Article revision 2010/08/08

Five Perl utility routines

David Walden

Abstract I had written a few relatively insignificant Perl programs before 2004. My first serious Perl program, developed in late 2004 and early 2005, involved lots of output of HTML. I needed some utility routines, particularly to allow me to use output templates. I was too lazy to find what I needed in the CPAN library, so I wrote some crude utility routines. I have since used these routines in several additional significant projects.

In this note I describe or mention five Perl utility routines that I have found to be useful. The first two routines are relatively significant and provide a rudimentary template capability. The last three routines are relatively insignificant.

1 Copytemplate

When I first started doing Perl programming,[1] I learned about Perl's multi-line quote (that is, literal) capability. For instance,

```
   print(OUT <<LITERAL);
<li><a href="final-count.txt">Total number of "reviews"</a>
</ul>
<hr>
<h3 id="TAGrecent">Most recent</h3>
   LITERAL
```

printed (to the output file referenced by the file handle `OUT`) everything from `<<LITERAL` to the matching `LITERAL`, discarding the stuff on the first line after `<<LITERAL`.

As I later began to develop a more complex program,[2] I learned about storing the multi-line literals in variables, for example,

```
   $outputstring = <<LITERAL);
<li><a href="final-count.txt">Total number of "reviews"</a>
</ul>
<hr>
```

1. To generate my movie website: http://walden-family.com/public/movie-index.htm
2. The website generator for *The PracTEX Journal* website:http://tug.org/pracjourn

```
<h3 id="TAGrecent">Most recent</h3>
   LITERAL
   print(OUT,$outputstring);
```

In this way I could reuse a multi-line literal without having to insert it in my program multiple times.

I soon wanted the ability to insert different values for parameters into such multi-line literal strings each time I output one. For instance, in the program that generates the TUG Interview Corner,[3] I need to output some HTML for a link to the interview, the name of the interviewee, and the date the interview was completed for each interviewee in a chronological list of interviewees. Thus, I defined a template as follows,[4]

```
##1## file name
##2## interviewee names
##3## date
$CHRONENTRY = <<'HTMLTEMPLATE';
<li><a href="##1##">##2##</a> (interview completed ##3##)
HTMLTEMPLATE
```

where the parameters to be filled in when the template is used are defined by strings of the form `##2##` (for the second parameter in this example).

I also defined a subroutine named `Copytemplate` which is called with an output file handle, the name of a variable containing a template, and the three (in this case) parameters being fed to the template, for example,

```
Copytemplate(OUT,$CHRONENTRY,"walden.html","Dave Walden","2006-09-20");
```

The routine `Copytemplate` is shown in the file http://walden-family.com/public/texland/perl-common-code.txt[5] Take a look at it. Using an unlikely-to-be-otherwise-used character string (i.e., `##i##`) to indicate parameters in the template was easier than attempting something more sophisticated such as named variables. Also, having a maximum number of parameters be explicit in the subroutine was the easiest approach.[6]

In fact, when I call this subroutine for the `$CHRONENTRY` template in the Interview Corner website generation program, I call it in a loop with three variables holding the information for each interviewee in turn:

3. http://tug.org/interviews
4. The first three lines are Perl comments lines to remind me of the order of parameters in uses of the template. Conveniently my notation for template parameters overlaps with Perl's notation for comments.
5. All five of the utility routines I will be talking about in this note are in that file.
6. As I said, in the abstract, I was too lazy to look up a capability for templates in CPAN; in fact, I had never used any existing template capability for any programming language. Thus, my way of thinking about templates was more or less like thinking about a simple macro processing system except that, instead of invoking defined macros by name, I call a macro evaluator with the name of the macro being invoked and the macro's call-time parameters.

```
Copytemplate(OUT,$CHRONENTRY,$record[1],$record[3],$record[2]);
```

You can see the HTML that results from using this and other templates at http://tug.org/interviews/ordered-list.html

A number of additional templates (but not all) for the Interviews website are show in the file http://walden-family.com/public/texland/perl-htmltemplates.txt

