Webify Documentation

John Lindsay Orr

February 6, 1999

1 Introduction

Webify is a tool for writing linked sets of webpages. It implements a model of a single main page, with numerous sidelines linked off it (and possibly more sidelines off them). The program provides tools to manage large numbers of web pages with a uniform look at feel, and to include images of text typeset in the TeX computer typesetting language.

Webify was designed to write a mathematics textbook on the web[1] and sucessfuly reduced the task of maintaining and cross-referencing over 400 HTML pages handling 60 Webify files. Webify was designed to include mathematics in webpages, using graphical images of TeX text (see "Webify and TeX" below), but its use for organizing large numbers of webpages is independant of its uses with mathematics. The main features of Webify are:

- Keeps track of numerous sideline documents in a single text document. These can range from short footnotes to lengthy chains of pages.

- Reduces the need for HTML (and TeX) preamble/header information by including a header files for all the documents in a set. This makes it easy to design a unified layout for a set of pages, and to change it for all the pages at once.

- User-defined references speed access to frequently-used text strings, such as HTML image tags to bullets or rulers.

- Webify is text-oriented and written in C, so can be used on a wide variety of platforms.

However, Webify is *not* a tool for the webpage novice. It is designed to speed the task of linking documents for someone who is already confidently writing their own HTML. Webify is an add-on on top of knowledge of HTML (and, possibly TeX), not a way around it. It will not take the place of an HTML editor, if that is what you are using. Moreover, it's main uses are for people who want to handle a suite of many linked pages, or who want to embed TeX text into webpages. It is virtually useless for writing a single page.

2 Writing a Webpage with Webify

2.1 A simple example

Figure 1 shows a simple hypertext document which states one theorem and references one sideline for the proof. This simple document shows many of the important features of Webify: Unless it's enclosed inside a `#tex` command, all text is in HTML (hence the need for `<p>` tags at paragraph breaks). The first segment of embedded TeX is the statement of Theorem 1. The argument of the `#tex` command is a label which can be referenced later in the document, to generate a `<img>` tag for the GIF file coming from the TeX insert. The same TeX insert is referenced in the main document and again in the sideline, by using the `#ref` command.

---

[1] *Analysis WebNotes*: `http://www.math.unl.edu/webnotes-bin/visit`

```
#mainhead(main.wfy)
#sidehead(side.wfy)
#texhead(tex.wfy)
Over two thousand years ago the Greeks knew that there are infinitely many
prime numbers. There are several proofs of this fundamental fact, and the
one we will give here goes back to Euclid:<p>
#tex(thm1)
{\bf Theorem 1.} There are infinitely many prime numbers.
#end
#ref(thm1)<p>
#link(thm1) Read the proof. #end

#sideline(thm1)
   We will prove:<p>
   #ref(thm1)<p>
   #tex
   {\bf Proof.}
   Suppose to the contrary there are only finitely many primes. List them as
   $p_1,\ldots, p_n$. Let $P=p_1p_2\cdots p_n$. Then $P$ must have a prime
   factor, yet clearly none of $p_1,\ldots,p_n$ can divide $P+1$.
   #end
#end
```

Figure 1: A sample Webify document

WARNING: If `#tex` is called with a reference name, as in Theorem1, *no* image tag is placed at the point `#tex` appears. That's why there's a `#ref` right after the statement of Theorem 1.

If there's not going to be any need to use a TeX insert more than once (such as with the proof of Theorem 1) then `#tex` can be called without a reference name. Webify chooses a name for the GIF file, and inserts an HTML image tag at the point that the `#tex` command appeared. That's why there's no `#ref` after the TeX segment that proves Theorem 1.

Note that the *same* name, `thm1`, was used to reference both the sideline and the TeX insert. This is possible because `#ref` and `#link` look for their labels in two different tables of references. Quite often a sideline and a TeX insert will have close enough a conceptual connection that it makes sense to give them the same name, and this allows the user to do this. (References generated by `#tex` are referenced by the `#ref` command, while references generated by `#sideline` are referenced by the `#link` command, so there is no possibility of ambiguity.)

