plainpkg

—

a "Minimal" Method for Making "Generic" Packages*

Uwe Lück†

September 19, 2012

Abstract

plainpkg.tex provides some rudimentary LaTeX-like package management for "generic" packages: (i) a (rather arbitrary) implementation of LaTeX's `\ProvidesFile` (support for readprov), (ii) an implementation of LaTeX's `\ProvidesPackage` that, in addition to (i), avoids loading twice, (iii) a simple implementation of LaTeX's `\RequirePackage` to allow nesting of package files with and without LaTeX and (iv) loading stacklet.sty for managing private letters with nested package files. Also, (v) a rather experimental `\ifltx` is provided indicating whether the format is LaTeX— or miniltx.tex has been loaded earlier ... A by-product is (vi) the helper `\withcsname` for `csname` constructs. The documentation also introduces a notion of "plainpkg packages" for a central explanation of how to make and work with "generic" packages based on plainpkg.

Related Packages: miniltx, maybeload; catoptions, pcatcode from amsrefs, texapi

Required Packages: stacklet.sty from catcodes bundle

Keywords: Macro programming, package management

Contents

*This document describes `plainpkg.tex`'s version v0.4a as of 2012/09/19.

†`http://contact-ednotes.sty.de.vu`

1 Purpose and Usage

1.1 Purpose

plainpkg.tex in the first instance is a tool for TEX macro packages to work with LATEX as well as with Plain TEX, perhaps even with other TEX formats. When LATEX seems to be missing, a definition for `\ProvidesPackage` is provided that avoids loading such a package a second time. Earlier (in the dowith package), I tried to “hide” the `\ProvidesPackage` command when it was not defined, the original motive was to have that command somewhere so its version information can be accessed by the readprov package. As such “generic” packages often use private LATEX internals, I thought that plainpkg also should offer a stack for handling category codes of `@` in nested package files. Rather than providing such a stack in `plainpkg.tex`, I use the more general stacklet.sty, because I have used different “private letters” in other nested package files that *require LATEX*, so such stacks should be accessible *without* plainpkg.

1.2 Installing: How and Why

The file `plainpkg.tex` is provided ready, like `stacklet.sty` (catcodes bundle), installation only requires putting both somewhere where TEX finds them (which may need updating the filename data base).[1]

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

1.3 Features

Besides providing stacklet's features—see the catcodes bundle documentation in `catcodes.pdf`—and a fallback definition `\ProvidesPackage` for running without LaTeX, some `\withcsname`, a conditional `\ifltx` as well as fallback definitions for `\RequirePackage` and `\ProvidesFile` are provided—see implementation sections for details.

1.4 Loading `plainpkg.tex`

`plainpkg.tex` may be loaded by `\input␣plainpkg` or—with LaTeX—by `\input{plainpkg}`. However, in a document source file this is useful only when so-called "weak plainpkg packages" according to Section 1.5 are loaded additionally. With LaTeX, the only effect will be that `\withcsname` works and that the stacklet package is loaded. So if you just want to have the stacklet functionality of support for private letters in nested package files, you better use `\RequirePackage{stacklet}` or `\usepackage{stacklet}` directly. The latter still is a little strange—it may be helpful for private letters other than `@` in a LaTeX document, or with "weak stacklet packages," a notion that I have not introduced yet.

1.5 Notion and Usage of "plainpkg Packages"

The main purpose of the present section is to have a central reference for all "generic" packages based on plainpkg, to avoid repeating details in the documentation of each single package of that kind.

1.5.1 The Notion: "Strong" and "Weak plainpkg Packages"

I introduce the notion of "**plainpkg packages**" for "generic" packages based on plainpkg.tex and requiring it.

Strong plainpkg packages (i) have filename extension `.sty` and (ii) contain `\input␣plainpkg`.

Weak plainpkg packages do not load `plainpkg.tex`, but their *documentation* says that they must be loaded *after* `plainpkg.tex` has been loaded. They have filename extension `.sty` as well.

My plainpkg packages will also contain `\ProvidesPackage{⟨pkg⟩}[⟨ver⟩]` (after `\input␣plpkgpkg`). A package loading `plainpkg.tex` and *not* containing `\ProvidesPackage` may work and be called a "plainpkg package", but the usefulness of such a practice, hmm, is in some sense discussed in Section 1.4.

"Weak" plainpkg packages are just an idea that came to my mind when I thought about the present documentation, at present I prefer *strong* plainpkg packages, I do not want to explain usage of weak plainpkg packages.

I like to place the `\input␣plainpkg` "right-adjusted" in the plain text file hoping this way the file information of the next `\Provides...` line is not overlooked.

