The journal of Apple technology.

Volume Number: 19 (2003)
Issue Number: 2
Column Tag: Section 7

Text-based File Formats

CSV, OML, XML, YAML...

by Rich Morin

BSD and OSX inherit a long tradition (stretching back into the earliest days of Unix) of using text files for data storage. Although there are some exceptions, most control, log, and other system data files are written in ASCII. This makes them easier to inspect, post-process, and even edit.

Apple, whose historic bent has been more toward binary file formats (e.g., the resource fork), seems to have adopted this idea wholeheartedly. In fact, they have gone a bit further, adopting XML (rather than line-oriented files) and Unicode (rather than ASCII). As a result, many OSX files are well structured, language-independent, and quite accessible to both humans and programs.

Many vendors (e.g., Microsoft) are also joining the XML caravan. Assuming that they document both the syntax and semantics of their interchange formats, we could see a dramatic change in possibilities for file interchange.

It is not clear, however, that XML is the Right Answer for all problems. Let's look at some of the alternatives, examining their strengths and weaknesses. Don't expect a comprehensive list; there are zillions of data formats in use. Here, in any event, are some that I would recommend.

CSV

Although CSV stands for "comma-separated values", commas are by no means essential to the idea. In fact, another term for this is "flat file format". Basically, the idea is that each line is a record and that some other delimiter (e.g., colons, commas, white space) is used to separate fields. Quotes or other devices are sometimes used to protect instances of the delimiter in the body of a field. Here are some examples:

```
/etc/crontab
  15 3 * * * root periodic daily
/etc/gettytab
  a|std.110|110-baud:\
        :np:nd#1:cd#1:uc:sp#110:
/var/log/netinfo.log
  Dec 21 17:59:33 cerberus netinfod ...
An_Excel_File.csv
  1,2,3
  "1,2,3","3,4,5"
```

Some files get a bit complex, adding syntax to support block structure, comments, line breaks, shell commands, etc. A highly-ornamented CSV file can begin to look like a "little language". Here is a fairly complex format, drawn from

/var/named/named.local:

```
$TTL 86400
@ IN SOA localhost. root.localhost. (
    1997022700 ; Serial
    28800      ; Refresh
    14400      ; Retry
    3600000    ; Expire
    86400 )    ; Minimum
  IN NS  localhost.
1 IN PTR localhost.
```

Many BSD control files, logs, and reports use white space to delineate fields, making the assumption that the included data will not contain spaces. This was never a safe assumption for path names, but the advent of OSX has made the problem all too real.

Try running "ps -axww" and see how man path names (for both commands and arguments) contain spaces. Then, consider how you would code up a way to determine which spaces are field separators and which are not. Not simple...

Of course, ps has other problems. Run "vi 1 '2 3' 4" in one window and "ps" in another. In the ps output, the COMMAND field looks like "vi 1 2 3 4", dooming any effort to parse it into a command path and distinct command-line arguments. Some sort of quoting convention is desperately needed here.

Blanks in path names make it impossible to parse certain log files, have already broken one Apple installer script, and could well foul up many BSD control files. It's probably too late to get Apple to back off from their use of embedded blanks in system path names, so you can expect to see some problems of this nature coming along...

OML

Although CSV is attractively simple, it may be too simple for your needs. For instance, you may need to support optional data, hierarchical structures, etc. On the other hand, you may not be ready for the formality of XML (Extensible Markup Language) and unwilling to design your own markup language (and parser).

OML (Ostensible Markup Language :-) is a powerful, simple, and convenient solution to this dilemma:

```
# This is a sample of file-system metadata.
<snap>
  <file>ASCII text</>
  <flags>f,avbstclinmed,,</>
  <lstat>303507,33200,1,1000,20,914,8</>
  <md5>2e1240f444fc3f984186fc5a4fd28eb0</>
  <times>1040087752,230,1740993,10890145</>
</snap>
```

This looks quite a bit like XML, but there are some small peculiarities. That comment, for instance, isn't legal XML. Nor is "</>" a legal termination for a tag. Is that really CSV syntax in the middle of some fields? Finally, where are the header lines?

