The Profession as a Culture Killer

Neville Holmes, University of Tasmania

Having a World Wide Web for each writing system would allow much better software than would a Universal Web.

Literate cultures are based on writing systems and print. In the early years of digital computing, machinery had a limited ability to print, but it didn't matter culturally as that printing almost entirely served to create commercial documents such as invoices and sales summaries. There were problems, though, in specialized areas like scientific programming.

Problems loomed larger in the mid-1960s when character sets with upper- and lower-case alphabets came into use on computers and in telegraphy. This enabled the processing of running text, but the typography was extremely crude. The computing profession and industry, however, saw this crudeness as simplicity and promulgated separate sets of similar crudity to cater to other cultural needs.

When personal computers became widespread, the computing industry—with the unthinking acquiescence of the computing profession—simply moved one of the main collections of crudities, the ASCII family, onto those machines. When the pressure for cultural extension grew impossible to ignore, a monster crudity, Unicode, subsumed the various small crudities. My essay entitled "Toward Decent Text Encoding" (*Computer*, Aug. 1998, pp. 108-109) advocated a quite different approach, one that could have been adopted as one of Unicode's subsets, but approaches like that were, to my knowledge, not seriously considered.

To my dismay, I recently read in a local newspaper that ICANN, the Internet Corporation for Assigned Names and Numbers, would begin allowing non-Latin characters in domain names. Getting it done will be immensely complicated, and the result will likely be chaotic. Worse, it shows a complete disregard for mankind's second greatest digital technology: writing.

I have a professional responsibility to protest against these developments, although I don't expect my protest to affect anything.

THE INTERNET

It turns out that ICANN, had started "the internationalization of the Internet's domain names" with its resolution of 25 September 2000 (icann.org/topics/idn) that it "must be … fully compatible with the Internet's existing end-to-end model and … preserve globally unique naming in a universally resolvable public name space."

This resolution seems rather strange. For example, the domain names are already international, and not only in having two-letter top-level domain names (TLDNs) for each country, from which Tuvalu profits greatly. What ICANN seems actually to have meant is to enable mixing of writing systems Unicode-style in domain names.

The delay in implementing this change is not surprising. Even confined to the Latin writing system, the domain name system seems to be lurching into chaos, what with commercial exploitation under the wonky protection of trade name legislation and the cutting loose of constraints on TLDNs.

The Internet is distinct from the Web, even though the Web is based on the Internet. The Internet is defined by the Internet Protocol (IP) binary names or addresses used for directing packets of data along paths within it. The unfortunate slowness of the transition from its IP version 4 to version 6 has confounded this end-to-end model. Its naming is, in the sense of being not necessarily fixed, no longer unique because there are too few IPv4 addresses to go around.

The Web is defined by its use of somewhat meaningful alphabetic domain names, which are used within uniform resource locators (wikipedia.org/wiki/Uniform_resource_locator) to point to where its resources are stored on the Web. In the URL *icann.org/topics/idn* the *icann.org* is the name of the "domain" where the resource *topics/idn* is stored.

When a program such as a Web browser needs to use a resource, it must have the Internet translate the domain name into an IP name so that the Internet can support the browser's use of the Web. To do this translation from meaningful name to binary name, the Internet provides a Domain Name System (wikipedia.org/wiki/Domain_name_system).

Thus the Internet uses culture-free binary names to manage its traffic, while the software that manages the Web uses cultural tokens as

Continued on page 110

Continued from page 112

components of the domain names that create the Web's upper-level structure.

From this, it can be seen that the Internet could well provide a separate DNS for each writing system without compromising, and maybe even helping, its end-to-end model and unique binary naming. This would effect a World Wide Web for each writing system, and I strongly believe this should be done. Indeed, it will probably happen anyway in the long run, but in a drawn out, unmanaged, and costly way.

If each writing system has its own Web, then each will have its own Web software such as browsers and searchers. These will be simpler because the needs will be simpler, and it will be possible to give better and more specific support to each system's typographical and compositional aspects. Webs can be practically supported and gradually developed for minor writing systems, systems that wouldn't have a chance were there to be a Universal Web. Anyone wishing to work in two writing systems would need only two sets of relatively simple Web software rather than one set of grotesquely complex software riddled with feature bloat.

Software needing to mix writing systems, for educational or scholastic use, could use markup for system switching and formatting, but this need not involve URLs and domain names. In any case, users could mix URLs for different Webs if needed because each DNS would translate the domain names for the different Webs into the underlying Internet addresses.

CULTURAL SOFTWARE

Many arguments favor providing separate support for different writing systems. Neglecting these arguments is tragic, but in the case of ICANN, it only continues the computing world's impoverishing of writing cultures, which has persisted since the early years of electronic computing.

