Polish texts in multilingual environments (a case study)

Janusz S. Bien
Institute of Informatics
Warsaw University
Banacha 2
02-097 Warszawa, Poland
JBSIEN@PLEARN.edu.pl

Abstract

There are two possible approaches to the more general use of software originally developed for one specific natural language: to create a version specific to another natural language, or to make a multilingual version preserving all or most of the features of the original, and additionally handling also the other language(s) in question. The second approach is much more difficult then the first, but for an important class of users, especially those from the academic milieu, the first approach is of little use: more and more papers are now being jointly written by multi-national teams communicating over the net; being able to use the same version both for national and international papers is an important advantage.

The present paper discusses the most important aspects of the multilingual adaption of \TeX{} from the point of view of typesetting Polish texts.

Introduction

Although \TeX{} is used successfully with various writing systems (including Hebrew, Arabic, Chinese and Japanese), we focus here our attention on Latin scripts. Although I am interested also in Cyrillic scripts (because Polish linguistic papers often quote some Cyrillic text — sometimes even Old Church Slavonic), I am afraid I have nothing interesting to say on this subject.

We limit also our attention to the \TeX{} program proper, but the accepted solutions should be propagated also to the appropriate utilities, first of all to Makeindex and Bib\TeX{}. It is worth noting that an international version of Makeindex was announced some time ago by Joachim Schroed ([53]). He informed me recently that he finished his work on it in 1992 and has no intention to develop it further; however, he is willing to act as a clearing-house for patches to the program (the program is under the GNU General Public License, so other volunteers can continue the work on it). Although by the time of the 1991 European \TeX{} conference International Makeindex was to be available from all good \TeX{} archives, it was never uploaded to the Comprehensive \TeX{} Archive Network for reasons which I do not understand. As for Bib\TeX{}, it is known that Oren Patashnik is working on version 1.0, but I am not aware of any public statement about the new Bib\TeX{} multilingual facilities.\footnote{I am looking forward to an official report from the \TeX{} Users Group on the present status of these programs.}

Preliminaries

What is \TeX{}? In my paper [5], I claimed that \TeX{} is first of all an unconventionally formulated set of standards; I think that such a meaning of \TeX{} is fully compatible with the general usage. The specific component (e.g. the \TeX{} program), extensions (e.g. Plain \TeX{}), implementations (e.g. em\TeX{}), distributions (e.g. 4\TeX{}) and installations are just more or less directly derived from these standards; by a \TeX{} system I mean a complete (from a given point of view) and usable installation, distribution or implementation.

Although an intrinsic feature of standards is the fact that they are publicly available (and publicly announced), the products using the standards are usually of commercial nature. However, the free availability of \TeX{} implementations is, in my opinion, \TeX{}'s enormous advantage. Therefore I focus my attention almost exclusively on those \TeX{} components and extensions which are freely distributed (e.g., according to the GNU licence).

In consequence I will not discuss below, among others, the multilingual \TeX{} developed by Michael Ferguson and first announced in [13] — although it is...
glyph A recognizable abstract graphic symbol which is independent of any specific design.4

glyph image An image of a glyph, as obtained from a glyph representation displayed on a presentation surface.

glyph shape The set of information in a glyph representation used for defining the shape which represents the glyph.

Personally I find this terminology inconsistent and counter-intuitive. The meaning of character is very abstract and too narrow for practical purposes, the notion of glyph is rather vague and the notion of font does not seem fully consistent with the meaning of glyph image used to define it; ISO graphical symbol is not a graphic symbol but just a printable character, and ISO glyph shape is not a glyph shape, etc. Moreover, some conflict and confusion between SC2 and SC18 terminology are now officially recognised and some attempts are being made to resolve them by developing the so-called “Character-Glyph Operational Model”.

Even if these attempts succeed, there are still important aspects of text not accounted for in the quoted standards. For example, while in Polish the character ó is a letter by itself with its own position in the alphabet (between o and p), in another language ó in can be treated as a letter o with an accent and sorted primarily as o. In consequence, although it would be nice to stick to some generally accepted terminology, using ISO standards for this purpose seems neither feasible nor reasonable. In the present paper we will therefore use a more intuitive common sense terminology, partially based on notions introduced in [9].

Written Polish. In 1136 a papal edict containing about 400 proper names in Polish (written, of course, in Latin) was sent to the Archbishop of Gniezno (the first capital of Poland) ([39, p. 31]; you can have a look at it e.g. in [57, p. 64]). Although some historians consider this date as the beginning of written Polish, the very first full Polish sentence (quoted in a Latin text) appeared in 1270 in a chronicle of a Cistercian monastery (cf. [57, p. 203]); the author spelled it as day ut ta pobrusa a ti pozivai, trying to render its pronunciation with the available Latin letters (the meaning of the sentence is roughly let me grind and take some rest). The first full text in Polish is a collection of sermons (called Kazania Świętockrzyskie) written down in the middle of the fourteenth century.

2 It seems that the status of Ferguson’s multilingual TeX changed recently (e.g. some of its features appeared in emTeX beta release 7), but I am not aware of any public information on this subject.

3 As far as I know, the Polish national standards body is a member of ISO/IEC JTC1 and also a member of Subcommittee SC2; it was formerly also a member of SC1 and has relatively recently joined SC18.

4 According to some sources, the notion of glyph was introduced in UNICODE (see later).
The first printed texts in Polish began to appear at the beginning of the sixteenth century. The center of the printing industry was Kraków (then the capital of Poland) and the printers were for obvious reasons very much interested in making Polish spelling as uniform and stable as possible. Their experiments were summarized by S. Murzyński in his work published in 1551 ([39, p. 362]); at that time using the ogonek diacritical mark for nasal sounds was already quite widespread.

The present Polish alphabet contains 9 national letters in both minuscule and majuscule forms:

a, ć, e, l, n, ọ, ś, ż, ź.

