The soul package

Melchior FRANZ

May 15, 1999

Abstract

This article describes the soul package\(^1\), which provides hyphenatable letterspacing (spacings out), underlining, and some derivatives such as majorules letterspacing that might be needed for high quality typesetting. All features are based upon a common mechanism that allows to typeset text syllable by syllable, where TeX's excellent hyphenation algorithm is used to find the proper hyphenation points. Two examples show how to use the provided interface to implement things such as 'an-a-lyzing syllables'.

Although the package is optimized for \texttt{HPEX} \(2\varepsilon\), it works with Plain \texttt{TEX} and with other packages, too. By the way, the package name soul is only a combination of the two macro names \texttt{\textbackslash so} (space out) and \texttt{\textbackslash ul} (underline)—nothing poetic at all...

Contents

1 Introduction 2
2 Typesetting rules 2
2.1 Theory 2
2.2 ... and Practice 2
3 Modes and options 3
3.1 \texttt{HPEX} \(2\varepsilon\) mode 3
3.2 Plain \texttt{TEX} mode 3
3.3 Command summary 3
4 Letterspacing 3
4.1 The macros 3
4.2 Some examples 5
4.3 Typesetting Fraktur 6
4.4 Dirty tricks 7
5 Underlining 7
5.1 Settings 7
6 How the package works 9
6.1 The kernel 9
6.2 The interface 9
6.3 Doing it yourself 10
6.4 Common restrictions 11
6.5 Known features (aka bugs) 12
7 The macros 14
7.1 The preamble 14
7.2 Common definitions 14
7.3 The \texttt{letterspacing} interface 18
7.4 The underlining interface 21
7.5 The \texttt{\textbackslash striking out} interface 23
7.6 The postamble 23
7.7 Additional hacks 23

---

\(^1\)This file has version number 1.3, last revised 1999/05/15.
I'd like to thank Stefan Ulrich for teaching me much about high quality typesetting, sending me dozens of error reports, and, finally, providing the 'example.cfg' configuration file. Without his help the package would only be half as good. And, no, he had nothing to do with minuscules letterspacing, underlining, and such...
1 Introduction

There are several possibilities to emphasize parts of a paragraph, where not all are considered to be good style. While underlining is commonly rejected, experts dispute about whether letterspacing should be used or not, and in which cases. If you are not interested in such debates, you may well skip over the next section.

2 Typesetting rules

2.1 Theory...

To understand the expert’s arguments we have to know about the conception of page greyness. The sum of all characters on a page represents a certain amount of greyness, provided that the letters are printed black onto white paper.

Jan Tschichold [5], a well known and recognized typographer, accepts only forms of emphasizing, which do not disturb this greyness. This is only true of italic shape, caps, and caps-and-small-caps fonts, but not of ordinary letterspacing, underlining, bold face type, and so on, all of which appear as either dark or light spots in the text area. In his opinion emphasized text shall not catch the eye when running over the text, but rather when actually reading the respective words.

Other, less restrictive typographers [6] call this kind of emphasizing to be ‘integrated’ or ‘aesthetic’, while they describe ‘active’ emphasizing apart from it, which actually has to catch the reader’s eye. To the latter group belong commonly despised things like letterspacing, demibold face type and even underlined and colored text!

On the other hand, Tschichold suggests to space out caps and caps-and-small-caps fonts on title pages, headings and running headers from 1 pt up to 2 pt. Even in running text readability of uppercase letters should be improved with slight letterspacing, since (the Roman) majuscules don’t look right, if they are spaced like (the Carolingian) minuscules.2

2.2 ... and Practice

However, in the last centuries letterspacing was excessively used, underlining at least sometimes, because the old Fraktur fonts could not use capitals or italic shape for emphasizing. This tradition is widely continued until today.

The Duden [1], a well known German dictionary, tells us how to space out properly: Punctuation marks are spaced out like letters, except quotation marks and periods. Numbers are never spaced out. The German syllable -sche is not spaced out in cases like “der Virchow sche Versuch”3. In the old German Fraktur fonts the ligatures ch, ck, sz (ß), and tz are not broken within spaced out text.

While some books follow all these rules [3], others don’t [4]. (In fact, most books in my personal library do not space out commas.)

\footnote{This suggestion is followed throughout this article, although Prof. Knuth already considered slight letterspacing with his \texttt{cmr10} fonts.}

\footnote{the Virchow experiment}
3 Modes and options

The soul package has a \LaTeXe mode, which is selected if the \texttt{documentclass} command can be found, and a plain \TeX mode, which is selected otherwise. These modes differ in some points:

3.1 \LaTeXe mode

This mode provides a package option \texttt{capsdefault} (see section 4.1) and two package options \texttt{nooverlap} and \texttt{overlap}, where the latter is selected by default. These options deal with the way underlines are typeset. They are described in section 5.3, but you’ll hardly ever need to know about them. The \LaTeXe mode provides an intelligent \texttt{\caps} command and makes commands ‘robust’ where it is desired. Furthermore, it tries to load a file ‘soul.cfg’, where local stuff is to be placed in. (See the file ‘example.cfg’, which implements a fairly complete \texttt{\caps} data base.)

3.2 Plain \TeX mode

This mode implements the respective options as commands \texttt{\overlap} and \texttt{\nooverlap}, and provides a simplified \texttt{\caps} command. The ‘fragile’ commands \texttt{\so}, \texttt{\caps}, \texttt{\ul}, and \texttt{\st} are to be protected by the user, if they are used in expanding environments such as \texttt{\write} arguments.

