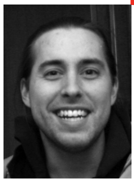
KRISTAPS DŽONSONS

fixing on a standard language for UNIX manuals



Kristaps Džonsons is a graduate student in theoretical computer science at KTH/CSC. He also writes open source software, such as mdocml (mandoc), sysjail, and the mult forks of OpenBSD and NetBSD, with the BSD.lv Project.

kristaps@bsd.lv

"A UNIX UTILITY WITH POOR DOCUmentation is of no utility at all." When sitting down to document utilities and file formats, devices, system calls, and games, there are many UNIX manual formats to choose from, each suffering from limitations.

In this article I survey the cadre of formats and propose fixing on a standard, a format optimally serving readers and writers. I begin by defining the applicable environment, where manuals are written and read, then enumerate criteria for a standard within that space. Among the formats surveyed, I determine that mdoc suffers the fewest limitations. mdoc is popular in BSD UNIX, but it is available pre-installed on any modern UNIX system, from GNU/Linux to Mac OS X to OpenSolaris.

First, it's important to ask: Why doesn't a standard already exist? In short, the current spread of formats—diverse as it may be—is *good enough*. UNIX users, programmers, and administrators tolerate the menagerie so long as the output of the man utility is roughly consistent. I propose that the benefits of fixing on a standard, from consistent authorship to powerful analytical tools, stipulate only a minimal burden of change: policy creation, education of authors, and slow migration from substandard formats.

The troff Condition

We generally associate the man utility with documentation, but, internally, it only locates manuals, invokes an output formatter, then pages to the screen. This formatter constitutes the primary mechanism of manual production. UNIX systems overwhelmingly use troff [2] as a formatter, usually in the form of a modern implementation such as GNU troff (groff) [3], or Heirloom troff [4]. I'll refer to "troff" as a stand-in term for any of these implementations.

I define a format as *reasonable* only if it's accepted by troff with specific, documented utility for formatting UNIX manuals. A format is *semi-reasonable* if it's indirectly accepted—losslessly transformed into an accepted form by an existing intermediate translation utility. In this study, I consider only reasonable and semi-reasonable formats.

An example of an unreasonable format is HTML, which is neither accepted by troff, losslessly translatable, nor has a UNIX manual mode. The panoply of common word-processing formats, such as the Open Document Format and Rich Text Format, are similarly unreasonable.

The roff me, ms, and mm macro packages, while accepted by troff and occasionally used for older manuals, are not considered as having a specific utility for UNIX manuals; thus, I consider them unreasonable. texinfo [5], while being used for general documentation, is also not specifically used for UNIX manuals and is therefore unreasonable.

Criteria

I define the set of standardization criteria as follows: *structural readability*, such that end users are presented with structurally consistent output between manuals; *syntactic regularity*, such that machines may disambiguously scan and parse input; and a rich *semantic encapsulation* for meaningful machine interpretation of contextual data.

We're comfortable with conventional man output: margin widths, text decorations, and so on. Structural readability stipulates consistent output given a heterogeneous set of input documents. Syntactic regularity is both a formal term, regarding grammar, and a subjective one, regarding the writer's ease of composition. In this article, I focus on the former: input languages must be reliably machine-parseable. Lastly, semantic encapsulation requires the annotation of information. Meaningful manual terms, such as function prototypes and cross-links, must be disambiguously annotated, as machines cannot reliably classify context in unstructured text.

By fixing on a language that meets these criteria, we guarantee maximum, meaningful exposure of our manual, and we expand the end user's documentation tool set—these days, necessarily constrained by the chaos of volatile conventions and irregular formats—with sophisticated tools for cross-referencing, formatting, and so on.

Survey: man and POD

troff accepts the "roff" type-setting language as input; however, direct usage of roff has been eclipsed by the use of macro packages simplifying the language—macros, like procedural functions, are a roff language feature allowing complex macro blocks to be referenced by a simple call. troff internally replaces these macros with roff during a pre-processing phase.

The man macro package became the first common format for creating UNIX manuals (predated by the mm and me packages) and established the backspace-encoded, 78-column display style enjoyed to this day.

```
.SH SYNOPSIS
.B find
[\fB\-dHhLXx\fR]
```

FIGURE 1: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH MAN

The fragment in Figure 1 illustrates a manual's synopsis section: the SH macro (all roff macros appear on lines beginning with the '.' control character) indicates section titles, and B applies a boldface type to its argument. The argument string is bracketed by boldface character escapes. In general, man macros describe the presentation of terms.

```
==head1 SYNOPSIS

    B<find> S<[ B<-dHhLXx> ]>
```

FIGURE 2: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH POD

The man format also forms the basis for the Perl "POD" (Plain Old Documentation) language, illustrated in Figure 2. POD, like man, is a presentation format.

Survey: mdoc

The other roff manual-formatting macro package is mdoc, which, beyond sharing common ancestry, is fundamentally different from man. Instead of annotating presentation, mdoc semantically annotates its terms. In Figure 3, for example, Op indicates an option string, usually displayed as enclosed in brackets, followed by a series of flags offset by the Fl macro. The proper presentation of these macros is managed by the formatter.

```
.Sh SYNOPSIS
.Nm find
.Op Fl dHhLXx
```

FIGURE 3: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH MDOC

Both man and mdoc are accepted natively by troff. POD is the default format for embedding manuals in Perl documents, and it translates directly into man with the perlpod utility for indirect acceptance by troff.

