1 0 0 1	Use Hilite Mouse Tracking

ESC [? Pm l - Reset (DECRST)

Ps =	9	Don't Send Mouse X & Y on button press
	4 0	Disallow 80 ↔ 132 Mode
	4 1	No more(1) fix (see curses resource)
	4 4	Turn off Margin Bell
	4 5	No Reverse-wraparound mode
	4 6	Stop logging
	4 7	use normal screen buffer
	1 0 0 0	Don't send mouse x & y on button press and release
	1 0 0 1	Don't Use Hilite Mouse Tracking

ESC [? Pm r - Restore DEC Private Mode Values. Value of Ps previously stored is retrieved. Ps values are the same as above.

ESC [? Pm s - Save DEC Private Mode Values. Ps values are the same as above.

Reports

The control sequences for obtaining cursor position and status reports are the same as VT100 control sequences. Even the terminal identification control sequence is the same except that the `iterm` responds by
ESC [?1 ; Ps C
Ps = 2 Base VT100, Advanced Video option (AVO)

Terminal parameters

Request	ESC [Ps x

Miscellaneous control sequences

ESC] 0 ; Pt BEL	Change Icon Name and Window Title to Pt
ESC] 1 ; Pt BEL	Change Icon Name to Pt
ESC] 2 ; Pt BEL	Change Window Title to Pt
ESC] 4 6 ; Pt BEL	Change Log file to Pt
ESC] 5 0 ; Pt BEL	Set Font to Pt
ESC l	Memory Lock (per HP terminals)
ESC m	Memory unlock (per HP terminals)
ESC [Ps;Ps;Ps;Ps;Ps T	Initiate hilite mouse tracking. parameters are func; startx; starty; firstrow; lastrow

- Ps - A single (usually optional) numeric parameter, composed of one of more digits.
- Pm - A multiple numeric parameter composed of any number of single numeric parameters, separated by ; character(s).
- Pt - A text parameter composed of printable characters.

A.2.2 Mouse tracking

The VT widget can be set to send the mouse position and other information on button presses. These modes are typically used by editors and other full-screen applications that want to make use of the mouse. There are three mutually exclusive modes, each enabled (or disabled) by a different parameter in the DECSET (or DECRST) escape sequence. Parameters for all mouse tracking escape sequences generated by `iterm` encode numeric parameters in a single character as value+040.

X10 compatibility mode sends an escape sequence on button press encoding the location and the mouse button pressed. It is enabled by specifying parameter 9 to DECSET. On button press, `iterm` sends

ESC [M CbCxCy

- Cb is button-1.
- Cx and Cy are the x and y coordinates of the mouse when the button was pressed.

Normal tracking mode sends an escape sequence on both button press and release. Modifier information is also sent. It is enabled by specifying parameter 1000 to DECSET. On button press or release, `iterm` sends

ESC [M CbCxCy

- The low two bits of Cb encode button information: 0=MB1 pressed, 1=MB2 pressed, 2=MB3 pressed, 3=release.
- The upper bits encode what modifiers were down when the button was pressed and are added together. 4=Shift, 8=Meta, 16=Control.

- Cx and Cy are the x and y coordinates of the mouse event. The upper left corner is (1,1).

Mouse hilite tracking notifies a program of a button press, receives a range of lines from the program, highlights the region covered by the mouse within that range until button release, and then sends the program the release coordinates. It is enabled by specifying parameter 1001 to DECSET. On button press, the same information as for normal tracking is generated; `iterm` then waits for the program to send mouse tracking information. All X events are ignored until the proper escape sequence is received from the pty:

ESC [Ps; Ps; Ps; Ps; Ps T

The parameters are func, startx, starty, firstrow, and lastrow. func is non-zero to initiate hilite tracking and zero to abort. startx and starty give the starting x and y location for the highlighted region. The ending location tracks the mouse, but will never be above row firstrow and will always be above row lastrow. (The top of the screen is row 1.) When the button is released, `iterm` reports the ending position one of two ways:

ESC [t CxCy - if the start and end coordinates are valid text locations.
ESC [T CxCyCxCyCxCy - if either coordinate is past the end of the line.

The parameters are startx, starty, endx, endy, mousex, and mousey. startx, starty, endx, and endy give the starting and ending character positions of the region. mousex and mousey give the location of the mouse at button up, which may not be over a character.

A.2.3 Tektronix 4014 mode

Most of these sequences are standard Tektronix 4014 control sequences. Graph mode supports the 12-bit addressing of the Tektronix 4014. The major features missing are the write-thru and defocused modes. The control sequences listed below do not describe the commands used in the various Tektronix plotting modes but does describe the commands to switch modes.

BEL	Bell (Ctrl-G)
BS	Backspace (Ctrl-H)
TAB	Horizontal Tab (Ctrl-I)
LF	Line Feed or New Line (Ctrl-J)
VT	Cursor up (Ctrl-K)
FF	Form Feed or New Page (Ctrl-L)
CR	Carriage Return (Ctrl-M)
ESC ETX	Switch to VT100 Mode (ESC Ctrl-C)
ESC ENQ	Return Terminal Status (ESC Ctrl-E)
ESC FF	PAGE (Clear Screen) (ESC Ctrl-L)
ESC SO	Begin 4015 APL mode (ignored by `iterm`) (ESC Ctrl-N)
ESC SI	End 4015 APL mode (ignored by `iterm`) (ESC Ctrl-O)
ESC ETB	COPY (Save Tektronix Codes to file COPYyy-mm-dd.hh:mm:ss)
(ESC Ctrl-W)	
ESC CAN	Bypass Condition (ESC Ctrl-X)
ESC SUB	GIN mode (ESC Ctrl-Z)
ESC FS	Special Point Plot Mode (ESC Ctrl-\)
ESC 8	Select Large Character Set
ESC 9	Select #2 Character Set
ESC :	Select #3 Character Set
ESC ;	Select Small Character Set

