ifnextok

—

`\IfNextToken` instead of `\@ifnextchar` Does Not Skip Blank Spaces*

Uwe Lück†

May 23, 2011

Abstract

The ifnextok package deals with the behavior of LaTeX's internal `\@ifnextchar` to skip blank spaces. This sometimes has surprising or for some users really *unwanted* effects, especially with brackets following `\\` where the user does *not* intend to specify an optional argument, rather wants that brackets are *printed*. The package offers commands and options for modifying this behavior, maybe limited to certain parts of the document source.
[It works!] It may also be useful with active characters in lieu of `\\`, e.g., the double quote `"` with `german.sty` or babel.

Keywords: macro programming, optional command arguments, manual line breaks

Contents

1 Installing and Calling **2**

2 The Package File **2**
2.1 Header (Legalize) 2
2.2 Outline 3
2.3 Caveats 3
2.4 The Main Command `\IfNextToken` 4
2.5 Patching Commands 4
2.6 Storing and Restoring 5
2.7 The Star Test 5
2.8 "Manual" Line Breaks 6

*This document describes version v0.1 of `ifnextok.sty` as of 2011/05/23.

†`http://contact-ednotes.sty.de.vu`

2.8.1 Outline of Implementation 6
2.8.2 "Normal" Manual Line Breaks 6
2.8.3 Manual Line Breaks in LaTeX Environments 7
2.9 Package Options 7
2.9.1 Behavior *without* Options 7
2.9.2 Option `newline` 8
2.9.3 Environments 8
2.9.4 "All Options" 8
2.10 Processing Options and Leaving the Package 8
2.11 VERSION HISTORY 8

1 Installing and Calling

The package file `ifnextok.sty` is provided ready, installation only requires putting it somewhere where TeX finds it (which may need updating the file-name data base).[1]

Below the `\documentclass` line(s) and above `\begin{document}`, you load `ifnextok.sty` (as usually) by

`\usepackage{ifnextok}` or by `\usepackage[`⟨*options*⟩`]{ifnextok}`

—⟨*options*⟩ described in Section 2.9.

2 The Package File

2.1 Header (Legalize)

```
1  \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2  \ProvidesPackage{ifnextok}[2011/05/23 v0.1 next token test (UL)]
3
4  %% Copyright (C) 2011 Uwe Lueck,
5  %% http://www.contact-ednotes.sty.de.vu
6  %% -- author-maintained in the sense of LPPL below --
7  %%
8  %% This file can be redistributed and/or modified under
9  %% the terms of the LaTeX Project Public License; either
10 %% version 1.3c of the License, or any later version.
11 %% The latest version of this license is in
12 %%     http://www.latex-project.org/lppl.txt
13 %% We did our best to help you, but there is NO WARRANTY.
14 %%
15 %% Please report bugs, problems, and suggestions via
16 %%
17 %%   http://www.contact-ednotes.sty.de.vu
18 %%
```

[1] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`

2.2 Outline

The ifnextok package deals with the behavior of LaTeX's internal `\@ifnextchar` to skip blank spaces. This sometimes has surprising or for some users really *unwanted* effects, especially with brackets following `\\` where the user does *not* intend to specify an optional argument, rather wants that brackets are *printed.* The package offers commands and options for modifying this behavior, maybe limited to certain parts of the document source. They are described in the sections below together with the presentation of the implementation.

As after multiletter commands blank spaces are skipped anyway (TeXbook p. 46f.), the package makes a difference only for one-symbol commands such as `\\`, or for active characters such as the double quote with `german.sty` and babel.

A little **overview:**

1. `\IfNextToken` is an alternative to `\@ifnextchar`, not skipping spaces (Section 2.4). This macro is the **low-level** backbone of all other modifications of LaTeX commands.

2. `\IfStarNextToken` is an alternative to `\@ifstar`, not skipping spaces, using `\IfNextToken` in lieu of `\@ifnextchar` (Section 2.7).

