annote
A tool to insert notes within text
(from the VIDEO project)
Revision: 1.30

Christian Queinnec
Université Paris 6 — Pierre et Marie Curie
LIP6, 4 place Jussieu, 75252 Paris Cedex – France

This documentation is written in English since it it the language for Computer Science as much as French is the official language for fencing, jumping, etc. This allows this documentation as well as the source programs to be read, debugged and enhanced from amateurs or afficionados from many countries. Actually these programs are released under the Gnu Public License[1] version 2.

The `annote` tool is part of the VideoC[-1-] set of tools used to produce an educational interactive CD to teach the C programming language. Its primary goal was to decorate code excerpts with popup explanation messages. The `annote` tool is a program that swallows a whole stream of characters, processes it and outputs the resulting text. Directives appear throughout the text; they allow to insert notes specified within independent annotation files. The `annote` tool thus provides a way to describe texts and/or programs accompanied by annotations without cluttering their definition.

One of the key principles of the VideoC[-1-] project was to be able to derive, from the same set of source files: a book and an interactive CD. Therefore the `annote` tool produces an output which is compatible with:

- TeX[-2-], the formating language of choice. An appropriate LaTeX[-3-] style allows to render these annotations as sort of minipage footnotes[-4-]. See below for the annotations associated to the funny numbers that appear in the margin.
- LaTeX2HTML [2] or HEVEA [3], it is also possible to obtain an HTML (mixed with Javascript and/or Java snippets) version with annotations rendered as pop-up windows automatically raised when the mouse is over some sensitive regions[-6-];
- `pdftex`, a PDF version may also be obtained. Contrarily to the HTML version, the PDF version is not animated.

The `annote` tool is particularly interesting when commenting excerpts of programs since nobody wants to pollute program files (and endanger their readability) nor their apparency on the screen (one can copy-paste snippets of code as they are seen not as they are encoded). The source program is rendered exactly as it is written and annotations are inserted in a non-obtrusive way. Moreover, annotations and annotated texts are kept inside different files therefore, the same program may be commented more than once; alternatively some annotations may apply to more than one program.

The `annote` tool itself and the other programs of the VideoC[-1-] project are available for download [4].

[1] `http://www.gnu.org/copyleft/gpl.html`
[2] `http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html`
[3] `http://para.inria.fr/~maranget/hevea`
[4] `http://www.infop6.jussieu.fr/cederoms/Videoc2000/clic/tools/VideoC.html`

-1- The project was funded by Université Paris 6. Its aim is to build an educational CD to teach the C programming language (hence the name of the project).

-2- TEX is due to Donald Knuth [Knu84].

-3- LATEX is due to Leslie Lamport [Lam86].

-4- A minipage footnote is a LATEX concept for those in the know [Lam86, p. 195]. Our annotations look like them but are rendered with fully redefinable LATEX macros.

-5- With the current CSS, mouse sensitive areas are blue. Roll your mouse over and see the attached annotation.

-6- When the mouse is over that region, a popup window with additional information automagically appears. If, without moving the mouse and after the window appeared, you click, the window is nailed on the screen. A second click in the sensitive area will let it vanish. The displayed information may contain hyperlinks, images and so forth.

1 Examples

Of course, the present documentation makes use of the `annote` tool itself and serves as a test file for `annote`. You may view it in Postscript [5] or in HTML [6]. Depending on the version various features are demonstrated. The Postscript and PDF versions come from TEX. They both contain a screen snapshot of «La Charte C» [7], another document processed by the `annote` tool better viewed with an HTML browser. See figure 1 for a snapshot.

Here is another example, a small excerpt of code taken out of the `annote.pl` program itself:

```
sub driver {                                                          -7-
   local($_) = @_;
   mask_without_notes_sections();                                     -8-
   read_all_annotation_files("\");                                    -9-
   introduce_missing_vicnotes();                                      -10-
   introduce_text_of_annotations("\\" . "ViCPlacePopUpHere");         -11-
   process_anchors();                                                 -12-
   unmask_without_notes_sections();                                   -13-
   return $_;
}
```

The annotations of the previous snippet of code appear below. They look better when they are raised by an hovering mouse.