ESC] Ps ; Pt BEL	Set Text Parameters of VT window
	Ps = 0 → Change Icon Name and Window Title to Pt
	Ps = 1 → Change Icon Name to Pt
	Ps = 2 → Change Window Title to Pt
	Ps = 4 6 → Change Log File to Pt
ESC ‘	Normal Z Axis and Normal (solid) Vectors
ESC a	Normal Z Axis and Dotted Line Vectors
ESC b	Normal Z Axis and Dot-Dashed Vectors
ESC c	Normal Z Axis and Short-Dashed Vectors
ESC d	Normal Z Axis and Long-Dashed Vectors
ESC h	Defocused Z Axis and Normal (solid) Vectors
ESC i	Defocused Z Axis and Dotted Line Vectors
ESC j	Defocused Z Axis and Dot-Dashed Vectors
ESC k	Defocused Z Axis and Short-Dashed Vectors
ESC l	Defocused Z Axis and Long-Dashed Vectors
ESC p	Write-Thru Mode and Normal (solid) Vectors
ESC q	Write-Thru Mode and Dotted Line Vectors
ESC r	Write-Thru Mode and Dot-Dashed Vectors
ESC s	Write-Thru Mode and Short-Dashed Vectors
ESC t	Write-Thru Mode and Long-Dashed Vectors
FS	Point Plot Mode (Ctrl-\)
GS	Graph Mode (Ctrl-])
RS	Incremental Plot Mode (Ctrl-^)
US	Alpha Mode (Ctrl- _)

A.2.4 Keyboard

- *Numeric keypad*

	Numeric	Application
0	0	ESC O p
1	1	ESC O q
2	2	ESC O r
3	3	ESC O s
4	4	ESC O t
5	5	ESC O u
6	6	ESC O v
7	7	ESC O w
8	8	ESC O x
9	9	ESC O y
-	-	ESC O m
,	,	ESC O l
.	.	ESC O n
Enter	CR/CR LF	ESC O M
PF1	ESC O P	ESC O P
PF2	ESC O Q	ESC O Q
PF3	ESC O R	ESC O R
PF4	ESC O S	ESC O S

- *Cursor control keys*

	Cursor	Application
Up	ESC [A	ESC O A
Down	ESC [B	ESC O B
Right	ESC [C	ESC O C
Left	ESC [D	ESC O D

- *Editing keypad*

Key	Code
Find	ESC [1 ~
Insert	ESC [2 ~
Remove	ESC [3 ~
Select	ESC [4 ~
Prev Screen	ESC [5 ~
Next Screen	ESC [6 ~

- *Function keys*

Key	Code	Code(Sun)
F1	ESC [11 ~	ESC [224 ~
F2	ESC [12 ~	ESC [225 ~
F3	ESC [13 ~	ESC [226 ~
F4	ESC [14 ~	ESC [227 ~
F5	ESC [15 ~	ESC [228 ~
F6	ESC [17 ~	ESC [229 ~
F7	ESC [18 ~	ESC [230 ~
F8	ESC [19 ~	ESC [231 ~
F9	ESC [20 ~	ESC [232 ~
F10	ESC [21 ~	ESC [233 ~
F11	ESC [23 ~	ESC [192 ~
F12	ESC [24 ~	ESC [193 ~
F13	ESC [25 ~	ESC [194 ~
F14	ESC [26 ~	ESC [195 ~
F15	ESC [28 ~	ESC [196 ~
F16	ESC [29 ~	ESC [197 ~
F17	ESC [31 ~	ESC [198 ~
F18	ESC [32 ~	ESC [199 ~
F19	ESC [33 ~	ESC [200 ~
F20	ESC [34 ~	ESC [201 ~

Appendix B

Code

This appendix contains the popular coding schemes used for internal representation of English and Indian scripts.

US ASCII code is a 7-bit code with 32 "control characters" and 96 "graphics characters". UK ASCII is the same as US ASCII except that the dollar sign is replaced by pound sign. The Dec Special Graphics character set is the same as ASCII character set except for the characters between 0x5f and 0x7e which are special line drawing characters.

ISCII (Indian Standard Code for Information Interchange) was standardized by DOE. The ISCII code contains only the basic alphabet required by the Indian scripts. All the composite characters are formed through combination of these basic characters. Immediate transliteration between different Indian scripts is possible, just by changing the display modes. In addition to the alphabets of Indian scripts the ISCII character set also contains the INV, ATR and the EXT code. INV character is used as a *consonant* and is used to for formation of composite characters which requires a consonantal base. ATR character followed by a displayable ASCII character, defines a font attribute applicable for the following characters. EXT followed by an ISCII character, defines a new character which can combine with previous ISCII character.

ISSCII-8 is a 8-bit code with lower 128 characters of the table containing the ASCII character set. ISSCII-7 bit code is meant for ISO compatible 7/8 bit environment, and 94 positions in the ASCII table is replaced by characters from ISCII character set. EA-ISCII is also a 7 bit code, however, it allows the mixing of Roman characters with Indian scripts. 'x' at the beginning of the word denotes the word in Indian script. In EA-ISCII 'x' is interpreted as follows:

- *double x* - xx is displayed as x. It is required for writing an English word beginning with x.
- *standalone x* - x which is preceded and followed by space or non alphabet, shows up as x rather than nukta.

