Publ. Astron. Soc. Aust., 1995, 12, 244–7

Making the Figaro Data Reduction System Portable

K. Shortridge[1], S. J. Meatheringham[2], B. D. Carter[3] and M. C. B. Ashley[3]

[1]Anglo-Australian Observatory, PO Box 296, Epping, NSW 2121, Australia
ks@aaoepp.aao.gov.au
[2]Mount Stromlo and Siding Spring Observatories, Australian National University, Private Bag, Weston PO, ACT 2611, Australia
[3]School of Physics, University of New South Wales, Sydney, NSW 2052, Australia

Received 1994 December 6, accepted 1995 February 9

Abstract: The Figaro data reduction system was originally written for DEC VAXes running VMS, and little attention was paid to making it portable. Recently, however, a cooperative effort between the AAO, MSSSO, UNSW, the UK Starlink network and Caltech has resulted in a version for UNIX. This new version has been run under VMS and three different versions of UNIX. The files produced by any version may be read directly by any other version, although Figaro has a particularly complex file format which contains an extensible, self-defining, hierarchical structure of data items. This complexity has allowed the addition of error and quality data, as well as specific structures used, for example, for echelle data. Figaro is written mainly in Fortran (with numerous DEC extensions) but there is also a significant use of C. While C and Fortran are reasonably portable, the way one is called from the other is less portable and needs careful handling. Ports to other systems are possible, with effort; a Macintosh version is being considered.

Keywords: data processing

1. Introduction

Figaro is an extensive set of programs for the reduction and analysis of astronomical spectra and images (Shortridge 1993). Raw data taken at either optical or infrared wavelengths can be reduced to fully calibrated observations, and the software allows considerable flexibility in the manipulation and visualisation of multidimensional data. Work started on Figaro at Caltech about 12 years ago. At that time, astronomical computing was dominated by the almost ubiquitous VAX 11/780 computer. Figaro—which was written to satisfy an immediate need, rather than with any thought that it might still be being used when VAXes were no longer the system of choice in many astronomy departments—was originally written to make as much use as possible of the facilities provided by the VAX and its VMS operating system.

Now, UNIX workstations are the dominant machines for data reduction in astronomical institutions. VAXes are still being used, but the VAX now looks dated. The basic instructions built into a VAX processor are relatively complex, and it is hard to build a very fast processor of such complexity. It is even harder to build a processor that is fast, complex, and also cheap. The UNIX workstations available nowadays have processors that execute very simple instructions, but do so extremely quickly. Complexity is traded for speed when the processor is built, and the complexity is provided later by the Fortran or C compilers, which use large numbers of these simple instructions to build the programs they are asked to compile. The VAX is a CISC (complex instruction set computer) machine in a world that is now dominated by the RISC (reduced instruction set computer) machines.

The VMS operating system that runs on the VAXes has now been ported to one particular RISC architecture, that of the DEC Alpha processor. However, the UNIX operating system, partly because it was written in C and so could be implemented relatively easily for any new processor, and partly because it was not originally a proprietary system, runs on almost all RISC workstations and is now the dominant system for astronomical work.

At the same time, many astronomers now have home computers with processors far more powerful than that of the original VAX 11/780. However, most of these are not used for astronomical data reduction, not because the machines are insufficiently powerful, but because the most common data reduction packages do not run on the operating systems (MS-DOS, MS-Windows, OS/2, Mac OS, etc.) that they use. Most of these home computers

1323-3580/95/020244$05.00

are based on CISC processors (the Intel 80x86, Motorola 680x0, and Intel Pentium), but a few RISC machines are starting to appear for home use (mainly based on the Power PC processor).

The common data reduction systems do not run on these machines for the same reason that systems such as Figaro have taken time to move from VMS to UNIX. Quite simply, it can be very difficult to port code from one system to another. This paper looks at some of the reasons for this, and discusses the way these various problems were addressed in the recent work done to make the Figaro system portable—to remove its dependence on the VAX processor design and its associated VMS operating system.

