The threadcol package*

Scott Pakin
scott+thrcl@pakin.org

January 6, 2013

1 Introduction

Consider the following situation: You have a two-column PDF file that you want to read on your computer (or tablet or whatever). Because you have a relatively small screen—and/or less-than-perfect eyesight—you zoom in to more easily read the text. You read the first column of the first page, then scroll back to the top of the page and over to the right to read the second column, then scroll left to read the first column of the second page, then scroll up and over to read the second column, and so forth. With all this distracting scrolling, it's easy to lose track of where you were or what you were reading.

The threadcol package helps you avoid this situation for the LaTeX documents you create. It puts every column into a PDF "article thread". A user can then opt to have his PDF reader automatically scroll through the document in proper reading order. In Adobe Acrobat/Reader this is accomplished simply by clicking any place in the document window where the mouse pointer is shown as a hand with an arrow in it. The user can scroll forward by clicking in the document window or pressing *Enter* and backward by shift-clicking or pressing *Shift+Enter*. See the Adobe Acrobat/Reader *Help* documents for more information.

Adobe Acrobat/Reader provide an Articles navigation panel that lists all of the document's article threads and lets the user jump to a specified thread. Figure 1 shows what the *Articles* panel looks like in Adobe Reader 9 on Linux. The panel may have to be displayed explicitly by the user. This can be done by right-clicking on the navigation-panel button list and selecting *Articles*. Alternatively, in Adobe Acrobat/Reader 9, one can also follow the *View* → *Navigation Panels* → *Articles* menu path or in Adobe Acrobat/Reader X and XI the *View* → *Show/Hide* → *Navigation Panels* → *Articles* menu path. If *Articles* appears as a pop-up, it can be docked simply by dragging the tab to the navigation panel.

I don't know if any PDF readers other than Adobe's provide special viewing of threads.

*This document corresponds to threadcol v1.0, dated 2013/01/06.

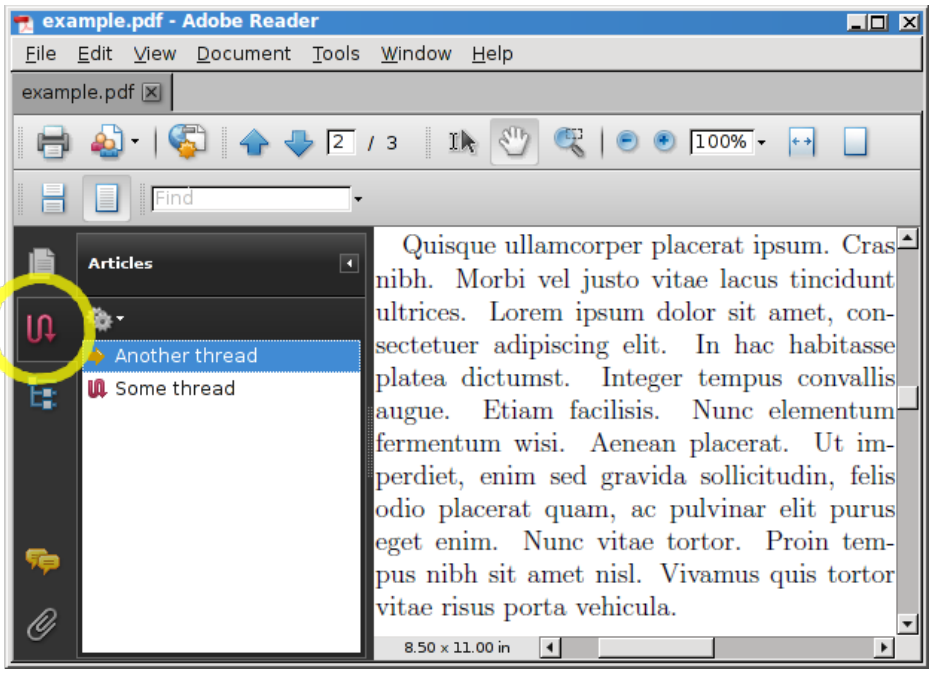

Figure 1: The *Articles* button and navigation panel in Adobe Reader 9

Although threadcol is most beneficial to two-column documents, it nevertheless also works with single-column documents—and even documents that switch between the two using `\onecolumn` and `\twocolumn`. In fact, although it is typeset in a single column, this document is itself threaded using threadcol. Section 4, which presents the commented threadcol source code, lies in a thread entitled, "Developer documentation". Sections 1–3 and 5 lie in a thread entitled, "User documentation". What this structure implies is that you can double-click "User documentation" in the Articles navigation panel then repeatedly click (or press *Enter*) on the document's text to read all of the user documentation while automatically skipping the developer documentation. Likewise, double-clicking on "Developer documentation" takes you right to that, bypassing all of the user documentation.