In the table the rightmost characters are formed by appending *nukta* to the corresponding Indian script character shown in the middle of the column.

B.1 ASCII 7-bit code

B.2 DEC special graphics

B.3 Indian script alphabet

B.4 ISSCII-8 code

B.5 ISSCII-7 code

B.6 EA-ISCII code

B.7 ATR chart

Appendix C

Inscript keyboard

The Inscript (Indian Script) keyboard overlay was standardized by DOE. It can be used on any QWERTY keyboard. The Indian script legends are shown on the right hand side of the key, as the left hand side has the English legends. It contains characters required for all the Indian scripts, as defined by ISCII character set. The overlay has been optimized from phonetic/frequency considerations. It is divided in two parts: the vowel pad on the left hand side, and the consonant pad on the right hand side.

Due to the phonetic/alphabetic nature of the keyboard, a person who knows typing in one Indian script can type in any other Indian script. The logical structure allows ease in learning, while the frequency considerations allow speed in touch typing. The keyboard remains optimal both from touch-typing and sight-typing points of view, in all Indian scripts.

Appendix D

User manual

The `iterm` is an X11R5-based VT102 and Tektronix 4014 terminal emulator, which supports the input and output of Indian and English scripts. It is an extension of `xterm`, and most of the functions are the same as original `xterm´s`, however, it has capabilities of displaying and entering text in Indian scripts, if compiled with -DITERM option. It also provides a status line, where relevant details are displayed.

Command to run `iterm`:

`iterm` [-toolkit option....] [-option]

The `iterm` has been designed to support all Brahmi-based Indian scripts - Devanagari, Punjabi, Gujarati, Oriya, Bengali, Assamese, Telugu, Kannada, Malayalam and Tamil. However, it is only configured to provide I/O of Devanagari scripts. User can configure `iterm` to support other Indian languages by making changes in the specification file. The default name of specification file is `specs` and should be present in `./config` directory.

It also supports variable width fonts in addition to fixed width fonts. Due to this, the text can be viewed in any font style, which was not possible in `xterm` as it supported only fixed width fonts.

The user can select between English and Indian scripts by pressing the various function keys, which can be customized (refer section on binding keys). The keyboard and display are independent of each other. `F1` and `F3` switches the keyboard and display mode respectively between English and Indian scripts. In addition it allows both 7 and 8 bit coding for Indian scripts. `F2` and `F4` changes between 7 and 8 bit coding (of Indian scripts) for keyboard and display respectively.

Termcap entries that work with `iterm` include "`iterm`", "`xterm`", "`vt102`", "`vt100`", and "`ansi`". The `iterm` automatically searches the termcap file in this order for these entries and then sets the "TERM" and "TERMCAP" environment variables.

D.1 Coding schemes

The English text is coded in ASCII. To allow all the existing applications to run, both 7 and 8 bit coding for Indian scripts are supported. The default coding schemes provided for Indian script are ISSCII-8 (Indian Script Standard Code for Information Interchange) and EA-ISCII (English Alphabet ISCII). However the user can specify his own 7 and 8 bit coding schemes. These details are to be provided in coding scheme file, the format for which is discussed below in the section of file formats.

Generally it is desirable to mix Indian script with English. For this purpose the 7-bit coding schemes have an escape character to switch between the two languages. However this escape sequence is optional.

D.2 Keyboard

Ordinary QWERTY keyboard is supported, with an overlay for Indian Script characters. Normally the keys of the QWERTY pad will generate English characters, but on pressing `F1` function key the Indian script characters will be generated. `F1` is

a toggle key, which selects the entry of Indian script or English characters. Inscript keyboard overlay as recommended by DOE is supported. However the keyboard can be mapped according to user's convenience. This mapping is to be provided in keyboard map file, the format of which is specified below. Also the user can choose between 7 or 8 bit coding for Indian scripts, by pressing of F2 function key (toggle key). The user can assign the functions of `F1` and `F2` keys to some other keys, by specifying in the resource database. See the section on binding keys for more details.

D.3 Display

The display can be set to show the text in Indian or English language. Pressing of `F3` key switches the display mode between Indian script and English. The `F4` function key can be used to select between 7 and 8 bit coding for display of Indian scripts.

D.3.1 Character sets

Special control sequences can also be sent to set the display. This can be done by setting G0, G1, G2, and G3 to the appropriate character set and invoking these into GL and GR group. The character sets supported by `iterm` are listed in Appendix A. The function keys `F3` and `F4` actually sets the current set (G0, G1, G2, G3) to the corresponding character (EA-ISCII/ISSCII-8) set. For mapping this functionality to some other keys, see the section on binding keys.

D.3.2 Display problems

Due to support of variable width font, the number of characters that can be displayed in a row cannot be determined. The maximum number of characters that

is supported to be displayed per row is equal to the width of the row. There are certain applications, like `vi`, which require to know about the number of characters that can be displayed in a row. These applications send only the specified number of characters to be displayed in a row. Due to this a sentence which can be completely displayed in the same row may be split over two rows. Or if large number of characters are specified then `iterm` may wrap it to the next line though the application may still think that it is displayed in the same row. This can cause editing problems. So to support these applications the number of characters to be displayed per row has to be judiciously chosen.

For calculating this, a font display symbol is considered to be a base character and the width of the screen divided by the width of the base character is assumed to be the number of characters that can be displayed per row. The user can choose this character and specify it in the font map file, format of which is specified in the section of file formats. The 7 bit coding generally contains an escape character for switching between the Roman and English text, which is normally not shown on the screen. Due to this additional character, the actual number of characters which are to be displayed in the same row under 7 bit mode should be more than when 8-bit code is used. The users can specify different base character for both seven and eight bit mode.