It will be probably a surprise for a Polish reader that, according to [18] and [28], none of them is specifically Polish, even if Kasubian is considered a dialect of Polish and not a different language. It is reasonably well known that q and ć are used also in Lithuanian and that ọ is used in many other languages (ranging from Hungarian and Czech to Spanish and Vietnamese), but some other letters are used also in Serbo-Croatian, Welsh and Wendish; in addition, the book [18] claims that almost all Polish letters are used in Sioux and some in Navaho. Even a letter so seemingly Polish-specific as ń was some time ago used in Catalan [16, p. 257]!

Although in some languages the diacritical marks can be omitted in special circumstances, this does not hold for Polish — all the letters should be always printed with the appropriate diacritics. Unfortunately, exceptions to this rule were introduced by computer and telecommunication technology — for many years output devices (e.g. line printers) with Polish character sets were prohibitively expensive, and Polish texts were printed without the diacritics; this custom still persists in electronic mail. A friend of mine calls such texts Pidgin Polish; I like this term and use it e.g. as a name of a conversion table for Eberhard Mattes’ dvispell program.

It should be stressed that the ogonek is not the same thing as the cedilla.

The repertoire of non-English letters in the Computer Modern fonts seems to be a direct consequence of the foreign names and words quoted in Knuth’s Art of Computer Programming. Thanks to such famous mathematicians as Łukasiewicz the stroke for l got a separate slot in the fonts, but — as before Lech Wałęsa there was no Pole of international fame with an ogonek in his name — no means were provided to typeset letters with ogonek. In consequence, standard TeX encourages some users to abuse the rules and use a cedilla instead of an ogonek. It results in such curiosities as the papers [54], which advocated in Polish the high quality of TeX output, but used throughout the text a cedilla instead of an ogonek and English hyphenation!8; ogonek and cedilla were also confused in [60], [48] and [58].

**Keyboard input**

Keying in multilingual texts There are several aspects of keying multilingual texts:

- how to key in a specific language,
- how to switch between languages,
- how to get the proper echo on the screen.

The echo on the screen (we do not discuss here hardcopy devices) is often the most limiting factor, because its character repertoire is determined by the code used (see below); quite often it forces the user to apply a kind of transliteration.

Usually the keyboard is designed for a specific language or alphabet; the only essential exception of which I am aware was the keyboard for COMECON RIAD and SM computers which allowed one to input both Latin and Cyrillic (and which was hated by users because the layout seemed quite random). If we are lucky, then the keyboard is adequate for one of the languages of our multilingual text; however, a typical situation in Poland is that the user inputs Polish and some other language on a keyboard of USA design.

We will now focus our attention on PC computers, as they are the most popular in Poland.

To cut a long story short, neither IBM in its National Language Support for DOS 3.30 introduced in 1988 ([24]) nor Microsoft in its Polish language support incorporated in later versions of MS-DOS (cf. [45]) offered solutions acceptable to Polish users.

---

8 Interested readers may look at my comments [3]. It is quite interesting that the author of [54] called himself later a professional ogonek designer [55].
Only in the Polish version of MS Windows 3.1 did Microsoft [46] recognize the current practice and offer the choice of two keyboard layouts: one compatible with typewriter standard (called therefore informally typist’s layout) and another one called programmer’s layout. The latter is principally the US layout with direct access to all ASCII characters, whilst Polish letters are obtained with Alternative Graphics Key, which is physically the right Alt key (or left Ctrl and left Alt pressed together — this feature, important though it is for Polish users, seems undocumented).

The natural language support in DOS allows switching between the national layout and the default layout, which is of little use for our purposes. However, during Windows 3.1 installation you can choose two arbitrary keyboard layouts as so-called primary and secondary layouts and then switch between them easily at run time. When necessary, the switch has to be accompanied by a parallel switch to a different font in an application; at least in theory this can be automated by appropriate macro definitions. In consequence it is now relatively easy to mix, e.g., Russian and Polish in a MS Word document; it can then be saved in Rich Text Format and input to TeX after appropriate conversion, but this is definitely too complicated a method to be recommended.

National standards Although this topic is only very loosely related to the subject of the present paper, I would like to mention it here because the work on a national standard for Polish computer keyboards is in progress and this is the right time to voice requirements, questions and comments. For more details, please refer to [10].

International standards ISO/IEC JTC1 Subcommittee 18, already mentioned above, several years ago started the work on the 9995 standard entitled Information technology - Keyboard layouts for text and office systems. The standard was finally approved in 1993 but for technical and organisational reasons has not yet been published, However, being involved for about a year in the activities of the Polish national standards body, I had the opportunity to look at the 1991 draft of the standard and at the later substantial changes. Although I find several fragments of the standard rather vague and confusing, it is definitely an important one and deserving of attention⁹.

The purpose of the 9995 standard is to cover almost all kinds of keyboard, including even numeric keypads in telephones and automatic tellers, and to obsolete earlier standards for typewriters, calculators and other office and data processing equipment. It consists of several parts:

1. General principles governing keyboard layouts.
2. Alphanumeric section.
3. Complementary layouts of the alphanumeric zone of the alphanumeric section.
4. Numeric section.
5. Editing section.
6. Function section.
7. Symbols used to represent functions, Symboles employés pour la représentation des fonctions (this part is bilingual).
8. Allocation of letters to the keys of a numeric keyboard.

The notion of case shift has been generalized by allowing selection between three so-called levels. The shift key is now called level 2 select and a separate key is to be provided as level 3 select. It is also possible to switch the whole layout, misleadingly called group, with the group select key, but the standard says nothing about its place on the keyboard. Every group can have all three levels. The level and group select keys are not to be confused with the Alternate key, the function of which — I guess — is application dependent. The standard intentionally says nothing about the default group 1, which should conform to the local standards or customs.

The most relevant area for the present paper is part 3 of the standard, which recommends two specific layouts. One of them is called complementary Latin group layout and is to be used when the primary layout is based on a non-Latin script (or when there is neither an established national standard nor an accepted custom). The second is called common secondary group layout and, when used with a typical national Latin layout or the complementary group layout mentioned above, allows one to input 333 different characters (including the space) of the 40 recognized European languages plus Afrikaans; to be more precise, these layouts allow one to input the full repertoire of the ISO/IEC 6937 standard discussed below.

