fref, reference formatter for *troff*

John Ankarström

Introduction

fref ("format reference") is a *troff* preprocessor that formats reference lists embedded in *troff* source code. It supports multiple languages and is macro package-independent.

fref has been developed to address concerns unsatisfactorily met by previous solutions. It differs from the much beloved *refer* in the following ways:

1. *fref* is a reference *formatter*, not a reference *resolver* like *refer*. It formats references specified in the *troff* source, not in an external database.
2. As a result of the first point, *fref* requires no macro package support, unlike *refer*, which relies on macro packages to do the formatting.
3. As a further result of the first point, *fref* is a much simpler program than *refer* and is thus easier for the user to extend according to his own preferences.

Because *fref* does not resolve references, it is not a good fit for reference styles that require elaborate or numbered inline citations.

Instead, *fref* is primarily designed for reference styles with simple inline citations, such as the Harvard system, that the user can type himself without much trouble. Indeed, for parenthetical citations, it is likely quicker to manually type the citation than to instruct the computer to generate it. In such cases, I hope you will find *fref* to be a very fitting solution.

If you do use the Harvard system, *fref* offers many benefits over *refer*. While the format of *refer* references depend on the macro package one uses, the output of *fref* is the same regardless of which macros are used. Furthermore, *fref* supports a richer variety of reference fields, such as translator and hypertext reference.

In summary, *fref* is a solution to the problem of reference list formatting, independent of the macro package used. It does not address the problem of inline reference resolution, which is largely a non-problem for users of the Harvard system.

Operation

The *fref* program takes *troff* source code on standard input and emits the same on standard output. Lines beginning with '%' (percent) are processed specially. Each such line denotes a field in a reference. For example, using the *-ms* macros:

```
.SH
References
.XP
%au  Baudouin de Courtenay, J.
%da  1972
%ti  The Difference between Phonetics and ...
%bo  A Baudouin de Courtenay Anthology
%ed  T. A. Sebeok et al.
%tr  Edward Stankiewicz
%ci  Bloomington
%is  Indiana University Press
.XP
... another reference ...
```

fref collects all percent lines until the next non-percent line, at which point a formatted reference is generated from the information specified in the fields. In the example above, the second .XP line will trigger *fref* to print a reference containing all the fields gathered thus far.

If, for some reason, a specified field is left out by *fref*, a warning is emitted on standard error. This might happen if an %ed (editor) field has been specified without a corresponding %bo (book) or %jo (journal) field.

If an unknown field is specified, *fref* dies.

The hard-coded reference style includes some language-specific words. The language (English by default) is controlled with the *-l* flag:

```
$ <example.t fref -lsv    # Swedish
$ <example.t fref -lru    # Russian
```

Examples

The reference listed earlier is rendered thus:

> Baudouin de Courtenay, J. (1972). The Difference between Phonetics and Psychophonetics. *A Baudouin de Courtenay Anthology* (ed. T. A. Sebeok et al.). Trans. Edward Stankiewicz. Bloomington: Indiana University Press.

The above reference was generated by processing the document's source code with *fref* before passing it to *troff*:

```
$ <fref.t fref | troff -ms | dpost | ps2pdf ->fref.pdf
```

Note that *fref* leaves the job of arranging the references alphabetically to the document author. The benefit is that the author is free to put arbitrary *troff* requests between references.

It is also possible to store references in a separate file, similarly to what is done with *refer*, and process the file separately from the main document:

```
$ <refs fref >refs.t
```

The resulting file refs.t might be included in the main document like so:

```
.SH
References
.so refs.t
```

The benefit of keeping the references in a separate file is that you can automate their alphabetical ordering:

```
#!/usr/bin/perl -n
$i++ if /^[^%]/ and $e[$i] =~ /^%/m;
$e[$i] .= $_;
END { print for sort { f($a) cmp f($b) } @e }
sub f {
        return $1 if $_[0] =~ /^%la.(.*)$/m;
        return $1 if $_[0] =~ /^%au.(.*)$/m;
}
```

If you keep your references in a separate file, you may want to consider using the .blm request to automatically start a new extended paragraph before every reference:

```
.SH
Reference
.blm XP
.XP
.so refs.t
.blm
```

That said, the benefit of storing the references in the document source itself is that you don't run the risk of losing them.

Reference fields

fref supports a large number of fields. All have two-letter names, much like *troff* requests:

%au *Author*. This is the only field of which more than one instance is allowed. Note that the author name is output as is.

%ad *Access date*. Printed after a hypertext reference.

%bo *Book*. Used for the book in which an article is published.

%ci *City*.

%da *Date (usually the year)*.

%ed *Editor*.

%hr *Hypertext reference*.

%jo *Journal*.

%la *Label*. Printed at the beginning of the reference, followed by an equals sign.

%lc *Reference-specific language* ("locale"). Temporarily overrides the *-l* flag and the default language of English for a single reference.

%no *TODO: Issue number*.

%is *Issuer, publisher*.

%pg *Page number(s)*.

%se *TODO: Series*.

%ti *Title*.

%tr *Translator*.

%vo *Volume*.

%xx *Extra information*. Printed at the end of the reference, but before any hypertext reference and access date.

Source code

Users of *fref* are encouraged to modify the source code according to their own requirements. *fref* is written in C using *lex* and contains circa 300 lines of code. Modification of the source code is required by endeavors such as changing the reference format, defining additional fields and adding support for additional languages.

The benefit of writing the formatting code in C is that the programmer has a full-fledged programming language at his disposal. For the string-based operations required by reference formatting, *troff* requests tend to fall short.