2 Rptovertemplate

The next routine at http://walden-family.com/public/texland/perl-common-code.txt is Rptovertemplate which stands for "repeat over template." A use of Rptovertemplate might look as follows:

```
my $newtemplate=Rptovertemplate($LEFTCOLUMNSTUFF,'(.*)\[1\[(.*)\]1\](.*)',\@staff);
Copytemplate(OUT,$newtemplate,$WIDTH,$firstbio,$firstname,$firsttitle);
```

Unlike Copytemplate which does parameter substitutions as it does output, the routine Rptovertemplate does parametet substitutions as it copies a template into another variable (as in the first line of code above); from there it can be output with Copytemplate (as in the second line of code above). The template $LEFTCOLUMNSTUFF used in the two-line example above is shown in Appendix A, and you can see the result of the two-line example in the left column at http://tug.org/pracjourn

Look in parallel at the $LEFTCOLUMNSTUFF template, the resulting web page, and the two lines of Perl above that assign a value to $newtemplate and then call Copytemplate with $newtemplate as one of the parameters. In particular, for instances of strings of the form ##2##, !!3!!, etc., in the $LEFTCOLUMNSTUFF template.

The line in the $LEFTCOLUMNSTUFF template

```
<td valign="top" width="##1##%" rowspan="2">
```

says to fill in the value of $WIDTH, a column width percentage, into the HTML as Copytemplate outputs the template as modified by Rptovertemplate.

Now look at the following lines in the $LEFTCOLUMNSTUFF template:

```
<b><a href="people/##2##">##3##</a>##4##<br></b>
[1[<a href="people/!!1!!">!!2!!</a>!!3!!<br>]1]
```

These turns into the following HTML:

```
<b><a href="people/lcarnes.html">Lance Carnes</a>, editor<br></b>
<a href="people/kchristiansen.html">Kaja Christiansen</a><br>
<a href="people/pflom.html">Peter Flom</a><br>
```

```
<a href="people/hhagen.html">Hans Hagen</a><br>
<a href="people/rlaakso.html">Robin Laakso</a><br>
<a href="people/tmiller.html">Tristan Miller</a><br>
<a href="people/tnull.html">Tim Null</a><br>
<a href="people/aogawa.html">Arthur Ogawa</a><br>
<a href="people/speter.html">Steve Peter</a><br>
<a href="people/yrobbers.html">Yuri Robbers</a><br>
<a href="people/wrobertson.html">Will Robertson</a><br>
```

This is done as follows. First the website generation program reads in a file containing the raw information on the editorial board. The contents of this file are:

```
#This is the staff list file for PracTeX

Lance Carnes|lcarnes.html|, editor|y
Kaja Christiansen|kchristiansen.html||n
Peter Flom|pflom.html||y
Hans Hagen|hhagen.html||n
Robin Laakso|rlaakso.html||y
Tristan Miller|tmiller.html||y
Tim Null|tnull.html||y
Arthur Ogawa|aogawa.html||y
Steve Peter|speter.html||y
Yuri Robbers|yrobbers.html||y
Will Robertson|wrobertson.html||y

#order the list in the order you want the names displayed
#vertical bars separate fields
#field 0 = name; field 1 = filename of bio in people directory; field 2 = title;
#field 3 = y or n for having written for journal
#separate multiple authors with "and"
```

(Ignore for for the purposes of this note the fourth, "y"or "no", field in the above data.) The website generation program parses that content into a list named `@staff` where each element in the list has three subelements, one each for the person's name, the name of the HTML file about the person in the `people` directory, and the person's title if any (although the file name and person name are reversed in order as they go into the `@staff` list). The data for the first person in the `@staff` list is removed and substituted in the HTML being output via the last three parameters of the `Copytemplate` call:

```
Copytemplate(OUT,$newtemplate,$WIDTH,$firstbio,$firstname,$firsttitle);
```

This first person is handled separately because the HTML for this person includes making the entry be in bold type.

