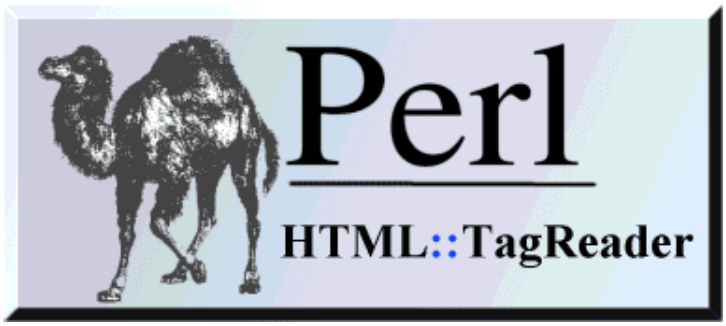
LinuxFocus article number 269
http://linuxfocus.org

Managing HTML with Perl, HTML::TagReader

by Guido Socher (homepage)

About the author:

Guido likes Perl because it is a very flexible and fast scripting language. He likes the motto of Perl "There's more than one way to do it" which reflects the freedom and possibilities you have when you go with opensource.



Abstract:

If you want to manage a website with more than 10 HTML pages then you will soon find out that you need some programs to support you. Most traditional software reads files line by line (or character by character). Unfortunately lines have no meaning in SGML/XML/HTML files. SGML/XML/HTML files are based on Tags. HTML::TagReader is a light weight module to process a file by Tag.

This article assumes that you know Perl quite well. Have a look at my Perl tutorials (January 2000) if you want to learn Perl.

Introduction

Traditionally files have been line based. Examples are Unix configuration files such as /etc/hosts, /etc/passwd .... There are even older operating systems where you have functions in the operating system to retrieve and write data line by line.
SGML/XML/HTML files are based on Tags, lines have no meaning here, however text editors and humans are somehow still line based.

Especially larger HTML files will usually consist of several lines of HTML code. There are even tools such as "Tidy" to indent html and make it readable. We use lines although HTML is based on Tags not lines. You can compare it to C-code. Theoretically you could write the entire code on a single line. Nobody does that. It would be unreadable.
Therefore you expect a HTML syntax checker to write "ERROR: line ..." rather than "ERROR after tag 4123". This is because your text editor allows you to jump easily to a given line in the file.

What is needed here is a good and light weight way to **process a HTML file Tag by Tag and still keep track of the line numbers**.

A possible solution

The usual way to read a file in Perl is to use the *while(<FILEHANDLE>)* operator. This will read data line by line and pass each line to the $_ variable. Why does Perl do this? Perl has an internal variable called INPUT_RECORD_SEPARATOR ($RS or $/) where it is defined that "\n" is the end of a line. If you set $/=">" then Perl will use the ">" as "end of line". The following command line Perl script will reformat html text to always end at ">":

perl -ne ’sub BEGIN{$/=">";} s/\s+/ /g; print "$_\n";’ file.html

A html file that looks like

```
<html><p>some text here</p></html>
```

will become

```
<html>
<p>
some text here</p>
</html>
```

The important issue is however not readability. For the software developer it is important that the data is passed to the functions in her/his code Tag by Tag. With this it will be easy to search for a "<a href= ..." even if the original html had "a" and "href" on separate lines.

Changing the "$/" (INPUT_RECORD_SEPARATOR) causes no processing overhead and is very fast. It is also possible to use the match operator and regular expressions as an iterator and process the file with regular expressions. This is a bit more complicated and slower but also very often used.

Where is the problem?? The title of this article said HTML::TagReader but now I have been talking all the time about a much simpler solution that does not require extra modules. There must be something wrong with this solution:

- Almost all HTML files in the world are faulty. There are millions of pages that contain e.g C code examples that looks on HTML code level like
  `if ( limit > 3) ....`
  instead of
  `if ( limit > 3) ....`
  In HTML "<" should start a tag and ">" should end it. None of them should appear on their own somewhere in the text. Most browsers will display both correctly and hide the error.

- Changing the "$/" effects the entire program. If you want to process another file line by line while you are reading the html file then you have a problem.

In other words it is only in special cases possible to use the "$/" (INPUT_RECORD_SEPARATOR).

Still I have a useful example program for you that uses what we discussed so far. It sets however "$/" to "<" because the web browsers can not handle a misplaced "<" as good as a ">". Therefore there are less web-pages with misplaced "<" than with misplaced ">". The program is called tr_tagcontentgrep (click to view) and you can also see in the code how to keep track of the line number. tr_tagcontentgrep can be used to "grep" for a string (e.g "img") in a Tag even if the Tag goes over several lines. Something like:

tr_tagcontentgrep -l img file.html
index.html:53: <IMG src="../images/transpix.gif" alt="">
index.html:257: <IMG SRC="../Logo.gif" width=128 height=53>

HTML::TagReader

HTML::TagReader solves the two problems with the modification of the INPUT_RECORD_SEPARATOR and offers also a much nicer way to separate text from tags. It is not as heavy as a full fledged HTML::Parser and offers what you want when processing html code: A method to read Tag by Tag.

Enough words. Here is how to use it. First you must write
use HTML::TagReader;
in your code to load the module. Then you call
my $p=new HTML::TagReader "filename";
to open the file "filename" and get an object reference returned in $p. Now you can call $p->gettag(0) or $p->getbytoken(0) to get the next Tag. gettag returns only Tags (The stuff between < and >) while getbytoken give you also the text between the tags and tells you what it is (Tag or text). With these functions it is very easy to process html files. Essential to maintain a larger website. A full syntax description can be found in the man page of HTML::TagReader.

