The mkswitch Package
Version 1.0

Alceu Frigeri[*]

May 2025

Abstract

This package offers two commands aimed at implementing a switch/case alike command.

Contents

1 Introduction

There are many ways of implementing a switch case programming structure. Notably, one can use `\str_case:nn` from *expl3*, or go over a loop using `\pdfstrcmp`, or construct an if-then-else tower, etc.

This implements a solution, somewhat based on [1], which (besides being simple) has the advantage of being constant time: once the cases are set up, suffice a single (internal) if (`\ifcsname`) to select the correct code to be executed.

Note: The implementation creates a `\csname` for each case, and it uses (at the end) the primitive `\ifcsname` to select the correct case.
Note: The coding is done using *expl3*, just for the sake of making it more "readable", in the package comments one can find an implementation using just TEX primitives.

2 Commands

Two set of commands are created, one to be used in a *expl3* code régime, and another set to be used in a user document.

2.1 User Document ones

`\mkswitch`

`\mkswitch ⟨switch⟩ {⟨default-code⟩}`

It will create a new switch ⟨switch⟩, which will expects a single argument. In case the argument doesn't corresponds to any defined case, ⟨default-code⟩ will be used. The resulting ⟨switch⟩ command is expandable, if ⟨default-code⟩ and ⟨case-code⟩ (added by `\addcase`) also are. This is just an alias to `\switch_new:Nn`

Note: `#1` can be used in ⟨default-code⟩.

`\addcase`

`\addcase ⟨switch⟩ {⟨case⟩}{⟨case-code⟩}`

It will add a ⟨case⟩ to a previously defined ⟨switch⟩ and associates ⟨case-code⟩ with it. ⟨case⟩ will be fully expanded at definition time. Once defined one can call `\switch {case}`, which will put said ⟨case-code⟩ in the input stream. This is just an alias to `\switch_addcase:Nnn`.

[*] https://github.com/alceu-frigeri/mkswitch

2.1.1 Example

First we create a switch, and associate a few (or more) cases. Note the possibility of using an auxiliary (fully expandable) macro/command when defining the cases.

```
\def\CaseAstring{case-A}
\mkswitch \myCase  {I~ don't~ know:~ #1\par}
\addcase  \myCase  {\CaseAstring} {A~ was~ used\par}
\addcase  \myCase  {case-B} {B~ was~ used\par}
```

To use the ⟨switch⟩, one just has to call it with ⟨case⟩ as an argument. Note the possibility of using an auxiliary macro/command (which has to be fully expandable) as a ⟨case⟩.

```
\def\somemacro{case-A}
\def\someothermacro{case-X}

If B, then \myCase{case-B}
If A, then \myCase{case-A}
If X, then \myCase{case-X}

if somemacro: \myCase{\somemacro}
if someothermacro: \myCase{\someothermacro}
```

If B, then B was used
If A, then A was used
If X, then I don't know: case-X
if somemacro: A was used
if someothermacro: I don't know: case-X

2.2 Expl3 ones

\switch_new:Nn

`\switch_new:Nn` ⟨switch⟩ {⟨default-code⟩}

It will create a new switch ⟨switch⟩, which will expects a single, type n, argument. In case the argument doesn't corresponds to any defined case, ⟨default-code⟩ will be used. The resulting ⟨switch⟩ command is expandable, if ⟨default-code⟩ and ⟨case-code⟩ (added by `\switch_addcase:Nnn`) also are.

Note: `#1` can be used in ⟨default-code⟩.

\switch_addcase:Nnn

`\switch_addcase:Nnn` ⟨switch⟩ {⟨case⟩} {⟨case-code⟩}

It will add a ⟨case⟩ to a previously defined ⟨switch⟩ and associates ⟨case-code⟩ with it. ⟨case⟩ will be fully expanded at definition time. Once defined one can call `\switch {case}`, which will put said ⟨case-code⟩ in the input stream.

2.2.1 Example

First we create a switch, and associate a few (or more) cases. Note the possibility of using an auxiliary (fully expandable) macro/command when defining the cases.

```
\ExplSyntaxOn
\def\CaseAstring{case-A}
\switch_new:Nn \TextCase       {I~ don't~ know:~ #1\par}
\switch_addcase:Nnn \TextCase  {\CaseAstring} {A~ was~ used\par}
\switch_addcase:Nnn \TextCase  {case-B} {B~ was~ used\par}
\ExplSyntaxOff
```

To use the ⟨switch⟩, one just has to call it with ⟨case⟩ as an argument. Note the possibility of using an auxiliary macro/command (which has to be fully expandable) as a ⟨case⟩.

```
\def\somemacro{case-A}
\def\someothermacro{case-X}

If B, then \TextCase{case-B}
If A, then \TextCase{case-A}
If X, then \TextCase{case-X}

if somemacro: \TextCase{\somemacro}
if someothermacro: \TextCase{\someothermacro}
```

If B, then B was used
If A, then A was used
If X, then I don't know: case-X
if somemacro: A was used
if someothermacro: I don't know: case-X

References

[1] Paul Gaborit. *Stack Exchange answer about Implementing Switch Cases*. 2012. URL: `https://tex.stackexchange.com/questions/64131/implementing-switch-cases/343306#343306` (visited on 12/10/2016).