3. Some "**patching**" commands aim at modifying existing (LaTeX) macros without specifying the resulting new definition explicitly (Sections 2.5 and 2.7). As a package writer, you just must know which macros need to be modified and specify their names as arguments for the patching macros.

4. There are low-level commands `\INTstore` and `\INTrestore` for undoing modifications of existing macros (Section 2.6).

5. There are **high-level** commands for modifying `\\` and selecting LaTeX **environments** to be affected (Section 2.8). Package **options** execute some of them.

(TODO: how command names are composed)

2.3 Caveats

Only a few of the commands have been tested so far, and usage together with amsmath may require special care or fail altogether.

Switching into "don't-skip-spaces" mode *two times* without switching back into normal mode in between won't work with this version (v0.1) of the package (TODO: permanent aliases). You will get the

`Argument of` ⟨*patching*⟩ `has an extra }.`

error. This also applies to commands that have been issued by package options.

2.4 The Main Command \IfNextToken

`IfNextToken`⟨*match*⟩`{`⟨*if*⟩`}{`⟨*else*⟩`}` is the obvious variant of LaTeX's internal `\@ifnextchar` executing ⟨*if*⟩ if actually the *"very next"* token is ⟨*mathc*⟩ and executing ⟨*else*⟩ otherwise. If ⟨*match*⟩ is *not* a *space token* (LaTeX's `\@sptoken`) but the next token *is*, ⟨*else*⟩ is executed; while `\@ifnextchar` tries matching the next token after ensuing space tokens.

```
\newcommand{\IfNextToken}[3]{%
    \let\nextok@match=#1\def\nextok@if{#2}\def\nextok@else{#3}%
    \futurelet\@let@token\nextok@decide}
\def\nextok@decide{%
    \ifx\@let@token\nextok@match \expandafter\nextok@if
    \else                        \expandafter\nextok@else
    \fi}
```

`\NoNextSkipping` now switches into "don't-skip-spaces" mode "altogether" (however ...):

```
\newcommand*{\NoNextSkipping}{\let\@ifnextchar\IfNextToken}
```

This appears so dangerous to me that I don't want to support it much right now. `\RestoreNextSkipping` just switches back to LaTeX's original version, so some support for amsmath may be missing here.

```
\newcommand*{\RestoreNextSkipping}{%
    \let\@ifnextchar\kernel@ifnextchar}
```

Actually, because `\NoNextSkipping` does not affect `\kernel@ifnextchar`, those of LaTeX's commands using the latter still will skip spaces (with package version v0.1).

2.5 Patching Commands

`\INTpatch`⟨*replacer*⟩⟨*macro*⟩ replaces something in the definition of ⟨*macro*⟩ according to the replacement macro ⟨*replacer*⟩. This seems to work with the macros I thought of. It does *not* work when (for replacing `\@ifnextchar`) (a) there are *more* `\@ifnextchar`s in the macro to patch, or when (b) an `\@ifnextchar` is inside a pair of braces.

```
\newcommand*{\INTpatch}[2]{%
  \expandafter\expandafter\expandafter \def
    \expandafter\expandafter\expandafter #2%
      \expandafter\expandafter\expandafter {%
        \expandafter #1#2#1}}
```

`\NextTestPatch`⟨*macro*⟩ replaces `\@ifnextchar` in the definition of ⟨*macro*⟩ by `\IfNextToken`.

```
\newcommand*{\NextTestPatch}{\INTpatch\nextok@patch}
\def\nextok@patch#1\@ifnextchar#2\nextok@patch{#1\IfNextToken#2}
```

2.6 Storing and Restoring

`\INTstore`⟨*macro*⟩ stores the meaning of the macro ⟨*macro*⟩ in a special name space.

```
\newcommand*{\INTstore}[1]{%
    \expandafter\let\csname\INT@name#1\endcsname#1}
\newcommand*{\INT@name}{INTstore.\expandafter\@gobble\string}
```

