UDC 520.88:519.682 *Astrofiz. Issled. (Izv. SAO)*, 1993, **36**, 132-137

BASIC PRINCIPLES OF FLEXIBLE ASTRONOMICAL DATA PROCESSING SYSTEM IN UNIX ENVIRONMENT

O.V.Verkhodanov, B.L.Erukhimov, M.L.Monosov, V.N.Chernenkov, V.S.Shergin
Special Astrophysical Observatory of the Russian AS,
Nizhnij Arkhyz 357147, Russia

Received March 4, 1992

ABSTRACT. *Methods of construction of a flexible system for astronomical data processing (FADPS) are described. An example of construction of such a FADPS for continuum radiometer data of the RATAN-600 is presented. A Job Control Language of this system is the Job Control Language of OS UNIX. It is shown that using basic commands of the data processing system (DPS) a user, knowing basic principles of job in OS UNIX, can create his own mini-DPS. Examples of such mini-DPSs are presented.*

Описываются методы построения гибкой системы обработки астрономических данных (ГСОАД). Приводится пример построения такой ГСОАД для данных первого облучателя РАТАН-600. Командным языком такой системы является командный язык операционной системы UNIX. Показывается, что пользуясь элементарными командами системы обработки данных (СОД), грамотный пользователь (т.е. знающий базовые принципы работы с ОС UNIX) может создать свою мини-СОД. Приводятся примеры построения таких мини-СОД.

INTRODUCTION

Each data processing system, whatever friendly it is, requires from the user some computer knowledge or training. And it is very often that the more friendly is the system, the higher knowledge is needed, and such systems are rather bulky. These faults are very frequent in systems, considered universal (MIDAS, AIPS, etc). Proce-

dures of start, commands of data editing, and execution of different operations demand knowledge of the basic level of a data processing system. If we bear in mind that each developed astronomical data processing system (ADPS) is "system in system", it is necessary to note then that knowledge of the commands of the operating system, in which ADPS is installed, is implied. The developed ADPS may also have another fault associated with a situation, when the principles of programming and working with images in this ADPS differ from the principles of the operating system which controls this ADPS.

Here we propose another approach of creation of a data processing system. This approach has been realized in the development of an ADPS for continuum radiometers of the RATAN-600 (Erukhimov et al., 1990, Verkhodanov, 1992 and Verkhodanov et al., 1992). We call this approach a module approach, and the system of reduction - a flexible astronomical data processing system (FADPS).

REALIZATION OF MODULE APPROACH

The main principles of work with programs of the data processing system and the rules of managing in the UNIX system are the same in this approach. Here the command-languages of the UNIX (**shell** and **cshell**) are the command languages of the ADPS. These languages have cycle, condition, and local transition operators. The process of data reduction is consecutive application of the operators (commands) to the user's data recorded as equidistant measurements and kept in separate files. Each operator corresponds to the definite command called "brick" or module. Having a set of such "bricks" the ADPS user familiar with the bases of the operating system UNIX can construct his own data processing system, or having typed a sequence of modules, get the necessary result.

Let, for example, some vector **D**, describing observations of a radio source, correspond to the observational data. We have to obtain the approximation curve as a Gaussian inscribed in a record of transit across the beam pattern of the telescope. The standard procedure of getting the result consists in consecutive application of two operators: background subtraction operator F_{bgd} and Gaussian-approximation operator F_{gauss}, i.e. as a result we have to obtain vector **R**:

$$\mathbf{R} = F_{gauss}\,(F_{bgd}(\mathbf{D})).$$

The procedure of application of the two operators will be written in the UNIX system by the following string:

bgd -w 10 < **D** ! **gauss** -n 20 > **R**,

where **bgd** is the program of background subtraction, **gauss** is the program of Gaussian approximation, the key **-w** shows that the next parameter is the size of the window for smoothing, the key **-n** shows that the next parameter is a level lower which there are

no inscribing of gaussians (see Verkhodanov et al., 1992 for more details about these and other programs). Symbols '<', '>' and '|' are the symbols of the job control language (JCL) of the shell of the UNIX meaning accordingly redirection of input (input from file **D**), redirection of output (output into file **R**) and a token of pipeline when the standard output of each command, but the last is connected by a pipe to the standard input of the next command. Naturally, this is valid when the operators serve as filters, i.e. they have only one input and only one output for the vector data. There may be any number of additional inputs for scalar parameters, and they may be set, as shown above, through the key words (e.g. the key -w 10). Besides the filters other program-operators may be used at the input, for example, the program **ravr** for robust summation of a great number of records (see hereafter), and others.

Thus combining sets of different program-operators and using the JCL of the UNIX the user constructs his own command strings to achieve a needed result.