D.4 Cursor

Cursor is displayed as a block surrounding the current character. There is a horizontal cursor which displays the logical position of the cursor. Whenever cursor is placed on a composite character, the actual character on which it is placed is shown on the status line.

D.5 Fonts

It supports both fixed width and variable width fonts. For viewing the Indian scripts any font can be specified. The fonts are not required to follow any standard. However the user has to specify the font table corresponding to that font in font map file, the format of which is given in the section of file formats.

D.6 Indian scripts

In total ten Indian scripts can be supported at a time. Dynamically, one can switch between the Indian scripts by selecting through the menu, which appears by pressing the following sequence: `Ctrl + 3rd` button. The default Indian script and the details of the Indian scripts can be specified in the specification file.

Each script can have its own coding, keyboard mapping and composition rules. Generally these will not change for different Indian scripts but one can use this feature to his own advantage. The user can specify two different languages say Devanagari1 and Devanagari2, both of which may have different keyboard mappings and coding schemes, but which actually represents the same script. So, if a user has files in two different coding scheme or the users wants to enter files using different keyboard mapping (different users may prefer different keyboard mapping), then he can dynamically switch between the two, just by clicking on the appropriate language in the menu. However, in this case the user will be able to use only 8 other Indian languages.

D.6.1 Syntax of Indian scripts

The Indian script characters can only be combined according to some rules. Every word consists of a number of syllables. To identify a correct syllable there are certain

rules, which are to be specified by the user. The default rule provided is as follows :

```
Word                 ::= {Syllable}[Cons-Syllable]
Syllable             ::= Cons-Vowel-Syllable | Vowel-Syllable
Vowel-Syllable       ::= Vowel[Modifiers]
Cons-Vowel-Syllable  ::= [Cons-Syllable]Full-Cons[Matra][Modifiers]
Cons-Syllable        ::= [Pure-Cons][Pure-Cons]Pure-Cons
Pure-Cons            ::= Full-Cons Halant
Full-Cons            ::= Consonant [Nukta]

Following conventions are used in the syntax given above :
::=  defines a relation.
{}   encloses items which may be repeated one or more times.
[]   encloses items which may or may not be present.
|    separates items, out of which only one can be present.
```

The above representation is in Backus Norm Form, however the users will have to represent these rules using some other representation, the details of which could be obtained in section on file formats. In the above syntax *nukta* can only combine with certain characters.

According to above rules if an invalid symbol is found then it is preceded by an "INV" (Invisible) character. This "INV" character should be present in the coding scheme supported.

D.7 Options

The `iterm` terminal emulator accepts all of the options supported by `xterm`, and in addition provides the following command line options.

- **-version:** This gives the `iterm` version.

- **-/+ se:** The escape sequence in 7 bit coding is shown or hidden.
- **-specifile:** The name of the specification file containing information about different Indian languages can be specified by this option. Complete path has to be provided.Default name of specification file is `./config/specs`.
- **-geometry:** It specifies the geometry in pixel width and height as opposed to `xterm` in which it denotes the character width and height.

For the various toolkit options and other options refer to `xterm` manual.

D.8 Resources

All the resource name and classes specified by `xterm` are supported. Besides these it understands the following resources:

- **showescseq** (*class* **ShowEscSeq**)**:** Specifies if the escape sequence is to be shown or hidden when 7 bit coding scheme is used for display of Indian scripts.
- **specfile** (*class* **SpecFile**)**:** It gives the name of the specification file containing information about different Indian languages supported. Complete path has to be specified.

D.9 Menu

It supports the menu provided by `xterm`, however, some extra information can be sent by the user to `iterm` with the help of these menus. The VT options menu contains an extra entry which can be used to specify whether the escape sequence in

7 bit mode is to be shown or hidden. The font menu contains 10 extra entries corresponding to 10 Indian languages supported by `iterm`. The users can dynamically choose between any of the languages. The name of the languages to be displayed in the menu can be specified by the user in the specification file. The user can also specify less than 10 languages.

D.10 Binding keys

By default, `F1` and `F2` function keys are used for selecting the keyboard mode, while F3 and F4 function keys are used to select the display mode. There are some functions to select these different modes. `change_mode_keyboard()` function changes the keyboard mode between English and the chosen Indian sctipt. Similarly `change_mode_display()` changes the display mode between English and the Indian script. `hindi_code_keyboard()` and `hindi_code_display()` functions change the keyboard and display coding respectively for Indian scripts between 7 and 8 bits. It is possible to rebind other keys to this action by changing the translation table. The default binding provided are:

```
~Meta <KeyPress>F1:change_mode_keyboard() \n\
~Meta <KeyPress>F2:hindi_code_keyboard() \n\
~Meta <KeyPress>F3:change_mode_display() \n\
~Meta <KeyPress>F4:hindi_code_display()
```

The keymap action can be used to add different keys for the above action. Following is the example for rebinding of the keys.

```
iterm*VT100.Translations: #override \
 ~Meta <KeyPress>F14: change_mode_keyboard() \n\
 ~Meta <KeyPress>F15: change_mode_display()
```

D.11 Configuration file

There is a main configuration file or the specification file, default name of which is `specs`, and it should be present in `./config` directory. However, with use of `-specfile` option or `specfile` resource the user can specify a different file. Besides specification file, there are other files which contains coding details, keyboard mapping, rules and font information. These files are listed in the specification file.

All the files follow a common format. Blank, newline and tabs are ignored. Comments can be present between "/*" and "*/". Every string should be followed by a blank and all the strings except where specified can have maximum of 15 characters. Rest of the string is ignored. % is a delimiter which should be present before starting of each new information. : and "– >" are used as delimiters.