In *Coded Character Sets, History and Development* (Addison-Wesley, 1980), Charles Mackenzie describes well the beginnings of the Latin writing system's sad story when brought under computation. Early printers could only use capital letters and a few special characters, all but one of which (the lozenge) were needed for commercial use in names and addresses, product names, and the like. Scientists and engineers using programming languages like Fortran were thus not only restricted to capital letters but also had only the hyphen as a basic mathematical symbol. This led to the replacement of the traditional arithmetic symbols by commercial ones: multiplication's saltire (×) by the asterisk (*), division's obelus (÷) by the virgule (/), and even addition's plus (+) by the ampersand (&), although users could pay extra money to get printer features that replaced ampersands with plusses.

Many arguments favor providing separate support for different writing systems.

The banditry of the computing industry and its profession became more pronounced when its developers introduced upper- and lower-case alphabets. While they still ignored the multiplication and division symbols, developers introduced a quite impoverished set of special characters in both ASCII and EBCDIC, mainly because the impact line printers of the time could deal only with a limited numbers of fixed-size characters.

When personal computers came along, developers adopted the limited ASCII character set, even though the PCs were accompanied by dot matrix printers that were not inherently limited like line printers and typewriter terminals. Many PC users put far more text onto their display screens than out to their printers anyway, and the screens also used dot matrices. The profession therefore imposed the PC's typographical poverty, not the hardware, particularly when cheap ink-jet and laser printers became common.

This was the theme: improve the graphics, colors, and images, but leave the character set in the dark ages of early digital computing. Capabilities available to old-fashioned letterpress printers were never passed on to ordinary text users.

Thus, I cringe when I see H2O and CO2 almost everywhere nowadays, online and in print. Reading technical texts, I shudder at a^2, 45uF, and >=. Even when I can read good online text with proper opening and closing quotes and em dashes, pasting those marks into my vi editor brings it to a sudden stop. When I get e-mail from Europe, the names are often difficult to interpret when they include special alphabetic characters. Although I can, with difficulty, include × and ÷ in the Word version of this essay, they aren't easy to get to, and getting them safely across to the final copy presents a challenge to the editorial staff.

Indeed, the neglect by computing professionals of the Latin writing system's culture brings about a multitude of problems. The Latin writing system was, until taken over by digital techniques, a rich graphical cultural artifact. With digital technology it could have been made even richer, but it has instead been made poorer through, in a word, theft.

The poverty is not just a property of the coding system. We have also been completely barbaric to culture with our keyboards. As a tiny example, there are three peculiarly chosen old-fashioned accents on my keyboard— ~ ` ^ —but I can't use them to key señor or caffè or Côte d'Azur in here, and certainly not café. It's ridiculous. And the Accenture people have it even worse.

With digital technology, we could have a single simple keyboard and accompanying software that would allow simple and complete support of all languages and disciplines that use the Latin alphabet (eprints.utas.edu.au/1564). It's just a matter of using our writing system's rich graphical culture and tradition. What we have now disgraces our profession.

OTHER CULTURES

Writing systems other than the one used here have been, if anything, even more culturally impoverished by the computing profession than the Latin alphabetic system.

The Chinese writing system provides perhaps the greatest contrast (wikipedia.org/wiki/Chinese_character). It's a very old system and has the wonderful advantage of being independent enough of the spoken languages that use it to let people who speak mutually unintelligible languages write intelligibly to each other. It's also faster to read and more economical of space than linear alphabetic writing systems.

In the computer age, however, users are crippled with qwerty keyboards, for which there are two main methods of keying in the Chinese (hànzì) characters: the pīnyīn alphabetic method and the wǔbǐzìxíng root method.

Under the pīnyīn method, the user keys in the Romanization of the official language, Mandarin, and software converts the syllables to characters. This is easy to learn, provided you know Mandarin. However, it's rather slow, especially if there's much need for disambiguation, which is only too likely as there are typically several characters for any of the relatively few pīnyīn syllables.

The wǔbǐzìxíng method takes advantage of the Chinese characters' graphical nature. Many characters are written with two or more separate components (roots) in a formal sequence, and dictionary sequence traditionally relies on this structure. Wǔbǐzìxíng keying also uses the structure and so is much faster than pīnyīn and usable by speakers of languages other than Mandarin. However, it too is superimposed on the qwerty keyboard and, because there are several hundred distinct roots, is so hard to learn that few take the trouble.

Clearly, a hànzì keyboard with, say, 64 root keys and three shift keys for each hand would make wǔbǐzìxíng style keying yet faster and much easier to learn.

But the qwerty keyboard is not the only cultural theft of the Chinese computing profession. All the coding schemes in use represent characters, not roots. Not only does this make dictionary sequencing difficult, it also makes the introduction of new characters impractical. Root encoding would free up both the written and spoken language for change, and it could even eliminate the need for ugly insinuation of alphabetic words into hànzì text.

The computing industry has savagely attacked written languages. The computing profession is gravely at fault in allowing this to happen. However, it's not too late to make amends. A good place to start would be to kill the Universal Web.

Neville Holmes *is an honorary research associate at the University of Tasmania's School of Computing. Contact him at neville.holmes@utas.edu.au. Details of citations in this essay, and links to further material, are at www.comp.utas.edu.au/users/nholmes/prfsn.*