

Crossing the Digital Divide: computer resources to aid minorities

Martin Hoskenⁱ and Melinda Lyonsⁱⁱ

Introduction

The term “digital divide” has been used frequently over the past five years to express the idea that certain people groups have less access to computing resources than others. Some aspects of the problem have been well explored in many documents. These include the financial problems of computer access, the lack of telecommunications resources in remote areas and a lack of basic literacy which would lead to computer use. These issues affect hundreds of millions of people around the world. A lesser-known problem is the lack of program and script support for minority groups, especially where those groups use non-Roman scripts.

Over the past fifty years, computer resources became available to many of the majority peoples of Asia, especially as their economies became important to the world economic system. Large corporations developed programs and script support for the major languages. However, linguistic minority groups have often been passed by in this important area. In some cases the specialized scripts of these minority peoples are not yet available on most computers. In other cases the representation of a minority language using the national script cannot be fully realized using currently available software.

This paper will outline the historical bases for this part of the digital divide. Some examples will be presented for the current situation. It will go further to outline the major components of a solution to the problem. Unesco and SIL have joined together to address some major aspects of the solution, and these will be discussed in the final section.

What is the Digital Divide?

The digital divide is the “disparity in access and use”ⁱⁱⁱ of various information and communications technologies. Many developed countries have noted the difficulty of providing computing and telecommunication resources to their poorest, illiterate or older populations. Also, those in rural areas are less likely to use these new technologies. In some instances language is also a factor related to computer or internet use, but in most cases the languages referred to are ones for which computer technologies are available. Even so, few documents on the digital divide from either the developed or developing world include language access as a factor in the digital divide.

There is yet another and deeper divide in ability to use information resources and technologies for those whose languages cannot currently be represented by available computing technologies. Millions of people speak and read languages which cannot be adequately or easily represented by computers. For these people, the problem is not one of access to telephone or computer, but of programs and fonts that enable them to use the new technology. Until minority peoples can work on computer like everyone else, the divide will not be overcome.

The Problem

The earliest computers were used exclusively for mathematical calculation. As computer technology developed, the earliest users required only the basic characters used to write European languages. This included the English alphabet plus some additional letters with accents or other marks, punctuation marks, and some additional characters such as currency symbols. At first, only special technicians had access to the new technology for writing and printing.

As computer technology was available to people in Eastern Europe and Asia, new problems were created of providing the characters for input and output in new languages. Technicians would handle the computer while general users were content with the printed documents that resulted. Data could not be easily shared between computers or systems, even though English users were creating complex databases. It was not until the decade of the 90's that Chinese language general database programs were available, that could be used to handle and share data readily.

Early Solutions

Few people in the 1980's were interested in developing an ability to represent minority languages on computer. Most who did were using Roman (European) letters, using the computer to produce printed literature for the minorities. A very small group of academics wanted to study minority Asian or African languages with unusual scripts. Often these people would request help from a technically savvy friend, who would enable simple input and output of the minority language. If unusual letter shapes were needed, a rough font would be produced to display or print.

During the eighties and early nineties individual solutions were common. It was cumbersome to produce multilingual documents. The letters for each script were on separate codepages in the Microsoft system. Each section of text needed to indicate the source of the letters. Combining text from different scripts on one page required a lot of work, or was impossible.

As large corporations expanded their operations to Asia, more users needed to combine Western and Asian scripts. Computer companies turned their attention to solving some of the problems. This enabled large sections of the population to use computers readily, but also created additional problems for minority groups using the national script.

Computer Issues in Asian Language Development

The history of language and literacy spread in Asia has created some unique problems for computer users in the region. Reading and writing spread from the Indian subcontinent across to Southeast Asia, with each new people group slightly modifying the letter shapes over time. The current result is that there are fifteen to twenty somewhat similar scripts in use across Southeast Asia. Four of these are national scripts, but many of the larger minority groups still use their own writing system for their own language. Most of these minority groups do not have adequate computer resources (see the website for the Script Encoding Initiative^{iv} for further details).

The more recent history of language development in Asia has created additional problems for those providing computing resources. Current plans and programs use the national language script as the basis for minority language literacy programs. Often, the national language is from a different language family than the minority group and its symbol system does not readily represent all the sounds of the minority language. In adapting the letters of the national language to represent the minority language, a language worker may use the symbols in a way which cannot be represented using current computer programs.

An example of the latter is seen when the Thai script is used to represent Katuic languages of Eastern Thailand. The Thai language has fifteen vowel glyphs used to represent the twenty-one distinctive vowels of Thai either singly or in specific combinations. The Bru language has a more extensive set of contrastive vowels, about forty or so. One proposed orthography uses the Pali dot as an additional vowel symbol to aid in distinguishing these vowels.^v However, the implementation of Thai using current software does not preserve this symbol where it is needed. It is considered a "mistake" and is eliminated.

Why is a General Solution Important?