[5]http://www.infop6.jussieu.fr/cederoms/Videoc2000/clic/tools/annote.ps
[6]http://www.infop6.jussieu.fr/cederoms/Videoc2000/clic/tools/annote.html
[7]http://www-spi.lip6.fr/~queinnec/cours/C/CharteC.html

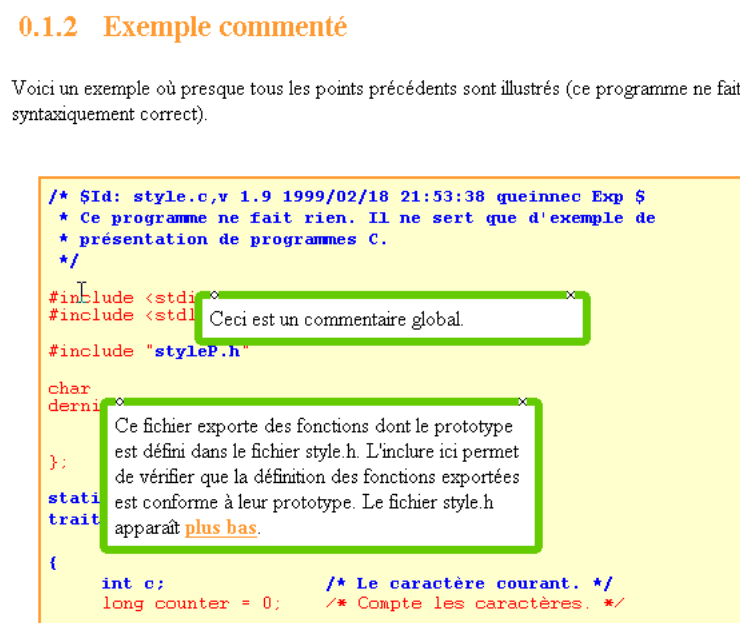

Figure 1: A screen snapshot. Annotations are bordered. The annotation above is temporarily popped-up (the cursor has disappeared but was hovering over the C commentary) while the annotation below has been nailed down.

-7- This subroutine is written in Perl as the rest of the `annote` tool.

-8- This routine hides the sections of text embedded between `\ViCWithoutNotes` and `\ViCWithNotes` macros.

-9- All annotation files specified by `\ViCAnnotation` macros are read and parsed.

-10- For all notes that are present in the annotation files but not referenced from a `\ViCNote`, insert such a macro based on the definition of the note.

-11- Process all `\ViCNote` and insert their associated explanation right before the next `\ViCPlacePopUpHere` marker.

-12- Embed mouse-sensitive regions between `\ViCAnchor` and `\ViCEndAnchor`.

-13- Restaure regions hidden by `mask_without_notes_sections`.

2 The current process

The `annote` tool is, currently, totally written in Perl*-14-* [WCS96]. The following phases are performed:

1. Command options are processed.
2. The whole text to annotate is read in memory.

3. Annotation files are read (they are specified with `\ViCAnnotation{`*file*`}` and they usually bear the `.ann` extension). `\ViCAnnotation` are ignored if embedded within `\ViCWithoutNotes` and `\ViCWithNotes` markers.

4. For all notes appearing in annotation files, `\ViCNote{`*notename*`}` macros are inserted if missing, except within `\ViCWithoutNotes` and `\ViCWithNotes` markers.

5. Annotations are introduced that is, `\ViCNote` macros are turned into `\ViCIndex` macros (notes are numbered at the same time) and the text explaining the note is inserted within a `\ViCPopUp` environment right before the next `\ViCPlacePopUpHere` marker. The mouse sensitive context is embedded within `\ViCAnchor` and `\ViCEndAnchor` macros.

6. The final text is output on the standard output stream.

A lot of bells and whistles is of course hidden here and there.