Another example: operation of averaging some vectors of data with preliminary background subtraction:

$$R = \sum_{i}^{n} F_{bgd}\ (D_i)\ ,\ i=1,n.$$

The following set of operators in the UNIX **shell** corresponds to this expression.

```
LIST=`ls D*`
BLIST=""
for i in $LIST
do  bgd -w 10 < $i > $i.B
    BLIST="$BLIST $i.B"
done
ravr $BLIST > R
```

Here **LIST** is the list of the files whose names begin with the letter D (hypothetically **D1**, **D2**, ...,**Dn**), **BLIST** is the list of the files with subtracted background data (names are formed by adding expansion **.B**), operators "**for**", "**in**", "**do**" and "**done**" describe the operator of cycle "**for**" of the language shell, **i** is the variable of the cycle, symbol '**$**' means "take the value of the variable", record LIST=`ls D*` where '`' means "assign to variable **LIST** the result of execution of command "ls D*" which gives the names of all files beginning with the letter D, **ravr** is the program of robust averaging (Erukhimov et al., 1990).

This example shows that programming with the JCL shell is simple enough. And if the user has worked in the UNIX system, then application of FADPS commands is not very hard. Programming (communication with FADPS) in this language is possible in both the interactive mode and using command files. It should be noted that the main procedures (e.g. averaging) are formed as command files or "bricks". So, for averaging the user can apply the program **aver**, and then the averaging command will be re-

corded in one string:

aver $LIST -r -o R,

here the key **-r** is the sign of robust averaging, and the key **-o** shows that the next parameter is the file name with the result of program execution.

A natural desire of every user is to have a great number of basic modules in the system. Although this question can be argued because the large list of command-operators gives rise to some confusion on the one hand, and on the other hand it makes no sense to create separate modules with similar operations which can be set by the input parameters.

In creation of "bricks" it is necessary to take into account some principles of writing such programs. First, if it is possible, the modules must work as filters, i.e., as it is noted above, to read data from the standard input and to record the result into the standard output. Second, program-operators must work with the data recorded in a certain standard format. We used F-format (Verkhodanov et al., 1991; 1993) close to the FITS format (Wells et al., 1981). And third, writing dialogue mode programs, it is important to foresee a possibility of parameter input with keys, i.e. the work of the program in the non-dialogue mode. Keeping to these rules the user or programmer can create his modules and include them into the data processing system. These principles were used in the data reduction for the experiment "COLD-90" (Parijskij et al., 1990) and in simulation of radio images (Verkhodanov and Khajkin, 1990).

A separate question is visualization of data processing results. The principal convenience of the ADPS is chiefly determined by a possibilitiy of visualization of the data and reduction results, and also to work with the data in the interactive graphic mode. In this case, of course, it is impossible to avoid creation of "system in system". The system UNIX is convenient because creation of such systems is facilitated by the FADPS, in this case there is no sense to make subroutines executing certain operations or corresponding to some operators. It is suffice to call and transfer control to some ADPS program. The result can be displayed graphically, i.e., this will result in saving human and computer resources.

In this way, for example, the program **fgr** was created. This program is a data processing system to be used with one-dimensional data of the RATAN-600 continuum radiometers in the graphic mode. This program employs different commands of the FADPS, and the ready result alone is shown on the display. A small fragment of the module scheme is shown in Fig. 1.

The structure of this small fragment shown in the picture is associated with the call of the program-calculator **fcalc** for F-files. The constructed system of menu makes easier for the user a search for the operation needed. Here is shown a transition:

Math (mathematics operations) $\longrightarrow$ **Fcalc** (scalar-vector operations) $\longrightarrow$ **Vector** (vector operations) $\longrightarrow$ needed operation.

Besides some operations unified by the item's name there are also operations of control, e.g. operation of recording the result into the operative memory, onto the hard disk, and operations of exit from the current menu.