There are several reasons for solving the general problem of providing computer resources for all languages in a freely available way. First, it will empower local people, enabling any and all to do local language development work. There will be no distinction between people groups based on available and unavailable resources. Second, it will reduce the need for special technical support for those doing language development from within or outside the minority group. As a large organization involved with language development, SIL has found technical support to be one of the great challenges. Finally, computer literacy or work skills learned in one environment will transfer readily across language boundaries. Minority people with computer skills will be able to work in the national or neighboring languages without any special or additional training.

Another reason for needing a general solution is illustrated by the situation mentioned above: the technology developed must allow adaptation by minorities who have different language needs than those of the national majority. The glyphs should permit combinations in any order needed to represent the sounds in the minority language.

One last feature of a general solution is that it would permit data storage and retrieval in any random or organized fashion. This will facilitate database and other information technology uses, perhaps allowing a minority user to store information on prices for local commodities over time, further improving local life. A worker in a local health facility could track births and deaths or other health information over time. But this is only possible when the local language can be used with other applications and programs.

Approach to the Problem

The solution to the problem of computer access for language development requires several components. The first and largest piece is a common storage system, encompassing all languages. The second and third parts of the problem are the ability to enter data into the system and to retrieve it—commonly these are keyboarding (as a method of entry) and rendering (the process of converting stored information to be viewed on the screen or printed on paper). The last two components of the large problem are the ability to analyze data and to convert data from one form to another. The focus of this discussion will be on the first three problems.

Encoding

Encoding is the means by which information is stored in the computer, using a special binary sequence for each character. As computing in different languages developed, each language used one codepage for all the characters and punctuation of the language. The computer system would tie the input and output to the 256 characters of a particular codepage.

The difficulties of using codepages to encode various languages came to international attention in the 1980s, as corporations were attempting to produce various versions for different countries. Users wanted to use more than one or two languages at a time for business interactions, but found themselves hindered. In 1988 a committee was formed to discuss this problem of multiple encodings. Four design principles were agreed to for the new encoding, which they called Unicode. These four principles stated:

- 1) It should be a universal standard covering all writing systems
- 2) It should avoid special coding sequences or states which might vary between systems
- 3) It should provide a uniform coding width for each character
- 4) The same value should always represent the same character

Following that meeting, Unicode was created in 1991. The Unicode encoding system has space for all writing systems, although some have not been added yet.

There are several minority languages of Asia yet to be added to the Unicode scheme, such as Cham, Khamti, Kayah Li, Lanna, and Pahawh Hmong. Proposals have been submitted to the Unicode Technical Committee for some of these languages, but work still needs to be done to finalize their encodings. The Script Encoding Initiative of the University of California at Berkeley has been set up to address this need, but welcomes input from researchers directly involved with the languages. The Script Encoding Initiative sees the need to incorporate minority scripts “For a minority language, having its script included in the universal character set will help to promote native-language education, universal literacy, cultural preservation, and remove the linguistic barriers to participation in the technological advancements of computing.”^{vi}

There were several design principles incorporated into Unicode which have importance for this discussion. First, one codepoint represents one character (or letter). Accents or letters that are placed over or under other letters are separate codepoints, and rules show how they are combined. When two or three symbols are combined for a single sound (such as “เ” “อ” “ะ” in Thai), these are each separate codepoints. Secondly, Unicode stores characters in logical order, or the order the sounds occur in the word. For languages which use right-to-left writing systems, this means that Unicode stores the right-most character first, unlike with European written order. Third, Unicode stores plain text—no formatting for text is a part of Unicode.

Input Methods

The second component of a general solution for minority computing is that of data input. Currently, the usual method of data input is by using a keyboard. Each key on the keyboard is tied to a particular codepoint using specialized software. The difficulties lie in assigning particular keystrokes to their codepoints, and in the cases where characters need to be combined.

When developing a keying system, two main ways are possible. The mnemonic system uses information printed on the keycaps to indicate what the key will type. For an English speaker typing Thai, the “L” key might have a lower case “ล” and the upper case might be the less common “ฬ”. A positional keyboard considers the relative positions of keys on the keyboard and often is a keyboard which has the frequent letters typed by lower case central keys, while less frequent letters are upper case or peripheral keys. For very large character sets, typing a single or pair of characters will open a candidate window. The correct character can be selected from the options presented.

For those needing more than a hundred or so different combinations, but not several thousand, modifier keys or dead keys can be used to provide additional combinations. Modifier keys are ones such as “Ctrl” “Alt” or the “Ctrl+Alt” combination. Often, however, these are used by the application programs, and their use for typing characters can be confusing to the application, or cause unexpected results. Dead keys are ones which display no character until a second character in the sequence is typed. This allows rapid input, but can cause confusion if one does not remember what has already been entered.

One solution to the problems with dead keys is to use the modifier key after the main character, with each result displaying on the screen so the typist knows what has been entered. This works well if each character or state can be displayed by the font, but not all characters can be displayed by all fonts. When an intermediate state is reached that cannot be displayed, it can cause confusion.

Another problem with each of the entry systems is how to delete specific characters represented by a Unicode string. When the modifier key system is used, each state may be deleted separately. However, when a dead key is used, there is no way to know how many keystrokes must be deleted to get to the state needed for further entry.