2.1 Command options

When the `annote` tool is invoked, its command arguments may be filenames or options. Options start with a dash. A single option is currently available:

-v be verbose. You can use it more than once to be even more verbose.

-g catch a glimpse of the stream to process before every phase. This option is useful for debugging when eager to know were are inserted anchors.

-d catch a glimpse of the current state of the stream in case of fatal errors. This option is useful for debugging. Pay attention some character codes (from 1 to 20) may appear in the output.

An unrecognized option triggers the printing of a small usage notice before exiting. Command arguments that are not recognized as options are considered to name files. All these files will be read, in order, into memory before the annotation process takes place. Missing files are reported.

Typical use is:

```
cat some files | annote.pl -v > document.tex
```
-15-

In this command, some files are gathered, processed by `annote.pl` with some verbosity then output in a TeX *-16-* file which may be further processed.

2.2 The big picture

The `annote` tool is currently named `annote.pl` and comes with numerous files that tailor the various phases through which a text is rendered on a screen. The big picture appears in figure 2.

This is a short introduction to these various files.

`annote.pl` is the Perl *-14-* script known as the `annote` tool.

`ViClib.pl` is a Perl *-14-* library used by `annote.pl` but also used by LaTeX2`HTML` *-17-*. It contains a lot of routines that may be of interest when using LaTeX2`HTML` *-17-*.

`videoc.sty` is the LaTeX *-18-* style file that gives a meaning to the macros inserted by `annote.pl`.

`videoc.perl` is the name of the LaTeX2`HTML` *-17-* style file that corresponds to the `videoc.sty` style file. It gives to LaTeX2`HTML` *-17-* a meaning for the macros inserted by `annote.pl`.

`videoc.hva` is the name of the HEVEA *-19-* style file that corresponds to the `videoc.sty` style file. It gives to HEVEA *-19-* a meaning for the macros inserted by `annote.pl`.

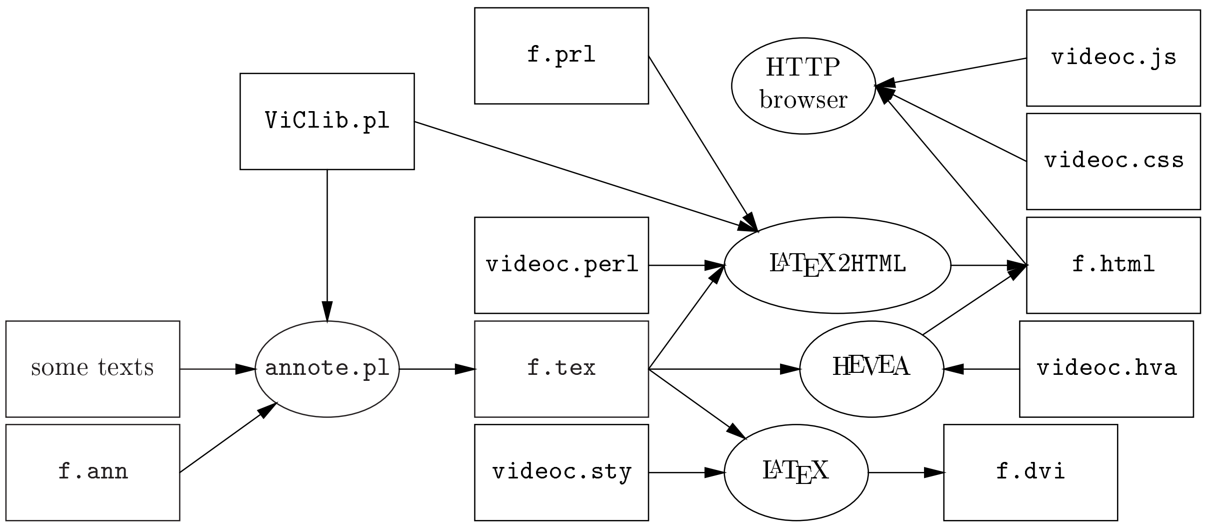

Figure 2: The place of the `annote` tool

`videoc.css` is the default Cascading Style Sheet referenced from the result of HEVEA *-19-* or LaTeX2HTML *-17-*. It defines however code must be rendered, how pop-up windows should be displayed, etc.