3.3 Command summary

or: Tribute to the impatient

Those commands marked with an asterisk are only accessible in \LaTeXe mode:

\begin{center}
\begin{tabular}{ll}
\texttt{\so{letterspacing}} & \texttt{letterspacing} \\
\texttt{\caps{CAPITALS, Small Capitals}} & \texttt{CAPITALS, SMALL CAPITALS} \\
\texttt{\ul{underlining}} & \texttt{underlining} \\
\texttt{\st{striking out}} & \texttt{striking out} \\
\texttt{\sodef\cs{1em}{2em}{3em}} & \texttt{define new spacing command \cs} \\
\texttt{\resetso} & \texttt{reset \so dimensions} \\
\texttt{\capsreset*} & \texttt{clear caps data set} \\
\texttt{\capsdef{\\\\}{1em}{2em}{3em}*} & \texttt{define (default) \caps data entry} \\
\texttt{\capssave\cs*} & \texttt{save \caps data set under name \cs} \\
\texttt{\setul{1ex}{2ex}} & \texttt{set \ul dimensions} \\
\texttt{\resetul} & \texttt{reset \ul dimensions} \\
\texttt{\setuldepth{y}} & \texttt{set underline depth to depth of y}
\end{tabular}
\end{center}

4 Letter spacing

4.1 The macros

\texttt{\so} The base macro for letterspacing is called \texttt{\so}. It typesets the given argument with a certain amount of \emph{inter-letter space} between every two tokens, \emph{inner space} between words, and \emph{outer space} before and after the spaced out text in case there
is a space preceding and following, whereby all kerning values are automatically reinserted at the right places. To enforce normal spaces instead of outer spaces, you can ‘hide’ preceding spaces with a \null before the \so command, and following spaces with any other token such as \relax or just an opening or closing brace afterwards.

The values are predefined for typesetting facsimiles mainly with Fraktur fonts. You can define your own spacing macros or overwrite the original \so meaning using the macro \sodef:

\sodef{cmd}{{(font)}}{{(inter-letter space)}}{{(inner space)}}{{(outer space)}}

The space dimensions, all of which are mandatory, should be defined in terms of \em letting them grow and shrink with the respective fonts.

Example: \sodef{an}{{.2em}}{{1em plus1em}}{{2em plus.1em minus.1em}}

\resetso after which you can type ‘\an{example}’ to get ‘e x a m p l e’. The \resetso command resets \so to its original meaning.

\caps For typesetting caps or caps-and-small-caps fonts there are two different \caps commands predefined with only slight spacing, which are mainly thought to be used in running text (see section 2.1). The following lines show the effect of \caps in comparison with the normal textfont and with small-capitals shape:

\normalfont DONAUDAMPFSCHIFFAHRTSGESELLSCHAFT
\scshape DONAUDAMPFSCHIFFAHRTSGESELLSCHAFT
\caps DONAUDAMPFSCHIFFAHRTSGESELLSCHAFT

In \texttt{plain} \TeX mode the \caps command is simply defined with \sodef. It executes a command \capsfont that is ignored by default and may be used to select a particular font.

Example: \font\capsfont=cmcsc10 \caps{Tschichold}

The \LaTeX version is slightly more complicated. It uses a small list as a ‘database’ to hold sets of standard values for different fonts, shapes, etc., which are then selected automatically.

\capsdef New fonts may be added to this list using the \capsdef command, which takes five arguments. The first argument describes the font with encoding, family, series, shape, and size, each optionally (e.g. OT1/cmr/m/n/10 for this very font, or only /ppl//12 for all palatino fonts at size 12 pt). The size entry may also contain a size range (5-10), where zero is assumed for an omitted lower boundary (-10) and a very, very big number for an omitted upper boundary (5-). The upper boundary is not included in the range, so, in the example below, all fonts with sizes greater or equal 5 pt and smaller than 15 pt are accepted (5 pt \leq \text{size} < 15 pt). The second argument may contain font switching commands such as \scshape, it may as well be empty or contain debugging commands (e.g. \message{*}). The remaining three, non-optional arguments are the spaces as described above.

Example: \capsdef{T1/ppl/m/n/5-15}{\scshape}{.16em}{.4em}{.2em}

The \LaTeX \caps command goes through the data list and takes the first matching set, so the order of definition is essential. There’s only one default set for all font combinations predefined, which can be overridden.

\capsreset The \capsreset command deletes all font sets except the default set, which can be overridden with a \capsdef command using the default identifier {/////}. 

4
This entry should be defined first, because it matches any font, so that no entry behind can ever be reached. The current \caps settings can be saved in a command sequence using the \capssave command. This allows to predefine different groups of \caps sets.

Example:

\capsreset
\capsdef{/cmss///}{10pt}{20pt}{30pt}
\capssave\widecaps
\capsreset
\capsdef{/cmss///}{.1pt}{.2pt}{.3pt}
\capssave\narrowcaps

\title{\caps{Yet Another Silly Example}}

If you have defined a bunch of sets for different fonts and sizes, you may lose control over what fonts are used by the package. With the package option \capsdefault selected, \caps prints its argument underlined, if no set was specified for a particular font and the default set had to be used.

4.2 Some examples

See also section 6.4.

| Ordinary text can be typed in as usual. | \texttt{\so{electrical\_industry}} | \texttt{electrical\_industry} |
| \texttt{\so{man\-u\-script}} | \texttt{man\-u\-script} |

| \texttt{- works as usual.} | \texttt{\so{le\_th\'e\{\^a\}t\'re}} | \texttt{le\_th\'e\{\^a\}t\'re} |
| Tokens that belong together have to be grouped, text inside groups is not spaced out. Grouped text must not contain hyphen points. | \texttt{\so{just\_an\{{\hbox{example}}\}}} | \texttt{just\_an\example} |

To prevent material with hyphen points from being spaced out, you have to put it in an \hbox (\mbox) with two pairs of braces around it. However, it’s better to end \texttt{spacing\ out} before and restart it afterwards.
### Punctuation marks are spaced out, if they are put into the group.

- `\{inside\}_{i} & \{outside\}_{o}. `inside. & outside.`

### Spaceout skips may be removed by typing `\<`. See also section 6.5. It's, however, desirable to put the quotation marks out of the argument.