There is one input program called Keyman that handles complex script input well. It works for those using Microsoft products, and handles the input using modifier keys. There is a second program for Open Source applications called SCIM, which is currently under development. It should work equally well for those using a Linux operating system environment.

Output Issues

After solving the problems of input and storage, generally data must be displayed on a computer screen or printed on paper. This process requires that an encoded string of characters be converted to a glyph string. Glyphs are the visual representations of the underlying characters. This process of converting encoded data to a visual presentation is called rendering. The basic process involves two steps: 1) identifying the glyph which corresponds to a Unicode string, and 2) rendering (printing or displaying) the glyph on the paper or screen at the proper location.

Some problems of the early rendering systems have been solved by the newer smart font technologies. Early technologies required the rendering system to use one codepoint for one glyph in one position, thus sometimes resulting in 4 different codes needing to be stored for a single tone mark, one for each position where it could be rendered. Smart fonts have a two stage processing approach, the first step is glyph substitution and the second is glyph positioning. Glyph substitution ensures that the correct glyph is selected in the right context (when there are specialized characters for different positions in a word, for instance). Glyph positioning fits the glyph in the right place relative to the baseline and other glyphs. In smart-font rendering these two steps are added, so that there is a four-step process: 1) converting the Unicode string to glyphs, 2) process the glyph string to a new glyph string, 3) position the glyphs relative to each other, and 4) rendering the glyph on paper or screen. For more information on rendering technologies, see Hosken, 2001b below.

Graphite

A new rendering system called Graphite has been developed by SIL because of problems and limitations of the previous systems. Graphite uses smart font rendering to enable the use of any glyph in any position needed for a minority script. It gives the font developer complete control of all aspects of script rendering. Those working with minority scripts can develop and use it independently of others. Because it can be used for any language, the skills developed are readily transferred to another language or script. For more information about the reasons Graphite was developed see Hosken 2002 “Rendering Thai as an Example of Using Graphite for Generalised Smart Font Rendering”,

Graphite technology is being incorporated into various products for word processing and Internet access. There is already a simple text editor called WorldPad which uses the Graphite rendering system. Development is in progress on a Graphite-enabled Mozilla web browser and a Graphite-enabled office suite called Open Office. Each of these tools will further enable minority peoples to use computing resources and promote language development.

Conclusion

The digital divide is a real barrier to those in the developing world who cannot use computing resources because no provision has been made for their unique scripts. This paper outlines many reasons why the current technologies are inadequate to the task of providing computing resources to small minority groups in Asia. The

new Graphite technology has the potential to provide an answer to this problem and allow many more people the chance to further develop their languages and communities.

Bibliography

- Department of Linguistics. U C Berkeley. 2003. Script Encoding Initiative.
<http://www.linguistics.berkeley.edu/~dwanders/alpha-script-list.html>
- Department of Linguistics. U C Berkeley. 2003. Script Encoding Initiative. Home Page
<http://www.linguistics.berkeley.edu/~dwanders/#List>
- Digital divide. 2002. National Office for the Information Economy, Commonwealth of Australia.
http://www.noie.gov.au/projects/access/Connecting_Communities/Digitaldivide.htm.
- Green, Julie and Feikje van der Haak. 2002. The Bru people in Khong Chiem, Ubon Ratchathani in Minority language orthography in Thailand: five case studies. Bangkok, Thailand: TU-SIL-LRDP Committee, Thammasat University.
- Hosken, Martin. 2002. Rendering Thai as an example of using Graphite for generalized smart font rendering. SNLP-Oriental COCODA 2002.
- Hosken, Martin. 2001. An Introduction to keyboard design theory: what goes where? In Implementing writing systems: an introduction, edited by Melinda Lyons. Dallas, Texas: SIL, Non-Roman Script Initiative, 121-137.
- Hosken, Martin. 2001. An Introduction to smart font rendering: glyph tracing for beginners. In Implementing writing systems: an introduction, edited by Melinda Lyons. Dallas, Texas: SIL, Non-Roman Script Initiative, 139-151.
- Lyons, Melinda. 2001. Graphite and WorldPad: tools for writing the world's other languages. In Techknowlogia, November-December 2001, 51-54.

ⁱ SIL International and Payap University

ⁱⁱ SIL International

ⁱⁱⁱ Digital divide. 2002. National Office for the Information Economy, Commonwealth of Australia.
http://www.noie.gov.au/projects/access/Connecting_Communities/Digitaldivide.htm.

^{iv} Department of Linguistics. U C Berkeley. 2003. Script Encoding Initiative.
<http://www.linguistics.berkeley.edu/~dwanders/alpha-script-list.html>

^v Green, Julie and Feikje van der Haak. 2002. The Bru people in Khong Chiem, Ubon Ratchathani in Minority language orthography in Thailand: five case studies. Bangkok, Thailand: TU-SIL-LRDP Committee, Thammasat University.

^{vi} Department of Linguistics. U C Berkeley. 2003. Script Encoding Initiative. Home Page
<http://www.linguistics.berkeley.edu/~dwanders/#List>