A Few Packages by Donald Arseneau

Uwe Lück[*]

April 5, 2010

Abstract

This document demonstrates the present (i.e., nicetext bundle release v0.4) capabilities of makedoc and niceverb to typeset LaTeX quality documentation from third-party package files having ASCII comments only, without modifying such package files manually.[1] Such packages usually mark comments by lines starting with '%␣'. This is somewhat difficult when the "comment mark" '%' may at the same time be used for "commenting out" in place of "true commenting." As opposed to substr.sty that conforms to the easier '%%␣' style, the packages nolbreaks and notoccite by Donald Arseneau exemplify the '%␣' commenting style.

The ASCII documentation of packages of this kind does not always clearly indicate when switching to `\tt` is appropriate. This problem is solved here by string replacements very specific to the package files. The setup for these replacements (in the source file arseneau.tex) still may be simplified.

Moreover, some similar packages cannot be properly typeset with nicetext at present because makedoc's loop is too rigid to deal with (i) ASCII indents, displays, lists and with (ii) instructions after `\endinput`. (On 2010/04/05, we try addressing (i) with wiki.sty, applied to Donald Arseneau's optional. A nicetext bug with replacing tildes shows up.)

It should be clear that all the package desriptions are Donald Arseneau's, nicetext just formats them.

Contents

[*]applying `http://ctan.org/pkg/nicetext`

[1]See `http://ctan.org/pkg/nicetext` for more on these packages.

1 nolbreaks

```
%  nolbreaks.sty  by Donald Arseneau
%  Public domain software -- please improve and submit to CTAN

\ProvidesPackage{nolbreaks}[2002/09/19 \space v 1.0 \space
   - no linebreaks in text]

```

Use `\nolbreaks{some text}` to prevent linebreaks in `some␣text`. This has the advantage over `\mbox{}` that glue (rubber space) remains flexible. It has the disadvantage of not working in all cases! Most common cases are handled here (`\linebreak` is disabled, for example) but spaces hidden in macros or `{␣}` can still create break-points.

Large pieces of text with no breaks can cause problems with paragraph justification. Giving the package option `[ragged]` allows a line before the unbreakable text to be cut short.

You should declare `\sloppy` in your document.

```
\let\nb@ragged\relax

\DeclareOption{ragged}{
  \def\nb@ragged{%
    \skip@\lastskip \unskip
    \nb@counter \lastpenalty
    \hskip \z@ plus 2cm\relax
    \penalty\nb@counter
    \advance\skip@ \z@ plus -2cm\relax
    \hskip\skip@
  }}
\ProcessOptions

\let\nb@@iwspace=\ %
\let\nb@@hskip=\hskip
\let\nb@@penalty=\penalty
\newcount\nb@counter
\ifx\langwohyphens\undefined
 \newlanguage\nb@lang
\else
 \let\nb@lang\langwohyphens
\fi

\DeclareRobustCommand{\nolbreaks}[1]{%
  \leavevmode
  \begingroup
    %  Apply flexible lead-in
    \nb@ragged
    %  Prevent hyphenation
    \language\nb@lang
    %  Disable commands that give breakpoints
    \let\ \nb@iwspace
    \let\hskip\nb@hskip
    \let\penalty\nb@penalty
    \let\language\nb@counter
    %  Prevent breaks in math
    \relpenalty\@M
    \binoppenalty\@M
    %  Prevent breaks at spaces (only outermost visible spaces)
    \@PreserveSpaces \@empty #1 \@PreserveSpaces
  \endgroup
}

\def\@PreserveSpaces#1 {#1\@ifnextchar\@PreserveSpaces{\@gobble}%
 {\@firstofone{\nb@@penalty\@M} \@PreserveSpaces\@empty}}

\def\nb@iwspace{\nb@@penalty\@M \nb@@iwspace}
\def\nb@hskip  {\nb@@penalty\@M \nb@@hskip}
\def\nb@penalty{\nb@@penalty\@M \nb@counter}

```

2 notoccite

```
% notoccite.sty        no t.o.c. cite     Jul 20, 2000
% Donald Arseneau      asnd@triumf.ca     TRIUMF, Vancouver, Canada,
% This is unrestricted software contributed to the public domain.
```

Ordinarily, cites used in titles or figure captions also appear in the table of contents and list of figures. If you then run `bibtex` using the `unsrt` (unsorted) style, they get numbered starting from 1, not the number they should have in the main text.