`\INTrestore`⟨*macro*⟩ *r*estores the meaning of ⟨*macro*⟩ that is expected to having been stored with `\INTstore`:

```
\newcommand*{\INTrestore}[1]{%
    \expandafter\let\expandafter#1\csname\INT@name#1\endcsname}
```

2.7 The Star Test

Before a LaTeX line-break command tests for an optional argument, it tests for a star using `\@ifstar`, which in turn invokes `\@ifnextchar`. So already `\@ifstar` needs to be modified. We do not so much want to change `\@ifstar` altogether, rather we will replace it at some places by a non-skipping variant `\IfStarNextToken`, using the patching command `\StarTestPatch`⟨*macro*⟩. (`\@ifstar` has an argument and therefore cannot be patched as nicely as the line-break commands.)

```
\newcommand*{\IfStarNextToken}[1]{\IfNextToken*{\@firstoftwo{#1}}}
\newcommand*{\StarTestPatch}{\INTpatch\nextok@starpatch}
```

The macro to be patched may contain a `\par` (`\@centercr` is an example), so we need `\long`:

```
\long\def\nextok@starpatch#1\@ifstar#2\nextok@starpatch{%
    #1\IfStarNextToken#2}
```

`\StoreStarSkipping` stores the current meaning of `\@ifstar` ...

```
\newcommand*{\StoreStarSkipping}{\INTstore\@ifstar}
```

... so that it can be restored by `\RestoreStarSkipping`:

```
\newcommand*{\RestoreStarSkipping}{\INTrestore\@ifstar}
```

`\NoStarSkipping` renders `\@ifstar` non-skipping altogether:

```
\newcommand*{\NoStarSkipping}{\let\@ifstar\IfStarNextToken}
```

This again seems to be so dangerous that it will not be supported much with package version v0.1 (by a package option).

2.8 "Manual" Line Breaks

2.8.1 Outline of Implementation

In the first instance, the present package aims at rendering `\\` a command that interpretes a left-hand square bracket as a start of an optional argument only if the bracket is not preceded by any other token (apart from the star in `\\*`), especially not by a space token.

Indeed, an author may expect that when a bracket opens in a *different* line than the `\\`, then it will be *printed* rather than interpreted as an *optional-argument delimiter* (the package author has been such an author some times). Now, when the bracket only is in a line *following* the line carrying the `\\`, the end-line character normally produces a space token (TEXbook p. 47), so the present idea of implementation will cover the case of a bracket in the next line.

In `latex.ltx`, the names of the commands implementing the line break have some "pivot" part ⟨*pivot*⟩ that we can use to patch them in a uniform way. They are two in each case: The first starts with `\@`⟨*pivot*⟩ and invokes `\@ifstar`, the second starts with `\@x`⟨*pivot*⟩ and invokes the left-hand-bracket test. Both of them need to be patched.

2.8.2 "Normal" Manual Line Breaks

If I had been aware of the difficulties of this part, I probably would not have started writing this package, hoping it would be the work of about an hour.

`\@xnewline` must be patched in order to get a non-skipping version of the bracket test, and this patch suffices for the optional-argument goal.

The `\@ifstar` call is in `\@normalcr`; the latter is invoked by the robust version of `\\`. However, LaTeX defines `\@normalcr` by a `\let` referring to the result of `\DeclareRobustCommand\\` ...

Things seem to be easier when `\\` calls `\@normalcr` instead of `\\␣` (the latter is the effect of `\DeclareRobustCommand`), we are **interchanging** the roles of `\\␣` and `\@normalcr` (**caution!**). Then we just need to control `\@normalcr`:

```
\def\\{\x@protect\\\protect\@normalcr}
```

(Another **Caveat:** I do not understand `\x@protect`.)