- `\{''\}<Pennsylvania<''\} `“Pennsylvania”`

### Numbers should never be spaced out.

- `\{1\}<3_{December}_{1995}\} `13 December 1995`

### `\slash`, `\hyphen`, `\endash`, and `\emdash` allow hyphenation before and after the break point.

- `\{input\slash_{output}\} `input/output`

### `\hyphen` must not be used for leading hyphens.

- `\{dots_{and}_{\hbox{-}}\} `\...\ and - jet`

### The `\~`-command inhibits line breaks. A space `\und` is mandatory here to mark the word boundaries.

- `\{unbreakable\~_{space}\} `unbreakable space`

### `\` works as usual. Additional arguments like * or vertical space are not accepted. Mind the space.

- `\{broken\`\line\} `broken line`

### The braces keep TeX from discarding the space.

- `\{pretty awful\{break\}_{test}\} `pretty awful test`

---

### 4.3 Typesetting Fraktur

The old German fonts\(^4\) deserve some additional considerations. As stated above, the ligatures ch, ck, sz (ß), and tz have to remain unbroken in spaced out Fraktur text. This may look strange at first glance, but you'll get used to it:

**Example:** `\textfrak{\{S\{ch\}u\{tz\}vorri\{ch\}tung\}}`

\(^4\)See the great old German fonts, which Yannis Haralambous kindly provided, and the oldgerm and yfonts package as their \TeX{} interfaces.
You already know that grouping keeps the soul mechanism from separating such ligatures. This is quite important for s; a*, and ”a. As hyphenation is stronger than grouping, especially the sz may cause an error, if hyphenation happens to occur between the letters s and z. (\TeX hyphenates the German word auszer wrongly like aus-zer instead of like au-szer, because the German hyphenation patterns do, for good reason, not see sz as ‘ß’.) In such cases you can protect tokens with the weird sequence e.g. \mbox{sz} or a properly defined command. The \ss command, which is defined by the yfonts package, and similar commands will suffice as well.

Especially the ‘ygoth’ font with its many ligatures is error-prone. You will have to assist the soul package in protecting or separating some of the ligatures as mentioned in section 6.4/number 6. This particular font, however, is probably too beautiful to get spaced out or underlined, anyway.

### 4.4 Dirty tricks

Narrow columns are hard to set, because they don’t allow much spacing flexibility, hence long words often cause overfull boxes. A macro—let us call it \magstylepar—could use \ss to insert stretchability between the single characters. The following columns show some text typeset with such a funny definition at the left side and under plain conditions at the right side, both with a width of 6 pc.

| Some magazines and newspapers prefer this kind of spacing because it reduces hyphenation problems to a minimum. Unfortunately, such paragraphs aren’t especially beautiful. | Some magazines and newspapers prefer this kind of spacing because it reduces hyphenation problems to a minimum. Unfortunately, such paragraphs aren’t especially beautiful. |
| --- |

Such a macro could only set one paragraph at once, it would be subject to the same restrictions as mentioned in section 6.4, so it would really be a dirty trick rather than a glorious novelty...

### 5 Underlining

The underlining macros are my answer to Prof. Knuth’s exercise 18.26 from his \TeXbook. :-) All said about the macro \ul is also true of the striking out macro \st, which is in fact derived from the former.

#### 5.1 Settings

\setul The predefined underline depth and thickness work well with most fonts. They can be changed using the macro \setul.

\setul{\{underline depth\}}{\{underline thickness\}}
Either dimension can be omitted, in which case there has to be an empty pair of braces. Both values should be defined in terms of \texttt{ex}, letting them grow and shrink with the respective fonts. The \texttt{\resetul} command restores the standard values.

Another way to set the underline depth is to use the macro \texttt{\setuldepth}. It sets the depth such that the underline’s upper edge lies 1 pt beneath the given argument’s deepest depth. If the argument is empty, all letters—i.e. all characters whose \texttt{\catcode} currently equals 11—are taken:

Examples: \texttt{\setuldepth{ygp}}, \texttt{\setuldepth\strut}, \texttt{\setuldepth{}}

\section*{5.2 Some examples}

See also section 6.4.

\begin{tabular}{lll}
\textit{Ordinary text can be typed in as usual.} & \texttt{\ul{electrical\_industry}} & \texttt{electrical industry} \\
& \texttt{\ul{electrical\_industry}} & \texttt{electrical industry} \\
\end{tabular}

\begin{tabular}{lll}
\texttt{\ul{\~a}} & \texttt{\ul{\~a}script} & \texttt{man-
script} \\
& \texttt{\ul{\~a}script} & \texttt{man-
script} \\
\end{tabular}

\begin{tabular}{lll}
\textit{Tokens that belong together have to be grouped. Grouped text must not contain hyphen points.} & \texttt{\ul{le\_th\_tre}} & \texttt{le \~the\_tre} \\
& \texttt{\ul{le\_th\_tre}} & \texttt{le \~the\_tre} \\
\end{tabular}

\begin{tabular}{lll}
\textit{The \texttt{\~a}command inhibits line breaks. A space is mandatory here to mark the word boundaries.} & \texttt{\ul{unbreak\_space}} & \texttt{unbreak-
able space} \\
& \texttt{\ul{unbreak\_space}} & \texttt{unbreak-
able space} \\
\end{tabular}

\begin{tabular}{lll}
\textit{The braces keep \TeX{} from discarding the space.} & \texttt{\ul{pretty\_awful\_break\_test}} & \texttt{pretty awful test} \\
& \texttt{\ul{pretty\_awful\_break\_test}} & \texttt{pretty awful test} \\
\end{tabular}

\section*{5.3 The dvips problem}

Underlining and \texttt{\sout{}} build up their lines with many short line segments. If you used the ‘dvips’ program with default settings, you would get little gaps on some places, because the \texttt{maxdrift} value allows the single objects to drift this many pixels from their real positions.

There are two ways to avoid the problem, where the \texttt{soul} package chooses the second by default:
1. Set the maxdrift value to zero, e.g.: \texttt{dvips -e 0 file.dvi}. (This is probably not a good idea, since the letters may then no longer be spaced equally on low resolution printers.)

2. Use the overlap option. This option causes the single line segments to overlap each other letting them stick out 0.5 pt to the left and to the right. The option \texttt{nooverlap} turns this overlapping off.

\begin{itemize}
    \item Use the commands \texttt{\nooverlap} and \texttt{\overlap} for non-\LaTeX{} packages. \\
    Unlike the \LaTeX{} options these commands can also be used after loading the package.
\end{itemize}

6 How the package works

6.1 The kernel

\texttt{Letterspacing}, underlining, and \texttt{strikout} use the same kernel mechanism. It typesets the given material in a 1sp wide \texttt{vbox} which provides that every possible hyphenation point leads to a new line within this box. After the number of all lines (i.e. syllables) is counted, and the respective lengths are stored (pass one: \texttt{analyzing}), the tokens are scanned again, and their length is added to a register. Always if the length of the next stored syllable is obtained (pass two: \texttt{reconstruction}), the required actions take place. These are controlled by the \texttt{interface}'.