The rest of the staff list (in `@staff`) is passed into the routine `Rptovertemplate` as follows:

```
my $newtemplate=Rptovertemplate($LEFTCOLUMNSTUFF,'(.*)\[1\[(.*)\]1\](.*)',\@staff);
```

The three parameters in the call to `Rptovertemplate` are a template to be modified (in this case the template `$LEFTCOLUMNSTUFF`), a pattern to be searched for in the template (in this case `(.*)\[1\[(.*)\]1\](.*)` using Perl's string search conventions), and a list to be repeated over (in this case in `@staff` using Perl's reference mechanism). `Rptovertemplate` copies the template `$LEFTCOLUMNSTUFF`) into `$newtemplate` until it finds the pattern specified in the second parameter. Use of the `(.*)\[1\[(.*)\]1\](.*)` pattern detects the following line in the `$LEFTCOLUMNSTUFF` template:

```
[1[<a href="people/!!1!!">!!2!!</a>!!3!!<br>]1]
```

`Rptovertemplate` then copies that line repeatedly to the end of the template being created in `$newtemplate`, replacing the parameter placeholders `!!1!!`, etc., by successive three-subelement items from the list. As with the `##1##` notation for `Copytemplate`, the `!!1!!` notation is used with `Rptovertemplate` because that sort of string is rare.[7] Once all of the elements of the list have been processed, `Rptovertemplate` copies the rest of the `$LEFTCOLUMNSTUFF` to `$newtemplate`. The code line

```
Copytemplate(OUT,$newtemplate,$WIDTH,$firstbio,$firstname,$firsttitle);
```

finally outputs `$newtemplate` (which now includes all but the first of the list of staff members), and during the output parameters `##1##` to `##4##` are filled in by the width and the three subelements of the first person on the staff list.

In the above example, the string to be repeated over was indicated by `[1[...]1]`. If there are multiple different strings to be repeated over, the following approach works, using successively `[1[...]1]`, `[2[...]2]`, and `[3[...]3]` to indicate the various strings to be substituted for:

```
my $newtemplate = Rptovertemplate($TOCTEMPLATE,'(.*)\[1\[(.*)\]1\](.*)',\@alist);
my $oldtemplate = $newtemplate;
$newtemplate = Rptovertemplate($oldtemplate,'(.*)\[2\[(.*)\]2\](.*)',\@blist);
$oldtemplate = $newtemplate;
$newtemplate = Rptovertemplate($oldtemplate,'(.*)\[3\[(.*)\]3\](.*)',\@clist);
Copytemplate($outfile,$newtemplate,$issueno,$issueyr,$toplefttitle,$width,$pubdate);
```

7. The subroutine could also be changed so the repeat pattern to search for does not have to be passed in as a parameter, but including the pattern was easier to debug and solves the problem of not having an escape mechanism for sequences that might be used for real in the template being copied.

First $newtemplate is created substituting the elements of @alist into the string indicated by [1[...]1] in $TOCTEMPLATE (the $TOCTEMPLATE template is shown in Appendix B with results shown at http://tug.org/pracjourn). Then that result is processed doing parameter substitutions with the elements of @blist into the string [2[...]2] in the original template (now part of the new template). Then it is done again doing substitutions into the string [3[...]3] with the elements of @clist. Finally, the resulting template is output with substitutions being made for instances of ##1## through ##5## that occurred originally in $TOCTEMPLATE and were successively passed along as the final version of $newtemplate was created.

The three lists were just my way of treating separately the Notices, Articles, and Columns which are separate blocks of HTML in a *PracTEX Journal* issue. I've considered changing Rptovertemplate so I can pass multiple lists to it, but so far I have not thought it worth the bother to try to think through the Perl code for doing that.

My two template routines are very ad hoc and limited in what they can do, but they have been sufficient for my needs as I have written several quite substantial Perl programs.

3 Three additional routines

Also in the file http://walden-family.com/public/texland/perl-common-code.txt is the routine Todaysdate. It merely looks up the current day, month, and year in the operating system and prints them in a sensible format. It takes an optional parameter which specifies day-month-year or month-day-year format. Typically I call this routine as a parameter in a call to Copytemplate:

```
Copytemplate(OUT,$FOOTERSTUFF,Todaysdate());
```

Another routines in the common-code file is Preprocessfilelines. It reads an entire file, strips out comments and blank lines, handles continuation lines using the well known backslash convention, and returns all the interesting lines as a list. For instance, if IN is an input file handle, then Preprocessfilelines might be called as follows:

```
my @lines = Preprocessfilelines(IN);
```

The final routine in the common-code file is Andnamestocommanames. This routine takes a string list of names in the BibTEX author list format, e.g., "Dave Walden and Karl Berry and Duane Bibby", and turns all but last "and" into a commas, e.g., "Dave Walden, Karl Berry, and Duane Bibby". (The routine does not correctly handle complex cases.)