The motivation for part 7 and some aspects of its content were presented by Alain LaBonté (an editor of the standard) in the paper [44]. He kindly provided me with the Postscript images of it looks as if controversial statements were simply removed.

⁹ It is interesting that in many details the final version is much less specific than the 1991 draft —
the symbols; after some post-processing the files are ready for inclusion in \TeX{} texts and a quick-and-dirty conversion to \METAFONT{} was done by Pawel Szymczak.\footnote{The files are available for anonymous FTP on \texttt{ftp.mimuw.edu.pl}; the Postscript version with full acknowledgments in \texttt{pub/users/jsbien/iso} and the \METAFONT{} version in \texttt{pub/users/szymczak/tex/mf}.}

The implementation of this standard will definitely facilitate keying of multilingual texts, as it seems much easier to learn only one secondary layout than to switch between various national layouts or to use some ad hoc method.

**Coded character sets**

**National and international standards** As was already said above, there are quite a lot of different standards concerning character sets; however, it is difficult to judge which of these are really used in practice and on what scale.

One of the codes which is used on a very large scale is the 7-bit code called ASCII (American Standard Code for Information Interchange). Although formally it is a standard (since 1977) only in the USA, much software simply requires it and forces its use worldwide. Its international variant is called ISO/IEC 646 (\cite{25}) and there exist some national equivalents, but (with a few exceptions) they were never serious competitors to the real ASCII. The reason why it was so well illustrated with \TeX{}, which uses for its own purposes practically all printable characters of the 7-bit code (we talk here about ASCII, but the same holds for e.g. EBCDIC), including those which are USA specific. Of course, the \TeX{} special characters can be reassigned to other codes, as in the case of a Swedish version of \TeX{}, but the price of incompatibility with the standard version is considered by most users as too high.

ASCII, ISO 646 and other 7-bit codes very soon appeared to be insufficient, and several approaches were taken to enlarge the repertoire of coded characters. The most straightforward method was to use 8 bits for a character. This resulted in a multipart standard ISO 8859 (\cite{29}, \cite{30} and others), defining several character sets. It is said that the selection of characters for each set is such that it satisfies the needs of several languages of a large, relatively coherent, geographical area (\cite{33}) but, at least for ISO 8859-1 (ISO Latin 1) and ISO 8859-2 (ISO Latin 2), a better name is the iron curtain syndrome (\cite{8}) — for example, ISO Latin 2 contains letters from German (because of the German Democratic Republic), Polish, Slovak, Czech etc., but no letters from e.g. French or Italian; ISO Latin 1 contains letters from German (because of the Federal Republic of Germany and Austria), French, Italian etc. but no letters from e.g. Polish. If my information is correct, at that time in the appropriate ISO committee all COMECON countries were represented by a Czechoslovak delegation, and the political division of Europe was reflected in the code tables with at least the tacit agreement of all parties involved. This acceptance was later confirmed by establishing ISO Latin 2 as a Polish national standard (\cite{51}).

Another approach to enlarging the repertoire consisted of allowing a special kind of two-byte representation for some characters, i.e. the coding of some characters with the use of non-spacing diacritics. This technique was used in ISO/IEC 6937; although the intended applications of this standard were Teletex and Videotex\footnote{Yes, for some strange reasons these words are spelled without a \texttt{t} at the end.}, they are especially useful when we want to code some text without really caring in which language it is written. A very important application of this kind is online library catalogues; for example, ISO 6937 is used by the VTLS system (originating from a project at Virginia Polytechnic Institute and State University) installed, among others, at the university libraries in Warszawa (Warsaw), Kraków and Gdańsk. In consequence, more and more of us will be using this code in the near future (cf. \cite{47}).

Still another approach consists of allowing code switching with appropriate escape sequences. The exact rules were formulated in ISO 2022 (\cite{26}) and a specific subset of these rules was described in ISO 4873 (\cite{27}). The ISO 2022 techniques are used in some terminals and terminal emulators.

**Industry standards** Besides formal standards there are also so-called industry standards introduced by manufactures dominating a given market. The PC code pages introduced by IBM and Microsoft belong in my opinion to this category: as far as I know, neither the ASCII-based PC code pages nor the IBM mainframe EBCDIC-based code pages were ever formally accepted as national or international standards.

The problem of Polish letters on IBM PC compatible computers caused, some time ago, a lot of discussions. The choice boiled down to the official 852 code page and the so-called Mazovia code introduced by a Polish PC clone manufacturer. I participated in the discussion (cf. \cite{7}) criticizing both
code pages; at first I had a slight preference for Mozavía but now I opt for 852 as a lesser evil. The reason is that in the meantime I have become aware that the repertoire of the 850 and 852 code pages is based respectively on ISO Latin 1 and ISO Latin 2; so, the iron curtain syndrome is common to ISO Latin 2 and the 852 code page, and therefore the solution to the problem should not be limited to the PC platform.

Another (perhaps the most important so far, in terms of potential applicability) industry standard is a two-byte encoding called UNICODE, which stands for unique, universal and uniform character encoding (56). The development of UNICODE was initiated in 1987 by Xerox Palo Alto Research Center and Apple, later supported also by Microsoft, Sun, Adobe and other manufacturers. However, it is no longer just an industry standard, because after long and hot discussions UNICODE was incorporated into the ISO 10646 standard described below.

Universal Character Set: The Universal Character Set defined in ISO 10646 (35) uses a four-dimensional coding space, consisting of 128 three-dimensional groups; each group consists of 256 two-dimensional planes, and each plane has 256 rows of 256 cells. In the canonical form called UCS-4, a character is coded by specifying in four subsequent bytes the group, plane, row and cell. The first plane (plane 0 of group 0) is basically a slightly revised MULTILINGUAL Plane; it can be used as a two-byte character set called UCS-2. Other planes are at present undefined.