A good option is to avoid cites in titles, and to specify optional caption text without cites:

```
  \caption[Picture of a bird.]{Picture of a bird \cite{audobon}.}
```

If you must use moving cites, you could manage them by deleting `.toc` and `.lof` files, then running `latex` once, then `bibtex`. However, the following definition fixes the problem so you don't need to worry about that.

NOTE: This definition works for the ordinary LaTeX definitions for `\cite` and others (`\addtocontents`, `\label`) but it may well fail when used with various packages for citations or cross references.

It works by locally setting `\@fileswfalse`, which is something like `\nofiles`, but `\@fileswfalse` does not affect `\label` or `\addtocontents`. `\nofiles` does most of its work by redefining `\protected@write`, and neither `\addtocontents` nor `\label` check for `\if@filesw`. `\cite` *does* check `\if@filesw`.

```
\ProvidesPackage{notoccite}[2000/07/20]
\def\@starttoc#1{%
  \begingroup
    \@fileswfalse
    \makeatletter
    \@input{\jobname.#1}%
  \endgroup
  \if@filesw
    \expandafter\newwrite\csname tf@#1\endcsname
    \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
  \fi
  \@nobreakfalse
}


```

3 optional

```
%
% O P T I O N A L . S T Y
% ~~~~~~~~~~~~~~~~~~~~~~~
% ver 2.2b  Jan 2005
%
% Enable multiple versions of a document to be printed from one source file,
% especially if most of the text is shared between versions.
%
% Copyright 1993,1999,2001,2005 by Donald Arseneau (asnd@triumf.ca).
% This software is released under the terms of the LaTeX Project Public
% License  (ftp://ctan.tug.org/tex-archive/macros/latex/base/lppl.txt).
% (Essentially: Free to use, copy, distribute (sell) and change, but, if
% changed, that fact must be made apparent to the user.)  It has a
% status of maintained.
```

How to Use

One way to use this package is to declare (for example)

```
\usepackage[opta]{optional}
```

at the beginning of your document, and flag optional text throughout your document like:

```
\opt{opta}{Do this if option opta was declared}
\opt{optb}{Do this if option optb was declared}
\opt{optx,opty}{Do this if either option optx or opty}
\opt{}{Never print this text!}
```

```
\opt{opta}{\input{appendices}}
\optv{xam}{Type: \verb|[root /]$ rm -r *|.}
```

Note that both the package option and the `\opt` argument can contain lists of options although, in practice, one or the other should be a single option name. Lists are allowed in both places to allow more flexibility in the style of use. (But making the definitions much more difficult, Grrr.)

Just as for `\includeonly`, you will have to edit the main document file to switch option codes (i.e., change the `\usepackage` line). There are, however, several ways to use this package without altering the main document file: separate files, file-name sensing, interactive prompting, and command-line option selection.

Typically, different versions of a document will require different document class and package setup, besides the different tags for optional.sty. In that case it is best to have a separate main file for each version of the document. Each stub file will declare the document class and load some packages (including this one) and then input the rest of the document from a file common to all versions.

```
\documentclass[A0]{poster}
\usepackage[poster]{optional}
\input{my_paper}
```

If the different opt-tags match the different stub file names (file poster.tex will typeset the `poster` version) then you can specify

```
\usepackage[\jobname]{optional}
```

Alternatively, this `\jobname` technique can make use of symbolic links, if your computer system supports them, by having a single main input file accessed under different names (and different `\jobname`s).

