The cybercic package*

Jared Jennings
jjennings@fastmail.fm

June 23, 2015

Contents

---

*This document corresponds to cybercic v2.1, dated 2015/06/24.

1 What's it for?

This package, when used in conjunction with cyber, adds IA control names to division names in the table of contents. (The cic stands for "controls in contents." *Division* means chapter, section, subsection, subsubsection, etc.)

As an example, suppose you have a section in your document like this:

```
% \section{Stuff about individual authentication}
%
% \documents{iacontrol}{IAIA-1} Bla, bla, bla...
%
```

Without cybercic, this creates a line in your table of contents that says something like, "3.2 Stuff about individual authentication ..... 34," and a section heading in the body of the document, on (in our example) page 34, that says something like, "3.2 Stuff about individual authentication."

When you add cybercic, the line in your table of contents will say, "3.2 Stuff about individual authentication (IAIA-1) ... 34." The section head in the body of the document will stay the same.

2 How to use it

In the preamble of your document, write `\usepackage{cybercic}`. But! You must be careful where you write it. If you use hyperref, you must use cybercic *after* hyperref. You must use cybercic after cyber, as well.

`\Cybercontrolsincontentsenabledfalse`
`\Cybercontrolsincontentsenabledtrue`

If there is a part of your document wherein you do not want section titles in the table of contents changed, at the beginning of that part write `\Cybercontrolsincontentsenabledfalse`. When you want controls-in-contents re-enabled, write `\Cybercontrolsincontentsenabledtrue`.

3 Caveats

First and foremost, when you use this package, and you also use hyperref to make a PDF with bookmarks in it, you cannot put *any* formatting in your section titles. This means you cannot write something like:

```
% \section{My \emph{Awesome} Section}
% \section{The {\tt /usr/var/tmp} directory}
%
```

You must content yourself with:

```
% \section{My Awesome Section}
% \section{The /usr/var/tmp directory}
%
```

This package is a horrible (but ingenious) hack. Do not use it unless you really, really need to. It may not play well with other packages.

If a section spans multiple pages, and compliance posture tags (like `\documents` or `\implements`, see `cyber` documentation) are strewn throughout, running page heads containing section names will only show IA control names mentioned before the page began. Patches gratefully accepted.

4 Implementation

```
\makeatletter
```

Given that we are using controlsincontents, sometimes it needs to be disabled for part of a document. The newif here is for that purpose.

ercontrolsincontentsenabledtrue
rcontrolsincontentsenabledfalse

```
\newif\ifCybercontrolsincontentsenabled
\Cybercontrolsincontentsenabledtrue
```

The word “division” here is a general term for a part, chapter, section, subsection, subsubsection, paragraph, or subparagraph.

Every kind of division has an optional argument, what should show in the table of contents instead of the title shown in the text. Our strategy is to set that optional toc name to a reference to a macro. The macro is first defined to contain the original name of the section, but it can be redefined later in the document to tack text onto the end of it.

`http://www.elektro.uni-miskolc.hu/~gati/references/latex/macro/tugpap1.pdf` introduces this sort of indirection.

Only the last definition of the division’s title macro that appeared in the document is expanded when the `.toc` file is written. So the table of contents then contains all the additions to the division’s title as well as the title itself. Unfortunately, at the part of the document where the division title is typeset in the text, none of the redefinitions have happened yet, so only the title which shows in the TOC can be amended using this method.

We need to name the macro that will contain each division’s title. Each name must be unique, so we use the counters for different kinds of sections; but each name must also be a valid identifier, so instead of using numbers we use letters. (Alph bombs out after it gets to number 26, so Roman it is.)

```
\newcommand \alphpart          {pt\Roman{part}}
\ifdefined\chapter
\newcommand \alphchapter       {\alphpart ch\Roman{chapter}}
\newcommand \alphsection       {\alphchapter se\Roman{section}}
\else
\newcommand \alphsection       {\alphpart se\Roman{section}}
\fi
\newcommand \alphsubsection    {\alphsection su\Roman{subsection}}
\newcommand \alphsubsubsection {\alphsubsection ss\Roman{subsubsection}}
\newcommand \alphparagraph     {\alphsubsubsection p\Roman{paragraph}}
\newcommand \alphsubparagraph  {\alphparagraph sp\Roman{subparagraph}}
```

Now we're going to replace `\@sect`. This macro is used for all divisions `\section` and smaller.

```
\let\skia@orig@sect\@sect
\gdef\@sect#1#2#3#4#5#6[#7]#8{%
    \ifCybercontrolsincontentsenabled
```