6.2 The interface

The package uses six interface macros that are to be defined according to the required task.

\begin{itemize}
    \item \texttt{\SOUL@preamble} \quad \texttt{P} \quad \texttt{executed once at the beginning}
    \item \texttt{\SOUL@interword} \quad \texttt{\ ο} \quad \texttt{executed between every two words}
    \item \texttt{\SOUL@everyhyphen} \quad \texttt{H} \quad \texttt{executed at every implicit hyphen point; It may access the letter kern in \texttt{\dimen@}, the hyphen kern in \texttt{\dimen3}, and the hyphen in \texttt{\box2}. This interface macro has to reinsert the hyphen kern, it may remove a character kern inserted by \texttt{\SOUL@everytoken}, if necessary.}
    \item \texttt{\SOUL@everytoken} \quad \texttt{T} \quad \texttt{executed after scanning a token; It may access the current token in \texttt{\SOUL@actual}, the next two tokens in \texttt{\SOUL@prefetch} and \texttt{\SOUL@pprefetch}, where \texttt{\SOUL@next} points to the first of them, which doesn’t contain an \texttt{\empty} token. The character kern is accessible via \texttt{\dimen@}. This interface macro is responsible for reinserting the character kern.}
    \item \texttt{\SOUL@everysyllable} \quad \texttt{S} \quad \texttt{executed after scanning a whole syllable; not used by the package definitions so far; If you want to access the whole syllable, you have to let \texttt{\SOUL@everytoken} collect the tokens.}
\end{itemize}
The above table’s middle column shows a mark that indicates in the following examples, when the respective macros are executed:

- **\SOUL@everytoken E** executed once at the end

<table>
<thead>
<tr>
<th>Word</th>
<th>\SOUL@everytoken</th>
<th>\SOUL@interword</th>
<th>\SOUL@everyhyphen</th>
<th>\SOUL@everyhyphen belongs to the left syllable.</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>\SOUL@everytoken</td>
<td>\SOUL@interword</td>
<td>\SOUL@everyhyphen</td>
<td>\SOUL@everyhyphen belongs to the left syllable.</td>
</tr>
<tr>
<td>Example</td>
<td>\SOUL@everytoken</td>
<td>\SOUL@interword</td>
<td>\SOUL@everyhyphen</td>
<td>\SOUL@everyhyphen belongs to the left syllable.</td>
</tr>
<tr>
<td>Beta Test</td>
<td>\SOUL@everytoken</td>
<td>\SOUL@interword</td>
<td>\SOUL@everyhyphen</td>
<td>\SOUL@everyhyphen belongs to the left syllable.</td>
</tr>
</tbody>
</table>

It’s only natural that these examples, too, were automatically typeset by the soul package using a special interface:

\DeclareRobustCommand*\an{%
  \def\SOUL@preamble{$^{^P}$}%
  \def\SOUL@interword{\texttt{\char127}}%
  \def\SOUL@postamble{$^{^E}$}%
  \def\SOUL@everyhyphen{$^{^H}$}%
  \def\SOUL@everysyllable{$^{^S}$}%
  \def\SOUL@everytoken{\SOUL@actual$^{^T}$}%
  \SOUL%
}

### 6.3 Doing it yourself

#### 6.3.1 Defining a new interface

Let’s define an interface that allows to typeset text with a centered dot at every hyphen point. The name of the macro shall be \sy (for syllables). Since the soul mechanism is highly fragile, we use the \LaTeX command \DeclareRobustCommand, so that the \sy macro can be used even in section headings etc.

\DeclareRobustCommand*\sy{%
  We only set \lefthyphenmin and \righthyphenmin to zero at the beginning. All changes are restored automatically, so there’s nothing to do at the end.

  \def\SOUL@preamble{\lefthyphenmin=0 \righthyphenmin=0 }%
  \let\SOUL@postamble=\relax

  We only want simple spaces. Note that they are not provided by default!

  \let\SOUL@interword=\space

\sy
Output the current token and the character kern.

\def\SOUL@everytoken{\SOUL@actual\kern\dimen0}\%

We would like to put a centered dot (\texttt{\textbullet}) at every implicit hyphen point except when the line is broken there, in which case there should be the hyphen character, anyway. The \TeX\ primitive \texttt{\discretionary} takes three arguments: 1. pre-hyphen material (\texttt{\box2} contains the current hyphen sign); 2. post-hyphen material; 3. no-hyphen material. The \texttt{\dimen0} kern that was inserted by the last \texttt{\SOUL@everytoken} command has to be removed. \texttt{\dimen3} contains the hyphen kern, which is not used by the CM/EC fonts, but, for example, by the \texttt{palatino} fonts.

\def\SOUL@everyhyphen{\kern-\dimen0\discretionary
\{\kern\dimen3\unhcopy\tw@\}{}
\{\hbox{\kern.5pt$\cdot$\kern.5pt}\}}\%

There’s nothing to do for \texttt{\SOUL@everysyllable}.

\let\SOUL@everysyllable\relax

Now that the interface is defined, we can start the mechanism.

\SOUL@

\textit{This lit\-tle macro will hard-ly be good e\-nough for lin\-guists, al\-though it uses \TeX’s ex\-cel\-lent hy\-phen\-a\-tion al\-go\-rithm, but it is at least a nice al\-ter\-na\-tive to the \texttt{\showhyphens} com\-mand.}

6.3.2 Modifying an interface

It’s of course not necessary to reinvent the wheel. The following example uses the underlining interface with a modified \texttt{\textbf{striking out}} preamble. Guess what it does... ;-(

\DeclareRobustCommand*\censor{%
\SOUL@ulbody
\def\SOUL@preamble{\setul{}{2.5ex}\SOUL@stpreamble}%
\SOUL@}

6.4 Common restrictions

The \texttt{soul} mechanism is quite complicated, so you shouldn’t be surprised that there are a couple of restrictions to bear in mind:

1. \texttt{soul} arguments must not contain more than one paragraph. In other words, they must not contain a \texttt{\par (endgraf)} command, but that shouldn’t really be considered to be a restriction.