Indexes are not normally read from start to finish so this document's index is omitted from both the "User documentation" and "Developer documentation" threads.

2 Usage

One of the threadcol package's goals is simplicity. All you have to do is include a

```
\usepackage{threadcol}
```

in your document's preamble for threadcol to do its work.

`\setthreadname` One bit of customization that threadcol does provide, however, is control over the name of the article thread. By default, this is "Entire document". However, if you prefer a different name, you can write `\setthreadname{`⟨*name*⟩`}` to change the name of the article thread to ⟨*name*⟩. It is in fact permissible to invoke `\setthreadname` multiple times and with different names. In this case, threadcol produces one thread per unique name. For example, the document shown in Figure 1 used two threads in the following manner:

```
\setthreadname{Some thread}
⟨Text for the first thread⟩
\setthreadname{Another thread}
⟨Text for the second thread⟩
\setthreadname{Some thread}
⟨More text for the first thread⟩
```

When the user clicks on the "Some thread" thread, navigation automatically bypasses all text in the "Another thread" thread and vice versa. One caveat is that in the current version of threadcol, `\setthreadname` issues a `\clearpage` command to ensure that text is assigned to the correct thread. Hence, threadcol cannot currently be used for sophisticated magazine- or newspaper-style documents with intertwined threads weaving through the pages. Still, it may be useful for coarsely divided units of text.

As a special case, specifying an empty thread name (i.e., `\setthreadname{}`) stops adding text to threads. This can be useful for a document's front matter and back matter, which may not belong in any thread's normal reading order.

3 Limitations

threadcol is a fairly simple package. As such, it has a number of limitations, including the following:

1. threadcol requires pdfLaTeX or LuaLaTeX; it does nothing when used with ordinary LaTeX or XeLaTeX. The package also relies on etoolbox, which requires ε-TeX support, but this is provided by all modern TeX distributions.

2. threadcol is incompatible with the fixltx2e and cuted packages. These packages redefine LaTeX's text-output routines in a manner that confuses threadcol.

3. Marginal notes (`\marginpar`) are not included in threads. The same is true for text that sticks out into the margin (e.g., using `\llap` or `\rlap`), such as the macro names in Section 4.

4. As mentioned in Section 2, threads currently must begin on their own page. Hence, `\setthreadname` forces a page break, which is often undesirable.

5. threadcol does not recognize columns created using the `multicols` environment from the multicol package. These appear to threadcol as a single column.

6. No attempt is made to preserve proper reading order beyond page and column order. For example, if a page ends with the first part of a sentence and the next page begins with a top float, the thread will present the text in that same order: the first part of the sentence, then the float, then the second part of the sentence. Even worse, threadcol is oblivious to footnotes that span columns; it will show the first column of text, including the first part of a long footnote, then the second column of text, which ends with the second part of the long footnote.

7. The package has not been tested thoroughly. Consequently, there are probably many more limitations than those listed above.

4 Implementation

This section is intended for developers and advanced users to learn how threadcol is implemented. Most readers can ignore this section.

We begin by loading a few helper packages upon which we rely: ifpdf to ensure that we have access to `\pdfstartthread` and `\pdfendthread` and etoolbox for the `\patchcmd` macro.

```
\RequirePackage{ifpdf}
\RequirePackage{etoolbox}
```

`\thrcl@thread@name` Define the name of the current thread. This is what shows up in the *Articles* navigation panel in Adobe Acrobat and Adobe Reader.

```
\def\thrcl@thread@name{Entire document}
```

`\setthreadname` Let the author change the name of the current thread (i.e., `\thrcl@thread@name`). The starred version suppresses the `\clearpage`.

```
\newcommand*{\setthreadname}{%
  \@ifstar{\gdef\thrcl@thread@name}{\clearpage\gdef\thrcl@thread@name}%
}
```

`\thrcl@threaded@box` This is a copy of the column box but surrounded by `\pdfstartthread` and `\pdfendthread`.

```
\newbox\thrcl@threaded@box
```

`\thrcl@box` Mimic TEX's `\box` primitive but wrap the given box within a `\pdfstartthread` and `\pdfendthread`. If `\thrcl@thread@name` is empty, however, invoke `\box` directly.

```
\def\thrcl@box#1{%
  \ifx\thrcl@thread@name\@empty
    \box#1
  \else
    \setbox\thrcl@threaded@box=\vbox{%
      \pdfstartthread name {\thrcl@thread@name}%
      \copy#1
      \pdfendthread
    }%
    \box\thrcl@threaded@box
  \fi
}
```