The Basic Multilingual Plane contains over 28,000 characters from various scripts of the world, including over 20,000 ideographic characters used in Japan, China and Korea. Non-ideographic scripts accounted for include Greek, Cyrillic, Armenian, Hebrew, Arabic, Devanagari, Bengali and others. Mathematical and technical symbols are also included, as well as various dingbats. Over 30,000 code positions are still unassigned, leaving considerable room for future development.

The first row of the Basic Multilingual Plane is just equivalent to ISO Latin 1, which in turn subsumes ASCII.

UCS assigns also unique names to most of its characters, and the codes are usually written as hexadecimal numbers; hence a can be referred as 0061 or as LATIN SMALL LETTER A, & as 00F3 or as LATIN SMALL LETTER O WITH ACUTE, a as 0105 or as LATIN SMALL LETTER A WITH OGONEK, etc.

Using UCS-4 or UCS-2 in the canonical form is not always feasible, so several so-called UCS transformation formats has been proposed. The most recent one is UTF-7 named A Mail-Safe Transformation Format (52), but more relevant for our purposes is the so-called File System Safe UCS Transformation Format (abbreviated FSS-UTF, UTF-2 or UTF-4) (59).

One of the problems with the original UCS is that the sequence of UCS codes may contain bytes which can be misinterpreted by present-day software, especially Unix-like operating systems and C programs. In particular, such a sequence may contain a zero byte as a part of a character code, while a zero byte is usually understood as the end of a string. Moreover, a full file name string coded in UCS may contain, as a part of a character code, a byte with the ASCII value for the slash character, while the slash character in this context usually separates directory names in the path to the file. This problem is solved by FSS-UTF, because the 7-bit ASCII characters are coded in single bytes without any change (the 8th bit is of course set to 0). Other characters are coded in sequences of 2, 3, 4, 5 or 6 bytes. The first byte of the sequence has at least the first two high-order bits set; the number of these bits determines the length of the sequence, while the remaining bits carry the proper information. All subsequent bytes in the sequence have the first two bits set to 10, and the remaining six bits carry the proper information. To get the original UCS value, we have to concatenate all the information bits. For example, FSS-UCS representation of 0061 is a single byte with the bit sequence 01100001, the representation of 00F3 consists in two bytes 11000011 10110011, and 0105 is represented as two bytes 11000100 10000101.

In consequence, the UCS strings coded in FSS.UTF can be processed by present day software without problems.

Internal and external localisation

Software and hardware manufacturers mean by ‘localisation’ the adaptation of their products to the local market, which usually includes adapting the software to the local language and customs. For the lack of a better word we can apply this term also to the national TeX versions. In my opinion it is quite important to distinguish clearly two different aspects of TeX (as well as other typesetting software) localisation.
By internal localisation I mean, for example, the possibility of using national characters in command names, changing the text of system messages and diagnostics and other facilities making the use of the system easier for a native speaker of a given language. By external localisation I mean the ability to produce a printed text in a given language, which adheres satisfactorily to the typographic conventions of this language. The key word here is satisfactorily, as in different situations our requirements may be different.

At one extreme we have the simplest case consisting of quoting an isolated word in a foreign language, usually a proper name of a person or a place; in such a situation we are usually fully satisfied by just an unambiguous rendering of national letters. To give a specific example, if one wants to spell correctly Lech Wałęsa in an English text, the exact shape and placing of the ogonek is not so relevant as the fact the the diacritical mark looks like ogonek and not like a cedilla or something else. If a user has such simple needs, then requiring him to install for this purpose special fonts (such as the PL fonts [38]), special formats (such as MgX/IMgX) or large packages (such as Babel [12]) does not seem reasonable. Furthermore, in such situation hyphenation is usually not a problem. If we know what the proper hyphenation is, we can enforce it with discretionary hyphens, but quite often we cannot be sure which hyphenation rules are appropriate for the given word: we may not know which natural language it comes from or, in the case of words of foreign origin, we may not know which one of the competing rules should be applied (e.g. how to hyphenate the name of a Polish linguist of world wide fame — Jan Niecislaw Baudouin de Courtenay?).

Another typical situation is when we have to quote more than a single word from another language, e.g. a few sentences or the titles of papers, books and periodicals etc. In this case automatic hyphenation in the proper language becomes essential, and a convenient way of foreign character input becomes desirable.

At another extreme we have the case of whole texts prepared in a foreign language. If possible, all the typographical conventions of the language in question should be observed: captions and dates (even if generated automatically) should use the proper vocabulary and format, etc. However, what really counts here is just the printed output: in which language were formulated the names of commands needed to accomplish the typesetting is completely irrelevant.

It is perhaps the right place to mention one aspect of Polish typesetting conventions which is not handled by TeX in an adequate way; although it probably concerns other languages, to the best of my knowledge the problem has not been seriously attacked. I mean the very popular way of emphasizing text in Polish which consists in spacing out words, phrases or sentences. (There is a German term for this, Sperrsatz, suggesting that the Germans too make use of such a convention[12]). The spaced-out words interact with punctuation marks in a rather subtle way: colon, semicolon, exclamation and question marks, hyphen and dash[13] are separated from the neighbouring letter by an interletter space, whilst point and comma are typeset close to the letter without any intervening space; furthermore, if the spaced-out text contains a number, the digits in the number are not spaced out ([11]). As the spaced out words are subject to the normal hyphenation rules, any long term solution has to use special (preferably virtual) fonts.

Although the internal and external localisations have definitely different purposes, they are often bundled together. There are of course serious reasons for so doing, especially in commercial environments: most Polish texts are written by Poles, French texts by native speakers of French and so on. In an ideal situation the localised versions should be mutually compatible, and it should be possible to join the texts prepared using them in a single multilingual document, but usually this does not seem possible without some compromises. In consequence, the bundling of internal and external localisation should not be forced on the user without some sound reasons. In particular, allowing Polish letters in macro names can be in conflict with some input conventions; as it has nothing to do with the quality of typesetting Polish texts, it should be possible to switch it on and off, depending on the user preferences.

