Development of the CCP4 software library

Martyn Winn, Charles Ballard and Eugene Krissinel December 2001

Aims

The CCP4 software suite is based around a library of routines which cover common tasks, such as file opening, parsing keyworded input, reading and writing of standard data formats, applying symmetry operations, etc. Programs in the suite call these routines which, as well as saving the programmer some effort, ensure that the varied programs in the suite have a similar look-and-feel.

Over the past 12 months, there has been a major effort to re-write much of the CCP4 library. The aims are:

- To implement a better representation of the underlying data model. For example, Eugene Krissinel's mmdb library acts on a data structure which represents the various levels of structure of a protein model. The new MTZ library encapsulates the crystal/dataset heirarchy that is increasingly being used by programs.
- To maintain support for existing programs. In particular, the existing Fortran APIs will be maintined, although they will now often be only wrappers to functions in the new library. It is hoped that many existing programs will be migrated to using the new library directly.
- To provide support for scripting. It is possible to generate APIs for Python, Tcl and Perl automatically from the core C code. Thus, much of the standard CCP4 functionality wil be available to scripts used e.g. in ccp4i or the molecular graphics project.

This incremental approach, maintaining the existing suite while improving the underlying code, puts constraints on what is possible, but is considered more appropriate for a collaborative project like CCP4.

Major components

mmdb

The "mmdb" library is designed to assist CCP4 developers in working with coordinate files. The major source of coordinate information remains the PDB files, although more information is becoming available in mmCIF format. The "mmdb" library will work with both file formats plus an internal binary format portable between different platforms. At the level of the library's interface function, there is no difference in handling different formats.

The "mmdb" library provides various high-level tools for working with coordinate files, which include not only reading and writing, but also orthogonal-fractional coordinate transforms, generation of symmetry mates, editing the molecular structure and some others. More information can be found on the mmdb project pages (http://msd.ebi.ac.uk/~keb/cldoc/).

cmtzlib

In the new formulation, reflection data is still held in a table format, but columns of data are arranged in a heirarchical manner. From the top down, the heirarchy is:

File -> Crystal -> Dataset -> [Datalist] -> Column

A `Crystal' is essentially a single crystal form, a `Dataset' is a set of observations on a crystal, and a `Datalist' (not yet implemented) is a grouping of associated columns. Note that the `Project' used in Data Harvesting (Newsletter #37) is now simply an attribute of the crystal.

The MTZ file format has been extended slightly to record this heirarchy. I have written a C function library to read/ write these extended MTZ files, and to manipulate a data structure representing the above data model. Some functions are derived from Jan-Pieter Abraham's solomon code, though they have been substantially altered (and therefore any problems are of my creation!).

One consequence of this formulation is that columns can now be identified in terms of their crystal or dataset, e.g.

```
LABIN  FP=NATIVE/FTOXD3 SIGFP=NATIVE/SIGFTOXD3
```

I have also written a Fortran API to the C library, which mimics the existing `mtzlib.f`. It should be possible to migrate existing Fortran programs to use this API with few/no changes. It is expected however, that future applications will use the core C functions which give better access to the data structure.

Some information on cmtzlib is available from http://www.ccp4.ac.uk/martyn/cmtz.html, but further developments will be integrated into the CCP4 library development.

cmaplib

Charles Ballard has written a C language library for the reading and writing of CCP4 format map files. A Fortran API mimics the existing `maplib.f`.

In a parallel development, there have been recent efforts to bring the MRC format for three-dimensional electron microscopy maps in to line with the closely-related CCP4 format map. For our part, we are considering suggested changes to the CCP4 map format which will be useful for EM applications. Some details can be found on the IIMS home page (http://msd.ebi.ac.uk/iims.html).

csymlib

The old implementation of symmetry held tabulated information in a manually-produced file `symop.lib`, together with other information distributed amongst routines in `symlib.f` (e.g. real space asymmetric unit limits in subroutine SETLIM). This set-up works in most cases, but was error-prone and difficult to maintain.

In the new formulation, `symop.lib` is replaced by another data file which is automatically generated. This is currently done using a short program which uses functions from `sgtbx` (part of the Computational Crystallography Toolbox, http://cctbx.sourceforge.net) . The new data file is more likely to be error-free, and is also more complete, in that many non-standard settings can be included easily. The new data file contains most quantities of interest, and only a few pieces of tabulated data are retained in the code (e.g. specifications of centric and epsilon zones).

I have written a new C library of functions to manipulate this symmetry information. When a spacegroup is identified by its name, number or operators, all the information connected with that spacegroup is loaded into memory, where it can be accessed easily. Wrapper functions mimic the old `symlib.f` routines.

ccp4_unitcell

The CCP4 library contains many routines concerned with the unit cell, in particular to do with tranforming between orthogonal and fractional coordinates. I have written C functions to perform the basic manipulations, with Fortran-callable wrappers to mimic old routines in `rwbrook.f` and elsewhere.

core library

Charles Ballard has re-written most of the core code. The files concerned are:

```
library_file.c         )
library_err.c          ) replacing library.c
library_utils.c        )
ccp4_general.c        ) to give CCP4 look-and-feel
ccp4_program.c        )
ccp4_parser.c           parser functions from Pete Briggs
ccp4_diskio.c          replacing diskio.f
```

These functions mostly support other code, but may be of use to application developers.

Support for scripting languages

I have used SWIG to generate interfaces to the python, perl and tcl scripting languages. These interfaces exist as automatically-generated wrapper functions, for example the files

```
cmtzlib_python_wrap.c
cmtzlib_perl_wrap.c
cmtzlib_tcl_wrap.c
```

provide cmtzlib functions which can be called from python, perl and tcl respectively. These wrapper functions, together with the main library code, are compiled into shared libraries which can be loaded by the scripting language:

```
ccp4module.so          loaded by       from ccp4 import *
ccp4_pl.so             loaded by       package ccp4_pl;
ccp4_tcl.so            loaded by       load ./ccp4_tcl.so ccp4_tcl
```

Much of the library functionality can be made available to the scripting language with virtually no effort. Access to complex datatypes requires a little more effort, as does wrapping C++ code, and this will be provided as required.

Other features

The new CCP4 library can be used in a variety of ways, depending on the functionality required and the language chosen. Therefore, facilities will be provided to do partial compilation, so that only the functions appropriate to the application are provided in the library. In any case, care is being taken to establish appropriate namespaces, to avoid conflicts in any programming environment. Many of the existing Fortran routines are being left unchanged, for example certain routines in ccplib.f and all of plot84driver.f. This is either because the routines are Fortran-specific, or simply because C equivalents have not been coded yet.

Acknowledgements

`cmtzlib` was originally based on Jan-Pieter Abrahams' functions in the program `solomon`. These have been substantially altered and added to, and so all problems are my creations, but I am grateful for the excellent starting point.

The formulation of this library has benefited from many discussions with Kevin Cowtan (who also provided some core functions), Eugene Krissinel, Airlie McCoy, and all members of the Daresbury team (Alun Ashton, Peter Briggs, Charles Ballard).