2.2   Header files

So far we haven't discussed the first three lines of the example above. These are used to load header files to give a standardized look to the head and foot of a document, and avod the need to keep writing standard header info for either HTML or TeX documents. Three differerent header files can be loaded, one for the mainline file, one to be used by all the sideline files and one for the TeX file. Figure 2 shows a simple header file which might be used by a series of documents.

If this document is loaded by the `#mainhead(`*filename*`)` command then the text above the `<!--TEXT-->` is inserted before all other HTML text in the main document, and the text after the breakpoint is appended at the end. If the document is loaded with `#sidehead(`*filename*`)`, the text will be used as header and footer for all sideline documents.

There is also a `#texhead` command, which loads header and footer commands for the texfile. Unlike the sideline

```
<html>
<head>
<title>#ref(TITLE)</title>
</head>
<body>
<a href="titlebar.conf"><img src="titlebar.gif" ismap></a>
<p><hr><p>
#sideline(titlebar.conf)
#plain
   default              http:#ref(THIS_PAGE)
   (20,20)  (80,80)     http:#ref(LAST_PAGE)
   (100,20) (160, 80)   http:#ref(NEXT_PAGE)
#end
<!--TEXT-->
<pre>
Page created by A. Hogarth Wands-Orr
Mail to: <a href="mailto:ishie@doghouse.net">Aloysius</a>
Last modidied: #ref(DATE)
</pre>
</body>
</html>
```

Figure 2: A sample header file

and mainline header files, the TeX header file should contain TeX macros, and header and footer material, such as the \documentstyle statements.

This sample header file illustrates two important features:

- There are a number of #ref commands, which will each be replaced by text in the final output. The labels which these refer to can be defined in the main Webify file, whuch lets the user customize aspects of the header and footer of a particular document. For example, to set date of last modification, the main Webify file should contain the line:

  ```
  #label(DATE)    January 27, 1996    #end
  ```

- The header includes a clickable image, which maybe shows buttons to jump to the next or previous page in a linked set. The clickable image needs a .conf file, which should have the URLs of these pages written in to it. This *.conf* file is generated by the sideline command in the header. The actual URLs for the next and last pages (and for the default of staying at the present page) need to be defined by #label commands in the main document.

  NOTE: The #plain command at the start of the sideline prevents any header files being loaded for that particular sideline. It also forces the sideline to be saved in a file with the exact filename given (otherwise an HTML suffix would be added).

3   Webify and TeX

Although the features of Webify will make it useful for preparing sets of webpages of all kinds, the program was specifically designed to allow for the seamless inclusion of TeX text into webpages. The philosophy adopted in Webify is twofold. That:

```
\documentstyle[12pt,amssymb]{article}
\pagestyle{empty}
\newcommand{\gif}[1]{\newpage\par\noindent}
\begin{document}

<!--TEXT-->

\end{document}
```

Figure 3: A TeX header file

- HTML does not support the sort of mathematical notation which is necessary for writing mathematical content on the web, and that the best way to overcome this is to embed in-line images of TeX text in pages.

- The platform-independance of the web means that mixing imaged TeX and regular HTML fonts side by side will always lead to undesirable results on some browsers.

Thus, the style that Webify is designed to support is interleaved paragraphs of HTML with paragraphs of embedded TeX. Doing this will minimize the contrast between disparate font sizes. It is possible to use Webify to embed equations or even single characters of TeX in the middle of a line of HTML, but the user will probably find that Webify's constructions are rather clumsy for that sort of use. However, in the author's optinion emdedding TeX in the middle of HTML produces unsightly shifts in font, and the designer of a page is almost always better working to keep TeX and HTML separate.

Webify uses the #tex command to mark a section of text as TeX. This text is excized from the HTML documents, and replaced by an HTML image tag pointing to a graphics file (by default a .gif) which is presumed to contain a graphics image of the excised TeX text. Webify on its own does *not* perform the conversion from TeX text to an image of the compiled TeX. What Webify does do is to collect all the excised segments of text into a single TeX file (with the same name as the Webify source file, but with a .tex extension). The intention is that this file will then be compiled by TeX (or LaTeX) and that the user will create graphics images of each of the TeX segements. This can be done automatically by means of a program such as tex2gif.