Acknowledgments

Karl Berry answered many questions about Perl as I wrote that first significant program in late 2004 and early 2005. No doubt he also gave me advice about good Perl programming style that I ignored, preferring to pretend I was still programming in Fortran in 1964.

Appendix 1

```
##1## is $WIDTH as a percent
##2## is filename of bio of first staff member
##3## is person's name of first staff member
##4## is title field of first staff member
# ^^1^^ is file name of bio for rest of staff members
# ^^2^^ is person's name for rest of staff members
# ^^3^^ is title field (more than just title) rest of staff members
$LEFTCOLUMNSTUFF = <<'HTMLTEMPLATE';

<table bgcolor="#ffffcc" width="100%" cellpadding=5 border=0>
<tr>
<td valign="top" width="##1##%" rowspan="2">

<p><b>The online journal of the TeX Users Group<br>ISSN 1556-6994</b><p>

  <hr noshade size=5>
  <b>About <i>The PracTeX Journal</i></b>
  <small>
<p><a href="info.html">General information</a>
<br><a href="submit.html">Submit an item</a>
<br><a href="stylefiles.html">Download style files</a>
<br><a href="copyright.html">Copyright</a>
<br><a href="mailto:pracjourn@tug.org">Contact us</a>
<br><a href="http://rss.softwaregarden.com/aboutrss.html">About RSS feeds</a>
<a href="rss/pracjourn_rss.xml"> <img src="rss/pracjourn_xml_lg.gif"></a>
<a href="rss/pracjourn_rss.html"> <img src="rss/pracjourn_html_lg.gif"></a>
</small>

  <hr noshade size=5>
  <b>Archives of <i>The PracTeX Journal</i></b>
```

```
  <small>
<p><a href="archive.html">Back issues</a>
<br><a href="authorindex/index.html">Author index</a>
<br><a href="titleindex/index.html">Title index</a>
<br><a href="pracjourn.bib">BibTeX bibliography</a>
</small>

<hr noshade size=5>
<p><b>Next issue</b><small><br>
Fall 2010
<br></small>

<hr noshade size=5>
<p><b>Editorial board</b>
<small>

<p>
<b><a href="people/##2##">##3##</a>###4###<br></b>
[1[<a href="people/!!1!!">!!2!!</a>!!3!!<br>]1]

<p><a href="others.html"><b>Other key people</b></a>

<p><a href="helpwanted.html"><b>More key people wanted</b></a>

</small>
</td>

HTMLTEMPLATE
```

Appendix 2

```
##1## is issue number
##2## is issue year
##3## is what is printed at top:  see $ARCHIVEISSUETOPLEFT or $CURRENTISSUETOPLEFT
##4### TOC width
##5## is publication date
#for [1[, [2[, and [3[:
# !!1!! is paperfile
# !!2!! is papertitle
```

```
# !!3!! is author
$TOCTEMPLATE = <<'HTMLTEMPLATE';
<tr><td valign="top" bgcolor=white cellpadding=32 colspan="2">
<table width="##4##" border=0>
<tr>
<td align=left><h3>##3##</h3></td>
<td align=right><h3>##2##, Number ##1##</h3></td>
</tr>
<tr>
<td align=left><h3> </h3></td>
<td align=right>[Published ##5##]</td>
</tr>
</table>
<dl>

<dt class="smallskip"><dd><b>Notices</b>

[1[<dt class="smallskip">
<dd><a href="!!3!!">!!1!!</a>
<br>  <font size=-1>!!2!!</font>
]1]

<dt class="smallskip"><dd><b>Articles</b>

[2[<dt class="smallskip">
<dd><a href="!!3!!">!!1!!</a>
<br>  <font size=-1>!!2!!</font>
]2]

<dt class="smallskip"><dd><b>Columns</b>

[3[<dt class="smallskip">
<dd><a href="!!3!!">!!1!!</a>
<br>  <font size=-1>!!2!!</font>
]3]

</dl>  </td></tr></table>
HTMLTEMPLATE
```