Though OML is seemingly designed to give hives to XML purists, it is also designed to work smoothly with existing XML tools. A couple of lines of Perl will strip out the comments and fill out the terminations. A quick pass through an XML parser extracts all of the named fields and attributes. Perl's split() operator, if need be, can break up the CSV data.

Since OML is pretty much a "roll your own" kind of thing, there isn't any real documentation. I'd suggest a look at "Doing it Simpler" (Leigh Dodds; www.xml.com) for some ideas, however.

XML

Despite any appearance to the contrary, I am quite a fan of XML. An enormous amount of meticulous and thoughtful effort (and some rather fancy computer science!) is going into creating "industrial strength" data formats and processing tools. In addition, the W3C (World Wide Web Consortium; www.w3.org) is being very careful to make sure that the official standards are open to all players.

For some projects, you really need to bring in power tools. The host of translators, validators, and other tools that XML provides can make otherwise impossible projects feasible, if not necessarily reasonable. Using XML also increases the chance that someone else's program will be able to parse your data. Given all of that, complaining about a bit of formality seems rather petty.

I won't try to cover XML here; there are shelves of books on the subject, with more coming out on a weekly basis. O'Reilly and Addison-Wesley have the broadest coverage; O'Reilly's XML web site (www.xml.com) is a good place to start your journey...

YAML

CSV has low overhead, is simple to read and edit, and handles lists and arrays well. OML and XML are a bit more bulky, but handle optional data and hierarchies smoothly. XML has an impressive suite of documentation, standards activities, and support software. Sometimes, however, you want low overhead, simplicity, and support for arbitrary data structures.

YAML (YAML Ain't Markup Language) fills this niche quite admirably. The syntax is simple and clean. The basic data structures are sequences (i.e., Perl arrays) and mappings (i.e., Perl hashes). YAML handles lists, arrays, and hierarchies easily; with a bit of extra work, it can handle arbitrary Perl data structures (e.g., cyclic graphs).

Here is the previous example, transliterated into YAML:

```
# This is a sample of file-system metadata.
snap:
  file:  'ASCII text'
  flags: 'f,avbstclinmed,,'
  lstat: '303507,33200,1,1000,20,914,8'
  md5:   '2e1240f444fc3f984186fc5a4fd28eb0'
  times: '1040087752,230,1740993,10890145'
```

A more idiomatic rendering, however, would look like:

```
# This is a sample of file-system metadata.
snap:
  file:  ASCII text
  flags: [ f, avbstclinmed, , ]
  lstat: [ 303507, 33200, 1, 1000, 20, 914, 8 ]
  md5:   2e1240f444fc3f984186fc5a4fd28eb0
  times: [ 1040087752, 230, 1740993, 10890145 ]
```

Aside from the fact that some spaces have been added after commas and the quotes have been eliminated (some turned into brackets), the second version looks very similar to the first. The resulting data structure is quite different, however; the bracketed lists have been turned into YAML sequences. This means that they don't have to be parsed in a follow-on step. Here is some access code, in Perl:

```
$file = $yaml{snap}{file};
$uid  = $yaml{snap}{lstat}[3];
```

YAML has several ways to write textual data. Here are some examples:

```
  - a simple text item
  - "double-quoted text\n "
  - 'single-quoted text'
- >
    This text
    is freeform.
- |
    This text
    isn't.
```

Although YAML has nowhere near the amount of documentation that XML has, there are some useful resources to recommend. The YAML web site (www.yaml.org) is the logical place to start; be sure to visit the YAML wiki. I'd also recommend a look at "Look Ma, No Tags" (Kendall Clark Grant; www.xml.com) for an informal introduction.

Rich Morin has been using computers since 1970, Unix since 1983, and Mac-based Unix since 1986 (when he helped Apple create A/UX 1.0). When he isn't writing this column, Rich runs Prime Time Freeware (www.ptf.com), a publisher of books and CD-ROMs for the Free and Open Source software community. Feel free to write to Rich at rdm@ptf.com.