`\thrcl@orig@outputpage` We want `\thrcl@outputdblcol` to use the original `\@outputpage` but all other invocations to use our modified `\@outputpage`.

```
\let\thrcl@orig@outputpage=\@outputpage
```

`\thrcl@patchcmd` Wrap etoolbox's 5-argument `\patchcmd` with a 3-argument version that uses hard-wired success and failure operations. The whole command is executed only if all of the previous `\thrcl@patchcmd` commands succeeded.

```
\def\thrcl@patchcmd#1#2#3{%
  \ifx\thrcl@patches@succeeded Y
    \patchcmd{#1}{#2}{#3}{}{\let\thrcl@patches@succeeded=N}%
  \fi
}
```

\thrcl@outputdblcol
\thrcl@outputpage
\thrcl@comdblflelt
\thrcl@patches@succeeded

Replace \box with \thrcl@box in LaTeX 2ε's \@outputdblcol and \@outputpage macros, which are used to output a column in, respectively, a two-column document or a one-column document. To avoid nesting of \pdfstartthread... \pdfendthread, we replace calls to \@outputpage with calls to \thrcl@orig@outputpage in \@outputdblcol.

Because there we need to apply multiple patches, we attempt to patch copies of \@outputdblcol, \@outputpage, and \@comdblflelt and replace the originals only if all changes are successful. We store "Y" in \thrcl@patches@succeeded if this is the case, otherwise "N".

```
\let\thrcl@outputdblcol=\@outputdblcol
\let\thrcl@outputpage=\@outputpage
\let\thrcl@comdblflelt=\@comdblflelt
\let\thrcl@patches@succeeded=Y
\ifpdf
```

We're in PDF-generating mode. Apply all of our patches.

```
  \thrcl@patchcmd
    {\thrcl@outputdblcol}%
    {\box\@leftcolumn\hss}%
    {\thrcl@box\@leftcolumn\hss}%
  \thrcl@patchcmd
    {\thrcl@outputdblcol}%
    {\box\@outputbox\hss}%
    {\thrcl@box\@outputbox\hss}%
  \thrcl@patchcmd
    {\thrcl@outputdblcol}%
    {\@outputpage}%
    {\thrcl@orig@outputpage}%
  \thrcl@patchcmd
    {\thrcl@outputdblcol}%
    {\@outputpage}%
    {\thrcl@orig@outputpage}%
  \thrcl@patchcmd
    {\thrcl@outputpage}%
    {\box\@outputbox}%
    {\thrcl@box\@outputbox}%
  \thrcl@patchcmd
    {\thrcl@comdblflelt}%
    {\box}%
    {\thrcl@box}%
  \ifx\thrcl@patches@succeeded Y
```

All of our patches succeeded. We can finally redefine \@outputdblcol, \@outputpage, and \@comdblflelt for real.

```
    \global\let\@outputdblcol=\thrcl@outputdblcol
    \global\let\@outputpage=\thrcl@outputpage
    \global\let\@comdblflelt=\thrcl@comdblflelt
  \else
```

Issue a warning message if any patch failed. This should happen only if a class or package redefines `\@outputdblcol`, `\@outputpage`, or `\@comdblflelt` in a manner incompatible with LaTeX 2$_\varepsilon$'s default definition.

```
    \PackageError{threadcol}{Failed to patch the output routine}{%
      The threadcol package needs to modify LaTeX's
      \protect\@outputdblcol\space macro to\MessageBreak
      incorporate support for PDF article threads.  These
      modifications failed,\MessageBreak
      presumably due to a class or package that redefined
      \protect\@outputdblcol\space in a\MessageBreak
      form incompatible with what threadcol expects.%
    }%
  \fi
\else
```

We're not in PDF-generating mode. Warn the author that the package will do nothing.

```
  \PackageWarningNoLine{threadcol}{%
    This package has an effect only when running\MessageBreak
    pdfLaTeX or LuaLaTeX and only when in\MessageBreak
    PDF-generating mode%
  }%
\fi
```

5 Future work

A future version of threadcol may address some of the limitations described in Section 3. In addition, it would great if the package could integrate seamlessly with the flowfram package, where PDF article threads would make a lot of sense.

One far more difficult change to implement would be to make threadcol cognizant of the "true" reading order, perhaps with help from the author. For example, the author could specify the point at which the thread should jump to a floating figure before jumping back to the corresponding text.

In practice, LaTeX 2_ε's output routines can be quite arcane, especially with regards to the handling of inserts (e.g., floats and footnotes). Getting threadcol to do more than what it currently does is likely beyond my current level of LaTeX expertise.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.