12 Actually the very first attempt of spacing out in TeX of which I am aware is H. Sommerfeld \sperrren macro, which was contained in a set of over 20 high density PC diskettes with various TeX stuff kindly provided to me by Klaus Thull in 1989 [6].

13 By the way, a traditional Polish dash is longer than en-dash and shorter than em-dash (perhaps because it is always used with adjacent spaces); I hope that the omission of a Polish dash in the Polish fonts and the Cork layout will be cured at the first opportunity.
Rendering Polish characters

Polish letters with ogonek belong to those few national letters which cannot be easily rendered using the Computer Modern fonts. It is practically impossible to construct a reasonable looking letter with ogonek using only a single CM font; therefore the interesting and elegant idea of Zlataška ([61]) of automatic generation of virtual fonts is not directly applicable to Polish — using a comma as an ogonek gives definitely ugly results, although somewhat better results can be achieved by the use of a dropped open-quote\cite{14}.

This does not mean, however, that it is totally impossible to construct acceptable Polish letters using CM fonts. A technique which emerged at Warsaw University Faculty of Mathematics, Informatics and Mechanics (with the principal contributions of Jerzy Ryll and Leszek Holenderski) yielded surprisingly good results and was incorporated into several \TeX macros and \LaTeX styles including the Babel package [12]; more information about it can be found in the documentation of the \ogonek package, submitted to the Comprehensive \TeX Archive Network in June 1994 and available in the \texttt{tex-archive/macro/latex/other} directory.

Of course, there were several attempts to design Polish letters in \METAFONT. One early attempt (of May 1989) was that of Mieczysław Prószyński, then a research worker at the Polish Academy of Sciences and now a co-owner of a prospering publishing house; his attempt deserves a mention here because he immediately placed his \METAFONT source in the public domain.

At present the best quality of Polish letters is provided by the PL fonts developed by Bogusław Jackowski and Marek Ryčko with some help from the professional typographer Roman Tomaszewski [38] and freely available as \METAFONT source; however, for historical reasons these fonts use a non-standard layout. Polish letters are also available in the DC fonts in the Cork layout ([21], [22]) developed by Norbert Schwarz, but their shapes are not satisfactory; it is rather obvious to me that improving them is not a task for Schwarz but for those Polish \TeX users who are interested in improving the typographical quality of Polish texts on a worldwide scale. In the meantime, just rearranging the PL fonts into the Cork layout would provide, in my opinion, immediate advantages for \LaTeX users.\cite{17}.

Font layout impact on \TeX processing

Hyphenation It was very well said by Michael Ferguson that \TeX was multiply unilingual, i.e. it could be adapted to almost any language, but only to a single language. Some of these limitations were overcome in \TeX3, but some of them remain petrified in the low-level design decisions. One such decision is that if a word is to be hyphenated, all its characters must be letters or ligatures from the same font. This requirement made it impossible to extend the \CM fonts by placing all of the missing characters in some supplementary fonts, and forced the creation of new fonts with new layouts.

The question of a standard extension of \CM fonts was raised by Yannis Haralambous in 1989 \cite{191}. I supported the principal idea but suggested a different approach to the problem in my paper [4]; later the discussion switched to electronic mail which was for me at that time practically not available. A new layout was finally agreed at the 1990 \TeX conference in Cork in Ireland \cite{14}. To allow for a greater number of national characters and diacritics, it was decided to make the new layout incompatible with the \CM fonts; the fact that the Cork layout is not yet in common use proves, in my opinion, that this was a mistake. Although the Cork layout is in principle better and more elegant, for many users switching to it still appears prohibitively difficult.\cite{15}

Towards a supplementary text layout As it was stressed recently on the net by Laurent Siebemann, the Cork layout does not solve the problems for all European nations and languages. To be specific, it does not allow hyphenation in the following languages:

- Catalan: missing l with middle dot (for more details cf. [16])
- Greenlandic: missing i with tilde (i, Í), a special Greenlandic letter kra in the shape of a small letter k, and the letter u with tilde (u, Ù).

\cite{15} Using the Cork layout is now quite easy with \LaTeX2e, but \LaTeX3 in practice requires big \TeX, and big \TeX on a PC with a processor other than 386 or higher can be prohibitively slow. If I understand correctly, the problem is not with the total size of the memory, but with its distribution to different data structures; let us hope that the implementors will make appropriate adjustments and that the users of XTIs and ATIs will not be deprived of the possibility to use the up-to-date \LaTeX version.
Polish texts in multilingual environments (a case study)

- Lapp (Lappish; Sami): missing g with cedilla (G, ґ)\textsuperscript{16} and the letter t with stroke.
- Latvian: missing a with macron (ā, Ā), e with macron (ē, Ė), g with cedilla (Ģ, Ģ), i with macron (ī, Ī), k with cedilla (ķ, Ķ), l with cedilla (ļ, Ķ), n with cedilla (ņ, N), o with macron (ō, Ū), r with cedilla (ŗ, Ū) and u with macron (ū, Ū),
- Lithuanian: missing e with dot above (ē, Ė), i with ogonek (ī, Ī) and u with ogonek (ū, Ū),
- Maltese: missing c with dot above (ċ, Ĉ), g with dot above (ġ, Ġ) and h with stroke (h),
- Welsh: missing w with circumflex accent (ū, Ū).

A letter is also missing from Afrikaans (small n with apostrophe) and several from Esperanto: c with circumflex accent (ĉ, Ĉ), g with circumflex accent (ĝ, Ĝ), h with circumflex accent (ĥ, Ħ), j with circumflex accent (ĵ, Ĉ), s with circumflex accent (ŝ, Š), u with breve (ū, Ū).