Various symbols used to explain the format of files are:

- <> indicates that the value conforming to the description in these brackets be specified. The value may be in form of string, characters or decimal values.
- { } means that the value can occur 0 or more number of times.
- [ ] indicates that it is optional.

D.11.1 Specification file format

Specification file describes the Indian languages `iterm` should support. Default Indian script name is specified. Normal and bold font names for each language is also mentioned. Various files containing coding details, keyboard mapping, type map, rules and fonts are to be specified for each language. Two different languages can specify the same files.

```
%<Default language>

%<Language name> : <normal font name> <bold font name> :
                   <file which provides coding detail>
                   <file which provides keyboard map>
                   <file containing the font details>
                   <file which gives the type map>
                   <file containing the rules>

%<Language name> : <normal font name> <bold font name> :
                   <file which provides coding detail>
                   <file which provides keyboard map>
                   <file containing the font details>
                   <file which gives the type map>
                   <file containing the rules>          }
```

```
%Devanagari
%Devanagari   :dvng10   dvng10   :iscii   keybd   font1   type   rule1
%Gujarati     :gujr10   gujr10   :iscii   keybd   font2   type   rule2
%Tamil        :tam10    tam10    :iscii   keybd   font3   type   rule3
```

Table 8: Syntax - specification file

Maximum of ten languages can be specified. All the entries are in form of string of characters. The maximum number of characters present in the font name and files could be upto 100. The fonts specified should be loaded else the default font specified for English text is used for displaying the text in that script.

Absolute or relative filenames may be specified. If relative filename is specified, then path prefix from `specs` file is prepended to the filename. The absolute filename begins with \.

D.11.2 Coding scheme file format

The coding scheme file contains 7 and 8 bit coding details for Indian scripts. All the characters in the set are given some descriptive names. This name is used to refer to the character in all other files. For example:

Descriptive Name	Character
%Chandrabindu	ँ
%Visarg	:
%Aa	आ
%I	इ
%Ka	क
%Kha	ख

Table 9: Example - descriptive names for Indian script characters

Corresponding to each character its equivalent 7-bit and 8-bit coding is provided. For each character there can be only one 8 bit code, while in 7 bit coding maximum of 5 codes can form one character. INV is a reserved keyword and there should be some character both in 8 and 7 bit which represent "INV".

```
%[<Escape character >]

{%<string description for characters>   ->
 <8 bit coding in decimal>              ->
{<7 bit coding in form of characters>}    }
```

%x		/⋆ Escape character (7-bit) ⋆/
%Chandrabindu	$->$ 161	$->$ A
%Visarg	$->$ 163	$->$ B x
%Aa	$->$ 165	$->$ C k
%I	$->$ 166	$->$ C l
%Ka	$->$ 179	$->$ D
%Kha	$->$ 180	$->$ E

Table 10: Syntax - coding scheme file

The escape character may be specified for 7-bit coding which allows switching between English and Indian scripts. Decimal value is to be specified for 8 bit, however, 7-bit codes are specified in form of characters. For 8-bit coding the users can specify any number ranging from 128 to 253. For 7-bit coding the range within which the coding can be specified is from 33 to 126.

D.11.3 Keyboard map file format

The keyboard map can be specified by indicating the correspondence between the characters on the keyboard and the characters in the Indian script. One key can generate a number of characters in Indian script. This helps users to easily enter the most commonly used conjuncts. The keys are written in form of characters while the Indian script character to which it maps is entered in the form of string description which was specified in the file containing the coding details. Every key can generate a maximum sequence of 10 characters.

```
{%<Keyboard char> -> { <string description for characters> } }
```

%&	$->$ Ka	Halant	Hard-Sha
%#	$->$ Halant	Ra	
%H	$->$ Pha		
%I	$->$ Gha		
%X	$->$ Chandrabindu		
%D	$->$ A		

Table 11: Syntax - keyboard map file

D.11.4 Font map file format

Font table and font characters used for determining the number of characters per row

```
%{ <font display codes to be moved to the beginning> }
%{ <font display codes to be moved to the end> }

%<font display codes to be considered as base char - for 7 bit coding>
%<font display codes to be considered as base char - for 8 bit coding>

{ %<type name>
{ %{<string description for characters>} -> { <font coding in decimal>} } }
```

```
% 69                          /* Characters to be moved to the beginning */
% 13                          /* Characters to be moved to the end */

% 107                         /* Base character for 7 bit code */
% 107                         /* Base character for 8 bit code */

% Conjunct
%Ka             Halant  Hard-Sha  -> 35
%Ja             Halant  Jna       -> 43

%Vowel
% A                              -> 97
% Aa                             -> 97   65

%Consonant
% Ka                             -> 107
% Kha                            -> 75

%Half-Consonant
% Ka             Halant          -> 63
% Kha            Halant          -> 72

%Matra
% Matra-Aa                       -> 65
% Matra-i                        -> 69

%Reph
% Ra             Halant          -> 13
```

Table 12: Syntax - font map file

are specified in font map file. Also list of characters to be moved to the beginning or to the end of the syllable are present in font map file.

Fonts are basically categorized into various user defined types. Each category contains several mappings. To generate the display symbols in font, font table is searched for matching entries. "Conjunct" is a reserved keyword and all mappings specified under conjunct are first searched for. The user may edit the present font file to add more *conjuncts*. The categories defined here are used in listing all the combination rules, details of which are present in rules section.