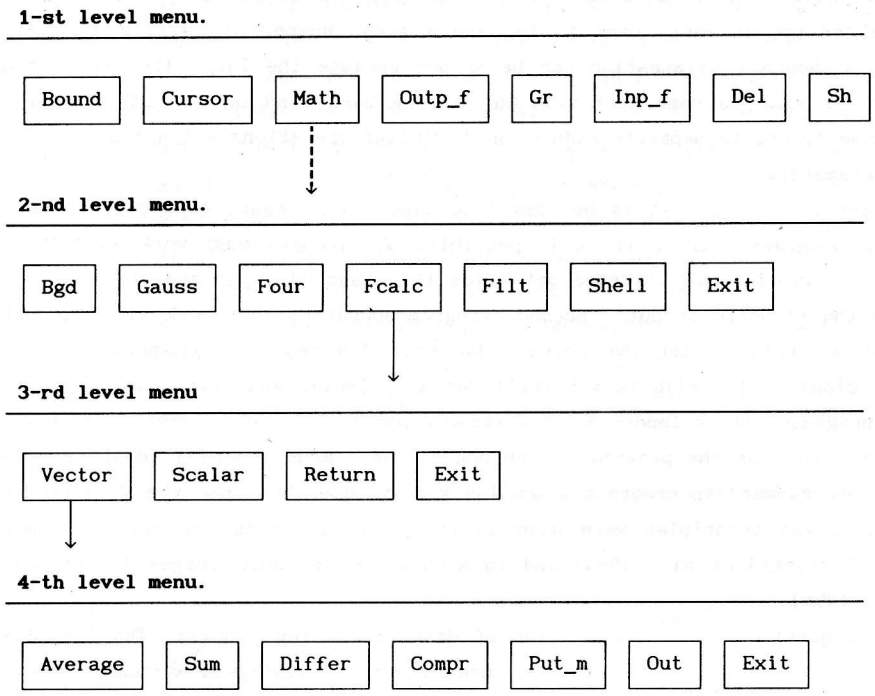

Fig.1. *Fragment of module scheme of the program* **fgr.**

The construction of the program is oriented exclusively on the described module principle of organization of the data processing system. Besides calling corresponding "bricks" such program is due to serve the user (system of different level menu and corresponding prompts) and to visualize a result. There is also a possibility of executing the program-module which is not described as an item of the menu. It can help to execute additional operations over the data: the program **fgr** makes a request for the program name (keys may be) after entry in the item **Filt** (filter) and transfers control to that program. The obtained result will be displayed.

Another question is work with data of different dimensions. Let us take again as an example the FADPS of continuum radiometers data of the RATAN-600 (Verkhodanov et al., 1992a, b). Here are realized possibilties of work with one-dimensional records of transits of radio sources through the beam of the RATAN-600 and work with two-dimensional surveys at this telescope. The optimal solution of this problem (i.e., the input data have different dimensions) is, of course, creation of program "bricks" working with different data types and distinguishing dimensions of these data after reading the headers of files. However, there are some procedures strictly

oriented on the data of defined dimensions, e.g. data visualization. In this case one has to create programs having similar problems, but different solutions because of different dimensions of input data.

In conclusion note that the flexibility of the FADPS is determined by two factors:

1) sufficient number of elementary program-"bricks" used for building the data processing system;

2) flexibility of the operating system UNIX whose command language is a strong "cement" in construction of this building.

Any user can create his own mini-ADPS out of program-"bricks" and JCL of the operating system.

It is important to note that this data processing system allows for new procedures to appear and is open for users and programmers.

We thank Yu.N.Parijskij, A.G.Gubanov, M.G.Mingaliev, N.S.Soboleva for the useful discussion when developing the FADPS for data of continuum radiometers.

REFERENCES

Erukhimov B.L., Vitkovskij V.V., Shergin V.S.: 1990, *Prepr. of SAO USSR AS*, **50**.

Parijskij Yu.N., Erukhimov B.L., Mingaliev M.G., Berlin A.B., Bursov S.N., Nizhelskij N.A., Naugolnaya M.N., Chernenkov V.N., Verkhodanov O.V., Chepurnov A.V., Starobinskij A.A.: 1990, *Proceedings of the NATO Advanced Research Workshop*, 437-442. Durham UK. Kluwer Academic Publishers, 1991.

Verkhodanov O.V, Khaikin V.B.: 1990, *22-nd All Union conference Radio telescopes and interferometers*, Erevan. Theses. 169 .

Verkhodanov O.V., Erukhimov B.L., Zhelenkova O.P, Likhvan P.O, Monosov M.L., Chernenkov V.N., Shergin V.S.: 1991, *Report of SAO USSR AS*, No. 200, 1-33.

Verkhodanov O.V.: 1992, *Preprint SAO RAS*, **75**, 102.

Verkhodanov O.V., Erukhimov B.L., Monosov M.L., Chernenkov V.N., Shergin V.S.: 1992a, *Preprint SAO RAS*, **87**, 38.

Verkhodanov O.V., Erukhimov B.L., Monosov M.L., Chernenkov V.N., Shergin V.S.: 1992b, *Report of SAO RAN*, No. 205, 1-177.

Verkhodanov O.V., Vitkovskij V.V., Erukhimov B.L., Zhelenkova O.P., Likhvan O.P., Monosov M.L., Chernenkov V.N., Shergin V.S.: 1993, *Preprint*, **89**SPb, 13-30.

Wells D.C, Greisen E.W., Harten R.H.: 1981, *Astron. Astrophys. Suppl. Ser.* 44, 363-370.