`videoc.js` is the JavaScript library used to animate pop-up windows. It knows how to handle incompatible browsers such as Communicator and Explorer but both of them should be at release 4 or more.

Some files customize the various filters:

`f.ann` is the annotation file containing the notes to be inserted in the text to process.

`f.prl` is the file that customizes the conversion from LaTeX *-18-* to HTML. It is written in Perl *-14-* and uses the resources of LaTeX2HTML *-17-* as well as these of `ViClib.pl`.

-14- Perl is due to Larry Wall [WCS96].

-15- Repeat this option to increase the verbosity.

-16- TeX is due to Donald Knuth [Knu84].

-17- LaTeX2HTML[8] is due to Nikos Drakos.

-18- LaTeX is due to Leslie Lamport [Lam86].

-19- HEVEA[9] is due to Luc Maranget.

[8] `http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html`
[9] `http://para.inria.fr/~maranget/hevea`

3 Format of annotation files

Annotations are held in files, usually named with an `.ann` extension. An annotation file roughly has an overall LaTeX[20] syntax (mimicing an XML structure) it may thus be edited with a LaTeX[20] mode within your favorite text editor.

An annotation file contains a series of notes and may end with an explicit `\endinput` macro. As in TeX[21], comments start with a percent sign and finish with the end of the line; unlike TeX[21], comments should only be preceded by whitespaces.

A note looks like a LaTeX[20] environment with a mandatory argument to hold the name of the note. Names of notes must be unique through an annotation file. Within a note may appear a number of fields. Typical uses of these fields appear in the following syntax:

```
\begin{note}{name }
   \begin{text}  a text spanning several lines   \end{text}
   \width{an absolute or relative width }
   \text-style{some style options }
   \what-style{some style options }
   \what  a regexp identifying the mouse sensitive region
   ... other fields ...
\end{note}
```

The possible fields of a note are listed in table 1.

field	**kind**	**argument**	**default**
`glimpse`	command	a file name	
`text-style`	command	a CSS directive	nothing
`what-style`	command	a CSS directive	nothing
`text`	environment	a LaTeX text	mandatory
`what`	command	a regexp	mandatory
`width`	command	an HTML size	50% by default

Table 1: Fields of notes

3.1 Fields of notes

Fields of notes are documented by order of importance. The annotation file associated to this very documentation is worth reading as an example.

text This is a mandatory argument. This is the text that will explain the sensitive region. It appears as a LaTeX[20] environment i.e., it starts with a `\begin{text}` and ends with a `\end{text}` macro. The text may contain images or hyperlinks.

what This field specifies the sensitive region. Pay attention, this field is specified as a Perl[22] regular expression (a regexp) and is written without being enclosed within braces. In the generated HTML version, when viewed by Communicator 4.x or Explorer 4.y, moving the mouse over this sensitive region will popup a window displaying the text of the note. If not specified, the sensitive region is the entire line where the `\ViCNote` is.

To ease hooking an annotation at a precise location, the regexp may also specify part of the context of the sensitive region. In that case, the sensitive region must be the first parenthetized sub-regexp. See examples below.

what-style This field specifies the style to apply to the sensitive region of the annotation. The content of this field appears within braces as a regular argument of a LaTeX[20] macro. The content is a CSS[23] directive. You may change colors, fonts etc. For instance a blue background may be specified as `\what-style{color: rgb(0,0,255);}`.

text-style This field specifies the style to apply to the explanation text of the annotation. The content of this field appears within braces as a regular argument of a LaTeX[-20-] macro. The content is a CSS[-23-] directive. You may change colors, fonts etc. For instance a blue background may be specified as `\text-style{color: rgb(0,0,255);}`. *Currently, you should not use this field with Communicator i.e., you should not use this field if you want to be portable.*

width This command specifies the width of the popup window. By default, this is half of the current size. Sizes appear within braces as a regular argument of a LaTeX[-20-] macro. Sizes are specified as in CSS[-23-] i.e., they may be relative as in `\width{66\%}` or absolute as in `\width{133px}` (which means 133 pixels).

glimpse This command may take the name of a file as argument otherwise it synthetizes a filename based on the name of the current note. This is a convenient field when debugging annotations since it dumps, in the specified file, the current state of the processed text right before handling the current note.