In consequence, it is perhaps time for the Sobiesszewo layout. It can be based on ISO 8859-10 (331)], which covers the following languages listed above: Greenlandic, Lapp, Latvian and Lithuanian. It covers also Danish, English, Estonian, Finnish, German, Icelandic, Norwegian, Faroese and Swedish, but does not cover the following languages from our list: Afrikaans, Catalan, Maltese and Welsh. Fortunately, as it is usual for ISO 8859 standards, the codes from 0 to 31 and slots from 128 to 159 are left unassigned. This hopefully should allow to accommodate the missing letters and still leave some slots free. Perhaps it would be a good idea to make the slots 0-31 at least partially compatible with the Cork layout. As for the slots 128-159, the decision would require further analysis.

Another starting point can be Baltic Rim Supplementary Set of characters, entered into the ISO register, maintained by European Computer Manufacturers Association (ECMA), on 1st April 1993 and assigned the registration number 179. The Baltic Rim character set covers Danish, English, Estonian, Finnish, German, Latvian, Lithuanian, Norwegian, Polish and Swedish languages; most of my comments to ISO 8859-10 apply also to this character set.

The obvious candidates for the free slots are the missing currency symbols, especially those which cannot be easily constructed, e.g. the peseta symbol.

\textsuperscript{16} Please note that the cedilla is placed under the majuscule and over the minuscule. It seems (cf. ISO 6937 [28]) that the Lapp and Latvian letter ģ (small g with a cedilla above) can be rendered as small g with acute accent (ģ).

A close look at the problem may also reveal other desirable supplements, e.g. the Polish dash mentioned earlier.

**Virtual fonts** It should be noted that there is now an official way of cheating \TeX about which fonts are actually present by using so-called virtual fonts ([43]). Virtual fonts have metric files of their own, but instead of METAFONT-generated images they use the instruction set of Device Independent files to construct the shapes of characters, usually referring to real (actually existing) fonts. Virtual fonts can be used just to rearrange the layout of characters or to view characters from several fonts as belonging to a single virtual font; this application of virtual fonts seems perfectly safe from the difficulties mentioned below, although introducing some overhead.

Virtual fonts can be also used to create composite character shapes from the components found in one or more real fonts; in particular, Laurent Siebenmann advocates (on the net and in the paper intended for this volume) the idea of re-implementing the fonts currently in use with so-called atomic fonts, i.e. containing only those characters which cannot reasonably be decomposed into reusable elements. Although this proposal has some advantages, it is not yet clear whether current \TeX implementations would process such fonts without a concomitant deterioration in the quality of output, caused by imprecise reconstruction of the character shapes due to the rounding problems during DVI file processing.

In any case, introducing a new font family with the new layout, e.g. the Sobiesszewo layout suggested above, does not mean doubling the disk space used up by the real fonts, because the new virtual fonts can borrow as much as possible from the existing Cork fonts (trading in this way time for space).

**Text input** Up to now we have discussed only one aspect of font layouts, namely the feasibility of hyphenation. However, the font layout has also a strong impact on text input. At first this problem was practically non-existent, because (with a few unimportant exceptions) there was one-to-one equivalence between a character on input, its ASCII code and a slot in a font for the image of the character. With 8-bit input permitted, a variety of 8-bit codes on one side and the 256 characters in the Cork layout allocated in a way only partially compatible with ISO Latin 1, a new problem appears: how to correctly associate the input codes with the proper font slots. We will discuss various approaches to this problem below, but at least in some cases it is possible to re-establish the one-to-one correspondence

\textsuperscript{16} Please note that the cedilla is placed under the majuscule and over the minuscule. It seems (cf. ISO 6937 [28]) that the Lapp and Latvian letter ģ (small g with a cedilla above) can be rendered as small g with acute accent (ģ).
by preparing a virtual font with characters allotted according to the input code used.

**TeX input and output**

This section is called "**TeX input and output**" to remind us that, besides the text to be printed, **TeX** also has other kinds of output, namely the log file and the messages sent to the screen. Although the problem of representing national letters in the log files and in messages is not crucial, it should be not completely forgotten.

**7-bit input and output.** Most file systems, and especially the most popular ones, do not allow file attributes describing the character code used. This means in practice that if we have an 8-bit file, without any external information we can be sure only of the meaning of the 7-bit ASCII subset (I assume here that we are not directly or indirectly involved with EBCDIC — if a file is sent by e-mail and an EBCDIC gateway is on its way, then some distortion may occur). Furthermore, 8-bit code pages may not allow us to use 8-bit input for all the languages in question. So, 7-bit input is not obsolete despite the fact that **TeX** can input 8-bit characters, and seems to be with us till **TeX** is able to input some form of Universal Character Set.\(^{17}\)

There are two approaches to 7-bit input, which I will call here multilingual and monolingual. The multilingual approach consists of using the generally accepted **TeX** commands. The initial set of such commands was provided with Plain **TeX** and **LT**-**TeX**, but for obvious reasons it missed the accents and letters not available in the CM fonts. Although the CM fonts were extended (at least conceptually) in 1990, at first there were no standard commands to typeset, for example, an ogonek. This was remedied in 1992, when the **TeX** Users Group Technical Working Group on Multiple Language Coordination WG-92-03\(^{18}\) recommended a set of **TeX** conventions concerning languages (cf. [29]). The conventions included the command names for typesetting letters and accents introduced in the extended layout; for example, the command \(	extbackslash a\k\) was assigned to the ogonek and the name justified as the last letter of the word ogonek (actually Jörg Knappen wrote in [40] that \(\&\) stands also for the first letter of the Scandinavian \textit{kvist}; it can be viewed also as the first letter of the German word \textit{Krummhaken}; I guess these words mean ogonek in the respective languages).

