La exp-testopt-Pakaĵo – ekspandebla varianto de `\@testopt`. *

Paul Ebermann†

6–a de marto, 2009

Ekspandebla varianto de `\@testopt` el la LaTeX-kerno, kaj iom pli ĝenerala (ankaŭ ekspandebla) `\@expandable@ifopt`, por rekoni ekziston de opciaj argumentoj (sen tuj doni defaŭlton). Ankaŭ `\newcommand`-varianto, kiu uzas tion.

Enhavo

1 Uzanta dokumentaĵo

Tiu ĉi pakaĵo enhavas unu makroon rekte uzebla de uzantoj, kvankam ankaŭ ĝin verŝajne pli ofte uzos pakaĵo-kreantoj. Krome ĝi enhavas du utilajn makroojn por pakaĵ-kreantoj (ne por uzantoj).

*Tiu dokumento rilatas al exp-testopt v0.3, de 2009/03/06.

†Paul-Ebermann@gmx.de

1.1 Makrooj

\expnewcommand `\expnewcommand*{`⟨*nomo*⟩`}[`⟨*paramnum*⟩`] [`⟨*defaŭlto*⟩`]{`⟨*kodo*⟩`}` – funkcias simile kiel `\newcommand`: Kreas komandon kun ⟨*num*⟩ parametroj (ĝis 9), kaj se ⟨*defaŭlto*⟩ estas donita, la unua parametro estos opcia kun ⟨*defaŭlto*⟩ kiel defaŭlta valoro. Diferenco al `\newcommand` nur ekzistas, se ⟨*defaŭlto*⟩ estas donita – tiam je voko de la makroo ni uzas ekspandeblan komparon por ekscii, ĉu unua parametro estas donita. (Tio havas la avantaĝon esti ekspandebla, sed la limigojn menciitajn sube.)

ndable@testopt `\@expandable@testopt{`⟨*ordono*⟩`}{`⟨*defaŭlto*⟩`}` – vokas ⟨*ordono*⟩n kun `[]`-parametro – se neniu plia estas donata de la uzanto, ni uzas ⟨*defaŭlto*⟩ anstataŭe. Pli detale: kontrolas, ĉu la sekva signo estas `[`. Se jes, iĝas ⟨*ordono*⟩, alikaze iĝas ⟨*ordono*⟩`[`⟨*defaŭlto*⟩`]`. (Tio funkcias simile kiel `\@testopt` el la LaTeX-kerno.)

pandable@ifopt `\@expandable@ifopt{`⟨*jes*⟩`}{`⟨*ne*⟩`}` – kontrolas, ĉu la sekva signo estas `[`. Se jes, vokas ⟨*jes*⟩, alikaze ⟨*ne*⟩.

1.2 Limigoj

Fakte tiuj makrooj ne kontrolas la sekvan simbolon (kio igus ĝin ne-ekspandebla pro uzo de `\futurelet`, kiel `\@testopt`), sed la sekvan makroo-argumenton (`#3`, fakte). Krome estas malfacile kompari ion, se oni ne povas difini/ŝanĝi ion.

Tio havas la sekvajn konsekvencojn:

- Ne eblas distingi `[` de `{[}` per nia algoritmo – ambaŭ estas traktataj kiel ⟨*Jes*⟩.
- La sekva argumento estu de la amika speco: Ĝi estu (se ĝi ne estas simple `[`) ĉeno el maksimume unu *danĝera* simbolo (kontrolsekvenco aŭ el danĝera kategorio) kaj poste simpla teksto (el kategorioj 10, 11, 12).

  Se ĝi konsistas el nur unu simbolo, tio ne estas problemo. Se ĝi konsistas el simpla teksto, tio ankaŭ ne estas problemo.

  En aliaj kazoj povas aperi strangaj erarmesaĝoj.
- Apero de pluraj spaco-simboloj antaŭ la `[` povas konfuzigi la programon, tiam ankaŭ aperos strangaj erarmesaĝoj.

Do, antaŭ la uzo de tiuj makrooj bone pripensu, ĉu la limigoj vin ĝenas.

(Krome eblas trompi nian algoritmon per kreado de pliaj makrooj en la nomspaco de `\exp-testopt@@`⟨*io*⟩`@`. Ne faru tion, krom se vi certas pri la konsekvencoj.)

