Some Problems with Accents in TEX: Letters with Multiple Accents and Accents Varying for Uppercase/Lowercase Letters

A.S. Berdnikov and O.A. Grineva

Institute of Analytical Instrumentation
198103 St.Petersburg, Rizskii pr. 26
email: `berd@ianin.spb.su`

Abstract. *The problems of using the internal command* `\accent` *as a tool for support of some Cyrillic writing systems is investigated. It is shown that the internal features of* `\accent` *prevent construction of some Cyrillic letters which require several accents simultaneously. A special macro which emulates the work of* `\accent` *by some other commands is suggested.*

The accents for I/i and J/j, which are different for uppercase and lowercase letters, are also considered. If-then-else *structures by use of which correct accents can be placed, depending on the letter case, are proposed. A similar technique can be used for case change of the Cyrillic "capital form" ligatures* **Љ** *and* **Њ**.

1. Introduction

A nearly inevitable feature of Cyrillic encodings is that some Cyrillic letters (for example, the majority of accented letters) will need to be assembled from pieces — accents, modifiers, composite elements, etc. There are cases (not too uncommon) where *several* elements should be attached to the same base character. The typical examples are stresses over accented letters, some specific characters from the Nivh language, multi-level accents specific for Saam, and so on:

Ё́, Ӣ́, Ӻ̋, Ӿ̇, Ӧ̄, Ӭ̅А̅, …

Another feature appears in the cases where an accent is positioned *between* two characters. For example, in Serbian the accents *acute*, *grave*, *double-grave* and *frown* (reverted breve) may be positioned like

б́рвно, мр̀твац, мр̏ква, зр̑тва, …

This effect can be simulated by placing the interletter accent above a special invisible character, as is available in the EC fonts, but in the present package we use a rule of zero thickness and height equal to the height of a lowercase or uppercase letter, as appropriate.

Finally, problems arise when we need to put accents above the letters "I/i" or "J/j" which contain dots for the lowercase form but not for the uppercase form. No problem arises unless there is automatic uppercase and lowercase conversion of the user-defined text, although, for example, the Khakassian uppercase "İ" could be unexpectedly transformed into a double-dotted "i̇" after `\lowercase{}` is used in a macro:

`\.I` = İ `\uppercase{\.I}` = İ `\lowercase{\.I}` = i̇

The final section also addresses a somewhat similar problem, which arises in the case of transformations of the ligatures **Љ**, **љ**, and **Љ**.

The primitive `\accent` command cannot be used for such constructions: it cannot analyse uppercase/lowercase characters, it puts accents only over the letters and atomic symbols ignoring the composite characters like `\hbox` or `\vrule` and it cannot put an accent over already accented characters. To overcome these restrictions and to support inside one encoding as many Cyrillic writing systems as possible, it is necessary to simulate the work of the command `\accent` by other TEX tools.

2. Internal working of the primitive `\accent`

Before we proceed to simulate the work of `\accent` we will examine the details of its behaviour. Its syntax is specified [2, 1] as:

`\accent`⟨*8-bit number*⟩ ⟨*optional assignments*⟩ ⟨*character*⟩

Character is a letter over which the accent should be placed. The *8-bit number* is the code which selects the character used as the accent, from the current font. The *optional assignments* between the *8-bit number* and the *character* could contain, in particular, the font-switching commands which enable the accent defined by *8-bit number* and the accented object defined by *character* to be taken from different fonts (although it may not contain `\setbox` assignments).