3.2 Examples of notes

The following are exemples of notes taken from the annotation file, (the `annote.ann` file since this file is named `annote.bk`), associated to this very documentation. The usual note simply contains a `text` field and a `what` field as follows. Every time the regular expression «CSS[-23-]» appears in the text, an annotation will be generated.

```
\begin{note}{css}
  \what CSS
  \begin{text} CSS stands for Cascading
               Style Sheet. \end{text}
\end{note}
```

Pay attention to the fact that the `what` field is a regular expression so backslashes, parentheses etc. must be handled cautiously. The following is an example of a more elaborate note with a background color:

```
\begin{note}{knuth}
  \what \\TeX{}
  \text-style{color: rgb{0,0,128);}
  \begin{text} \TeX{} is due to Donald Knuth \cite{Knuth84}. \end{text}
\end{note}
```

When the `what` field only contains words, pay attention to the interleaving separators since the words may be split apart by an end of line. For instance, here is a note that resists to a split between two words (it also illustrates a comment used in a note):

```
\begin{note}{a}
  % Pay attention: these two words may be split apart on two lines.
  \what sensitive\s*regions
  \begin{text}
        When the mouse is over that region, a popup window with
  additional information automagically appears. If, without moving the
  mouse and after the window appeared, you click, the window is nailed
  on the screen. A second click in the sensitive area will let it
  vanish. The displayed information may contain hyperlinks, images and
  so forth.
  \end{text}
\end{note}
```

Observe that notes specified with a `what` field appear any time the `what` regexp occurs in the text to be processed.

To ease positioning an annotation, one may use a slightly more complex `what` field to precise the context of the sensitive region as done in the following note.

```
\begin{note}{v-option}
  \what annote.pl (-v)
  \begin{text} Repeat this option to increase the verbosity. \end{text}
\end{note}
```

-20- LaTeX is due to Leslie Lamport [Lam86].

-21- TeX is due to Donald Knuth [Knu84].

-22- Perl is due to Larry Wall [WCS96].

-23- CSS stands for Cascading Style Sheet [10].

4 The resulting text

The `annote` tool swallows all its input stream then processes it. This section describes the annotation process and the contributing macros. These macros appear in table 2.

`\ViCAnnotation{`*filename*`}`
`\ViCNote{`*notename*`}`
`\ViCPlacePopUpHere`
`\ViCWithNotes`
`\ViCWithoutNotes`
`\ViCIndex[`*number*`,`*filename*`,`*notename*`]`
`\ViCAnchor[`*style*`][`*number*`,`*filename*`,`*notename*`]` *what* `\ViCEndAnchor[]`
`\begin{ViCPopUp}{`*number*`,`*filename*`,`*notename*`}{`*width*`}{`*style*`}` *text* `\end{ViCPopUp}`

Table 2: Macros read/written by the `annote` tool

First, all annotation files are read in memory. Annotation files are specified with the `\ViCAnnotation` macro. This document specifies for instance, `\ViCAnnotation{annote.ann}`. It is harmless to cite an annotation file more than once.

An annotation is inserted any time a `\ViCNote` macro appears. This macro mentions, as argument, the name of the note to insert, for instance, «`css`» in `\ViCNote{css}` specifies that the note named «`css`» must be inserted.

All the notes of all annotation files are examined and, if there is no associated `\ViCNote` explicitly mentioned in the text, the `where` field is used to determine all the locations where to insert the corresponding `\ViCNote`s. An error is signalled if a note cannot be placed somewhere in the text.