2. Fonts can \textit{not} be changed within a \texttt{soul} argument. Instead you have to stop spacing out and underlining, etc., change the font, and then restart it. It’s, however, better to avoid such cases at all.
3. The input text must not contain discretionary hyphens. Thus you have to handle cases like the German word Zu\textbackslash discretionar\-\{y\}\{k-\}{}\{c\}ker by yourself.

4. The soul mechanism doesn’t recognize -, --, and ---. Instead, you have to use the commands \textbackslash hypen, \textbackslash dash, and \textbackslash emdash, respectively. The command \textbackslash slash is internally redefined and works as usual.

5. The mechanism needs $\textbackslash u_{10}$ (a normal space with $\textbackslash catcode$ 10) to separate words. Thus, you have to keep \TeX\ from discarding spaces after commands, e.g.: \textbackslash so\{first line\textbackslash break\}second line\}

6. Ligatures are generally separated. Since the width of a ligature may differ from the overall width of the concerned characters, these might be displaced. Although the effect is hardly visible with most fonts, you can iron it out, if you either force the characters together using an \textbackslash mbox, or separate them explicitly using a \textbackslash > in between.

Some ligatures cause displacements though, which are not neglectable. The ‘ygoth’ font, for example, replaces ‘a’ and ‘e’ by a much narrower ‘æ’ character. That’s why you should either type \textbackslash so\{\textbackslash mbox\{ae\}\textbackslash ra\textbackslash risch\}, or \textbackslash so\{a\textbackslash er\textbackslash risch\}. Unfortunately, both versions disable automatical hyphenation, so you have to give some hints. (This particular problem doesn’t encounter with fonts where ‘æ’ is created by a command \textbackslash ae rather than by an entry in the \textit{ligtable}.)

7. Ambigous ligatures can cause troubles, which you can avoid by deciding whether you mean \textbackslash so\{ff\textbackslash f\} or \textbackslash so\{f\textbackslash ff\}, but this is supposed to be a German problem only.

8. Commands that are based on the soul mechanism must not be nested. If you really need such, put the inner stuff in a box, and use this box.

\texttt{\newbox\anyboxname
\sbox\anyboxname{ \so{the worst} } }
\texttt{\ul{This is by far\{\usebox\anyboxname\}\example\!}}

yields: This is by far the worst example!

6.5 Known features (aka bugs)

There’s only one error message for the moment. It warns about failed reconstruction due to different length results in pass one (analyzing) and pass two (reconstruction).

Possible reasons are:

- You protected a hyphen point only with braces: ‘input’ would normally be hyphenated ‘in-put’. If you typed (for some mysterious reason) \textbackslash so\{i\textbackslash np\textbackslash ut\}, then pass one will see the hyphen point and thus report two syllables ‘in’ and ‘put’, while pass two will desperately try to reconstruct the length of ‘in’ with a token ‘i’ and a token ‘np’. You can solve the problem by typing \textbackslash so\{i\textbackslash \textbackslash mbox\{np\}\textbackslash ut\}
or, of course,
\DeclareRobustCommand*\np{\mbox{np}}
\so{i\np ut}

- You used -, --, or ---, instead of the commands \hyphen, \endash, and \emdash, respectively.

- You used the \inputenc package and stated a compound character in a section heading, caption, etc. The \inputenc package allows to use e.g. ‘ä’ instead of ‘"a’ in an input file, and that’s usually no problem for \soul. But if you use such a character in e.g. a section heading, that character gets decomposed when it is written to the .toc file. If that file is read in to typeset the table of contents, \soul issues an error. You can work around this cumbersome error by putting braces around that character, e.g.: \section{\so{Ger{ä}t}}

- Quite unlikely: You forgot the funny \| command at word boundaries: Some fonts have built-in kerning with the boundary character. The EC-font’s German opening quotes, for example, are followed by a certain kern, except when a word begins after them. Here again, the two passes disagree on how to hyphenate the argument. You can solve this problem by putting a \| command after the quotes to remove the unwanted kern.
\so{noch ein {,,\|}dummes{''} Beispiel}

This is a somewhat silly example, since you should have typed
\so{noch ein {,,}\<dummes\<{''} Beispiel},
anyway, in which case the \| would not have been necessary.

The \soul mechanism recovers from these errors by simply omitting the rest of the current syllable. To make finding the responsible syllable easier, a black square like ■ is put right after it.

References


7 The macros

7.1 The preamble

This piece of code makes sure that the package is only included once. This is automatically provided by \LaTeX{}, but not necessarily by other packages.

\[\expandafter\ifx\csname SOUL@\endcsname\relax\else\expandafter\endinput\fi\]

The following lines decide whether the package was loaded by \LaTeX{} or by another package, in which case the \LaTeX{} commands have to be provided (somehow). Older \LaTeX{} versions are not recognized as \LaTeX{}, but that shouldn’t be a problem.

\[\expandafter\ifx\csname documentclass\endcsname\relax\chardef\atcode=\catcode'@\catcode'@=11\def\DeclareRobustCommand*{\def}\def\providecommand*{\def}\def\DeclareOption#1#2{\expandafter\def\csname#1\endcsname{#2}}\PackageError#1#2#3{{\newlinechar'^^J\errorcontextlines\z@\edef\{\errhelp{#3}}\errmessage{Package \unexpanded{\@name} error: \unexpanded{#2}}}\else\NeedsTeXFormat{LaTeX2e}\ProvidesPackage{soul} [1999/05/15 v1.3 letterspacing/underlining (mf)]\fi\]

7.2 Common definitions

This macro starts the whole process. It expects that the interface macros (see 6.2) are already defined properly. The interface macros \SOUL@preamble{} and \SOUL@postamble{} are executed here.

\SOUL@ This macro starts the whole process. It expects that the interface macros (see 6.2) are already defined properly. The interface macros \SOUL@preamble{} and \SOUL@postamble{} are executed here.

\SOUL@ This macro starts the whole process. It expects that the interface macros (see 6.2) are already defined properly. The interface macros \SOUL@preamble{} and \SOUL@postamble{} are executed here.
Calls \texttt{\SOUL@handleword} as long as there is some material left.

\def\SOUL@donext{\expandafter\SOUL@handleword\SOUL@material\@@}

\texttt{\SOUL@handleword} Splits a word from \SOUL@material, and calls \SOUL@word for every word. The interface macro \SOUL@interword is executed in between.