If the assignments are not followed by a *character* (where *character* is a token with `\catcode=11` or `\catcode=12`, `\char` command, or a command created through `\chardef` or `\noboundary` commands), TEX treats `\accent` as equivalent to `\char`⟨*8-bit number*⟩ command. Otherwise the character which follows the assignment is accented by the character that corresponds to the *8-bit number*. The accenting algorithm is the following[1]:

1. If the *character* has a height equal to `\fontdimen5` (the ex-height) for the font from which the accent is selected, the accent is centered horizontally over the box corresponding to the *character* with no additional vertical shift.
2. If the *character* has a height unequal to the ex-height of the font from which the accent is drawn, and the font is an upright one (i.e. the slant `\fontdimen1` for the font where the accent is selected is equal to `0pt`), the accent `\char`⟨*8-bit number*⟩ is placed in an `\hbox` which is vertically shifted up or down, as appropriate, by the difference of the height of the *character* and that ex-height. It is also centered horizontally over the *character* box.
3. If the *character* has a height unequal to `1ex` and the font is slanted, i.e. the value `\fontdimen1` for the font where the accent is selected is not equal to `0pt`, the accent `\char`⟨*8-bit number*⟩ is placed in an `\hbox` and vertically shifted as described above. Horizontally, the center of the accent box is displaced from the center of the *character* by the amount of the vertical shift multiplied by the font slant `\fontdimen1` (which is nominally a *slant-per-pt* value). Note that both of these are signed quantities, so that the shift may be in either direction.
   The additional horizontal shift proportional to the slant value is an essential component of TEX's accenting algorithm, ensuring that the accent is correctly positioned over a slanted character regardless of its height.

Positioning of the accent box over the accented character is performed by explicit kerns inserted by TEX. The kerns are inserted in such a manner that the horizontal size of the accented character is not affected; however, the vertical size of the accented character is corrected for the height of the resulting construction.

[1] This description is slightly adapted from text extracted by Bernd Raichle from the original Web source of TEX.

The fact that the command `\accent` inserts explicit kerns prevents TEX from using its automatic hyphenation algorithm, as always happens for words with explicitly inserted kerns. It is essential that the commands for the correct alignment of the accent with respect to the accented character are inserted in the output stream first — i.e., *before* the main character. If the italic correction command `\/` is placed after the accented character, but it is separated from the main character by a kern, an `\hbox` command, etc., it fails to define the proper italic correction shift.

3. Macro implementation of `\accent` for multiple accents

By definition the command `\accent` can put accents only over atomic letters or symbols. In particular, it cannot put an accent over a character which already bears an accent (at least in text mode). The other disadvantage is that the accent and the accented character should usually be taken from the same font. Although the syntax of `\accent` enables one to use font-switching commands, doing so is not a simple matter.

A simple macro emulating the work of `\accent` and which overcomes some of its restrictions is suggested in [3][2]:

```
\def\ifnnull#1{\def\inner{#1}\ifx\inner\empty\else}
\def\genaccent#1#2#3{%
   \leavevmode\setbox0=\hbox{#3}%
   \vbox{\offinterlineskip
         \ifnnull{#1}\hbox to \wd0{\hss#1\hss}\kern 0.2ex\fi
         \vbox to \ht0{\copy0\vss}%
         \vtop{\null\vbox to \dp0{\vss}%
           \ifnnull{#2}\kern 0.2ex \hbox to \wd0{\hss#2\hss}\fi
         }%
   }%
}
```

This enables us to put as many accents as we wish[3] above (`#1`) and below (`#2`) an arbitrary TEX object (`#3`). It is also a simple matter to select the accents used for this purpose from a separate font without needing to take care about the current font's status. Unfortunately the command does not take account of

2 The original commands are modified a little to fit the purposes of this paper.

3 Defined as an `\hbox` if necessary.

the fine tuning for slanted fonts, and it also breaks down the work of the italic correction `\/` (the main character is deeply hidden inside a `\vbox`).