Survey: DocBook

The DocBook [6] suite, like troff, is a general-purpose typesetter. Unlike troff, its input language, also called "DocBook," is based on XML (historically, SGML). DocBook has a schema for annotating UNIX manuals, illustrated in Figure 4, translating into man with docbook2x and docbook-to-man for further compilation by troff.

```
<refsynopsisdiv>
        <cmdsynopsis>
                <command>find</command>
                <arg choice="opt">
                        <option>dHhLXx</option>
                </arg>
        </cmdsynopsis>
</refsynopsisdiv>
```

FIGURE 4: FRAGMENT OF FIND MANUAL SYNOPSIS SECTION AS FORMATTED WITH DOCBOOK

The necessary complexity of processing XML demands a significant infrastructure of compilers and schemas to correctly transform materials. docbook-to-man (which operates only on SGML DocBook) requires an SGML parser, the appropriate DTD files, and a driving script. Importantly, existing tools for translation only produce man-lossy transition from semantically encoded to presentation-encoded documents.

Evaluation

The criteria described earlier in this article were structural readability, syntactic regularity, and semantic encapsulation. I noted that these criteria only apply to reasonable or semi-reasonable formats.

By virtue of being directly accepted by troff, mdoc and man are both eminently reasonable. DocBook and POD, on the other hand, require specialized utilities to translate input into man. Although these utilities must in general be downloaded and installed, their popularity makes them readily available on most systems, and thus they are semi-reasonable.

The matter of structural readability may be reduced to the author's level of influence on presentation. DocBook and mdoc manage presentation, while man and POD must be styled by the author. Given a non-uniform distribution of authors, it's safe to say that mdoc and DocBook satisfy readability more readily than the presentation languages. In other words, an author's control over function prototype styling will almost certainly produce varied output.

Syntactic regularity is both grammatical and structural. DocBook, by virtue of XML, follows a context-free grammar (upon combination with the tag schema); mdoc, man, and POD are context-sensitive. The matter of structural regularity, on the other hand, is largely subjective; some prefer the terseness of roff macros, while others prefer more descriptive DocBook tags.

In general, it's safe to say that DocBook's context-free foundationspromotes its syntactic regularity above the others. The matter of structural regularity, while important, remains a subjective matter.

The last criterion, semantic encapsulation, is by far the most significant in terms of meaningful analysis of data. POD and man, as with any presentation language, are semantically opaque: beyond using heuristic analysis, the content of these manuals is closed to machine interpretation.

```
\fIvoid\fP \fBexit\fP \*(lp\fIint\fP\*(rp
```

FIGURE 5: FUNCTION PROTOTYPE ENCODED IN MAN

DocBook and mdoc, however, are rich with semantic meaning; by careful analysis of the parse tree, machines can cross-link references, group terms, and perform many other useful operations. Figures 5 and 6 illustrate presentation and semantic encapsulation, respectively.

```
.Ft intmax_t
.Fn imaxabs "intmax_t j"
```

FIGURE 6: FUNCTION PROTOTYPE ENCODED IN MDOC

As noted earlier, DocBook's translation tools don't currently produce mdoc, which amounts to a lossy transform. Thus, while DocBook itself may be semantically rich, its intermediate format, and thus troff input, is not.

The format fitting all criteria with the fewest limitations is mdoc, featuring a reasonable, semantically rich language for manual data annotation. The lossy translation of DocBook to man, as well as its requirement of downloading additional processing tools, render it substandard.

The man and POD formats, as presentation languages, are opaque to machine interpretation. I consider this an insurmountable limitation, since it prohibits meaningful analysis of manual data.

Adoption

The hindrance of mdoc's widespread adoption is as much due to its poor exposure beyond the BSD UNIX community as to the limited semantic functionality of its popular compiler, groff.

Documentation for the mdoc format is, at this time, constrained to templates, the formidable mdoc.samples manual distributed with most BSD UNIX operating systems, and the minimal mdoc manual in general UNIX systems. Furthermore, unlike man, which exports few macros, the complexity of mdoc, with well over 100 available macros, makes introductory reference materials critical.

Although serving to format mdoc manuals for regular output, groff offers no semantic-recognition features: for example, HTML output (via grohtml) correctly cross-referencing manual references. This is a matter of groff's design, which internally translates mdoc into a presentation-based intermediate form, thus losing the semantic annotations of the input.

Fortunately, groff's limitations are being addressed by the mandoc [1] utility, which exports a regular syntax tree of mdoc input (and man, within the limitations of presentation encoding) for analysis. The issues of good introductory documentation and exposure, unfortunately, remain unsatisfied.

Conclusion

By using mdoc to write manuals, powerful documentation analysis is made considerably easier—arguably, by using man, POD, or a similar presentation format, meaningful analysis isn't possible at all. This is demonstrated by the total lack of manual analysis beyond the man, apropos, and whatis utilities, and various patchwork presentation services (such as man.cgi [7] and man2web [8]) in use today. Attractive, cross-referenced hypertext references, section-by-section querying of local manual sets, and other possibilities arise by fixing on mdoc, possibilities hindered by the preponderance of presentation-based, opaque languages.

REFERENCES

[1] mdocml: http://mdocml.bsd.lv.

[2] troff: http://www.troff.org.

[3] groff: http://www.gnu.org/software/groff/.

[4] heirloom: http://heirloom.sf.net/doctools.html.

[5] texinfo: http://www.gnu.org/software/texinfo/.

[6] docbook: http://www.docbook.org.

[7] mancgi: http://www.freebsd.org/cgi/man.cgi/help.html.

[8] man2web: http://man2web.sf.net.