#1 is the kind of division, e.g. section or subparagraph. #2 is the depth of this kind of division in the hierarchy (1 is a chapter, 2 is a section, etc)

```
    \ifnum #2>\c@tocdepth
```

The kind of division we're starting is too detailed to show up in the table of contents. Just do the usual `@sect` thing. (Before this special case was written, any use of `\subsubsection` resulted in an `\inaccessible` error.)

```
        \skia@orig@sect{#1}{#2}{#3}{#4}{#5}{#6}[#7]{#8}
```

A side effect of not redefining `\last@division` and `\last@divtitle` here as we do below is that any `\addtosectionname`s that happen will add to the name of the last-declared TOC-worthy division - which will be a larger division that contains this one.

For example, assume `\setcounter{tocdepth}{1}`, i.e. only chapters are shown in the TOC. Now if we `\chapter{One}` `\section{Onedotone}` `\addtosectionname{Bla}`, Bla will be added to One, not onto Onedotone.

```
    \else
```

This division will show up in the TOC. Prepare it to have text added onto its name there. Let's capture a unique name for the section being created:

```
        \edef\last@division{#1\csname alph#1\endcsname}
```

(An example expansion would be paragraphptchseIsussIIp.) Now we construct a name for a macro which will contain the title of this section.

```
        \edef\last@divtitle{\csname titleof\last@division \endcsname}
        \edef\last@divaddedto{\csname addedto\last@division \endcsname}
```

Now we define that macro. The expandafter causes `\last@divtitle` to be expanded before the `\def`, so that we are not defining `\last@divtitle`, but the thing it expands to—in our example, we are defining `\titleofparagraphptchseIsussIIp`. Mind-bending, eh?

```
        \expandafter\def\last@divtitle{#7}
```

Now we call the old `@sect`, giving `\last@divtitle` as the section title to use in the table of contents. But we don't want to expand it right now, because it would just expand to #7. So we put a `\noexpand` first. But we don't want its value to be exactly "`\last@divtitle`", because that one gets redefined all the time, and only the last definition would be used in writing the toc file, so the name of every section in the toc would end up the same... This expansion stuff is a bit fiddly, isn't it? So instead of using `\last@divtitle`, as above, we have to write its expansion out here.

```
        \skia@orig@sect{#1}{#2}{#3}{#4}{#5}{#6}[\noexpand\csname titleof\last@division\endcsname]
    \fi
    \else
```

If we're in this `else`, Cybercontrolsincontentsenabled is not true. Just do the usual section thing.

```
      \skia@orig@sect{#1}{#2}{#3}{#4}{#5}{#6}[#7]{#8}
    \fi}
```

And we need to add the same machinery to `\@chapter`—if it exists. (Some document classes do not define it.)

```
\ifdefined\@chapter
\let\skia@orig@chapter\@chapter
\gdef\@chapter[#1]#2{%
    \ifCybercontrolsincontentsenabled
    \edef\last@division{chapter\alphchapter}
    \edef\last@divtitle{\csname titleof\last@division \endcsname}
    \edef\last@divaddedto{\csname addedto\last@division \endcsname}
    \expandafter\def\last@divtitle{#1}
    \skia@orig@chapter[\noexpand\csname titleof\last@division\endcsname]{#2}
    \else
    \skia@orig@chapter[#1]{#2}
    \fi}
\fi % defined @chapter
```

Fix PDF bookmarks by using the expansion of the title macro, not its name.

When you use hyperref and you tell it to make PDF bookmarks, the bookmarks it writes to the PDF file have to have very normal sorts of names. I think you may get to use UTF-8, but you can't have any text formatting. The way hyperref makes sure of this is by temporarily defining all of the LaTeX formatting macros you may want to use, like `\textsf`, to do nothing, so that they expand to their contents rather than some TeX code that makes the formatting happen plus the contents, and then evaluating the line that's supposed to go in the table of contents (the optional argument to `\@section` or `\@chapter` which defaults to the first argument) expanding only those macros. As I recall. It's been six months since I figured it out.

But we've set the section name as it should appear in the contents to a macro, and not a macro that hyperref is prepared for. So the PDF bookmark for section 1.2 which begins on page 4 says `1.2 \titleofsectionptchseI  4` instead of `1.2 My Cool Section (IAIA-1)  4`.

Our hamfisted solution to this is to go ahead and fully expand the parameter. That means that the `\titleofsectionptchseI` macro is expanded, to "My Cool Section (IAIA-1)." But it also means that all of those guards that hyperref had to deal with what happens if I write `\section{My \emph{Awesome} Section}` are gone, and suddenly if there is any formatting in any section names, bad things happen. So this is the part of the code that necessitates that all division titles have no formatting in them at all.