This scheme could be modified. Suppose the accent is defined as the first parameter of some macro, and the main symbol to be accented as its second parameter. The commands:

```
\setbox0=\hbox{#2}
\setbox2=\hbox to \wd0{\hss#1\hss}
\raise\ht0\box2\kern-\wd0 \unhbox0
```

do the necessary work:

— put the accent above the main character,
— align it properly in the horizontal direction (at least for upright fonts),
— do not change the width of the main character while the height of the composite construction is corrected properly,
— leave the main character as the last object in the output stream so that the italic correction command `\/` can have access to it,
— enable use of arbitrary TeX constructions for accents and accented characters,
— conserve kerning with the characters followed after the accented character (unfortunately the kerning with the *previous* character is destroyed).

The only thing to do is to add the fine horizontal tuning which is not a problem once we can calculate it. The horizontal shift is derived from multiplying a height dimension value by `\fontdimen1`, which is itself a dimension value (though it is in effect physically dimensionless, being a slant value per point). TeX does not provide a single command for multiplying two dimension values.

A macro which converts the `\fontdimen1` of the current font into a text string `\slant@value` without the suffix `pt` at its end can be coded, for example, as[4]:

```
\begingroup
  \catcode‘P=12 \catcode‘T=12
  \lowercase{\endgroup
    \def\strip@pt#1PT{#1}}
\edef\slant@value{\expandafter\strip@pt\the\fontdimen1\font}
```

and used as:

[4] We use a trick, suggested by Bernd Raichle and based on `\kslant` shown on p.375 of the TeXbook, to make the macro `\slant@value` which is fully expandable; the name `\strip@pt` is that used in the LaTeX kernel for this function.

```
\setbox0=\hbox{#2}
\setbox2=\hbox to \wd0{\hss#1\hss}
\dimen2=\slant@value\ht0
\kern\dimen2
   \raise\ht0\box2\kern-\wd0
\kern-\dimen2 #2
```

4. A set of higher order commands for multiple accents

To create an universal set of accenting commands it is necessary to take into account the following factors:

— The symbol used as accent could be an ordinary `\hbox`, but it could be raised in advance by `1ex` as well. To use the latter as an accent it is necessary to delete this additional height, and for slanted fonts it is necessary to delete the additional horizontal shift as well.
— Although accents are typically placed above characters, there are cases where modifiers are placed below the character, at the baseline of the character, and over the character to make composite objects.
— Accents which are placed above or below the character may need additional vertical space between the accent and the character.
— Other modifiers that need to be joined to the character may need to be shifted closer to the character to ensure that no gap appears.
— Symbols like comma may need to be raised above the baseline before they can be used as an upper accent.
— Accents can be placed before the letter at the beginning of a word, after the letter at the end of a word, or between two letters in the middle of a word.
— The accents "*overline*" (*macron*) and "*underline*" may be longer than the width of a single letter.

To satisfy these requirements, the style file ACCENTBX.STY contains the following commands:

`\upaccent{`*accent*`}{`*main-symbol*`}` — put *accent* above *main-symbol*.
`\dnaccent{`*accent*`}{`*main-symbol*`}` — put *accent* below *main-symbol*.
`\baseaccent{`*accent*`}{`*main-symbol*`}` — put *accent* at the baseline below symbol *main-symbol*.
`\nullaccent{`*accent*`}{`*main-symbol*`}` — the *accent* is centered with respect to the height of the *main-symbol*.

`\upaccentbar{`*main-symbol*`}` — put a horizontal bar above the whole width of *main-symbol*.

`\dnaccentbar{`*main-symbol*`}` — put a horizontal bar below the whole width of *main-symbol*.

`\markchar` — make an invisible rule with height `1ex`.

`\MARKCHAR` — make an invisible rule with height equal to the capital height of the current font.

`\marktwochar{`*char*$_1$`}{`*char*$_2$`}` — make an invisible rule with height the larger of the heights of the two argument characters, and with depth the larger of the depths.

We can use a variety of TEX constructions as the *accent* and *main-symbol* — mainly those that can be used inside `\hbox`. In particular, an accented letter can be used as the main symbol (i.e. multiple commands `\upaccent`, `\dnaccent`, `\baseaccent`, etc., may be used). It is essential to note that if the final symbol in a chain of accenting commands is one with `\catcode` 11 or 12, the italic correction command `\/` will work correctly if it follows after the composite construction.