Every time a `\ViCNote{`*notename*`}` is found, it is transformed into a `\ViCIndex[`*number*`,`*filename*`,`*notename*`]` where *number* is the number given to the note (in the LaTeX *-24-* binding, this number appears in the margin), *filename* is the name of the annotation file from where comes the note and *notename* is the name of the note. A `ViCPopUp` environment will be inserted in the rest of the text; its argument are the reference of the note (that is, a *number*, a *filename* and the *notename*), the width and the

[10] `http://www.w3c.org/`

CSS-*25*- style to use in the popup window. Finally the mouse sensitive context starts with an opening `\ViCAnchor[`*style*`][`*number*`,`*filename*`,`*notename*`]` macro and ends with a `\ViCEndAnchor[`*number*`,`*filename*`,`*notename*`]` macro. Observe that arguments of the `\ViCIndex`, `\ViCAnchor` and `\ViCEndAnchor` macros are bordered by square brackets instead of curly brackets. Despite the usage in LaTeX-*24*-, these arguments is mandatory, not optional as it may seem. This convention had been taken to ease recognizing the arguments within code contexts.

The annotation is turned into a `ViCPopUp` environment. This environment will appear, after the `\ViCNote` that calls for it, right before the next `\ViCPlacePopUpHere` macro. A `ViCPopUp` environment contains everything that is necessary to generate a minipage note or a pop-up window later on.

When splitting a document into numerous independent HTML pages, pay attention to place at least one `\ViCPlacePopUpHere` per generated page. Since LaTeX2HTML-*26*- tends to process pages within independent Unix processes, it may also be necessary to let `\ViCAnnotation` macros be present in these pages either.

-*24*- LaTeX is due to Leslie Lamport [Lam86].

-*25*- CSS stands for Cascading Style Sheet [11].

-*26*- LaTeX2HTML[12] is due to Nikos Drakos.

4.1 Examples

Consider, for instance, the paragraph you are reading-*27*-. I make it long enough so that two notes-*28*- can simultaneously apply on it: one of them being explicitly called. Here is how I originally wrote it:

-*27*- Yes, this paragraph!

-*28*- The two notes are named «this-paragraph» and «two-notes».

```
Consider, for instance, the paragraph \ViCNote{this-paragraph}
you are reading. I make it long enough
so that two notes
can simultaneously apply on it: one of them being explicitly called. Here is
how I originally wrote it: \par
%%% QNC: Place the text of previous annotations here:
\ViCPlacePopUpHere%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The result of the `annote` tool looks like:

```
\ViCAnchor[][29,annote.ann,this-paragraph]Consider, for instance, the paragraph
you are reading\ViCEndAnchor[29,annote.ann,this-paragraph]. I make it long enough \ViCIndex[29,annote.ann,this-paragraph]
so that \ViCAnchor[][30,annote.ann,two-notes]two notes\ViCEndAnchor[30,annote.ann,two-notes]\ViCIndex[30,annote.ann,two-notes]
can simultaneously apply on it: one of them being explicitly called. Here is
how I originally wrote it: \par
%%% QNC: Place the text of previous annotations here:
\begin{ViCPopUp}{29,annote.ann,this-paragraph}{50\%}{}
 Yes, this paragraph!
\end{ViCPopUp}
```

[11] http://www.w3c.org/

[12] http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html

```
\begin{ViCPopUp}{30,annote.ann,two-notes}{50\%}{}
 The two notes are named «this-paragraph» and
  «two-notes».
\end{ViCPopUp}
\ViCPlacePopUpHere%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

In this excerpt you may see the introduction of the \ViCNote{two-notes} triggered by its «what» field containing the words *so that two notes*[-31-]. The two \ViCNotes are then transformed into \ViCIndex accompanied by ViCPopUp environments and \ViCAnchor and \ViCEndAnchor characterization of mouse sensitive regions[-32-].

-31- The two notes are named «this-paragraph» and «two-notes».