1.5.2 How to Load a plainpkg Package

For loading a plainpkg package ⟨*generic*⟩`.sty` from within some file ⟨*loading*⟩, we have the following cases:

from within the LaTeX document preamble of ⟨*loading*⟩:
`\usepackage{`⟨*generic*⟩`}`[2]

not intended for LaTeX: `\input␣`⟨*generic*⟩`.sty`

possibly with LaTeX("generic"): `\Require{`⟨*generic*⟩`}`
—and `plainpkg.tex` should have been loaded before,
recommendation: ⟨*loading*⟩ a plainpkg package ⟨*lodyng*⟩`.sty` itself.

Note: The optional argument as in `\RequirePackage{`⟨*generic*⟩`}[`⟨*date*⟩`]` is **not supported** (at present)!

1.5.3 How to Make a plainpkg Package

Section 1.5.1 tells what rather is *required* for a plainpkg package, and Section 1.5.4 summarizes what additionally *works* in a plainpkg package, due to plainpkg's *features.*

1.5.4 What a plainpkg Package May Contain

A plainpkg package may contain

- `\ProvidesPackage`, `\RequirePackage` (without optional argument),
- `\ifltx`, and `\withcsname`;
- stacklet commands properly paired: for each "private letter" ⟨*char*⟩, place
  - `\PushCatMakeLetter\`⟨*char*⟩ above its first use and place
  - `\PopLetterCat\`⟨*char*⟩ after the last use, above `\endinput`.
  - If ⟨*char*⟩ is `@`, `\PushCatMakeLetterAt` and `\PopLetterCatAt` are recommended instead.

1.5.5 Other "plainpkg Files"

As plainpkg also provides a fallback definition for `\ProvidesFile`, another notion could be that of a "plainpkg file" ⟨*file*⟩ that (i) has an arbitrary filename extension, (ii) is loaded by `\input␣`⟨*file*⟩ or, with LaTeX, `\input{`⟨*file*⟩`}` and (iii) may contain what is allowed according to Section 1.5.4, apart from `\ProvidesPackage`. As an obvious example, all the document source files such as ⟨*part*⟩`.tex` may start with `\ProvidesFile`, or certain `.def` files could be considered "plainpkg files."

[2] . . . or even `\RequirePackage{`⟨*generic*⟩`}` . . .

2 Comparison with miniltx and maybeload

Without LaTeX, the definitions of `\ProvidesPackage` and `\RequirePackage` are by no means copies from LaTeX, as they are in miniltx. Rather, `\ProvidesPackage` will work like maybeload's `\thisfileis`.—maybeload was made for "LaTeX," too, according to its comment. But that rather was pre-2_ε LaTeX. plainpkg might also have been called "maybeload2e", as we are essentially combining maybeload's functionality with fall-back support for LaTeX 2_ε's basic package commands. But of course, that name would not reflect loading stacklet, whose purpose also has been to have as little as possible above `\ProvidesPackage`.

3 The Package File

3.1 Header—Bootstrapping and Legalese

The first line is for Section 3.3. Next I want to have `\Provides...` info at the top of the file, but such a command hasn't been defined yet. `\def\filename` etc. could be bad as well, overriding `\filedate` of a package that loads plainpkg.tex.

```
\ifx\plainpkginfo\undefined
  \gdef\plainpkginfo{\ProvidesFile{%
                                                    %
                       plainpkg.tex}                [%
                       2012/09/19 v0.4a plain package management]}
%%
%% Copyright (C) 2012 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below --
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3c of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% We did our best to help you, but there is NO WARRANTY.
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

3.2 Purpose and Usage

... of the file `plainpkg.tex` is described in Section 1 of the documentation file `plainpkg-doc.pdf` generated from `plainpkg-doc.tex` (... in case somebody is reading the plain text of `plainpkg.tex`).

3.3 Avoiding Being Loaded Twice

Continuing the first conditional:

```
\else
```

Keeping the `\endinput` *outside* the conditional:

```
    \expandafter \endinput
\fi
```

The earlier idea to close the conditional more below conflicted with avoiding `\input␣stacklet` under readprov.

3.4 \global

A funny idea of my earlier makedoc in the nicetext bundle was that its macro definitions should be *local*, for *preprocessing* documentation files—results being written to files. Well, but when `makedoc.sty` loads the present `plainpkg.tex`, the latter's definitions should *last*. Therefore, v0.4 renders all definitions *global*. I.e., `\def` is replaced by `\gdef`, `\edef` by `\xdef`, and `\let` assignments are prefixed with `\global`.

3.5 A Tool for \csname

`\withcsname⟨cmd⟩` is a little helper with `\csname` that will be used by stacklet as well. Usage actually should be about as

`\withcsname\`⟨*letters*⟩␣⟨*chars*⟩`\endcsname`

or

`\withcsname\`⟨*letters*⟩⟨*non-letter-char*⟩⟨*chars*⟩`\endcsname`

(but better don't expect that `@` were a non-letter-char!) or

`\withcsname\`⟨*non-letter*⟩⟨*chars*⟩`\endcsname`

(tokenization ...) and should result in a sequence of two tokens—in the dowith notation—

?⟨*letters*⟩ ?⟨*chars*⟩ or ?⟨*letters*⟩ ?⟨*non-letter-char*⟩⟨*chars*⟩

or ?⟨*non-letter*⟩ ?⟨*chars*⟩ ...

