A Rational International Keyboard Layout for the Latin Script and its Application to Polish

Marcin Woliński

April 25, 2011

1 Introduction

In my work I need to program in several programming languages, to write texts in Polish and English, and to do some typesetting. All ASCII characters that are present on a typical keyboard are necessary for programming even those completely useless from typographical point of view. For typesetting one needs various variants of quotes, dashes, special writing symbols like the section and copyright signs. When writing scientific stuff I often need to quote person names from various languages, often using accents. Characters such as the bullet • (U+2022) and the raised dot · (U+00B7) can make your plain text emails look much more attractive.

The characters which we'd like to have available are a plenty. So the obvious problem is to remember their positions on the keyboard. An interesting feature of the layout proposed here is that the symbols present on a physical keyboard are used as mnemonics for the additional bindings.

From my point of view the main problem with Polish keyboard layouts currently offered by XKeyboardConfig project (and so available, e.g., in Ubuntu) is the lack of typographically proper Polish quotation marks. The version of XKeyboardConfig data present in Ubuntu 10.10 includes Polish opening quote only in Dvorak-like layouts, and no layout has guillemet signs («»). Some discussion of this problem has taken place on XKeyboardConfig list in 2006 but unfortunately it was suspended before conclusion.

2 Beginner's Guide to X Window Ways of Entering Text

Linux (or rather the X Window System) has several mechanisms for entering text using the keyboard. We will discuss them in turns.

First of all pressing a typical key results in a character, e.g. a lowercase Latin letter. The number of characters accessible this way is multiplied thanks to the use of modifier keys. The Shift key provides for inputting capital letters. In the case of Polish keyboard a second modifier key is used, called „AltGr" or „grey Alt" (the key was indeed of darker shade on keyboards of AT PC). This modifier allows to place two additional characters on each alphanumerical key.

So the full binding of a key looks like this: $\boxed{\begin{matrix}\text{A} & \text{Ą}\\ \text{a} & \text{ą}\end{matrix}}$. In the context of X Window we speak of four "levels" of the key: $\boxed{\begin{matrix}2 & 4\\ 1 & 3\end{matrix}}$. Level 1 is accessed without modifiers, level 2 is accessed with Shift, level 3 with AltGr (which is called "third level chooser" in the X lingo), and level 4 with Shift+AltGr. The number of levels is not limited to four by X Window, but the layout has to be possible to remember, so four levels is already a lot.

It's worth noting that the level 3 chooser does not have to be bound to AltGr. In fact I strongly encourage all touch typists to have two third level shifts on both sides of the keyboard

— analogous to two Shifts. For me the “windows keys” are a perfect choice. This variant among many others may be selected in keyboard options independently of the layout used.

Mechanical typewriters used the mechanism of dead accents to extend the set of available characters. Some keys of a mechanical typewriter had printable characters on them but when pressed did not move the carriage. That way the type was printed at the same position as the type on the next key pressed. The mechanism was used to add accents to base letters. Dead keys are present also on computer keyboards (Windows users know at least the annoying dead tilde key: it doesn’t show up when you try to type a tilde).

Dead keys on the computer keyboard follow almost verbatim the mechanical implementation. If you press, e.g., a dead accent acute key ´ (not the apostrophe key which is not dead) and then the letter e the keyboard driver will react by emitting the character é. A difference is that the keyboard driver actually replaces the two characters corresponding to the keys pressed with one new character. The new glyph is not realised by overprinting basic glyphs. We get a single character é as if it were present on one of the keys. Note that the order of hitting keys is retained — first the dead modifier, then the base character. In the case of of a mechanical typewriter this was required by the mechanics, on a computer it is an arbitrary convention.

There are other ways in which a sequence of key presses may get mapped to a single character. The next is the compose mechanism. To use it one has to bind the *compose* function to one of the physical keys on the keyboard (various possibilities for that are available in keyboard options). The compose key behaves a bit differently from modifier keys. You do not keep the compose key pressed while hitting the other keys comprising the sequence. Here are some examples of compose sequences: Comp. s s → ß, Comp. o r → ®, Comp. t m → ™, Comp. - d → đ, Comp. o e → œ, Comp. % o → ‰, Comp. f s → ſ, Comp. ^ 2 → ^{2}, Comp. 3 4 → ¾. The full list is contained in the file `/usr/share/X11/locale/en_US.UTF-8/Compose` (in fact this file also defines dead characters).

The next way of entering characters uses the so called advanced input methods. As far as I understand, this idea was born in the context of Far East languages, where the number of characters is too large to bind them to the keys of a standard keyboard. One needs some tools to quickly select needed characters. These tools are language specific, so they are created separately for each language (or rather script), and so we have several input methods specialised in various scripts.