Special objects for use as *accents* may be created with the following commands:

`\aboxadjust{`*accent*`}` — lower by `1ex` a character which is usually the argument of an `\accent` command — i.e. one which is placed over the object with height `1ex`. The horizontal shift built into accents in slanted fonts is subtracted as well.

`\aboxsplit{`*accent*`}` — insert above and below the *accent* `0.2ex` of white space.

`\aboxjoin{`*accent*`}` — close up above and below the *accent* by `0.1ex`.

`\aboxsplitup{`*accent*`}` — insert `0.2ex` of white space only *below* the symbol used as *accent*, so that it becomes suitable as an *upper* accent separated by a clear blank space from the character.

`\aboxsplitdn{`*accent*`}` — as `\aboxsplitup`, except that the space is inserted *above* the symbol, so that the symbol is ready to be used as a *lower* accent.

`\aboxjoinup{`*accent*`}` — delete `0.1ex` of white space *below* the symbol used as *accent*, so that it is suitable as an *upper* accent joined with the main body of the character.

`\aboxjoindn{`*accent*`}` — as `\aboxjoinup`, except that the white space is subtracted *above* the symbol, and the *accent* is designed to be used as a *lower* accent joined with the character.

`\aboxnull{`*accent*`}` — make a box with zero height and zero depth but with the same baseline and contents as the argument.

`\aboxbar{`*symbol*`}` — create a horizontal bar which is as wide as is necessary to overline or to underline the composite *symbol*. The commands `\upaccentbar` and `\dnaccentbar` are defined with its help as:

```
\def\upaccentbar#1{\upaccent{\aboxsplit{\aboxbar{#1}}}{#1}}
\def\dnaccentbar#1{\dnaccent{\aboxsplit{\aboxbar{#1}}}{#1}}
```

5. Uppercase/lowercase sensitivity

5.1. Accents over *I* and *J*

The letters *I* and *J* should be in a separate class from all other letters used with accents. The essential difference is that the lowercase form implicitly contains the *dot* accent while the uppercase form does not. To make correct accented versions of lowercase *i* and *j* TEX Latin encodings contain special characters *dotless-i* and *dotless-j*, usually selected by the commands `\i` and `\j`. A "correctly" working `\accent` command should analyse the uppercase and lowercase forms of one of these letters, and select the proper glyph to produce the correct form of letter even after `\uppercase` and `\lowercase`.

When we consider the cyrillic writing systems, the capital *Ukranian I (Ï)* should be produced by `\"I`, while the lowercase *Ukranian i (ï)* uses the command `\"\i`. On the other hand, the Khakassian letter *dotted-I*, which has a dot in its uppercase as well as in its lowercase form, requires `\.I` to produce the uppercase variant *İ*, and an ordinary `i` to produce the lowercase variant *i*. Since it is assumed that the user can distinguish which letter he/she wants and can separate the uppercase and lowercase variants explicitly, it seems that the existence of two types of specification of lowercase and uppercase forms does not produce any troubles. The problems only really start when `\uppercase` and `\lowercase` get involved:

İ = `\.I`	`\uppercase{\.I}` = İ	`\lowercase{\.I}` = i̇
i = `i`	`\uppercase{i}` = I	`\lowercase{i}` = i
Ï = `\"I`	`\uppercase{\"I}` = Ï	`\lowercase{\"I}` = ï
ï = `\"\i`	`\uppercase{\"\i}` = ï	`\lowercase{\"\i}` = ï

Although in principle `\uppercase` and `\lowercase` can be encountered anywhere, in practice only `\uppercase` (and that only in LATEX headers) is used outside explicit user control. Roughly the LATEX mechanism for automatically created headers works as follows:

1. The header text *some-header-text* containing the text and TEX commands is created elsewhere.
2. The object *some-header-text* is expanded into non-expandable tokens.
3. The uppercase transformation over the list of non-expandable tokens is performed. The latter means that characters are substituted by their `\uccode` values while TEX primitive commands remain unchanged.
4. The result is substituted into the header of the page and the list of tokens is processed by TEX as normal.