D.11.5 Type map file format

Word written in any Indian script is composed of syllables. Syllables are a sequence of characters combined according to some rules. The user can provide the rules for finding valid syllables. The word is scanned for syllables and any symbol which does not form a part of valid syllable is preceded by an "INV" character.

To specify the rules first of all a type map is to be provided. This map categorizes the character set. For example the set may be categorized into *vowels, consonants, matras,* etc. A particular character not included in this file is assigned the default type specified by the user. Maximum of 50 different types can be present. "Begin" and "End" are reserved keywords and cannot be used. Refer to table 13 for the syntax of type_map file.

D.11.6 Rules file format

This file contains the syllable rules which lists the valid syllables. It also contains the combination rules, which specifies the mapping between input characters and output display symbols. The combination rules follow the syllable rules. "Begin"

```
 %<default type>
{%<type>        ->  <string description for character> }
```

%Invalid	
%Type_Vowel	$->$ A
%Type_Vowel	$->$ Aa
%Type_Vowel	$->$ I
%Type_Vowel	$->$ Ii
%Type_Consonant	$->$ Ka
%Type_Consonant	$->$ Kha
%Type_Consonant	$->$ Ga
%Type_Consonant	$->$ Gha
%Type_Cons-r	$->$ Ra
%Type_Modifier	$->$ Anuswar
%Type_Modifier	$->$ Chandrabindu
%Type_Modifier	$->$ Visarg
%Type_Matra	$->$ Matra-Aa
%Type_Matra	$->$ Matra-I
%Type_Matra	$->$ Matra-U
%Type_Halant	$->$ Halant

Table 13: Syntax - type map file

and "End" are reserved keywords and are used to denote the beginning and end of syllable or word.

Syllable rules

Now using the categories in type map file syllable rules can be specified. There are a number of rules, each having some name. User can specify all combination of categories representing valid syllables.

```
For example:
      R0: Type_Vowel Type_Modifier
      R1: Type_Vowel
```

Every type indicates one character of that type in the character set. If there can be combination of same type, then the type has to be written down required number of times. Some rules are very complicated. To specify these rules, the total combination specifying a syllable can be split in many rules. The user can specify that a particular rule does not indicate a valid syllable but it has to combine with some other rules to form a complete syllable. To specify this he states what all rules can follow this rule.

```
For example:
     R2:Type_Consonant Type_Halant -> R3 R4
     R3:End
     R4:Type_Consonant
```

If there are no rules present after $->$ then it indicates that the combinations present in that rule signifies a valid syllable.

```
{%<rule number>: {<character types>} -> {<rule number>} }
```

%R0	:Type_Vowel Type_Modifier	->
%R1	:Type_Vowel	->
%R2	:Type_Consonant Type_Halant	-> R3
%R3	:Type_Consonant Type_Halant	-> R4
%R4	:Type_Consonant Type_Halant	-> R5 R6 R7 R8
%R5	:End	->
%R6	:Type_Consonant Type_Matra Type_Modifier	->
%R7	:Type_Consonant Type_Matra	->
%R8	:Type_Consonant Type_Modifier	->
%R9	:Type_Consonant	->

Table 14: Syntax - syllable rules

- *Combination rules*

```
{%{ <character types> }   -> {<font display code types>}}
```

%Begin	Type_Cons-r	Type_Halant	End	– > Half-cons
%Begin	Type_Cons-r	Type_Halant		– > Reph
%Type_Halant	Type_Cons-r			– > Rkar
%Type_Consonant	Type_Halant	End		– > Consonant Halant
%Type_Consonant	Type_Halant			– > Half-cons
%Type_Vowel				– > Vowel
%Type_Modifier				– > Modifier
%Type_Matra				– > Matra
%Type_Consonant				– > Consonant
%Type_Numeral				– > Numeral
%Type_Punctuation				– > Punctuation
%Type_Halant				– > Halant
%Type_Nukta				– > Nukta

Table 15: Syntax - combination rules

The input string of characters combine in some way to form display symbols of certain category in the font table. These can be specified in form of combination rules. These rules basically states that these combination of character from various categories (specified in type map file) in the input string will generate the font codes belonging to certain categories (specified in font map file). This helps in determining the location in the font table where characters of that particular type are present. Also it helps to genertate the font characters according to the given context. The string which matches is replaced by font display code. Generation of font codes are generally context sensitive. This requirement of the language is met by specifying the combination rules.

%Type_Invalid		/* Default value if some character is not specified */
%Type_Modifier	->	Chandrabindu
%Type_Modifier	->	Anuswar
%Type_Modifier	->	Visarg
%Type_Vowel	->	A
%Type_Vowel	->	AA
%Type_Vowel	->	I
%Type_Vowel	->	II
%Type_Vowel	->	U
%Type_Vowel	->	UU
%Type_Vowel	->	RI
%Type_Vowel	->	E
%Type_Vowel	->	EY
%Type_Vowel	->	AI
%Type_Vowel	->	AYE
%Type_Vowel	->	O
%Type_Vowel	->	OW
%Type_Vowel	->	AU
%Type_Vowel	->	AWE
%Type_Cons-nukta	->	KA
%Type_Cons-nukta	->	KHA
%Type_Cons-nukta	->	GA
%Type_Consonant	->	GHA
%Type_Consonant	->	NGA
%Type_Consonant	->	CHA
%Type_Consonant	->	CHHA
%Type_Cons-nukta	->	JA
%Type_Consonant	->	JHA
%Type_Consonant	->	JNA
%Type_Consonant	->	HARD_TA
%Type_Consonant	->	HARD_THA
%Type_Cons-nukta	->	HARD_DA
%Type_Cons-nukta	->	HARD_DHA
%Type_Consonant	->	HARD_NA
%Type_Consonant	->	SOFT_TA
%Type_Consonant	->	SOFT_THA
%Type_Consonant	->	SOFT_DA
%Type_Consonant	->	SOFT_DHA
%Type_Consonant	->	SOFT_NA

