The F-package for Input/Output

Volker Blobel

II. Institut für Experimentalphysik der Universität Hamburg

The **F-package** is a general stand-alone package for machine-independent input-output of data blocks. It is used for almost all data of the **H1** collaboration's experiment at the *ep* collider **HERA**. Automatic conversion between the different word formats of various computers is performed during the exchange of records via networks or tapes. The record key of F-package records allows by means of a machine-independent interpreter language the fast access to a subset of records and to a subset of data blocks within records. The record key may include assignments to physics classes. Special database-like applications (for example handling of calibration constants) are supported by keyed-access and ordered-access.

Principles

Efficient input/output of data and transfer between different computers is an essential requirement of today's large High-Energy-Physics experiments, where *distributed computing* plays a large role. Based on the long experience with the BOS dynamic memory management system [1] a new Input/Output package [2] had been proposed and realized within the H1 collaboration for the experiment at the *ep* collider HERA. The F-package itself is a general *experiment-independent stand-alone* package, which is easily interfaced to program systems. It has been interfaced to BOS and to LOOK [4], a system used for the event display and general data analysis (histograms etc.) within H1.

in \ out	IEEE	IBM	VAX	DEC
IEEE	-	x	x	x
IBM	x	-	x	x
VAX	x	x	-	x
DEC	x	x	x	-

ALLIANT	IBM RS 6000	MIPS
HP-APOLLO	IBM-MVS	SGI
DEC station	IBM-VM	SUN
HP RISC	MacIntosh	VAX

Figure 1: The matrix of possible word-format conversions (indicated by x), available on each computer. A table of the different machine versions is shown on the right.

One principle, followed in the design of the F-package, is the automatic conversion of the recorded data to the different word formats of different computers. In contrast to other systems, which know *local* and *exchange* files, no such distinction is made in the F-package. Instead, there is the whole matrix of conversion options on all computers, as shown in fig. 1, allowing for example to read a VAX file and to write an IEEE file on an IBM-MVS system. Each physical record contains the full format information needed to convert the record *in-place* by a call to a subroutine; thus the conversion can be done by any component of a distributed system, for example by a general server.

Another principle is the efficient use of disk space and the minimization of the load on networks. The optimal value of the length of physical records can be selected for a given application (whereas there is no limitation on the size of logical records). In general the data analysis in high energy physics requires the frequent access to subsets of data, for example to events of certain physics classes or to certain runs. A solution to the data-selection problem had been realized already several years ago in the framework of the IO-part of BOS, and is successfully used [3]. In addition to the (usually big) data file a short so-called index file (also called event-directory) is introduced, which carries only certain identifiers of the logical record (run and event#, physics classes) and the (physical) record number(s) on the data file. Requested (logical) records are selected from the information on the index file; the record number, available from the index file, is then used to access the (physical) record(s) on the data file. In the case of disk files (or staged files) the Fortran direct-access is used, in the cases of tape files fast skipping to the requested records is done. Disk files with fixed-length records written in sequential mode can be read with direct mode. Experience [3] has shown that reading a subset of records from a large file via index file means almost no overhead w.r.t. CPU time and load on the network compared to reading a file which contains only the selected records. In the F-package the methods applied already in the IO-part of BOS are *generalized.* A support of database-like applications (search, replace, delete records) has been included.

Another principle followed in the design is user-friendliness. Portability of application code is an important aspect. Since (1) there are several small differences in the IO part of different Fortran77 dialects, and (2) there are very many options within the package, an *interpreter language* for the basic IO functions like OPEN, and for the steering of all options has been included in the package, to allow an identical user-interface on different platforms.

The record and file structure

Logical F-package records carry a header and one or more data blocks, each described by a format similar to the Fortran format statement. The record header contains a record key as shown in fig. 2. The elements of the record key, the record name (eight characters) and two record numbers, and the record classification are used in the various selection options. The data part of a logical record consists of a number of data blocks, each identified by a name (eight characters) and a number, and described by a format; the data part of each block may have any length. In a given application the parts of the logical record (record header, block header, data) are transmitted to the system by a set of calls and mapped to the physical records.

