5.3.7.1.2.2. *zincCif*

zincCif is a Perl script that takes a ZINC-format file (again from standard input or as a name on the command line) and pretty prints the corresponding CIF to standard output. Often, the pipeline `cifZinc a.cif | zincCif > b.cif` produces a more attractive CIF than the original.

5.3.7.1.2.3. *zincGrep*

The shell script *zincGrep* is the utility most requested by those seeing a CIF for the first time. It allows a regular-expression search of a ZINC-format file (or a CIF specified on the command line, which is converted to a ZINC-format file first) and reports the block name, data name, index and value. For example, if the file describing a triangle in the last section were called simple.cif, the command `zincGrep _name simple.cif` would produce

```
object _name            ;\ntriangle\n;
```

5.3.7.1.2.4. *cifdiff*

cifdiff is a C-shell script that takes two CIFs and lists the differences between them. Unlike the standard Unix utility *diff*, which compares files line-by-line, *cifdiff* can determine differences that are independent of reordering and white-space padding.

This script takes each CIF, converts it to a ZINC-format file, then sorts it, first based on the data-block name, then (keeping the loops together) on the data name. It then removes the last field (which is not part of the CIF) and stores the remainder in temporary files. It then runs the standard *diff* program against these reordered temporary files. This is remarkably effective both in finding any differences and in providing the context (it names the block and data name as well as the value) needed to understand the differences.

Fig. 5.3.7.2 illustrates two CIFs that are very different in the presentation of their contents, but have only a small difference of substance in their content. Fig. 5.3.7.3 indicates the output from *cifdiff* that identifies the changed data.

5.3.7.1.2.5. *zb*

zb is a small (less than 200 lines) Tcl/Tk program (Ousterhout, 1994) that provides a simple graphical front end to a ZINC-format file or CIF allowing the user to browse through the contents. Multiple files can be viewed simultaneously, as can multiple data blocks, on any X terminal. *zb* recognizes command-line argument file names in the form *.cif as being in CIF format and converts them to ZINC format automatically.

5.3.7.1.2.6. *zincNl*

zincNl is a Perl script that takes a ZINC file and creates a Fortran-compatible namelist file allowing for easy access to any CIF by Fortran programs without the need for extensive I/O libraries or reprogramming. As with *zb* above, it will automatically convert a CIF to a ZINC-format file if it needs to.

For a more substantial tool providing CIF input functions in Fortran and C, see the discussion below of *CifSieve* (Section 5.3.7.2).

5.3.7.1.2.7. *zincSubset*

zincSubset is another C-shell script which is very short but very useful. It allows a user to generate a custom subset of any ZINC-format file (or CIF) simply by listing the desired data blocks and data names. The script has two file arguments, the first of which specifies a file with regular expressions that specify what is to be included in the subset, and the second of which is the ZINC-format file itself (or standard input). It allows two options: '*-c*' to remove comments and '*-v*' to invert the sense of the search.

It converts the CIF into a ZINC-format file, uses the Unix *grep* program to search through the ZINC-format file for patterns that appear in the regular-expression file and pretty prints the result. For example, the command

```
zincSubset defs cif_core.dic > cifdic.defs
```

will produce a subset of the core CIF dictionary that contains only the names and definitions when the file named 'defs' contains two lines with the TAB-surrounded word '`_name`' and the TAB-surrounded word '`_definition`'.

All the tools listed in this section operate by design in concert with each other, providing the opportunity for generating increasingly complex tools. For example, to generate the Fortran namelist input file with only certain data items, a pipeline of *zincSubset* and *zincNl* will suffice. As more tools are developed, the range of applications will increase many-fold.

```
data_sample
     _title  'scatter graph'
     _description
; A collection of x, y coordinates of points
  drawn in specified colours
;
     loop_
         _x  _y  _colour
          0   0   red
          1   1   red
          2   4   red
          3   9   orange
          4   16  orange
          5   25  orange
      _status complete
```

(*a*)

```
data_sample
_status complete
loop_    _y  _x  _colour
          0   0   red
          1   1   red
          2   2   red
          9   3   orange
          16  4   orange
          25  5   orange
_title  'scatter graph' _description
; A collection of x, y coordinates of points
  drawn in specified colours
;
```

(*b*)

Fig. 5.3.7.2. Two example files in CIF format differing greatly in layout but little in content: (*a*) sample1.cif, (*b*) sample2.cif.

```
% cifdiff sample1.cif sample2.cif
18c18
< sample         _y      2       4
---
> sample         _y      2       2
```

Fig. 5.3.7.3. Output from *cifdiff* comparing files sample1.cif and sample2.cif of Fig. 5.3.7.2. The entries in each line comprise the data-block name, the variable name, the zero-based index of the occurrence of the value in a looped list and the value. Hence it is the *third* value of **_y** in the loop that has changed.