Webify also places a TeX macro (by default \gif{*image*.gif}) before each segment of TeX. It's up to the user to provide a definition of this macro, but a good use is to make it simply break to a new page, so that each segment of TeX is put on its own page. The name of the TeX segment is passed to the macro, under the assumption that the program (or person) cutting out the processed images will need to know the name of the file to save each image in.

The command #texhead loads a TeX header file which should contain a definition of \gif{} together with any TeX or LaTeX header material. Figure 3 shows an example of a minimal TeX header file. Note that the second segment of TeX text was created by a #tex command with no optional argument, so it is given an auomatically-generated name (based on the main file name, which is here assumed to be test.wfy). Figure 4 shows the TeX file that Webify would produce if the Tex header file in Figure 3 were used by the sample file shown in Figure 1.

4   Summary of Commands

#sideline(*optional title*)...#end

This marks the enclosed text as a sideline document. A new HTML document is created and the text is parsed according to Webify's rules. If a title is provided then the sideline document takes that name (with a suffix added), and the title is recorded to be used for reference by #link.

NOTE: A sideline on its own does not generate an HTML anchor in the document the sideline appears in. Use #link to reference sideline documents.

4

```
\documentstyle[12pt,amssymb]{article}
\pagestyle{empty}
\newcommand{\gif}[1]{\newpage\par\noindent}
\begin{document}

\gif{thm1.gif}{\bf Theorem 1.} There are infinitely many prime numbers.

\gif{test_001.gif}
{\bf Proof.}
Suppose to the contrary there are only finitely
many primes. List them as $p_1,\ldots, p_n$. Let
$P=p_1p_2\cdots p_n$. Then $P$ must have a prime
factor, yet clearly none of $p_1,\ldots,p_n$ can
divide $P+1$. Contradiction.

\end{document}
```

Figure 4: Texfile ouput

NOTE: If #sideline is called with no title, then a title is automatically generated for it. This sideline is referenced by the first subsequent #link which appears with argument, after which the reference to it is lost. However, an HTML document is always generated, even if it is never referenced.

#link(*reference*)...#end

The reference should correspond to the title of a sideline document. This routine creates an HTML anchor linking to the sideline document referred to, and encloses the text between #link() and #end within the anchor.

NOTE: The sideline that a #link refers to may appear in the Webify document before that link. The same does not apply to #ref commands.

#tex(*optional title*)...#end

The enclosed text is added to the TeX document. The text is prefaced by an user-defined TeX macro (the default is \gif{name.gif}) which references the title of the text section. Note that the title has a suffix added to it. The default is .gif, but can be changed by setting an environment variable.

The macro \gif{name.gif} is defined in the standard TeX header file provided with Webify. It causes each segment of text to be printed on a separate page, in preparation for use with the GIF extraction program. The extraction program, tex2gif, also recognizes \gif{name.gif} to create a GIF with the same name.

NOTE: If #tex is called with no title, a unique title based on the title of the mainline document is generated for it.

IMAGE TAGS: If #tex is called with no title then an HTML image tag referring to a GIF file with the same name as was generated for the TeX segment is placed in the current HTML document, at the point at which the #tex command was found. If #tex is called with a title then an image tag is created for the TeX segment, but it is not written into the document. Instead it is placed on the list that #ref refers to, and may be inserted many times in different documents,by #ref commands.

#label(*tag*)...#end

The #label command records a label for future use with the #ref command. The text between the #label command and its corresponding #end is recorded verbatim, with no processing of Webify commands that appear in it. When the text saved by a #label command is inserted into a document with a #ref command, the text is parsed by Webify at that point, and all Webify commands (including other #ref's) are expanded.

NOTE: #label may be called more than once with the same argument. Subsequent calls will replace the current text associated with that argument with the new text.

NOTE: There are a number of reserved words that have a special meaning to webify when a label is set for them. These are discussed under the heading Environment Variables.

#ref(*reference*)

A call to #ref causes the #ref tag to be replaced with the text enclosed inside the correponding #label command. The text is then parsed by Webify for any commands it may contain.

The #label/#ref commands are extremely versatile, and allow a lot of short-cuts in document-preparation:

- A list of #label commands can be included in a header file, providing 'macros' to customize a style of document. This could include frequently-used image-tags such as GIFs for bullets or separator lines.