(This is not quite what the LATEX source code does, but the result of "true" macro is close to this description.)

Long-established formats such as *Plain*, $\mathcal{AMS}$-TEX and LATEX 2.09 use the `\uppercase` and `\lowercase` commands in this sort of operation, while LATEX 2_ε uses `\MakeUppercase` and `\MakeLowercase`, whose behaviour is somewhat different. The transformation of the characters in old formats only uses the `\uccode`–`\lccode` values, while the LATEX 2_ε commands use an additional list (`\@uclclist`) of characters which use their own rule different from `\lccode`–`\uccode`. In particular, while the commands `\uppercase` and `\lowercase` do not affect `\i` or `\j`, LATEX 2_ε makes explicit provision for them so that `\MakeUppercase` treats them as might be hoped.

The letter *dotless-i* remains constant after `\MakeLowercase`, and is transformed into capital *I* by `\MakeUppercase`. The letter *I* is not transformed by `\MakeUppercase`, but `\MakeLowercase` transforms it into ordinary *i*. Finally, the letter *i* is transformed into *I* after `\MakeUppercase`, and is unchanged by `\MakeLowercase`. Similar transformations happen with $\jmath \longrightarrow J \longleftrightarrow j$. These properties allow the user to create text that will generate headers and footers properly as long as only one `\MakeUppercase` is used in the exercise, but creates funny effects if the user employs explicit `\uppercase`–`\lowercase` within the text[5]. The optional variant where a stress is put above some of these letters makes life even more unhappy.

The following macros analyse whether the text is being transformed by `\uppercase` or `\lowercase` commands and put the correct form of the letters. The commands `\i` and `\I` create the letters $\imath/I$, which are correctly processed by `\uppercase`–`\lowercase`, if those commands are carefully enough applied. Commands `\ii` and `\II` create the letters $i/\dot{I}$ with corresponding properties:

```
\edef\temp{\the\i} \chardef\idotless=\temp\relax
\def\i{\ifnum‘i=‘\i \idotless \else I\fi}
```

5 The problem becomes even more complex taking into account that there is also a letter $\dot{I}$ in the T1 encoding. It is converted to ordinary *i* by `\lowercase`, and into *I* after subsequent `\uppercase`.

```
\def\I{\ifnum`I=`\I I\else \idotless \fi}
\def\ii{\ifnum`i=`\i i\else \.I\fi}
\def\II{\ifnum`I=`\I \.I\else i\fi}
```

Finally, the ordinary letters `i` and `I` correspond to the case where the lowercase form has a dot while the uppercase form does not. These macros work because the control sequence name `\I` in the the construction ``` `\I ``` (and similarly `\i`) is conserved under `\uppercase`–`\lowercase` transformation, while the expression `` `I `` (and `` `i ``) is converted appropriately. It is worthwhile noting that even though the *then* and *else* parts of the macro are also affected by `\uppercase` and `\lowercase` the results are still correct, since the clause that is changed is always the one not selected:

`\uppercase\expandafter{\i}`=I	`\uppercase\expandafter{\ii}`=İ
`\lowercase\expandafter{\i}`=ı	`\lowercase\expandafter{\ii}`=i
`\uppercase\expandafter{\I}`=I	`\uppercase\expandafter{\II}`=İ
`\lowercase\expandafter{\I}`=ı	`\lowercase\expandafter{\II}`=i

It is more or less evident that the combination "lowercase-i-without-dot"$\longleftrightarrow$ "uppercase-I-with-dot" is of no interest, although a corresponding macro could be created.