Most of this definition of addcontentsline comes from the hyperref package. For this to work properly, cyber must be loaded *after* hyperref.

```
\@ifpackageloaded{hyperref}{%
    \gdef\addcontentsline#1#2#3{% toc extension, type, tag
      \begingroup
    \let\label\@gobble
    \let\textlatin\@firstofone
    \ifx\@currentHref\@empty
      \Hy@Warning{%
        No destination for bookmark of \string\addcontentsline,%
        \MessageBreak destination is added%
      }%
      \phantomsection
    \fi
    \expandafter\ifx\csname toclevel@#2\endcsname\relax
      \begingroup
        \def\Hy@tempa{#1}%
        \ifx\Hy@tempa\Hy@bookmarkstype
          \Hy@WarningNoLine{bookmark level for unknown #2 defaults to 0}%
        \else
          \Hy@Info{bookmark level for unknown #2 defaults to 0}%
        \fi
      \endgroup
      \expandafter\gdef\csname toclevel@#2\endcsname{0}%
    \fi
    \edef\Hy@toclevel{\csname toclevel@#2\endcsname}%
```

Unlike hyperref, expand #3 so we don't try to use a macro name as the title of the bookmark. But #3 is likely a `\numberline{...}`, and at the time this is expanded, it seems that `\numberline` is some LaTeXy thing, not a hyperreffy thing, and hyperref complains about all the TeX code in the expansion of #3. So we make sure that i s going to just put the number in.

```
    \let\saved@numberline\numberline
    \ifHy@bookmarksnumbered
      \let\numberline\Hy@numberline
    \else
      \let\numberline\@gobble
    \fi
    \edef\Hy@expandedtag{#3}
    \let\numberline\saved@numberline
```

Now, about writing that bookmark.

```
\Hy@writebookmark{\csname the#2\endcsname}%
      {\Hy@expandedtag}%
      {\@currentHref}%
      {\Hy@toclevel}%
      {#1}%
    \ifHy@verbose
      \typeout{pdftex: bookmark at \the\inputlineno:
        {\csname the#2\endcsname}
        {\Hy@expandedtag}
        {\@currentHref}%
        {\Hy@toclevel}%
        {#1}%
```

```
          }%
        \fi
        \addtocontents{#1}{%
          \protect\contentsline{#2}{#3}{\thepage}{\@currentHref}%
        }%
      \endgroup
    }
}

```

That blank line just above seems to be necessary.

To avoid restating IA controls in appended-to section names, we'll need a substring search function.

From the substr package, version 1.2, 2009/10/20, copyright 2000, 2005, 2009 Harald Harders, available from CTAN:

—

expands the first and second argument with `\protected@edef` and calls #3 with them:

```
\newcommand\su@ExpandTwoArgs[3]{%
  \protected@edef\su@SubString{#1}%
  \protected@edef\su@String{#2}%
  \expandafter\expandafter\expandafter#3%
  \expandafter\expandafter\expandafter{%
    \expandafter\su@SubString\expandafter
  }\expandafter{\su@String}%
}
```

tests if #1 in #2. If yes execute #3, else #4

```
\newcommand*\IfSubStringInString[2]{%
  \su@ExpandTwoArgs{#1}{#2}\su@IfSubStringInString
}
\newcommand*\su@IfSubStringInString[2]{%
  \def\su@compare##1#1##2\@nil{%
    \def\su@param{##2}%
    \ifx\su@param\@empty
      \expandafter\@secondoftwo
    \else
      \expandafter\@firstoftwo
    \fi
  }%
  \su@compare#2\@nnil#1\@nil
}
```

—

End substr package excerpt.

Now, to append some text to the name of the section we're in, we just have to redefine the macro whose name is the value of `\last@divtitle`. In order to add to it without infinite recursion, we use `\edef`, which expands its body before defining. Again, rather than defining `\last@divtitle` itself, we want to define the macro it names. The expandafter does this.

```
\def\addtosectionname#1{%
    \def\skia@yes{yes}
    \expandafter\ifx\last@divaddedto\skia@yes
      \IfSubStringInString{#1}{\last@divtitle}{}{%
        \expandafter\edef\last@divtitle{\last@divtitle , #1}%
      }
    \else
      \expandafter\edef\last@divtitle{\last@divtitle ---#1}
    \fi
    \expandafter\def\last@divaddedto{yes}}

\makeatother
```