2. Portability of Code

There are a number of things that can make it difficult to port a program from one system to another. The main ones, at least for a system such as Figaro, are as follows:

- *The instruction set.* If any code makes explicit assumptions about the basic instruction set implemented by the processor, this code will need to be completely rewritten for a new system. Any code written in assembler—written explicitly in terms of the processor instruction set, rather than in a processor-independent language like Fortran or C—obviously comes under this heading, but it is also possible to write Fortran or C that has some implicit assumptions about the details of the processor. (The number of bits used to hold an integer value, for example, influences what range of integer values can be handled.)
- *The operating system.* Programmers often find that they need facilities that are not provided by the programming language they are using. Often these are available through calls to subroutines that are provided by the system and work directly with the operating system. However, these differ from system to system, and code that contains such calls will have to be modified if it is to run on a new system. Different operating systems have quite different conventions for the way files are named. Some will allow multiple versions of files with the same name, others do not. Figaro made great use of the fact that VMS supported 'file mapping', where an operating system call can set up a one-to-one correspondence between the bytes of a file and a range of memory addresses. This allows very efficient file access on systems that support it, but not all systems do.
- *Language extensions.* A lot can be accomplished using absolutely standard Fortran, for example. However, almost all manufacturers provide extensions to the standard languages, and these often make programming so much easier that they are hard to resist. For example, standard Fortran 77 restricts the programmer to variable names of no more than 6 characters, a restriction few programmers heed. The 'DO ... END DO' loop construct is not part of standard Fortran 77, but is used by most programmers nowadays.
- *Inter-language calls.* Most but not all of Figaro is written in Fortran, which is the traditional scientific programming language. There were a few very small routines written in VAX assembler, but most of the non-Fortran Figaro code is in C, which is a better language to use for system-level operations, particularly when the system is UNIX, which is itself written in C. A program written entirely in standard Fortran or standard C will port easily to most machines, but there is no standard way for a Fortran program to call a C routine. All systems allow this to be done, but all do so in different ways.
- *Windowing systems.* Nowadays one expects images to be displayed in resizable windows in multiple colours with scrollbars and various mouse-operated controls. A standard is emerging here, at least for large systems: VAXes and UNIX workstations now all support the X windowing system, although often in different ways. However, X is generally not used by the Macintosh or MS-Windows, so is not supported on home computers.

3. Figaro Itself

The Figaro design, particularly the file format used, was influenced by work done by Starlink, the UK astronomical data analysis network. As a general purpose astronomical data reduction system, Figaro has two particular advantages. It provides a very flexible hierarchical format for data files, making it easy to introduce things like error and data quality information, and it is particularly easy for users to write their own Figaro programs.

However, Figaro was written to run efficiently on a VAX, and as a result was never written to be portable. Indeed, it made enthusiastic use of non-standard facilities provided by VMS. It had some routines in VAX assembler. It used a large number of the non-standard extensions to Fortran provided by the VMS compiler. The Fortran code had a number of calls to VMS operating system routines. Figaro had many cases of Fortran routines calling C, since the hierarchical file format is implemented through Starlink's Hierarchical Data System (HDS) and this is written in C. It was originally written to support a disparate set of image display systems, but not the standard X Window system.

4. Making Figaro Portable

Recently, effort has been put into reworking the Figaro code to remove—or at least to separate—its VAX-dependent elements[1]. Although the initial aim was to get a version that would run under UNIX, the intention was not to turn it into a UNIX system. Rather, it was to turn it into a portable system in the literal sense of a system that was able to be easily ported to a variety of different machines.

The result is a version of the traditional Figaro system that now runs under VMS and a number of versions of UNIX: SunOS, Solaris, and ULTRIX, with an OSF/1 version being planned. Parts of the code have even been ported to an Apple Macintosh, running under the native Mac OS, but this is very experimental and incomplete.

The use of HDS means that Figaro files can be read on any machine. A file written using VMS Figaro on a disk shared with a UNIX system may be read directly by the UNIX version of Figaro on that UNIX machine. Any machine dependences are handled by the HDS routines. A UNIX Figaro file may be transferred over the network to a Macintosh and read by the experimental Macintosh Figaro code.

The porting philosophy adopted was to accept that there had to be different versions of some sections of code for different systems. These were then encapsulated in small routines so that different versions of these routines could be provided for the various systems.

The C language includes a pre-processor that allows the C code to be modified automatically as part of the compilation process, depending on flags supplied to the compiler. This means that alternative blocks of code can be used, depending on the system on which the code is to run. Small changes can be made by defining individual symbols differently on different systems. This is convenient, but can lead to code that is hard to follow, since the source files contain different versions of the same code for different systems.