Advanced input methods are sometimes useful also in the context of European languages. For example among the input methods offered by GNOME there is a method for transliterated Cyrillic alphabet (y a → я) and another for phonetic symbols of IPA (n g → ŋ).

To conclude this review we should also mention the possibility of inputting characters using their numbers. For example in GNOME one can access any Unicode character by its hexadecimal number using the prefix Ctrl+Shift+U. For example: Ctrl+Shift+U 2 0 3 0 → ‰.

There is a multitude of ways for entering text in X Window. However, each of these tools seems to have its specific use, similar to various carpenter’s tools. One does not expect to have one tool suitable both for cutting wood and fastening screws.

The keys bound directly to characters are the fastest means for typing. But a large character set mapped this way needs much learning. This is only worth the effort for characters used on a regular basis. Dead characters seem to be a comfortable way of extending the character set with seldom used accented characters. Each accent can be combined with several base characters. The drawback is that this takes more keystrokes. The compose mechanism seems a good way to access special characters which we don’t use often. The compose sequences seem rather easy to remember or even guess. They are definitely easier to remember than Unicode numbers. Transliterated input methods are a nice way to type a short passage of Cyrillic text from time to time, e.g, Cyrillic titles in a bibliography of a scientific article. But if one has contact with Cyrillics on a regular basis it is probably wiser to learn a regular Cyrillic layout.

3 The Proposed Layout

The layout proposed here is opportunistic: we take the basic QWERTY layout for granted and we decide to stay as close as possible to what is engraved on the keys of a typical keyboard. The proposition concerns the set of symbols bound to the third and fourth level of keys, including a rich set of dead accents.

We decide to keep all ASCII characters in place, even those completely useless from the typographical point of view: ~ ` ^ 6 " '. The grave, tilde, and circumflex/caron characters in ASCII are stand-alone accents which do not have any use in typesetting. One needs combining variants of these accents (defined in Unicode) to put them over letters. These characters are, however, needed in programming as part of syntaxes of programming languages. If we use typographically correct quotation marks for a string literal, e.g., “a string”, a C++ or Java compiler will react with an error message. We have to use straight quotes in this context: "a string". The single straight quote/apostrophe is used, e.g., in SQL and Perl. Unix shells use both apostrophe and the “backquote”, which according to Unicode is in fact a standalone grave accent. None of programming languages I know allows to denote subtraction with the minus sign (U+2212), one has to use ASCII hyphen (U+002D) for that. So all these ASCII characters have to remain accessible.

For me an important aspect of a keyboard layout is ability to remember it. Obviously, since some characters are used on a regular basis and some only from time to time, one tends not to remember some parts of the layout. To make remembering easier I decide to follow the following rules:

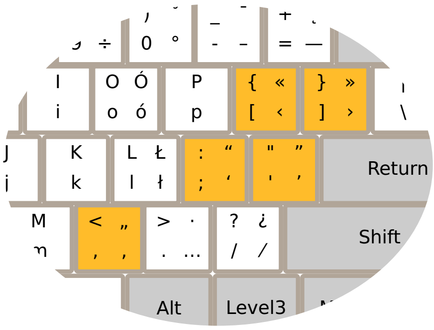
- Additional non-alphabetic symbols will be placed on keys which already bear non-alphabetic characters.
- Additional (accented) letters will be placed on alphabetic characters.
- Dead accents will be placed on the fourth level of the top row of the keyboard and only there.

The characters that I missed most are proper quotation marks. The problem of quotation marks is complicated due to large variability in different languages. The English quotation marks look like “this” and ‘this’ (however the British tend to use single ‘marks’ and only for ‘“quotes” within quotes’ they use double ones while Americans do it the “‘other’ way around”). German quotes are like „this“ and Polish like „this”. Moreover the guillemets are used differently in « French » than in »German«.

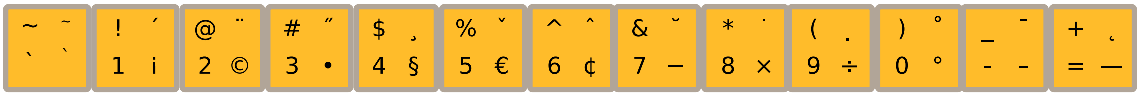
We see here two possibilities: to make the mapping language dependent and assign an opening quote (whichever it is) to one key and closing quote to another. But that would be impractical if one needs to often switch between, e.g., English and Polish. So a better idea is to have all types of quotes available simultaneously.