-32- When the mouse is over that region, a popup window with additional information automagically appears. If, without moving the mouse and after the window appeared, you click, the window is nailed on the screen. A second click in the sensitive area will let it vanish. The displayed information may contain hyperlinks, images and so forth.

5 Bindings

This section describes how to process the result of the annote tool.

5.1 Binding with TEX or pdftex

This is quite simple, just process the resulting stream with LATEX. Of course, the videoc.sty package should be used to give a meaning to the inserted macros (see table 2). It is quite simple to redefine these macros provided some simple TEX-knowledge.

5.2 Binding with HEVEA

This binding is quite simple. Just include the videoc.hva style file when processing the TEX[-33-] file. HEVEA[-34-] is tremendously faster than LATEX2HTML.

5.3 Binding with LATEX2HTML

This is more complex since it has to drive the LATEX2HTML[-35-] engine. As shown on figure 2, I write a .prl file to customize this documentation with respect to LATEX2HTML[-35-]. Here follow some hints on how LATEX2HTML[-35-] work:

1. customization and styles are loaded,
2. the entire text is read and pre-pre-processed,
3. verbatim-like parts are removed from the text and hidden somewhere,
4. environments are translated into html (the text is therefore a mixture of TEX[-33-] and HTML),
5. commands are processed,
6. verbatim-like parts are inserted back into the text embedded within <PRE> and </PRE> tags,
7. the entire text is post-post-processed and output.

A lot of LaTeX-*36*- style files are already defined by LaTeX2HTML. These files contain Perl-*37*- functions definitions specifying how they are translated into HTML. A LaTeX-*36*- environment e is defined by a `do_env_`e function (this function receives as single argument the content of the environment) that returns the translated environment. A LaTeX-*36*- command c is defined by a `do_cmd_`c function (this function receives as single argument the rest of the text) that returns the rest of the text (part of which is not yet parsed). There exists some macros to retrieve arguments (optional or mandatory) (see examples in hte `annote.prl` file).

To process the result of the `annote` tool, it is necessary to pre-pre-process the text (there is a function of the `ViClib.pl` library that allows to add new pre-pre- or post-post- processing functions) and leave enough information for the following passes. Some treatments have to be done again on those parts of the text that were hidden in verbatim-like sections.

-*33*- TeX is due to Donald Knuth [Knu84].

-*34*- HEVEA[13] is due to Luc Maranget.

-*35*- LaTeX2HTML[14] is due to Nikos Drakos.

-*36*- LaTeX is due to Leslie Lamport [Lam86].

-*37*- Perl is due to Larry Wall [WCS96].

6 Caveats

It is not recommended that a similar anchor occurs twice in a source line. For instance if you want to annote every occurrence of a word such as "the", chances are that two "the" occurs on the same line and this leads to a problem of generation. Just break your source line to avoid this problem. This problem might be corrected in some future.

Another restriction inherited from HTML is that it is not possible to embed an anchor within an anchor. Therefore, it is not possible to specify a mouse sensitive region including an anchor.

7 Problems

This section should not exist! Unfortunately, there are at least the following problems that are summed up here so I cannot forget about them.

- The text-style field works but resets Communicator for all following annotations to a transparent background color which I was not able to change.

8 Conclusions

The best way to use the `annote` tool is probably to experiment with it once this document is read. The `annote.bk` file and its companion `annote.ann` annotation file are self-describing examples. These files are part of the distribution[15] of the `annote` tool. Of course, to browse the HTML version will also help to see the result.

Happy annotations!

[13] `http://para.inria.fr/~maranget/hevea`
[14] `http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html`
[15] `http://www.infop6.jussieu.fr/cederoms/Videoc2000/clic/tools/VideoC.html`

References

[Knu84] Donald Ervin Knuth. *The TEX Book*. Addison Wesley, 1984.

[Lam86] Leslie Lamport. *A Document Preparation System, LATEX User's Guide & Reference Manual*. Addison Wesley, 1986.

[WCS96] Larry Wall, Tom Christiansen, and Randal L Schwartz. *Programming perl*. O'Reilly & Associates, Inc., second edition, 1996.