- Conversely, if #ref commands are included in a header file, they can serve as 'variables' in the header, which can be set by including the correponding #label commands in the main document. For example, to write a footer to a document which gives the date of last modification, include a #ref(DATE) command in the footer section of the header file, and include #label(DATE)...#end in the main document.

NOTE: A #ref command can only appear after the #label command that it referes to. This means in the previous example, the #label(DATE)...#end in the main document must come before the #mainhead command.

NOTE: The header document for the TeX file is searched for #label and #ref commands, but the TeX inclusions in the main document are not searched. TeX has its own macro system which far supasses this one, so there shouldn't be any significant loss in this restriction.

NOTE: The labels from #label and #link are recorded on different lists, so the same tag can be used for both commands without ambiguity. This is useful, since conceptually one often want to use a tag like thm1 for both the TeX insert which contains its text, and for the sideline where the theorem is proved.

#mainhead(*file path*)
#sidehead(*file path*)
#texhead(*file path*)

Loads a file to be used as the header file for, respectively, the mainline, sideline, and TeX files. More than one occurrence of the same header command is allowed. If that happens, all the #label commands in each of the files will be recorded, all the sidelines will be generated, and their labels recorded. However, only the text from the most recent header command of a particular type will be used to generate the basic header text.

In fact, a header file contains the text that will be used both for the footer and the header of a document. The two parts are separated by a breakmark, which can be set in the environment veraible BREAKMARK. The default breakmark is <!--TEXT-->.

hash

Produces a '#' mark.

#plain

When `#plain` is present in an HTML document, it suppresses the inclusion of *any* additions to the text in the file. This means that the prevalent header for the file is ommitted (not very useful for the main document, but important for a sideline, since all sideline files otherwise load the same header file). Moreover, the document is saved in a file with exactly the name it is given in the document declaration (i.e. no suffix is added).

NOTE: The main use envisaged for this command is to generate a `.conf` file as a sideline.

`#comment...#end`

The text between the commands is ignored by the Webify program.

NOTE: You may also include an optional comment after the `#end` of any command, using the format `#end(arbitrary comment here)`. This enables the user to remind themselves what that `#end` matches, for instance:

`#sideline(theorem3)`

`...`

`#end(sideline theorem3)`

The only restriction is that this comment may not span more than one line, and any parenthesis included must balance properly.

5   Environment Variables

The following variables customize various aspects of webify's behavior. The program will read their values from the UNIX environment, and they can be set (or reset) with the `#label` command from inside a document. Moreover, the variables; `HEADER_PATH`, `PAGES_PATH`, and `PAGES_URL`, can be set on the command line with the syntax;

```
webify -d<var_name>=<value> <file_name>
```

`PAGES_URL`

Specifies the URL prefix that will be included in all references to sideline documents.

`IMG_URL`

Specifies the URL prefix that will be included in all references to gif images.

`PAGES_PATH`

Specifies the location in the file structure in which all Webify's output files will be written. If none is specified, the default is the same directory as the input file. If the directory does not exist, it will be created.

NOTE: This directory will be created, no matter how dumb of a thing it is to do. This means that if you set `PAGES_PATH` to be an absolute path, but forget the initial slash, that complete path will be created underneath the current directory. Or, if you put odd characters in the `PAGES_PATH`, the directory will be created with those odd characters in its name.

`BREAKMARK`

Specifies the tag that marks the middle a header file. The default is `<!--TEXT-->`.

TEXMACRO

Specifies the name of the macro for the TEX segments in the document. The default is `gif`.

DEBUG

Turns debugging mode on. This forces all output to `stdout` as opposed to writing to files.

IMGSUFF

Specifies the suffix that will be appended to references to images created from TEX sections. The default is `.gif`.

HTMLSUFF

Specifies the suffix that will be appended to the main HTML and sideline HTML files. The default is `.htm`.

HEADER_PATH

Specifies the path header files are included in. When loading header files, the current directory will first be searched, and then the path set in `HEADER_PATH`.

There are also two special variables, which cannot be set but are always maintained by the program. These are:

THIS_URL

Contains the full URL of the current file.

THIS_PATH

Contains the full path to the current file.