Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

Filtering KPNO LaTeX Observing Proposals with Perl

David J. Bell

National Optical Astronomy Observatories, Tucson, AZ 85726-6732

Abstract. An automated observing proposal processing system has been in use at KPNO during the past three years. LaTeX proposal templates are filled out by users and sent to KPNO via electronic mail. Observer and proposal information fields in these files are well-marked with LaTeX tags, thus allowing automated extraction and importation to observatory databases.

A significant complication of this process is that although the fields are well-marked, the information they contain often arrives in a variety of formats that must be recognized and standardized. Perl's regular expression and text manipulation capabilities make it an excellent tool for performing these functions. This paper outlines the filtering system in use at KPNO and discusses some of the ways Perl has proven useful for parsing LaTeX documents.

1. Introduction

The Kitt Peak observing proposal handling system (Bell et al. 1996) is an automated system for distributing, receiving, and processing LaTeX observing forms and associated PostScript figures. It has already handled several thousand files for Kitt Peak alone, and has also been in use at two other observatories. Twice a year, as proposals are arriving, information must be quickly imported and updated in observatory databases. In the past this was performed through manual entry while inspecting each paper copy—a tedious and sometimes overwhelming task, particularly when over 100 proposals arrive on the final day.

This paper describes a new and better approach. The proposals are run through a filtering program that locates the desired fields in the proposal form, optionally parses each field into several database subfields, and rearranges the information into a standardized format. When confused about an entry, the program attempts to make a good guess but also flags that item for human inspection. Such an approach can save much time, while still ensuring the accuracy of imported data.

2. Why LaTeX?

LaTeX is a widely and freely available text formatting language that gets significant use throughout the astronomical community. Even inexperienced users can fill in a well-designed template form using any text editor, and submit it with any e-mail program. The completed forms serve a dual role: they can be printed to produce nicely formatted paper documents, and, since needed information is well-tagged by the structural markup, they can also be processed by automated scripts to extract data fields. If needed, they can be edited locally (impractical with PostScript documents, for example), and by modifying a single style file one can quickly reformat hundreds of documents.

However, problems can occur during automated filtering. Some of the fields, such as addresses, consist of just one line on the form, but need to be split into several subfields for the database. The form could be changed, but there are already thousands of existing documents in the present format, and it would be inconsiderate to force users to split entries into many subfields that are all recombined on the printed form. A second problem is that users often embed commands into the fields, and it would be confusing if we mailed out envelopes with raw TEX in the addresses. The biggest drawback, however, is that there are no limits over what users type into the fields, whereas the database needs standardized formats.

3. Why Perl?

Perl is a widely and freely available scripting language that is best known for its many uses for system administration tasks and, more recently, as being *the* CGI-programming language. What makes it so good at these things is that it is a superb text-processing language. It includes some of the most efficient and powerful regular expression and string manipulation operators available anywhere, allowing it to quickly locate and manipulate myriads of tiny parcels of text. Powerful string manipulation code can be written quickly and compactly, making it great for processing LATEX files (for instance, the popular `latex2html` program is a Perl script). Perl has a familiar C-like syntax and a very forgiving nature. If users leave fields blank, or type in full sentences when the form asked for an integer, the script doesn't bomb out, but can easily be programmed to do something reasonable and move on.

4. Filtering Strategies and Examples

Many of the fields extracted from the forms require very little processing other than reformatting, and this can often be done in just a few lines of code. For example, the lines:

```
$phone = "($1) $2-$3 $4" if ($phone =~
        /^\D*(\d{3})\D*(\d{3})\D*(\d{4})\s?(.*)$/);
```

will recognize a US phone number in practically any likely format, such as "800–5551212ext123," and standardize it to "(800) 555-1212 ext123." Some more interesting examples will be discussed in the following sections.