Table 16: Default categories - type map file

%Type_Consonant	$->$	NA	%Type_Numeral	$->$	1
%Type_Consonant	$->$	PA	%Type_Numeral	$->$	2
%Type_Cons-nukta	$->$	PHA	%Type_Numeral	$->$	3
%Type_Consonant	$->$	BA	%Type_Numeral	$->$	4
%Type_Consonant	$->$	BHA	%Type_Numeral	$->$	5
%Type_Consonant	$->$	MA	%Type_Numeral	$->$	6
%Type_Consonant	$->$	YA	%Type_Numeral	$->$	7
%Type_Consonant	$->$	JYA	%Type_Numeral	$->$	8
%Type_Cons-r	$->$	RA	%Type_Numeral	$->$	9
%Type_Consonant	$->$	HARD_RA			
%Type_Consonant	$->$	LA			
%Type_Consonant	$->$	HARD_LA			
%Type_Consonant	$->$	ZHA			
%Type_Consonant	$->$	VA			
%Type_Consonant	$->$	SHA			
%Type_Consonant	$->$	HARD_SHA			
%Type_Consonant	$->$	SA			
%Type_Consonant	$->$	HA			
%Type_Cons-inv	$->$	INV			
%Type_Matra	$->$	MATRA_AA			
%Type_Matra	$->$	MATRA_I			
%Type_Matra	$->$	MATRA_II			
%Type_Matra	$->$	MATRA_U			
%Type_Matra	$->$	MATRA_UU			
%Type_Matra	$->$	MATRA_RI			
%Type_Matra	$->$	MATRA_E			
%Type_Matra	$->$	MATRA_EY			
%Type_Matra	$->$	MATRA_AI			
%Type_Matra	$->$	MATRA_AYE			
%Type_Matra	$->$	MATRA_O			
%Type_Matra	$->$	MATRA_OW			
%Type_Matra	$->$	MATRA_AU			
%Type_Matra	$->$	MATRA_AWE			
%Type_Halant	$->$	HALANT			
%Type_Nukta	$->$	NUKTA			
%Type_Punctuation	$->$	VIRAM			
%Type_Numeral	$->$	0			

Table 17: Default categories - type map file

%R0 :	Type_Vowel Type_Modifier	− >
%R1 :	Type_Vowel	− >
%R2 :	Type_Consonant Type_Halant R8 R9 R10 R11 R12 R13	− >
%R3 :	Type_Cons-nukta Type_Halant R8 R9 R10 R11 R12 R13	− >
%R4 :	Type_Cons-inv Type_Halant R8 R9 R10 R11 R12 R13	− >
%R5 :	Type_Cons-r Type_Halant R8 R9 R10 R11 R12 R13	− >
%R6 :	Type_Cons-nukta Type_Nukta Type_Halant R8 R9 R10 R11 R12 R13	− >
%R7 :	Type_Cons-inv Type_Nukta Type_Halant R8 R9 R10 R11 R12 R13	− >
%R8 :	Type_Consonant Type_Halant R14 R15 R16 R17 R18 R19	− >
%R9 :	Type_Cons-nukta Type_Halant R14 R15 R16 R17 R18 R19	− >
%R10 :	Type_Cons-inv Type_Halant R14 R15 R16 R17 R18 R19	− >
%R11 :	Type_Cons-r Type_Halant R14 R15 R16 R17 R18 R19	− >
%R12 :	Type_Cons-nukta Type_Nukta Type_Halant R14 R15 R16 R17 R18 R19	− >
%R13 :	Type_Cons-inv Type_Nukta Type_Halant R14 R15 R16 R17 R18 R19	− >
%R14 :	Type_Consonant Type_Halant R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31 R32 R33 R34 R35 R36 R37 R38 R39 R40 R41 R42 R43 R44	− >
%R15 :	Type_Cons-nukta Type_Halant R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31 R32 R33 R34 R35 R36 R37 R38 R39 R40 R41 R42 R43 R44	− >

Table 18: Default syllable rules

%R16 :	Type_Cons-inv Type_Halant R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31 R32 R33 R34 R35 R36 R37 R38 R39 R40 R41 R42 R43 R44	− >
%R17 :	Type_Cons-r Type_Halant R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31 R32 R33 R34 R35 R36 R37 R38 R39 R40 R41 R42 R43 R44	− >
%R18 :	Type_Cons-nukta Type_Nukta Type_Halant R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31 R32 R33 R34 R35 R36 R37 R38 R39 R40 R41 R42 R43 R44	− >
%R19 :	Type_Cons-inv Type_Nukta Type_Halant R20 R21 R22 R23 R24 R25 R26 R27 R28 R29 R30 R31 R32 R33 R34 R35 R36 R37 R38 R39 R40 R41 R42 R43 R44	− >
%R20 :	End	− >
%R21 :	Type_Consonant Type_Matra Type_Modifier	− >
%R22 :	Type_Consonant Type_Matra	− >
%R23 :	Type_Consonant Type_Modifier	− >
%R24 :	Type_Cons-inv Type_Matra Type_Modifier	− >
%R25 :	Type_Cons-inv Type_Matra	− >
%R26 :	Type_Cons-inv Type_Modifier	− >
%R27 :	Type_Cons-r Type_Matra Type_Modifier	− >
%R28 :	Type_Cons-r Type_Matra	− >
%R29 :	Type_Cons-r Type_Modifier	− >
%R30 :	Type_Cons-nukta Type_Matra Type_Modifier	− >
%R31 :	Type_Cons-nukta Type_Matra	− >
%R32 :	Type_Cons-nukta Type_Modifier	− >
%R33 :	Type_Cons-nukta Type_Nukta Type_Matra Type_Modifier	− >
%R34 :	Type_Cons-nukta Type_Nukta Type_Modifier	− >
%R35 :	Type_Cons-nukta Type_Nukta Type_Matra	− >
%R36 :	Type_Cons-nukta Type_Nukta	− >
%R37 :	Type_Cons-inv Type_Nukta Type_Matra Type_Modifier	− >
%R38 :	Type_Cons-inv Type_Nukta Type_Modifier	− >
%R39 :	Type_Cons-inv Type_Nukta Type_Matra	− >
%R40 :	Type_Cons-inv Type_Nukta	− >
%R41 :	Type_Consonant	− >
%R42 :	Type_Cons-inv	− >
%R43 :	Type_Cons-r	− >
%R44 :	Type_Cons-nukta	− >