Another scheme is to invoke LaTeX with the command line such as:

```
latex \def\UseOption{opta,optb}\input{file}
```

(with quoting appropriate to your operating system) then options `opta` and `optb` will be used in addition to any options specified with the `\usepackage` command.

You can prompt yourself to specify the option(s) with every run through LaTeX:

```
\usepackage{optional}
\newcommand{\ExplainOptions}{man = users manual, check = checklist,
      ref = reference card, post = poster.}
\AskOption
```

The definition of `\ExplainOptions` is optional; it only serves to help the person who answers the question. The `\AskOption` is also optional; it will be executed automatically whenever optional.sty sees no list of options. This method is too tedious to use much.

The normal restrictions forbidding special characters in package options and reference tags apply also the the tags used by the `\opt` command.

These are not `comment` macros: The optional text must be well-formed with balanced braces, even if not printed. The `\opt` command *is* completely `expandable` which means it is robust and can even be used in messages (`\typeout`).

As usual, `\verb` commands and verbatim environments cannot be used in the argument to `\opt`. For this purpose there is a variant form of `\opt` called `\optv` (optional verbatim) which may have a limited class of verbatim material in the argument. It can do so by leaving the braces around the argument, which may have undesired side effects. For an `\optv` argument to be successfully ignored, the verbatim material must have balanced braces etc.

The `\opt` command is only intended for small sections of text. If you need to optionally include whole sections or chapters, put that material in a separate file, and `\opt`-ionally use an `\input` command:

```
\opt{internal}{\input{prog_listings}}
```

```
%===================== END INSTRUCTIONS =======================

\ProvidesPackage{optional}[2005/01/26 ver 2.2b; \space
  Optional inclusion/omission]
```

Initialize used-option-list to `\@gobble` to eat the comma when the first entry is `appended`.

```
\@ifundefined{UseOption}{\let\UseOption\@gobble}{}
\DeclareOption*{\edef\UseOption{\UseOption,\CurrentOption}}
\ProcessOptions
\AtBeginDocument{\Opl@Setup}

\newcommand*\opt[1]{\if\Opl@notlisted{#1}\expandafter\@gobble
   \else \expandafter\@firstofone \fi}

\newcommand*\optv[1]{\if\Opl@notlisted{#1}\expandafter\@gobble\fi}

```

This initial definition forces immediate setup if `\opt` used in the preamble

```
\def\Opl@notlisted{\fi \Opl@Setup \if\Opl@notlisted}

\newcommand\AskOption{%
 \@ifundefined{ExplainOptions}{}{\typeout{\ExplainOptions}}%
 \typein[\UseOption]{Specify which optional text to process:}%
 }

\def\Opl@Setup{%
 \ifx\UseOption\@gobble\AskOption\fi
 \let\Opl@notlisted\@empty % initialize list of checks
 \@for\@tempa:=\UseOption\do{%
```

```
    \ifx\@tempa\@empty\else\expandafter\Opl@oneop\expandafter{\@tempa}\fi}%
  \ifx\Opl@notlisted\@empty \PackageWarning{optional}%
    {No options were selected, so all optional text will be printed}%
    \let\opt\@secondoftwo
  \else
    \typeout{Using optional text marked with \UseOption. }%
    \toks@\expandafter{\Opl@notlisted}%
    \edef\@tempa{\def\noexpand\Opl@notlisted####1{,\the\toks@,}}\@tempa
  \fi
  \let\Opl@Setup\@empty \let\Opl@oneop\undefined
  \let\AskOption\undefined \let\ExplainOptions\undefined
}
\begingroup
\catcode`\Z= 3 % special delimiter
\gdef\Opl@oneop#1{%
  \@ifundefined{Opl@Match@#1}{%
    \toks@\expandafter{\Opl@notlisted}%
    \edef\Opl@notlisted{\the\toks@ \csname Opl@Match@#1\endcsname ,####1,#1,Z}%
   \@namedef{Opl@Match@#1}##1,#1,##2Z{##2}%
  }\relax
}
\endgroup
\endinput
```