I strongly believe that these recommendations should be extended further to cover also the letters from the supplementary layout discussed above; in other words, they should cover the full repertoire of ISO 6937. Furthermore, these commands should be available as standard in **IAT**-**TeX**, in the worst case producing a message that a given character is not available in the given encoding.\(^{19}\) At the moment the commands proposed in [29] are available in **IAT**-**TeX** only after switching to **T1** encoding, and this fact is described only in the documentation of the fontenc package.\(^{20}\)

As for a monolingual approach to 7-bit input, it is convenient to use a kind of universal diacritical mark, as this makes the text easier to type and, what is more important, more legible. As my first experience was with **TeX** 0.8 when **IAT**-**TeX** was not yet available ([2]), I used at first the commercial at sign (\texttt{\&}). Later I followed the **german**.sty convention of using a double quote (" ") for this purpose, as it is practically the only character without any special meaning (admittedly it is used to introduce a hexadecimal constant, but the risk of conflict is small; moreover, as reported by Braams [12, p. 61], this problem was solved by Bernd Raichle in german style version 2.3e). Another convention used in Poland (introduced in **LEX**/**LATEX** and still available in **MP**/**IM** packages) consists of using a slash (/) as the prefix character; it is easier to type if no keyboard macros are used but involves some incompatibilities with standard Plain **TeX** and **IAT**-**TeX**.

Unfortunately, as Polish has both \(z\) with dot above and \(\&\) with acute accent, the problem arises which of them should be accessed by the prefixed letter \(z\) and how the other one should be accessed. I faced this problem already as a student ([11]) and decide to give priority to the principle of analogy: as the prefixed letters \(c, n, o, s\) refer to the respective letters with an acute accent, the prefixed letter \(z\) should be accessed by \(z\) with dot above.

\(^{17}\) A good example of the consequences of the lack of a standard can be seen in **TeX** and TUG News Vol. 1 No 4 on p. 11, where System przygotowywania dokumentów. Przewodnik użytkownika i podręcznik is rendered as System przygotowywania dok. Przewodnik użytkownika i podręcznik.

\(^{20}\) Perhaps it is described more explicitly in the new edition of **IAT**-**TeX** manual, but I have not yet seen it.
z should refer to ż. The subsequent problem of accessing the letter ż I solved on phonetic principles: as ż is pronounced in the same way as rz, I decided that the prefixed letter r refers to ż; the only drawback of this solution is that confusing ż and rz is a common spelling error in Polish, and this convention may further encourage this confusion.

Another solution to this problem, which is very popular, followed another line of thought: the letter ż is much more frequent in Polish then ż, therefore it was ż which was allocated to the prefixed letter ż while the letter ż was allocated to the arbitrarily selected letter x.

I have assumed here that the universal diacritical mark precedes the letter it modifies. Such prefix notation is very easy to implement in TeX, but a postfix notation is also theoretically possible. The advantage of an appropriately constructed postfix notation is the preservation of Polish alphabetical order when sorting according to the ASCII collating sequence. However, the only practical way to implement it in TeX is to use ligatures; this is not only prohibitively difficult for a typical user, it also has some intrinsic disadvantages described in [38]. I am not aware of anybody using postfix notation for TeX, but Philip Taylor informed me that Pierre MacKay made a strong case for postpositive accents at TUG '94.

Thanks to the availability of numerous tools for the definition of keyboard macros, there is no intrinsic relationship between the conventions of 7-bit input and the keyboard layout used; however, it is much more convenient if they are closely related. Many variants of the 'programmer's layout' including the official one provided in MS Windows 3.1 PL, usually use the z and ż keys — together with some special key such as Alt — to input respectively ż and ż. However, I am quite happy with my own choice, as it allows me to avoid conflicts with Alt-x usage in MS Kermit and Emacs-like editors.

8-bit input and output Although 8-bit output in TeX is a great advantage, it is not a panacea. To get the expected results, the input codes should be interpreted in the proper way and properly associated with the font characters.

Let us note first that a notion of character and letter in the TeX context is quite ambiguous and rather confusing. It seems useful to distinguish at least the following notions:

external input character An element of the input character set in use. A character representing a natural language letter (this we treat as a primary, undefined notion) can be called a coded letter.

internal input character This is the translation of an external character code by the TeX program using the xord array [41, p. 10]; the array may happen to be modified by a TeX code page (see below). The internal input characters are assigned category codes; a character with an assigned category code is called a character token.

In particular, a character can be assigned the category letter, i.e. category code 11; however, it may be very strongly stressed that the only practical meaning of this category consists in allowing TeX's input mechanism to recognize control sequences. In consequence, assigning the category letter to a character representing a national letter allows one to easily create control sequences with names which are meaningful in a given language, but has no impact on the actual text processing.

internal output character This is the character to be output to the screen or to a text file; it can come from the TeX program source through the pool file or be dynamically created during the program run from a text character (see below).

external output character This is the translation of an internal output character code by the TeX program using the xchr array [41, p. 10]; the array may happen to be modified by a TeX code page (see below). The external output characters are usually written to the log file or displayed on the screen.

internal text character This is an abstract notion which has no direct equivalent in TeX's data structures but nevertheless it seems useful. By an internal text character I mean a character token which has been associated (during horizontal list creation) with a specific font and a specific language number (designating a set of hyphenation patterns). The following properties, actually attributed just to the character codes, in my opinion pertain to the internal text characters: lower case code, upper case code, space factor code and math code.

internal text letter Intuitively speaking, an internal text character is a 'real' letter if it has both the majuscule and minuscule forms; speaking more technically, by internal text letters we mean such internal text characters as can be subjected to the hyphenation algorithm (if hyphenation is not disallowed by the \hyphenchar setting). Probably for reasons of efficiency, the actual condition is much more
technical: a character representation is a letter in this sense if it has been assigned a non-zero minuscule equivalent with the \texttt{\textsc{lower case code}} attribute (the minuscule letters have just to point to themselves, and the zero value means that there is no lowercase equivalent).

External text character. This notion cover both a character box in the sense of [42, p. 63], ready to be shipped out to a DVI file, and its representation in the DVI file. It is associated with a specific slot in a specific font and has also such individual properties as height, depth, width and static correction. External text characters are conceptually derived\textsuperscript{21} from internal text characters by using the data from a TFM file, and in particular by executing a ligature program if appropriate. By an external text letter we mean the external text character derived from a single internal text letter.