\def\SOUL@handleword#1 #2\@@{\def\SOUL@material{#2}\let\SOUL@penalty\allowbreak\if$#1$\else\expandafter\SOUL@word{#1}\if$#2$\else\unskip\SOUL@penalty\SOUL@interword\fi\fi}

\texttt{\SOUL@word} This macro does the real hard work. The huuuuuge kern will help to catch errors.

\def\SOUL@word#1{\bgroup\def\SOUL@toks{#1\kern.5\maxdimen}\relax\relax}

Now the given word is typeset in \texttt{\box0} under circumstances, which enforce hyphenation at every possible point.

\def\SOUL@exhyphen{\kern\dimen@ii\penalty\z@}
\SOUL@analyzesyllables This macro decomposes \box0 removing box after box from the bottom. The length of a hyphen is subtracted from every box width except from the last, the lengths are stored in \SOUL@syllablelens separated by a slash. The number of syllables is counted in \count0.

\def\SOUL@analyzesyllables{%
  \setbox\z@=\vbox{\unvcopy\z@\unskip\unpenalty
  \global\setbox\@ne=\lastbox}%
  \ifvoid\@ne
    \let\SOUL@\relax
  \else
    \setbox4=\hbox{\unhbox\@ne}%
    \dimen@=\wd4
    \ifnum\count@>\z@
      \advance\dimen@-\dimen@ii
    \fi
    \edef\SOUL@syllablelens{\the\dimen@/\SOUL@syllablelens}%
    \advance\count@\@ne
    \let\SOUL@\SOUL@analyzesyllables
  \fi\SOUL@}

\SOUL@donextsyllable This macro asks for the length of the next syllable to be set, and outputs tokens until the syllable is complete. The interface macro \SOUL@everysyllable is executed after scanning a whole syllable. The interface macro \SOUL@everyhyphen is executed at every hyphen point.

\def\SOUL@donextsyllable{%
  \ifnum\count@>\z@
    \advance\count@\m@ne
    \dimen@=\z@
    \SOUL@getsyllabellength
    \let\SOUL@hflag=y%
    \SOUL@donexttoken
    \SOUL@everysyllable
    \ifnum\count@>\z@
      \ifx y\SOUL@hflag
        \SOUL@everyhyphen
      \fi
    \fi
    \let\SOUL@\SOUL@donextsyllable
  \else
    \let\SOUL@\relax
  \fi\SOUL@}

\SOUL@getkern This macro detects the inter-character kern between parameter \#1 and \#3 and returns it in \#2, which has to be a \dimen register.

\def\SOUL@getkern#1#2#3{%
  \setbox4=\hbox{#1#3}\wd4
  \setbox4=\hbox{#1\null#3}\advance\dimen@-\wd4}

\SOUL@donexttoken This macro outputs all tokens and returns, when the current syllable is complete. First of all, the tokens in \SOUL@prefetch and \SOUL@pprefetch are shifted, and a new token is read to \SOUL@pprefetch.
\def\SOUL@donexttoken{%
  \let\SOUL@actual\SOUL@prefetch
  \let\SOUL@prefetch\SOUL@pprefetch
  \SOUL@gettoken

  Now \SOUL@next is set equal to \SOUL@prefetch or to \SOUL@pprefetch. The character kern between \SOUL@actual and \SOUL@next is taken and stored in \dimen@, after which the current token’s length is added to the length register. \dimen3 is set to the hyphen kern.

  \expandafter\ifx\SOUL@prefetch\empty
    \let\SOUL@next\SOUL@pprefetch
  \else
    \let\SOUL@next\SOUL@prefetch
  \fi

  \SOUL@getkern\SOUL@actual\dimen@\SOUL@next
  \setbox\z@\hbox{\SOUL@actual}\
  \advance\dimen5\wd\z@

  \SOUL@getkern\SOUL@actual{\dimen3}\SOUL@hyph
  \let\SOUL@\relax

  If the current syllable length is yet obtained or, at least, after adding the hyphen kern, we stop, otherwise we add the character kern and continue. If we already got too far, issue an error message and recover.

  \ifdim\dimen5=\dimen4
    \SOUL@everytoken
  \else
    \dimen7=\dimen3
    \advance\dimen7=\dimen5
    \ifdim\dimen7=\dimen4
      \SOUL@everytoken
    \else
      \ifdim\dimen5=\dimen4
        \SOUL@error
        \count@=\z@
      \else
        \advance\dimen5=\dimen@
        \SOUL@everytoken
      \fi
    \fi
  \fi}

\SOUL@error\{\vrule\@height.8em\@depth.2em\@width1em%
\PackageError{soul}{Reconstruction failed}{}
\}

\SOUL@getkern\SOUL@actual\dimen0\SOUL@next
\setbox\z@\hbox{\SOUL@actual}\
\advance\dimen5=\wd\z@}
\SOUL@error\{\vrule\@height.8em\@depth.2em\@width1em%
\PackageError{soul}{Reconstruction failed}{}
\}

These macros split one length information from the macro \SOUL@syllablelens and return it in \dimen4.