The following ten characters should be mapped: “, ”, „, «, », ‘, ’, ‚, ‹, ›. They form five pairs of single and double quotes of the same kind. We will put single quotes on level three and double on level four by analogy to the symbols engraved on the " ' key.[1] The most natural key for the English closing quote is the apostrophe/quote key. An obvious place for the low opening quote is the comma key. We put the opening English quote on the semicolon key only because it is adjacent to the closing one. The French quotes are placed on the keys, which are close to regular quotes and which form a visible opening/closing pair: { [and }].

[1] This is a change with respect to my original design, where double quotes were put on the more easily accessible level 3. However, after a year of using it I found that I still make errors when using the ASCII quotes. My fingers have quickly mastered the layout after the change.



Let us now examine the top row of the keyboard. Since these are non-alphabetical keys we will place additional non-alphabetical characters on level three. The fourth level of these keys will be occupied by dead (combining) accents. (Accents as a rule are placed above base characters, so the fact that they are accessed using Shift fits here nicely.)



This row is organised according to a set of associations based mainly on the shape of symbols engraved on the particular keys. Let us start with the accents:

- The first key from the left, $\boxed{\substack{\tilde{}\\ \grave{}}}$, is non-typical since it bears two accents on it. As an exception we place dead accents grave and tilde respectively on level 3 and 4 of this key.
- The acute accent obviously has to be adjacent to grave, so we place it on the $\boxed{\substack{!\\ 1}}$ key. For many people acute is the first accent which comes to mind, so the digit one is a good place for it.
- The next accent that comes to mind for me is umlaut / diaeresis. And it consists of two dots. So we place it on the key $\boxed{\substack{@\\ 2}}$.
- The adjacent key $\boxed{\substack{\#\\ 3}}$ will have on it the Hungarian variant of the umlaut (or double acute, ő). Note that the character # has two diagonal strokes almost like ˝, while @ is round like the dots of the regular dieresis.
- The key $\boxed{\substack{\$\\ 4}}$ will have the *accent cédille* (ç), since its shape resembles the curve of the dollar sign.
- The next three keys are organised around the “hat” printed above the digit 6. This key is the obvious place for the “hat” accent — *circonflexe* (î). To the left (on $\boxed{\substack{\%\\ 5}}$) we place its opposite, i.e., Czech háček (ě, cf. straight diagonal line of the percent character) and to the right (on $\boxed{\substack{\&\\ 7}}$) we allocate the round analogue of this accent — breve (ŏ, cf. round curves of the ampersand).
- The star on $\boxed{\substack{*\\ 8}}$ is the best approximation for a single point that we can find in the top row, so we place the dot accent on this key (ė).
- We place the dot-below accent (ọ) on the adjacent key $\boxed{\substack{(\\ 9}}$.
- The key with the digit zero $\boxed{\substack{)\\ 0}}$ is the best place for the ring accent (Czech kroužek, ů).
- The hyphen/underline key will obviously contain the macron accent (ō).
- Finally, we place the ogonek (Polish tail) “accent” at the end of the row ($\boxed{\substack{+\\ =}}$). After all a tail is usually attached at an end of a creature.

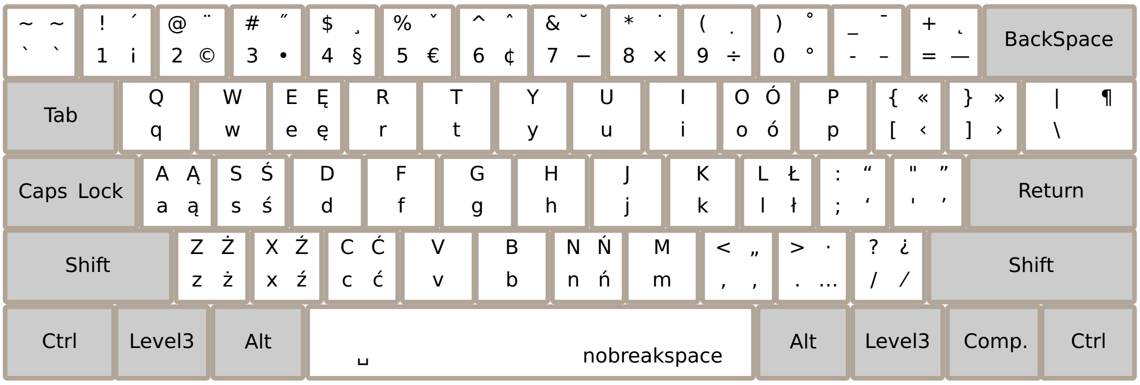
The symbols on the third level are organised in the following manner:

- The key $\boxed{!\atop 1}$ containing exclamation mark is equipped with the Spanish reversed exclamation mark ¡.
- Since the copyright sign © resembles @ we place it on the $\boxed{@\atop 2}$ key,
- The number sign # can be associated with enumerations, so we place the bullet • used for itemisations on the key $\boxed{\#\atop 3}$.
- The dollar sign $ is similar in shape to the section sign §, so the latter goes on the $\boxed{\$\atop 4}$ key.
- The Euro sign is placed on $\boxed{\%\atop 5}$ since we have seen it engraved there on several keyboards.[2]
- We place the ¢ character adjacent to € (although I'm not sure it is really used with Euro).
- Next three symbols are organised around the star symbol. A star is one of possible symbols for multiplication, so we place the times sign × on the key $\boxed{*\atop 8}$. Around it we allocate other arithmetic operations in the standard order: addition, subtraction, multiplication, division. The plus already has its place on the keyboard, so we are left with minus[3] − which goes to the $\boxed{\&\atop 7}$, times × on $\boxed{*\atop 8}$, and divide ÷ on $\boxed{(\atop 9}$.
- The key with zero is the best place for the degree symbol °.
- The next two keys will contain dashes. On the key with a single line (-, hyphen) we allocate the en-dash –, and with two lines (=, equals) we place the em-dash —.

We bind some more non-alphabetical characters to the following keys:

- As a counterpart to the question mark we have a Spanish reversed question mark ¿.
- For the slash / we have fraction sign ⁄ (in carefully crafted fonts these two angular lines have different slope and length: the correct ½ vs. the ugly 1/2).
- The key containing the dot (full stop) will contain the ellipsis sign … on level three, and with Shift a shifted (centered) dot ·.
- Pilcrow (paragraph sign) ¶ will be a counterpart for the vertical bar.

The following diagram depicts a complete Polish layout based on the above. Polish accented letters have been added on level 3 and 4 of some alphabetical keys in the same way as is done in currently available layouts. It is, however, worth noting, that the layout of non-alphabetic keys can easily be a basis for a layout of another Latin script language.



[2] I have seen a keyboard with € engraved on the $\boxed{\$\atop 4}$ key, but that is a great spot for §. Some keyboards have € on the key $\boxed{e}$, but that is an alphabetic key, moreover on a Polish keyboard we need it for the letter ę.

[3] This is a different dash than hyphen -, endash –, and emdash —!

The proposed layout does not include script digits nor fractions. The set of superscript digits present in the current layouts is ridiculously small: ^{1}, ^{2}, and ^{3}. Probably it's a heritage of the Latin 1 codepage or Postscript fonts in Adobe Standard Encoding. However, today's OpenType fonts often contain all superscript *and* subscript digits. Mapping them all on the standard keyboard does not seem practical or perhaps even feasible (maybe on the numerical keyboard?). A similar problem concerns fractions: Latin 1 includes the characters ¼, ½, ¾, but Unicode has 15 characters representing irreducible proper vulgar fractions with denominators 2, 3, 4, 5, 6, and 8. Which of them to map on the keyboard? How to remember their layout? I think that the right solution is to map none and access fractions and script digits with the compose mechanism, which turns out to be quite comfortable and logical.

Perhaps a good idea for the Polish layout would be to include as many characters from the languages of neighbouring countries as possible. We mean mainly the characters which are not easily composed from a base character and an accent.

In XKB configuration files the proposed layout has been realized in two steps: the layout `latin(intl)` provides a mapping for basic Latin script enriched with dead accents and non-alphabetic symbols. Then the definition for `pl(intl)` includes that layout and adds specifically Polish alphabetical characters. That way it is easy to add analogous layouts for other languages using Latin script, which we advocate for.

4 Summary

I use the layout presented here since 2007. It was presented at the Polish TeX Users' Group meeting in 2008 with positive reaction from the audience. I know of several people who use and praise this layout, so it seems to be high time to include it in the XKeyboardConfig database.

The layout comprises the full set of Polish characters as well as 14 dead accents allowing to access practically all accented characters used in European languages, a comprehensive set of quotation marks and dashes, selected writing symbols. An important feature is the use of symbols engraved on the physical keyboard as mnemonics.

The proposed layout of accents and writing symbols is generic. It can be the base for a layout for other European languages. Only the alphabetic part would need to be adapted.

The files defining the layout and installation instructions are available at
`http://marcinwolinski.pl/keyboard`.