Table 19: Default syllable rules

%Begin Type_Cons-r Type_Halant End	->	Half-cons
%Begin Type_Cons-r Type_Halant	->	Reph
%Type_Halant Type_Cons-r	->	Rkar
%Type_Cons-nukta Type_Nukta Type_Halant End	->	Nukta-cons Halant
%Type_Cons-nukta Type_Nukta Type_Halant	->	Half-nukta-cons
%Type_Cons-nukta Type_Nukta	->	Nukta-cons
%Type_Cons-nukta Type_Halant End	->	Consonant Halant
%Type_Cons-nukta Type_Halant	->	Half-cons
%Type_Consonant Type_Halant End	->	Consonant Halant
%Type_Consonant Type_Halant	->	Half-cons
%Type_Cons-r Type_Halant End	->	Consonant Halant
%Type_Cons-r Type_Halant	->	Half-cons
%Type_Cons-inv Type_Halant	->	Half-cons
%Type_Vowel	->	Vowel
%Type_Modifier	->	Modifier
%Type_Matra	->	Matra
%Type_Consonant	->	Consonant
%Type_Cons-nukta	->	Consonant
%Type_Cons-r	->	Consonant
%Type_Numeral	->	Numeral
%Type_Punctuation	->	Punctuation
%Type_Halant	->	Halant
%Type_Nukta	->	Nukta
%Type_Cons-inv Type_Halant Type_Cons-r	->	Consonant Rkar
%Type_Cons-inv Type_Nukta Type_Halant	->	Consonant Nukta Halant
%Type_Cons-inv Type_Nukta	->	Consonant Nukta
%Type_Cons-inv Type_Matra Type_Modifier	->	Consonant Matra Modifier
%Type_Cons-inv Type_Matra	->	Consonant Matra
%Type_Cons-inv Type_Modifier	->	Consonant Modifier
%Type_Cons-inv	->	

Table 20: Default combination rules

References

[1] Barbara F. Grimes, *Ethnologue: Languages of the world*, ISBN: 0-88312-815-2 (paper) 0-88312-823-3 (hardcover), Twelfth edition, 1992, http://www.sil.org/ethnologue/ethnologue.html.

[2] Yashwant Malaiya, *Language and scripts of India*, http://www.cs.colostate.edu/~malaiya/scripts.html.

[3] *Indian Script Code for Information Interchange - ISCII Standard*, IS 13194 : 1991, Bureau of Indian Standards, Manak Bhawan, 9 Bahadur Shah Zafar Marg, New Delhi, December 1991.

[4] *Gist Multi-lingual Card user's guide*, Quark Computers Pvt. Ltd., C-1, Sarvodaya Nagar, Kanpur, December 1992.

[5] Sumant Narayana Pattanaik, Satyajit Nath and S.P. Mudur, *Computer processing of Indian scripts - a pure consonant approach*, National Centre for Software Technology(NCST), Bombay.

[6] David B. Lewis, *x-faq/part 1-6*, ftp://ftp.x.org/contrib/faqs, 1995.

[7] *Online manual page for xterm - terminal emulator for X*, X11R5, Massachusetts Institute of Technology.

[8] *Online manual page for kterm - kanji terminal emulator for X manual*, X11R6, XXI working group in Japan Unix Society, Japan.

[9] Nabajyoti Barkakati, *X Window System Programming*, ISBN-81-203-0958-8. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, second edition, 1995.

[10] Adrian Nye, *Xlib Programming manual*, ISBN 0-937175-11-0 in Nutshell Handbooks, O'Reilly and Associates, Inc., 632 Petuluma Avenue, Sebastopol, CA 95472, second edition, July 1990.

[11] Adrian Nye, *Xlib Reference manual*, ISBN 0-937175-12-9 in Nutshell Handbooks, O'Reilly and Associates, Inc., 632 Petuluma Avenue, Sebastopol, CA 95472, second edition, July 1990.

[12] Adrian Nye and Tim O'Reilly, *X Toolkit Intrinsics Programming manual*, ISBN 0-937175-56-0 in Nutshell Handbooks. O'Reilly and Associates, Inc., 632 Petuluma Avenue, Sebastopol, CA 95472, second edition, September 1990.

[13] Staff of O'Reilly & Associates Inc, *X Toolkit Intrinsics Reference manual*, ISBN 0-937175-57-9 in Nutshell Handbooks. O'Reilly and Associates, Inc., 632 Petuluma Avenue, Sebastopol, CA 95472, second edition, September 1990.