4.1. Names and Addresses

Name and address entries on the form need to be split into subfields for the database. LATEX codes are stripped out (e.g., diacritical marks) or replaced (e.g., non-English characters). Names are split into an array based on punctuation and whitespace, and then compared to lists of titles to be thrown away, or

```
\name{Prof.~Dr.~Ant{\^o}nio-Ryan M.\ VAN DE W\O RF, Jr.}
\address{\small Dept.~of Phys.~\& Astr.; Mail Stop 16; rm 101;
     VICTORIA B.C.\ V8 x4 m6~~Canada}

FN: Antonio-Ryan
MI: M
LN: Van De Worf, Jr
A1: Dept. of Phys. & Astr
A2: Mail Stop 16, rm 101
CY: Victoria
ST: BC
ZC: V8X 4M6
CO: CANADA
```

Figure 1. LaTeX Name and Address Fields and Filter Output. Formatting and special-character codes have been stripped and the entries correctly parsed into subfields. The province, postal code, and country name have been standardized.

```
\telescope{ 4.0-meter~~~}
\instrument{Prime focus camera with the new 4$\times$2 CCD mosaic}

TE: 4m
IN: PF
DE: MOSA
```

Figure 2. LaTeX Telescope and Instrument Fields and Filter Output. Regular expression hashes have been used to identify items and return standardized database codes.

```
\optimaldates{2/16-3/2, 4/14-5/1 or 5/13-31}
\acceptabledates{21DEC1996---07JUN1997\hfil}
\optimaldates{\it Feb.~1st--27th or March 2nd--23rd, 1997}
\acceptabledates{Late sept.\ through early-april}

OD: Feb 16 - Mar  2, Apr 14 - May  1, May 13 - May 31
AD: Dec 21 - Jun  7
OD: Feb  1 - Feb 27, Mar  2 - Mar 23
AD? Sep 20 - Apr 10
```

Figure 3. LaTeX Date Fields and Filter Output. Since the last range is somewhat vague, the filter has flagged the field for human inspection.

likely surname components that need to be recombined into a last-name field. Address parsing is more difficult, due to widely varying punctuation and foreign addresses. For this reason, lists of cities and countries from past proposals are first searched, and once a city is found, the parsing is almost always right. When new cities show up, the script makes an attempt at parsing based on punctuation, flags the data, and logs the new city for possible addition to the search list. See Figure 1.

4.2. Telescopes, Instruments, and Detectors

The primary problem in these cases is in recognizing the many ways in which the same item can be requested. This is achieved by stepping through associative arrays in which a standardized code is the key and a regular expression that matches various forms of that item is the value. For instance the hash element

```
RCSP => r[-\.\s]*c[-\.\s]*sp
```

can be used to map user entries such as "r-c spectrograph" or "R. C. Spec" to the standardized code "RCSP." When new instruments become available, the script can be quickly updated simply by adding a new code-regex pair to the hash. See Figure 2.

4.3. Observing Dates

Interpreting date strings requires parsing a string into several date ranges, then each range into dates, and finally each date into a day and month. Commonly used English words like "through" and "or" are first replaced with symbols like "-" and "," respectively, before splitting on these symbols. Unneeded information, such as years and ordinal abbreviations, are removed. Months are then standardized with a series of substitutions, so that the strings "09/", "SEP", "september", etc., will all be turned into "Sep". Once this month string is pulled out, what's left is hopefully a day number—if not, special cases like "mid" are considered. See Figure 3.

5. Conclusion

A Perl script has been developed for filtering KPNO LaTeX observing proposals. Although the desired information arrives in a large variety of formats, Perl's powerful text manipulation capabilities allow the script to accurately identify and reformat entries for database import. Only a few standardization problems remain, and these may be eliminated in the future with pre-configured menus and buttons on an HTML form—such an interface to the LaTeX template is currently being designed.

References

Bell, D., Biemesderfer, C. D., Barnes, J., & Massey, P. 1996, in ASP Conf. Ser., Vol. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes (San Francisco: ASP), 451