Abstract

Summary: We present a set of Perl modules for the flexible and robust parsing and editing of EMBL/SWISS-PROT databases.

Availability: The Web page at http://www.sanger.ac.uk/Software/PerlModule/ provides information about downloading the SPEM and PrEMBL modules, and provides links to documentation and example code.

Contact: mrp@sanger.ac.uk

Introduction

The major biological sequence databases are distributed in partially structured flat file formats. There is a recurrent need of informaticians to parse these files automatically, in particular to retrieve information from EMBL (Stoesser et al., 1998; http://www/ebi.ac.uk/embl.html) and SWISS-PROT (Bairoch and Apweiler, 1998; http:/www.expasy.ch/). At the Sanger Centre, we routinely construct and submit EMBL entries. Currently, the formatting of annotation and comments is done by hand. As the rate of sequence production continues to increase, we need to be able to automate the generation and correction of EMBL entries, within our heterogeneous computer environment.

There are already tools to query these biological databases, although, to our knowledge, none that allow their editing. SRS (Etzold et al., 1996) is a powerful system able to construct and execute queries across multiple databases. However, it does not support in-place editing, requires appreciable system resources (both computationally and for storage), and it is not easy for a naïve user to form these queries. The XYLEM package (Fristensky, 1993) contains no functionality for editing the feature table in place, and the parser cannot be modified. A relational database could store the EMBL database. However, we would still have to add functionality to parse the entries into the database, edit them within it and print them out again. This approach requires third party software of the database itself. Another option is the NCBI toolkit, which uses ASN.1. We know of no parsers for this toolkit to EMBL format and, in addition, as it is C based, the toolkit is not suitable for the rapid development of scripts.

The lack of a robust and portable method to provide large-scale parsing and editing of EMBL entries led us to develop the SPEM and PrEMBL object Perl modules. Perl was chosen due to its speed and the ease with which Perl scripts can be developed, and its strong cross-platform support, including UNIX, Win32 and Macintosh. SPEM is the set of object Perl modules that encapsulate the parsing of EMBL and SWISS-PROT formatted text files, and for editing or creating entries. There is one concrete implementation of SPEM, called PrEMBL, which manipulates files in the EMBL format. The PrEMBL modules provide a standard interface for parsing and editing the EMBL database. Users can interact with the modules without needing a comprehensive knowledge of the database format. It is possible to process the whole of the EMBL database, entry by entry, on a modest workstation using these modules. Entries are represented as a series of objects, creating a natural mapping to other object-based representations. The recent adoption of a CORBA standard for a biological database (Rodriguez-Tomé et al., 1997) illustrates the trend for object-based processing of biological databases.

Overview of SPEM

There are three base classes provided in SPEM. These are SPEM::Parser, SPEM::Set and SPEM::Entry. They encapsulate the core behaviour and content of a stream parser, a set of entries and an individual entry, respectively. The modules work on the premise that a SPEM formatted entry is made up of a set of objects that know how to transform themselves to and from a text stream. A particular object might be responsible for more than one line in the stream. The SPEM::Set holds a set of such entries and the SPEM::Parser is responsible for producing such a set from a text stream, each object registering its interest in a particular tag (e.g. ID or CC). As any of the object definitions are easy to change, the parser can be easily modified to construct different representations of the text stream.
Fig. 1. Example Perl script using the PrEMBL modules. This script processes all of the entries returned from an SRS query. All sequences <400 nucleotides are discarded. A comment is added to the remaining entries, and all of the repeat features are edited out. Finally, each edited entry is written out to STDOUT.

The EMBL implementation of SPEM

Figure 1 gives a brief overview of some of the ways that the PrEMBL modules can be used. The PrEMBL modules define the necessary objects to represent the EMBL database which are registered with a SPEM::Parser. Additional methods are provided in the PrEMBL::Entry object to retrieve the common objects of an entry, e.g. the ID field or the feature table.

The PrEMBL feature table implementation

For most applications, by far the most informative part of an entry is the feature table. The feature table is a single object responsible for all the FT lines in the EMBL file. This is the region that most people need to parse. Therefore, there is a rich functionality associated with the feature table object. The feature table is composed of a list of features, each with a key, a location, a bounding box and a list of qualifiers. The qualifiers are presumed to have a name and a value, where appropriate. No attempt is made to enforce the requirement that the qualifiers are legal EMBL components. The feature table can be searched by location, or by text key, or by a user-defined method.

Future development

It is hoped that this framework can be extended to encompass the other flat file databases that employ similar formats, such as SWISS-PROT and TrEMBL. This would provide informaticians with uniform access to these data sources, and promote code reuse between both individuals and projects. Also, because the SPEM specification implicitly includes the ability to reproduce the entries and database from within the program, it will be invaluable in the construction and large-scale editing of these types of resources.

References