\def\SOUL@getsyllablelength{%
  \expandafter\SOUL@splitlen\SOUL@syllablelens\@@}
\def\SOUL@splitlen#1/#2\@@{%
  \dimen4=#1\def\SOUL@syllablelens{#2}}
\SOUL@gettoken These macros split a single token from the macro \SOUL@toks and return it in the macro \SOUL@pprefetch.
\SOUL@splittok
\SOUL@gettoken\%\expandafter\SOUL@splittok\SOUL@toks\@@\def\SOUL@pprefetch{\#1}\def\SOUL@toks{\#2}
\SOUL@imhyphen These macros are needed to handle implicit and explicit hyphens. The macro \SOUL@exhyphen has to come with every explicit hyphen (see below).
\SOUL@exhyphen \SOUL@imhyphen\SOUL@exhyphen\SOUL@exhyphen\penalty\exhyphenpenalty
\SOUL@allowhyph This macro is part of the \texttt{german} package. It allows hyphenation after non-letters by simply ending the word with an invisible and unbreakable skip.
\SOUL@allowhyph\SOUL@allowhyph\SOUL@allowhyph\penalty\@M \hskip\z@skip
\SOUL@hyphen These macros allow the soul mechanism to recognize explicit hyphen commands. This is necessary, since it has to subtract the hyphen width from all implicitly hyphenated syllables, except from the last.
\SOUL@hyphen\SOUL@endash\SOUL@slash
\SOUL@endash\SOUL@slash
\SOUL@sobody Define the interface macros. All tokens are output by \SOUL@everytoken so there's nothing to do for \SOUL@everysyllable. The macro \SOUL@interword requests a simple space, knowing that this space will be increased by the macro \SOUL@preamble.
\SOUL@sobody\let\SOUL@preamble\SOUL@sopreamble\let\SOUL@interword\space\let\SOUL@postamble\SOUL@sopostamble\let\SOUL@everyhyphen\SOUL@soeveryhyphen\let\SOUL@everysyllable\relax\let\SOUL@everytoken\SOUL@soeverytoken
\SOUL@sopreamble Set the proper \texttt{spaceskip}, fix the \textit{inner space}. This can't be done globally because it depends on the current font. Switch to horizontal mode, and add an \textit{outer space} in case it comes right after a space.
\SOUL@sopreamble\spaceskip\SOUL@soinner\skip@\SOUL@soskip\let\empty\leavevmode\ifdim\lastskip>.25em\unskip\hskip\SOUL@soouter\fi
Add *outer space* at the end in case one of \( \text{ } \) (a normal space, or any character with category code 10), \( \text{ } \), *space*, ~ (unbreakable space), or \@xobeysp follows.

\def\SOUL@sopostamble{%
\global\skip@\SOUL@soouter
\aftergroup\SOUL@socheck}
\def\SOUL@socheck{
\futurelet\SOUL@next\SOUL@sodospace}
\def\SOUL@sodospace{\bgroup
\def\{\let\{= }
\def\~##1{\hskip\skip@\egroup}\
\ifx\SOUL@next\%
\def\~ {\hskip\skip@\egroup}\
\else\ifx\SOUL@next~
\def\~~{\nobreak\hskip\skip@\egroup}\
\else\ifx\SOUL@next\%
\else\ifx\SOUL@next\space
\else\ifx\SOUL@next\@xobeysp
\else
\let\~\egroup
\fi\fi\fi\fi\fi\~}
\SOUL@soeveryhyphen Insert the \texttt{discretionary} command that removes the inter-letter kern and provides the hyphen kern and the hyphen character in case hyphenation takes place.
\def\SOUL@soeveryhyphen{%
\ifx\SOUL@prefetch\SOUL@noskip\else
\discretionary{\advance\dimen3-\dimen@}{\ifdim\dimen3=\z@\else\kern\dimen3\fi}
\unhcopy\tw@}{\{}%
\fi}
\SOUL@soeverytoken If the actual token is an implicit hyphen (\-) or a separator (\>), do nothing. Else, if it is the \texttt{noskip} command (\<), remove the last skip. In all other cases output the token followed by the inter-letter kern and a *spaceoutskip*.
\def\SOUL@soeverytoken{%
\ifx\SOUL@actual\SOUL@imhyphen\else
\ifx\SOUL@actual\SOUL@sep\else
\ifx\SOUL@actual\SOUL@noskip
\unskip\unpenalty
\else
\SOUL@actual
\ifdim\dimen@=\z@\else\kern\dimen0\fi
\nobreak\hskip\skip@
\fi
\fi
\fi}
\SOUL@noskip Define the command sequence that indicates that no *spaceoutskip* should appear and the separator that can be used to break ligatures.
\def\SOUL@noskip{\<}
\def\SOUL@sep{\>}
\SOUL@setso Set the *spaceoutskip*, *inner*, and *outer space*.
\def\SOUL@setso#1#2#3{%
\sodef Define a new ‘robust’ letterspacing command.

\resetso (Re-)Define the standard \so command with convenient default settings.

The \caps commands are to be defined different depending on whether \LaTeX or another package is used.

\capsreset Delete the caps data base and insert the default set.

\capsdef This macro adds an entry to the \caps data base.

\capssave This macro saves the current \caps settings in a macro with the given name. The settings remain unchanged.

These macros find the first matching entry in the \caps data base.
\gdef\SOUL@caps{\SOUL@sobody
  \def\SOUL@preamble{\SOUL@setso{#7}{#8}{#9}#6\SOUL@sopreamble}%
  \SOULq}%
\fi}
\def\SOUL@chk#1#2{\
  \if$#1$\else\def\~{#1}\fi
  \ifx#2\~\else\let\SOUL@match=0\fi
}\fi}
\def\SOUL@dimchk#1#2{\if$#1$\else\SOUL@rangechk{#2}#1--\@ne\@@\fi}
\def\SOUL@rangechk#1#2-#3-#4\@@{
  \count@=#4\n  \ifnum\count@>\z@\n    \ifdim#1\p@=#2\p@\else\let\SOUL@match=0\fi
  \else\n    \dimen@=\if$#2$\z@\else#2\p@\fi\n    \ifdim#1\p@<\dimen@\else\let\SOUL@match=0\fi\n    \dimen@=\if$#3$\maxdimen\else#3\p@\fi\n    \ifdim#1\p@<\dimen@\else\let\SOUL@match=0\fi
  \fi}
\caps
The \LaTeX\ \caps version gets its settings from the \caps data base.
\DeclareRobustCommand*\caps{\bgroup
  \def\\##1{\expandafter\SOUL@capsfind\##1/}\
  \SOUL@capsbase\aftergroup\SOUL@caps\egroup}
\def\SOUL@capsdefault{\capsdef{////}\SOUL@capsdfltfnt
  {.028em@plus.005em@minus.01em}\
  {.37em@plus.1667em@minus.1em}\
  {.37em@plus.111em@minus.14em}}
\let\SOUL@capsdfltfnt\scshape
\fi
\ul
\ul
\SOUL@ulbody

The underlining interface
\ul
\SOUL@ulbody
\SOUL@ulbody
\ul
\SOUL@ulbody
\SOUL@ulbody

\SOUL@ulbody
Set up the underlining interface and define the \ul command.
\ul
\SOUL@ulbody
\SOUL@ulbody
\SOUL@ulbody
\SOUL@ulbody
\SOUL@ulbody
\SOUL@ulbody