2 Ekzemplo

Jen ekzemplo de la uzo por krei komandon `\beispiel` kun opcia argumento.

```
212 ⟨*test⟩
213 \makeatletter
214 \errorcontextlines␣=␣20
```

```
\beispiel
\def\beispiel{%
  \@expandable@testopt\beispielImpl{Default}%
}
\beispielImpl
\def\beispielImpl[#1]{%
  \fbox{#1}%
}
\makeatother

\expnewcommand*{\ekzemplo}[1][Defa\u␣ulto]{%
  \textbf{(#1)}%
}

\expnewcommand*{\ekzemploDu}[2][defa\u␣ulto]{%
  \ensuremath{\frac{\mbox{#1}}{\mbox{#2}}}%
}

  \beispiel{egal}␣\beispiel[bla]
  \beispiel$␣e=m␣c^2$␣\beispiel[$␣e=m␣c^2$]

  \ekzemplo\ekzemplo[Bla]
  \expandafter\ekzemplo\space[bla]
  \expandafter\expandafter\expandafter\ekzemplo\expandafter\space%
       \space[dua␣bla]
  \ekzemplo
  [nova␣linio]
  \ekzemplo

  [du␣novaj␣linioj]
  \ekzemplo[blub]

  \ekzemploDu{testo}
  \ekzemploDu[bla]{testo}

⟨/test⟩
```

Kontraŭe al kreado de tia komando per `\newcommand`, la komando estas tute ekspandebla (nu, krom la enhavo de `\beispielImpl`), anstataŭ robustigita per malfruigita ekspandado. Tiel ekzemple eblas uzi valorojn de variabloj kun ilia aktuala valoro, ankaŭ se la komando mem estos poste skribota en dosieron. (Hmm, iom komplika, ĉu?)

La supra kodo donas jenan rezulton:

Default egal bla Default $e = mc^2$ $e = mc^2$
(Defaŭlto)(Bla) (bla) (dua bla) (nova linio) (Defaŭlto)
[du novaj linioj] **(blub)**
$\frac{\text{defaŭlto}}{\text{testo}}$ $\frac{\text{bla}}{\text{testo}}$

3 Implementado

271 ⟨∗package⟩

3.1 Ĉefaj makrooj

Niaj du ĉefaj makrooj nun aspektas tiom similaj, ke ni povas implementi `testopt` per `ifopt`.

ndable@testopt
```
280 \newcommand*{\@expandable@testopt}[2]{%
281   \@expandable@ifopt{#1}{#1[#2]}%
282 }
```

`\@expandable@ifopt` nun faras la laboron.

pandable@ifopt
```
286 \newcommand{\@expandable@ifopt}[3]{%
287   \expandafter\ifx%   \ifx komparas la difinojn de du makrooj, nome ...
288     \csname␣exp-testopt@@\string#3@\endcsname%   ... makronomo kreita
          %   el la tria argumento (= la signo poste).
290     \exp@testopt@opt@%   ... kaj tiu antaŭdifinita makroo.
291     \afterfi{#1#3}%   En la jes-kazo, vokas la unuan argumenton (kun la tria),
          %   ...
293   \else%   ... en la ne-kazo la duan.
```

Sed tiam ni ankoraŭ devos eltrovi, ĉu la tria konsistis el unu aŭ pluraj tokenoj, por meti `{...}` ĉirkaŭ ĝin en la dua kazo. (Se ĝi nur estas unu tokeno, meti `{...}` povas esti malhelpa, se ĝi fakte ne estas argumento al `#2`, sed ekzemple io kiel `$`, aŭ sekva makroo.)

```
299     \@expandable@ifOneToken#3\@expandable@ifOneToken%   Tiu ĉi makroo
          %   provas eltrovi tion.
301     {\afterfi{#2#3}}%
302     {\afterfi{#2{#3}}}%
303   \fi%
304 }%
```

La ideo por tia ekspandebla kontrolo de la sekva ≫signo≪ estas, ke ni kreas el la ≫signo≪ makroonomon, kaj komparas la signifon de ĝi kun signifo de konata makroo.

3.2 Komparendaj makrooj

Por tio ni kreas iujn nomojn, kiuj estas supozeble ne uzata de iu alia, kun signifoj, kiuj ankaŭ supozeble ne erare aperas.

p@testopt@opt@
```
316 \def\exp@testopt@opt@{<<<[>>>]}%   tio estas la nomo, kies valoron ni komparas
      %   kun aliaj kandidatoj.
318 \@namedef{exp-testopt@@\string[@]{<<<[>>>]}%   Tio – \exp-testopt@@[@ –
      %   estas la makroo, kiu estos trovota en la jes-kazo.
```

Ni ankoraŭ devas eltrovi, ĉu la argumento estas nur unu tokeno aŭ pluraj, por povi meti `{...}` en la dua kazo. Ni uzas similan trukon por tio.

ble@ifOneToken
```
325 \long\def\@expandable@ifOneToken#1#2\@expandable@ifOneToken#3#4{%
```

```
  \expandafter\ifx%
     \csname␣exp-testopt@@\string#2@\endcsname%
     \exp@testopt@empty@%
     \afterfi{#3}%
  \else
     \afterfi{#4}%
  \fi
}
```

testopt@empty@

```
\def\exp@testopt@empty@{<<<>>>}%    tio estas la nomo, kies valoron ni
        %    komparas kun aliaj kandidatoj.
\@namedef{exp-testopt@@@}{<<<>>>}%    Tio – \exp-testopt@@@ – estas la
        %    makroo, kiu estos trovota en la kazo, ke #2 estas malplena.
```

3.3 Komando-difinoj

Ni volas krei varianton de `\newcommand`, kiu uzas nian ekspandeblan metodon por opciaj argumentoj, anstataŭ la robustan.