`\StoreNewlineSkipping` *stores* the skipping behavior of `\\` outside special environments:

```
\newcommand*{\StoreNewlineSkipping}{%
    \INTstore\@normalcr \INTstore\@xnewline}
```

`\RestoreNewlineSkipping` *re*stores the skipping behavior of `\\` outside special environments:

```
\newcommand*{\RestoreNewlineSkipping}{%
    \INTrestore\@normalcr \INTrestore\@xnewline}
```

`\NoNewlineSkipping` *suppresses* skipping blank spaces with `\\` outside special environments:

```
\newcommand*{\NoNewlineSkipping}{%
    \StarTestPatch\@normalcr \NextTestPatch\@xnewline}
```

2.8.3 Manual Line Breaks in LaTeX Environments

The macros in the present section should modify LaTeX's `\\` in environments (⟨*env*⟩ being one of:) `center`, `tab`, `array`, and `tabular`. These *environment names* are the expected *arguments* of those macros. However, argument `center` also affects the `flushleft` and `flushright` environments, and `array` and `tabular` should also affect their enhanced variants from other LaTeX packages. When this internal structure of LaTeX changes, the present section may become obsolete ...

`\INTactOnEnv{`⟨*action1*⟩`}{`⟨*action2*⟩`}{`⟨*env*⟩`}` is the backbone of these macros. ⟨*action1*⟩ and ⟨*action2*⟩ are one of

`\INTstore`, `\INTrestore`, `\StarTestPatch`, `\NextTestPatch`.

⟨*action1*⟩ deals with `\@ifstar`, ⟨*action2*⟩ deals with `\@ifnextchar`:

```
\newcommand*{\INTactOnEnv}[3]{%
    \expandafter#1\csname  @#3cr\endcsname
    \expandafter#2\csname @x#3cr\endcsname}
```

`\StoreSkippingCRs{`⟨*env*⟩`}` *stores* the skipping behavior of `\\` in environments ⟨*env*⟩:

```
\newcommand*{\StoreSkippingCRs}{%
    \INTactOnEnv\INTstore\INTstore}
```

`\RestoreSkippingCRs{`⟨*env*⟩`}` *r*estores the skipping behavior of `\\` in environments ⟨*env*⟩:

```
\newcommand*{\RestoreSkippingCRs}{%
    \INTactOnEnv\INTrestore\INTrestore}
```

`\NotSkippingCRs{`⟨*env*⟩`}` *suppresses* space skipping of `\\` in environments ⟨*env*⟩:

```
\newcommand*{\NotSkippingCRs}{%
    \INTactOnEnv\StarTestPatch\NextTestPatch}
```

Do these commands work?
[Or do they not?]

2.9 Package Options

2.9.1 Behavior *without* Options

If the package is called without any option, it only defines `\IfNextToken`, `\IfStarNextToken` and the other package-writer or user commands, without actually changing behavior of any LaTeX command.

2.9.2 Option `newline`

Package option `newline` stores and disables space skipping for `\\` in "normal" mode according to Section 2.8.2:

```
\DeclareOption{newline}{\StoreNewlineSkipping\NoNewlineSkipping}
```

2.9.3 Environments

The next package options are just the environment names according to Section 2.8.3 (`center`, `tab`, `array`, `tabular`). Option ⟨*env*⟩ stores and disables the skipping behavior of `\\` in ⟨*env*⟩ environments. We abuse the our temporary macro `\nextok@match` from Section 2.4:

```
\def\nextok@match#1{%
    \DeclareOption{#1}{\StoreSkippingCRs{#1}\NotSkippingCRs{#1}}}
\nextok@match{center}
\nextok@match{tab}
\nextok@match{array}
\nextok@match{tabular}
```

2.9.4 "All Options"

Package Option `all` has the same effect as using the `newline` option and the environment package options `center`, `tab`, `array`, and `tabular` at once.

```
\def\nextok@match#1{\csname ds@#1\endcsname}
```

(... must not be changed before `\ProcessOptions` ...)

```
\DeclareOption{all}{%
    \nextok@match{newline} \nextok@match{center}
    \nextok@match{tab} \nextok@match{array} \nextok@match{tabular}}
```

2.10 Processing Options and Leaving the Package

```
\ProcessOptions
\endinput
```

2.11 VERSION HISTORY

```
v0.1   2011/05/23   very first
```