Fortran does not provide this facility, but a Fortran program can be run through the C pre-processor to get the same effect. We have used this facility, but very sparingly. Generally, we have tried to avoid too much use of the pre-processor facilities, preferring to have different source files for different systems. Where a pre-processor construct is used, we always use a functional test, for example, 'Does this compiler allow files to be opened "readonly"?' rather than 'is this the ULTRIX compiler? If not, is it the SunOS compiler?, etc.' This is easier to read, and does not require changes to the source code when a new system is added.

In many cases, what might have been a significant difficulty has been obviated by commercial pressures. Since VMS was so dominant in scientific computing, other manufacturers were forced to provide many of the facilities of VMS in order to tempt VMS users over to their systems. This is most obvious in the case of the Fortran compilers—although Figaro makes cavalier use of VMS Fortran extensions, almost all of these are supported by the Fortran compilers on the systems to which we have ported Figaro.

The following list of ways in which the code was modified for portability corresponds to the set of porting problems listed earlier:

- *The instruction set.* Routines in VAX assembler were replaced by equivalent C routines, although the more efficient assembler routines can still be used in the VAX version. Very few other instruction-set problems have arisen, although the different ways in which floating point numbers are represented by different machines, and the byte-ordering (the esoteric question of which of the numerous bytes of an integer is the most significant) have had to be handled.
- *The operating system.* Most systems provide the facilities required by Figaro, but in different ways. Operating-system-dependent operations, such as renaming a file, have been removed to separate routines and different versions of these have been provided for the different systems. Most recent versions of UNIX even support 'file mapping', although Starlink's HDS is able to emulate this for systems that do not provide it.
- *Language extensions.* As mentioned above, commercial pressures have almost removed this problem. The 'INCLUDE' statement, commonly used to read the same set of common definitions into a number of source programs, is provided by all systems, despite being non-standard. However, it has to be given a filename, and the syntax of filenames differs from system to system. We have changed all the Figaro 'INCLUDE' statements to the minimal format 'INCLUDE name' where 'name' is a single symbol with no non-alphabetic characters—not even a period. On all systems this can be made to point to the required file to be included: under VMS it is a logical name, under UNIX it is a symbolic link.
- *Inter-language calls.* These are handled by a Starlink package, 'CNF', which provides a set of macro definitions that can be used by the C pre-processor to manipulate the definitions of C

[1] A portable version of Figaro is available from Starlink, but this is a version of Figaro that runs under the ADAM environment, which provides a quite different interface to Figaro and is not strictly a port of the traditional Figaro system. However, a deal of the work done for this system was incorporated into the portable Figaro described in this paper, and the code for a Figaro application is the same in both versions; only the underlying frameworks differ.

routines and the syntax of calls to Fortran routines from C. These macros differ from machine to machine.

- *Windowing systems*. The Figdisp display server written by Sam Southard provides support for image and line graphics displays using X-windows. This runs on UNIX and can run under VMS, although the original VMS display code for use with other display systems is also still available. Caltech's PGPLOT library was used for line graphics, and a portable version of this has been available for some time now.

5. Conclusion

Making Figaro portable was not a trivial task. However, the problem was much simpler than it might have been given the VMS dependences of the original code. This was helped to a large extent by the fact that commercial pressures and the former dominance of VMS encouraged UNIX vendors to put VMS support into their products, particularly the compilers. The use of VAX-specific Fortran, which might have been expected to pose a porting problem, turned out to have been an advantage. The use of common packages such as Starlink's HDS, or Caltech's PGPLOT, which were ported by their originators, also helped significantly.

The latest version of Figaro is available through anonymous ftp from the Anglo-Australian Observatory. Details can be obtained from ks@aaoepp.aao.gov.au.

Acknowledgments

We would like to acknowledge the contribution made to this project by Horst Meyerdierks—based at the Royal Observatory, Edinburgh, under the Starlink aegis—who worked closely with us on the Figaro port and supplied the code modifications he had made for the Starlink ADAM version of Figaro. Sam Southard of Southard Software Systems in California, who had been involved with the earlier Caltech port of Figaro to UNIX (one that had concentrated mainly on making the system run on the SunOS version of UNIX), also provided valuable help, advice and code.

Shortridge, K. 1993, in Astronomical Data Analysis Software and Systems II, ed. R. J. Hanisch, R. J. V. Brissenden & J. Barnes, ASP Conf. Ser., 52, 219