1.10. Notes on Perl and Pod

The Perl programming language[1] is quite a difficult one for the prg2lout program to deal with, and our boast that programs can be included with 'absolutely no modifications' is not quite true for Perl.

Here is the complete list of problem areas. In most cases the effect is to get the formatting wrong over a short region, which is not perhaps so disastrous; and it should be easy to modify your Perl program without changing its meaning, to work around these problems. After all, in Perl there is always more than one way to do it.

1. *Here-documents* such as

   ```
   <<"EOF"
   These lines will be read as though
   enclosed in double quotes
   EOF
   ```

   will be handled correctly only if the string used to terminate the document is one of *EOF*, *EOT*, *END*, and the empty string. This terminating string may be enclosed in all three kinds of quotes, or in none. Anything after the << symbol on the same line will be treated as a string, which means that stacked here-documents will not be printed properly.

2. When prg2lout is scanning the program looking for the beginning of a lexical unit, it may come upon a / character, and this *initial* / (not subsequent ones in the same lexical unit) it finds difficult to interpret, since it may be the beginning of a regular expression, to be formatted like a string, or it may be a complete lexical unit denoting division. The program chooses the regular expression (or equivalently, string) interpretation if the / character is immediately preceded by q, qq, qx, qw, qr, m, s, y, or tr. It also chooses the regular expression interpretation if the / character is immediately preceded by zero, one, or two space or tab characters, which are themselves immediately preceded by a complete lexical unit which is either =, =~, !~, split, or if. Otherwise it chooses the division interpretation. In the rare cases where this rule fails, you can force prg2lout to choose the regular expression interpretation by placing an m in front of the initial / (this does not change the meaning of the program), and you can force the division interpretation by placing at least three spaces before the / character.

3. The prg2lout program recognises complete substitution operators, such as s/abc/ABC/ and s{abc}{ABC}, as individual lexical units, and it formats them as it does strings. However, its rule for deciding where these units end is rather simple-minded: at the start, it works out what character finishes the unit (in the examples above, / and } respectively), and scans along until it reaches the second occurrence of this character not preceded by the \ escape character. This is inadequate in two respects. Firstly,

   ```
   s{{}}{}
   ```

   is a complete legal substitution expression in Perl, but prg2lout will think it ends after the second }, and so format the rest of it wrongly. Secondly, and more seriously,

---

[1] My thanks to Mark Summerfield for help with Perl and Pod.

```
s{abc}<ABC>
```

is also allowed in Perl, but prg2lout does not understand that the finishing character can change in this way, so in this example it will keep scanning forever, looking for the second } character, which is disastrous.

Further work may eliminate some of these problems.

The Pod language is used by Perl programmers for creating documentation, and may be found within Perl programs or standing alone. Lout supports both arrangements without any special action by the user. At the beginning of the perl setup line, the following line has been placed:

```
@SysInclude { pod }
```

Thus, asking for Perl always gives you Pod as well. If you are using your own setup files for both languages, it is probably better to break this connection by deleting this line from your copy of the perl setup file and placing

```
@Include { mypod }
@Include { myperl }
```

at the start of your document in the usual way.

Because Pod is a documentation language rather than a programming language, the setup file options listed in Section 1.4 are mostly of no relevance. So for Pod only these have been discarded and replaced by a completely different set of options, controlling such things as the size of headings and the gaps between list items, which you can find documented in the pod setup file.

The only known problem with Pod formatting is that the E<> constructs enclosing a character number or a 'non-numeric HTML entity' (to quote the author of Pod) will appear verbatim rather than being replaced by the characters they represent.