Font character. This is the character representation in a font with some properties of its own, including the actual height, depth and width of the character image, which may be different from those specified in the TFM file. It may be convenient to further distinguish internal font characters (the atomic font elements) and external ones, which can be composed by adding diacritics, overprinting etc.

From the point of view of a typical user, the \TeX fonts and their layouts are fixed. He can perhaps have some limited choice of the coded character set used (but sometimes he may be willing to process a text prepared with some other character set). There are several means which can be used to provide appropriate translation of the external input characters to the external output, text and font characters.

External translation. Although it is relatively easy in UNIX environments to provide appropriate filters for \TeX primary input and output, there is no efficient filter implementation on some platforms including DOS. In the filter version, this technique allows one to keep only one copy of the files, but there is a risk of confusion when \TeX may read a file directly. In the general case the translated duplicates must be maintained for all relevant files. The advantage of this method is that it imposes practically no limitations, and in particular national letters can be used in control sequences if desired.

Internal translation. By assigning to some characters the ‘active’ category, we can make them a single character macro call; in consequence, we can translate them into almost anything we wish. However, as a character can have only one category code, in some cases a national character code cannot get the ‘letter’ category and hence be used in the names of control sequence. With this exception, this is a very convenient technique from a user’s point of view, especially if the category code assignment is performed by appropriate styles or packages which are already available on the net in several variants. It should be noted that a reference to such a package in the source text serves also another purpose: that of documenting the character set used.

\TeX code pages. To the best of my knowledge, the primary motivation of this non-standard \TeX extension was to make use of the PC 8-bit code pages with the older (7-bit) \TeX. A translation table is usually provided as a specially prepared binary file during format generation and stored with the format. Although it was almost an ideal solution before \TeX 3, now I do not think it is worth recommendation. The two disadvantages which are important for me are the following: firstly, if you occasionally want to process a text in a different code than usual (say, in Mazovia instead of 852) then a special format has to be generated (or prepared in advance wasting disk space); secondly, there is no easy way to determine which code page a given format has been generated with, and that can lead to serious confusion. So, if the \TeX code pages are to be used at all, then I strongly recommend that an appropriate message be set up in the \texttt{everyjob} register to report the code in use.

Virtual fonts. As mentioned already, it is possible to reference the real fonts through a virtual font with layout fully parallel to the character set used; from the theoretical point of view this approach is very general (by transferring the translation to the virtual font mechanism, it avoids possible conflicts relating to the use of national letters in control sequences) and I do not see at the moment any intrinsic disadvantages.

To the best of my knowledge, the virtual font approach is not used in practice. One possible
reason which is no longer valid was the difficulty of changing all the fonts in a document; this is now no problem with the New Font Selection Scheme being a standard feature of \LaTeX, but for the users of other \TeX formats this can be still a serious obstacle. Another reason is purely practical: there are few convenient tools for virtual font preparation; however, for this special case such a tool does not seem difficult to create. The more substantial objection is the overhead introduced by virtual fonts in present day \TeX implementations, but I hope that \TeX implementors can reduce it substantially.

Multi-byte character codes At present multi-byte character codes are used only in some special applications and the text editors using them are now only of an experimental nature. However, it seems worth noting that the active character mechanism allows one to use some of them as \TeX input. In particular, successful experiments were reported on the net with printing library data with non-spacing diacritics and UNICODE texts in FSS-UTF format. It is worth mentioning that the use of multi-byte character codes both for input and output is supported in the \Omega system; \Omega is a very promising \TeX extension designed by Yannis Haralombous and John Placide ([49], [50], [23]); it uses two-byte Universal Character Set (ISO 10646) for internal representation, and user-defined finite automata are used to translate between input, output and internal character codes.

Conclusions

Let me summarize the main points of my paper:

- There is at least one still unsolved (although, I hope, solvable) problem of high quality printing in Polish and other languages, i.e. spacing out (letter spacing used for emphasis as opposed to justification).
- The limits of the Cork layout are now clearly perceived and it seems to be the right time to think about supplementing it by a secondary layout.
- The standard for commands to typeset all characters of the ISO 6937 repertoire should be developed and these commands should be a standard feature of \LaTeX.
- The use of virtual fonts for the general handling of 8-bit input seems worthy of further investigation.

Let me conclude with a personal remark. My first decision to use \TeX was made long ago, when from a Stanford University report I learned about a powerful, freely available, typesetting system. I started to use it as soon as it was possible and developed a love-hate attitude: I used only \TeX because of its obvious advantages but I was permanently irritated by its various idiosyncracies. Recently I was forced by circumstances to use, also in a systematic way, a renowned commercial WYSIWYG editor, and — to my surprise — I found its user [un]friendliness on a practically equal level; understanding the behaviour of the WYSIWYG editor in sophisticated situations is perhaps even more difficult than in \TeX, where at least in theory everything can be looked up in the source or by reference to one of the books on \TeX's operation. So despite some doubts in the past, I made again the decision to stick with \TeX as my primary typesetting tool.

Acknowledgement

The author is very much obliged to the reviewer Philip Taylor for his comments and his help to improve the English language of the present text.

References


Euro\TeX Proceedings 1994

15
Językoznawcze Uniwersytetu Gdańskiego nr 16, r. 1992., s. 51-59.


[23] Yannis Haralambous, John Plaice. First applications of \Omega: Greek, Arabic, Khmer, Poetica, ISO 10646/UNICODE, etc. TUG 94 preprints.


[33] ISO/IEC 10367 Information technology — Standardized coded graphic character sets for use in 8-bit codes.


16
[40] Jörg Knappen. Summary on Polish (La)TeX. Info-Tex@SHSU.edu and TeX-Euro@vri.unz.uni-heidelberg.de, 20th November 1992.


[46] Microsoft Windows. Podręcznik użytkownika [no date, no place, no publication number].


[52] UTF-7. A Mail-Safe Transformation Format for Unicode. Internet Request for Comments no 1642. Available by anonymous FTP e.g. from src.doc.ic.ac.uk.