Jen la originalo el latex.ltx (ltdefns.dtx, por esti preciza, CTAN-versio de 2006-05-21).

(354) ⟨∗latex2e⟩

Handle the second optional argument.

\@xargdef

(356) `\long\def\@xargdef#1[#2][#3]#4{%`
(357) `  \@ifdefinable#1{%`

Define the actual command to be:
`\def\foo{\@protected@testopt\foo\\foo{default}}`
where `\\foo` *is a csname generated from applying* `\csname` *and* `\string` *to* `\foo`*, ie the actual name contains a backslash and therefore can't clash easily with exisiting command names. "Default" is the contents of the second optional argument of* `(re)newcommand`*.*

```
(365)       \expandafter\def\expandafter#1\expandafter{%
(366)           \expandafter
(367)           \@protected@testopt
(368)           \expandafter
(369)           #1%
(370)           \csname\string#1\endcsname
(371)           {#3}}%
```

Now we define the internal macro ie `\\foo` *which is supposed to pick up all arguments (optional and mandatory).*

```
(374)         \expandafter\@yargdef
(375)           \csname\string#1\endcsname
(376)            \tw@
(377)            {#2}%
```

```
(378)                 {#4}}}
(379) ⟨/latex2e⟩
```

Kaj ni nun devas ŝanĝi la markitan parton por anstataŭe uzi nian metodon. Tio signifas, ke ⟨*nomo*⟩ nun estas difinita kiel `\@expandable@testopt\`⟨*nomo*⟩`{`⟨*defaŭlto*⟩`}`, anstataŭ `\@protected@testopt`⟨*nomo*⟩`\`⟨*nomo*⟩`{`⟨*defaŭlto*⟩`}`.

(Kompreneble ni ne redifinas `\@xargdef`, sed difinas nian propran varianton, por uzi anstataŭe – ni ja ne volas ŝanĝi la bazan funkcion de `\newcommand`.)

`\@exp@xargdef`

```
393 \long\def\@exp@xargdef#1[#2][#3]#4{%
394   \@ifdefinable#1{%
395     \expandafter\def\expandafter#1\expandafter{%
396       \expandafter%
397       \@expandable@testopt%
398       \csname\string#1\endcsname%
399       {#3}}%
400     \expandafter\@yargdef%
401       \csname\string#1\endcsname%
402       \tw@%
403       {#2}%
404       {#4}%
405   }%
406 }
```

Kaj nun kelkaj paralelaj difinoj al `\newcommand`, `\new@command` kaj `\@newcommand`. `\expnewcommand` estas la ĉefa komando.

`\expnewcommand`

```
415 \def\expnewcommand{%
416   \@star@or@long\expnew@command%
417 }
```

`\expnew@command`

```
418 \def\expnew@command#1{%
419   \@testopt{\exp@newcommand#1}0%
420 }
```

`\exp@newcommand`

```
421 \def\exp@newcommand#1[#2]{%
422   \@ifnextchar␣[{\@exp@xargdef#1[#2]}%    Ĉi tie ni vokas nian varianton
          %    anstataŭ \@xargdef.
425                 {\@argdef#1[#2]}%    Tiu estas la originalo.
426 }
```

3.4 Helpaj makrooj

`\afterfi{`⟨*kodo*⟩`}` – saltas ĝis post la sekva `\fi`, kaj ekzekutas ⟨*kodo*⟩n tie.

Kopiita el gmutils.

`\afterfi`

```
436 \@ifundefined{afterfi}{%
437   \def\afterfi#1#2\fi{%
438     \fi#1%
```

```
439       }%
440 }{}%
```

Jam fino :-)

```
446 \endinput
447 ⟨/package⟩
```

(Jes, vere fino.)

4 Ŝanĝoj

v0.1
- Ĝenerale:
  - CheckSum 37, 0
  - Unua versio, 0

v0.2
- Ĝenerale:
  - CheckSum 90, 0
  - Preta por publikigo, mi kredas., 0
- `\exp@testopt@empty@`:
  - Aldono de `\expnewcommand`., 337

v0.3
- `\@exp@xargdef`:
  - Cim-korekto: `\exp@new@command` renomita al `\expnew@command`, 406
- Ĝenerale:
  - CheckSum 139, 0
  - Post pluraj cimkorektoj nun vere preta ..., 0
  - Testo ankaŭ por `\expnewcommand`, 0

5 Indekso

Kurzivaj nombroj indikas la lokojn, kie la indeks-ero estas priskribita, substrekitaj nombroj indikas la lokon de la difino, la aliaj estas uzoj.