Here is now a real example program. It prints the document titles for a number of documents:

```
#!/usr/bin/perl -w
use strict;
use HTML::TagReader;
#
die "USAGE: htmltitle file.html [file2.html...]\n" unless($ARGV[0]);
my $printnow=0;
my ($tagOrText,$tagtype,$linenumber,$column);
#
for my $file (@ARGV){
  my $p=new HTML::TagReader "$file";
  # read the file with getbytoken:
  while(($tagOrText,$tagtype,$linenumber,$column) = $p->getbytoken(0)){
  if ($tagtype eq "title"){
    $printnow=1;
    print "${file}:${linenumber}:${column}: ";
    next;
  }
  next unless($printnow);
  if ($tagtype eq "/title" || $tagtype eq "/head" ){
    $printnow=0;
    print "\n";
```

```
    next;
  }
  $tagOrText=~s/\s+/ /; #kill newline, double space and tabs
  print $tagOrText;
  }
}
# vim: set sw=4 ts=4 si et:
```

How does it work? We read the html file with $p->getbytoken(0) when we find <title> or <Title> or <TITLE> (they are returned as $tagtype eq "title") then we set a flag ($printnow) to start printing and when we find </title> we stop printing.
You use the program like this:

htmltitle file.html somedir/index.html
file.html:4: the cool perl page
somedir/index.html:9: joe's homepage

Of course it is possible to implement the tr_tagcontentgrep from above with HTML::TagReader. A bit shorter and easier to write:

```
#!/usr/bin/perl -w
use HTML::TagReader;
die "USAGE: taggrep.pl searchexpr file.html\n" unless ($ARGV[1]);
my $expression = shift;
my @tag;
for my $file (@ARGV){
  my $p=new HTML::TagReader "$file";
  while(@tag = $p->gettag(0)){
    # $tag[0] is the tag (e.g <a href=...>)
    # $tag[1]=linenumber $tag[2]=column
    if ($tag[0]=~/$expression/io){
      print "$file:$tag[1]:$tag[2]: $tag[0]\n";
    }
  }
}
```

The script is short and does not have much error handling but otherwise it is fully functional. To grep for tags that contain the string "gif" you type:

taggrep.pl gif file.html
file.html:135:15: <img src="images/2doc.gif" width=34 height=22>
file.html:140:1: <img src="images/tst.gif" height="164" width="173">

One more example? Here is a program that will strip all the <font...> and tags from html code. These font tags are sometimes used in massive amounts by some poorly designed graphical html editors and cause lots of problems when viewing the pages on different browsers and with different screen sizes. This simple version strips all font Tags. You can change it to remove only those that set fontface or size and leave color unchanged.

```
#!/usr/bin/perl -w
use strict;
use HTML::TagReader;
# strip all font tags from html code but leave the rest of the
# code un-changed.
```

```
die "USAGE: delfont file.html > newfile.html\n" unless ($ARGV[0]);
my $file = $ARGV[0];
my ($tagOrText,$tagtype,$linenumber,$column);
#
my $p=new HTML::TagReader "$file";
# read the file with getbytoken:
while(($tagOrText,$tagtype,$linenumber,$column) = $p->getbytoken(0)){
  if ($tagtype eq "font" || $tagtype eq "/font"){
    print STDERR "${file}:${linenumber}:${column}: deleting $tagtype\n";
    next;
  }
  print $tagOrText;
}
# vim: set sw=4 ts=4 si et:
```

As you can see it is very easy to write useful programs with just a few lines.
The source code package of HTML::TagReader (see references) already contains some applications of HTML::TagReader:

- tr_blck -- check for broken relative links in HTML pages
- tr_llnk -- list links in HTML files
- tr_xlnk -- expand links on directories into link on index files
- tr_mvlnk -- modify tags in HTML files with perl commands.
- tr_staticssi -- expand SSI directives #include virtual and #exec cmd and produce a static html page.
- tr_imgaddsize -- add width=... and height=... to <img src=...>

tr_xlnk and tr_staticssi are very useful when you want to make a CDrom from a website. The web server will e.g give you http://www.linuxfocus.org/index.html even if you typed only http://www.linuxfocus.org/ (without the index.html). If you however just burn all the files and directories on a CD and access the CD with your web browser directly (file:/mnt/cdrom) then you will see a directory listing instead of index.html and this happens not only once but everytime you klick onto a link that points to a directory. The company that made the first LinuxFocus CD made this mistake and it was terrible to use the CD. Now that they get the data via tr_xlnk the CDs are working.

I am sure you will find HTML::TagReader useful. Happy programming!

References

- The man page of HTML::TagReader
- Perl tutorial: Perl III (January 2000)
- The tr_tagcontentgrep program (the one not using HTML::TagReader): tr_tagcontentgrep (txt) or tr_tagcontentgrep (html)
- The source code of HTML:TagReader:
  http://cpan.org/authors/id/G/GU/GUS/
  or
  http://main.linuxfocus.org/~guido/
- Tidy is essential if you do web design: tidy, a utility to syntax check html
  How to use tidy? Easy:
  `tidy -e file.html`
  will print html errors
  `tidy -im -raw file.html`

will edit the file and indent it nicely. It will also correct faults (as far as tidy can guess what was meant).

2005-01-14, generated by lfparser_pdf version 2.51