```
\gdef\withcsname#1{\expandafter#1\csname}
```

2012/08/27 I realize that from the three files I made this weekend (plainpkg.tex, stacklet.sty, actcodes.sty), a single ?`withcsname` token appears in macro replacement "texts" (actcodes doesn't use it at all)—wondering whether I should remove it TODO—however, it improves readability of the files.

3.6 \Provides..., \ifltx

```
\ifx\ProvidesPackage\undefined
```

... or can it be \relax, cf. concern in german.sty?

We expect that \ProvidesFile and \ProvidesPackage are used with the trailing "optional" argument:

```
  \gdef\ProvidesFile#1[#2]{\wlog{#1 #2}}
```

\ProvidesPackage gets maybeload functionality. v0.2 aims at saving a few tokens. And we form a token name containing an @ without changing its \catcode.

```
  \xdef\ProvidesPackage#1{%
    \noexpand\withcsname
            \withcsname\noexpand @providespkg\endcsname %%  ‘ ’ v0.3
                       ver@#1.sty\endcsname{#1}}        %% .sty v0.3
```

... so \ProvidesPackage{⟨*chars*⟩} should result in

$$?\texttt{withcsname}\,?\texttt{@providespkg}\,\mathsf{in}_k^*(\texttt{ver@}\langle chars\rangle)\,?\texttt{endcsname}\,.\{\,\mathsf{in}_k^*(\langle chars\rangle)\,.\}$$

where $\mathsf{in}_k^*(\chi)$ is the tokenization of χ with current \catcode function k; and this should further result in

$$?\texttt{@providespkg}\,?\texttt{ver@}\langle chars\rangle\,.\{\,\mathsf{in}_k^*(\langle chars\rangle)\,.\}$$

```
%   \show\ProvidesPackage
```

The first tokens of the next code line result in ?gdef ?@providespkg:

```
  \withcsname\gdef @providespkg\endcsname#1#2[#3]{%      %%  ‘ ’ v0.3
    \ifx#1\relax \ProvidesFile{#2.sty}[#3]%
                 \xdef#1{#3}%
```

... like LaTeX, while maybeload consumes less memory.

```
    \else        \expandafter \endinput
    \fi }
```

Moreover, if \ProvidesPackage has not been defined before, neither LaTeX is present nor miniltx has been loaded, so \ifltx is rendered \iffalse (the construction ensures proper skipping by the outer conditional, and \ifltx rather than \iflatex alludes to considering miniltx and to our worries):

```
  \global \expandafter \let \expandafter
    \ifltx \csname iffalse\endcsname
```

If \ProvidesPackage has been defined, how come? Not from plainpkg.tex which is not loaded twice. Rather from LaTeX or miniltx:

```
\else
  \global \withcsname \let ifltx\expandafter\endcsname  %% 2012/08/26
                      \csname iftrue\endcsname
\fi
```

Now \ProvidesFile for plainpkg.tex can be executed:

```
\plainpkginfo
```

3.7 \RequirePackage

Without LaTeX ...

```
\ifltx \else                                          %% 2012/08/25
```

`\RequirePackage` simply is ...

```
  \gdef\RequirePackage#1{\input #1.sty}
\fi
```

3.8 \catcode Stacks

... (for private letters) are provided by stacklet.sty ...

```
\RequirePackage{stacklet}
```

3.9 Leaving and HISTORY

```
\endinput
```

VERSION HISTORY

```
v0.1   2012/08/22   very first
v0.2   2012/08/23   refinements with \csname
v0.3   2012/08/24   \oncsname -> \withcsname;
                    loading ‘stacklet.sty’ with LaTeX;
                    doc. extended
       2012/08/25   doc. "installing" mv. here;
                    aligning first conditional + label, \ifltx
       2012/08/26   explaining \withcsname;
                    corrected second \ifltx;
                    bug fix "ver@#1.sty"!!!
       2012/08/27   account for ‘@’ being letter; doc. fix
                    remark on keeping \withcsname
v0.4   2012/08/27   \global
       2012/09/15   doc. was \Providespackage
       2012/09/16   doc: endorsing \RequirePackage more clearly
       2012/09/17   bug fix: \def\plain... -> \gdef\plain...
                    (for local makedoc processing)
       STORED SEPARATELY
v0.4a  2012/09/19   moving documentation outside, \label..., ?gdef
```