\SOUL@ulpreamble
Set the spaceskip and the underlinethickness and -depth.
\SOUL@ulpreamble
\SOUL@ulpreamble
\SOUL@ulpreamble
\SOUL@ulpreamble
\SOUL@ulpreamble
\SOUL@ulpreamble

21
Spaces are expandable and shrinkable, underlined skips.

```
\def\SOUL@ulinterword{\SOUL@ulleaders\hskip\spaceskip}
```

**\SOUL@uleverytoken** Output every token and character kern overlapped with the matching underline. The *overlap* option adds .5pt at every side.

```
\DeclareOption{nooverlap}{% 
  \def\SOUL@uleverytoken{% 
    \setbox\z@\hbox{\SOUL@actual\ifdim\dimen@=\z@\else\kern\dimen@\fi}% 
    \dimen@ii\wd\z@
    \unhbox\z@\llap{\SOUL@ulleaders\hskip\dimen@ii}%%
  }}
```

```
\DeclareOption{overlap}{% 
  \def\SOUL@uleverytoken{% 
    \setbox\z@\hbox{\SOUL@actual\ifdim\dimen@=\z@\else\kern\dimen@\fi}% 
    \dimen@ii\wd\z@\advance\dimen@ii\p@ 
    \unhbox\z@\llap{\SOUL@ulleaders\hskip\dimen@ii\kern-.5\p@}%%
  }}
```

**\SOUL@uleveryhyphen** Offer an underlined hyphen character, if hyphenation takes place, or a possibly dropped out underlined inter-character kern otherwise.

```
\def\SOUL@uleveryhyphen{\discretionary 
  \advance\dimen3-\dimen@ \ifdim\dimen3=\z@ \else\kern\dimen3\fi 
  \setbox4\hbox{\unhcopy\tw@} 
  \rlap{\SOUL@ulleaders\hskip\wd4}\box4{} 
  \hbox{\SOUL@ulleaders\hskip\dimen@}}
```

**\setul** Allow setting the *underlinedepth* and the *underlinethickness*.

```
\def\setul#1#2{% 
  \if$#1$\else\def\SOUL@@{#1}\fi 
  \if$#2$\else\def\SOUL@ulthickness{#2}\fi 
```

```
\resetul Provide convenient default settings.

```
\def\resetul{\setul{.65ex}{.1ex}}
```

**\SOUL@ulleaders** This macro is actually providing the underlines. It has to be followed by a \hskip.

```
\def\SOUL@ulleaders{\leaders\hrule\@depth\dimen3\@height\dimen6}
```

**\setuldepth** Set the *underlinedepth* according to the argument. Take all letters (including the commercial 'at') if no argument is given.

```
\def\setuldepth#1{% 
  \setbox\z@\hbox{\tracinglostchars\z@ 
  \ifx\SOUL@@\empty 
    \loop 
    \ifnum\catcode\count@=11\char\count@\fi 
  \ifnum\count@<\@cclv 
    \advance\count@\@ne 
  \repeat 
  \else 
    #1% 
  \fi% 
  \hskip\dimen@\advance\dimen@\p@ 
  \xdef\SOUL@uldepth{\the\dimen@}}
```

```
7.5 The striking-out interface

\st Striking out is nearly the same as underlining, ...

\DeclareRobustCommand*{\st}{\SOUL@ulbody
\let\SOUL@preamble\SOUL@stpreamble
\SOUL@}

\SOUL@stpreamble ... only the lines have to be raised 0.5 ex.

\def\SOUL@stpreamble{%
\dimen@=\SOUL@ulthickness
\dimen@i=-.5ex
\advance\dimen@i-.5\dimen@
\edef\SOUL@uldepth{\the\dimen@i}%
\SOUL@ulpreamble}

7.6 The postamble

Set the default options and values. In case we are in \LaTeX\ mode, add the \texttt{capsdefault} option, reset the \texttt{caps} data base, and include the local configuration file. Finally: exit.

\resetso
\resetul
\expandafter\ifx\csname documentclass\endcsname\relax
\catcode'@=11 % <-- TeX
\else
\DeclareOption{capsdefault}{%
{\AtBeginDocument{%
\def\SOUL@capsdfltfnt#1{\SOUL@ulbody\SOUL@ulpreamble}}}%
\ExecuteOptions{overlap}
\capsreset
\InputIfFileExists{soul.cfg}{\PackageInfo{soul}{Local config file soul.cfg used}}{}}
\ProcessOptions
\fi
\endinput

7.7 Additional hacks

\superspaceout If you are spacing out a lot, you are probably doing something wrong. If you are sure you aren’t, you may type in the following lines, after which you can access the \texttt{so} command by simply enclosing the respective words in circumflexes. Type “\texttt{space out}” to \texttt{space out}. This construction is fragile and can’t be used in floating arguments like section headings etc. It may, however, be used in mathematical environments without interfering with the circumflex’s superscript function.

\makeatletter % <-- LaTeX
% \chardef\atcode=\catcode'@ \catcode'\%=11 % <-- TeX
\gdef\superspaceout{\catcode'\=active
\def'\relax
To allow ^\text{x} constructions you have to execute this macro first. It restores the circumflex's superscript \texttt{\textbackslash catcode}.

\texttt{\textbackslash def\textbackslash offsuperspaceout\{\textbackslash catcode\textbackslash ^\textbackslash =7 \}}

\texttt{\textbackslash def\textbackslash subunderline}{\textbackslash catcode\textbackslash _\active}
\texttt{\textbackslash gdef\textbackslash subunderline}{\textbackslash catcode\textbackslash _\active}
\texttt{\def\textbackslash _\relax}
\texttt{\ifmmode\let\textbackslash next\sb\else\let\textbackslash next\SOUL@ul\fi\textbackslash next}\%
\texttt{\def\SOUL@ul\textbackslash ##1\textbackslash _\{\textbackslash ul\textbackslash ##1\}}
\}
\texttt{\makeatother} % \texttt{\textbackslash LaTeX}
\%
\texttt{\textbackslash catcode\textbackslash @\texttt{\textbackslash =\texttt{\textbackslash atcode}}} % \texttt{\textbackslash TeX}