Similar commands are intoduced for $\jmath/j$ and $J/\dot{J}$. It is essential that the commands `\i` and `\I`, `\ii` and `\II`, `\j` and `\J`, `\jj` and `\JJ` create the correct forms when the commands `\uppercase` and `\lowercase` affect them. They also work correctly when there are additional upper and lower accents created by the commands from Section 3.

These commands do not have the required effect under transformation by LaTeX 2_ε commands `\MakeUppercase` or `\MakeLowercase`: the solution is, however, very simple. The commands should be defined as encoding-dependent (so as to ensure robustness and to make use of the T1 encoding İ character if possible) and pairs "`\i\I`" and "`\ii\II`" should be added to the `\@uclclist`.

The rules created by the commands `\markchar` and `\MARKCHAR` described in Section 4 also transform into each other by similar application of `\uppercase` and `\lowercase`.

5.2. Ligatures "L+soft-sign" and "N+soft-sign" sensitive to uppercase/lowercase conditions

A similar problem of case-sensitivity exists in the writing systems for Serbian, Macedonian, Itelmen and Nganasan languages which use the ligatures "*L+soft-*

sign" and "*N+soft-sign*": **Љ**,**љ**,**Њ**,**њ**.As well as these uppercase and the lowercase variants shown above there is also a "title" variant[6] which is composed from the uppercase **Л** and **Н** and the bowl from the *lowercase* soft-sign **ь**. These ligatures (**Љ** and **Њ**) are used for titles where the first letter would otherwise be the uppercase variant while the rest of the text is composed of lowercase letters.

The command `\uppercase` applied to the title letter should transfer it into the normal uppercase form, and `\lowercase` — into normal lowercase form. Changing `\lccode`–`\uccode` values for the title letters (which could help in this case) is not allowed in LaTeX 2ε, and the list `\@uclclist` used in LaTeX 2ε to make upper/lowercase conversion is of no use since different assignments would be necessary for `\MakeUppercase` and `\MakeLowercase`.

The following macro uses the same idea as that used in Section 5.1 to distinguish `\uppercase` and `\lowercase` modes and to produce the correct letter as the output:

```
\def\makeCYRlje{%
   \ifnum`x=`\X        \CYRLJE  % \uppercase has been applied
   \else \ifnum `X=`\x \cyrlje  % \lowercase has been applied
   \else               \CYRlje  % natural title form
   \fi\fi
}
```

Here `\CYRLJE` corresponds to the uppercase form Љ, `\cyrlje` to the lowercase form љ, and `\CYRlje` to the title form. Similar commands can be defined for the letter "N+soft-sign".

Acknowledgements

We are grateful to Olga Lapko, Mikhail Kolodin and Andrew Janishevskii for numerous useful discussions resulting in the appearance of the style file ACCENTBX.STY. The examples of TeX programming demonstrated by Kees van der Laan helped to make the package ACCENTBX.STY more effective and compact. The discussions with Jörg Knappen helped to understand better the base ideology of accents in T1 and EC/DC fonts. The extensive comments and recommendations of the referee of this paper, Bernd Raichle, enabled to improve greatly the description of the `\accent` algorithm in Section 2 and to correct numerous errors in suggested macro. Finally, Robin Fairbairns spent a lot of

[6] The Dutch ligature *IJ* also can exhibit a title form.

his time the polishing-English in this paper. We express our warmest thanks to all these TEXnicians and TEXperts.

This research was partially supported by a grant from the Dutch Organization for Scientific Research (NWO grant No 07–30–007).

Bibliography

[1] Victor Eijkhout. *TEX by Topic, a TEXnician's Reference.* Addison-Wesley, 1991.

[2] Donald E. Knuth. *The TEXbook.* Addison-Wesley, 1984 (reprinted with corrections in 1989).

[3] D.Salomon. NTG's Advanced TEX course: Insights & Hindsights. *MAPS* special issue, 1994.