Database-like applications. Certain additional operations are provided for database-like applications with F-package files, for example for the handling of calibration data, with the so-called *keyed* and *ordered* access. Reading of records is in the *order of the keys*, either forward or backward (regardless of the order of writing); prior to reading the file can be positioned to any key, allowing search operations in an efficient way. Records are replaced by simply writing the new version to the file. The special operation are

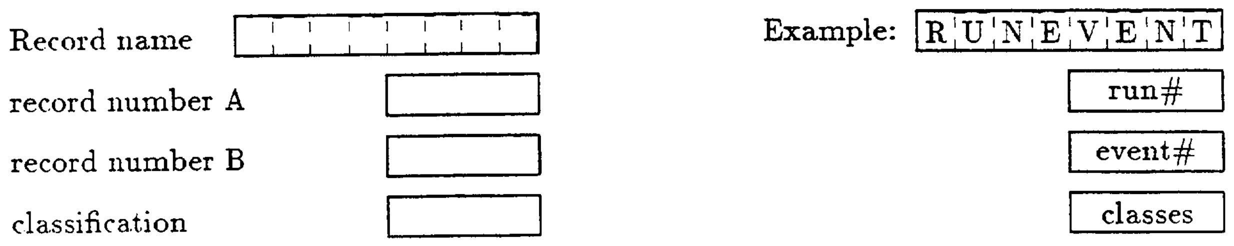

Figure 2: Components of the record key. The assignment used by the H1 collaboration for event data is shown on the right.

realized on the IBM-MVS system with the "keyed access" extension of VS Fortran. The simultaneous write-access, available with this extension, is an important property for the central application in the H1 database (MDB)[5]. In a test version of the F-package the special operations are available with standard Fortran too, simulating the keyed access with Fortran direct access, called "ordered access". A copy of records between files with different access methods is possible without any loss of functionality, since the basic record structure is identical for the different access methods.

Network access. F-package files can be transmitted over networks by standard methods (FTP, NFS). Access to single records on another system is possible with a server (FPserver), which is part of the F-package. The remote file access has been implemented in a standard client-server model. There exist versions of the FPserver for IBM/MVS, for UNIX-workstations including DEC-stations and for Macintosh PC's. The network part of the F-package has been written in C and is based on the Sun Remote Procedure Call (RPC) library with the underlying transmission protocol TCP/IP. The FPserver is running on the DESY IBM mainframe and on some UNIX computers since more than a year.

The interpreter language

The interpreter language is used to steer all options of the F-package in batch or interactive applications. The rules for the source form are identical to Fortran90 with blanks being significant. Statements start with a keyword. They are interpreted and immediately executed. Examples for OPEN statements are:

```
OPEN  INPUT  FILE = "F99ABC.DATA1,F99ABC.DATA9"  HOST=DSYIBM
OPEN  OUTPUT  ACTION=WRITE  FILE = "F99ABC.NEW"  SPLITMB=200
```

In these examples INPUT and OUTPUT are symbolic file names, used also in IO-calling sequences and selections statements. Multi-file input and I/O via a server (keyword HOST) are supported. Other options allow for example staging or the output of files of limited length with automatic generation of file names (keyword SPLITMB defines the space limit in megabytes).

There is a variety of possible *selection* criteria for records; certain criteria can also generate an end-of-data condition for reading. The statement to select records (or more general: to select data) start with the keyword RSELECT, and are followed by the symbolic file name and one or several selection specifications. A simple example with (read) selection of certain records, specified by their record number, is

```
        RSELECT  INPUT  RECORD = 1:3, 90:100, STOPMB = 10.
```

Only records 1 to 3 and 90 to 100 are read from the input file. The end-of-data condition is reached after record 100 or after reaching in total 10 Megabytes (keyword STOPMB). Further selection options make use of the information in the record key. They allow to select records with certain record names and (assuming event data with the assigment by H1 shown in fig. 2) from certain runs or certain events. In the latter case the complete record key is specified; the statement

```
        RSELECT  INPUT  RECKEY = 'RUNEVENT'  3033 17,                  &
                        RECKEY = 'RUNEVENT'  3035 113
```

has the effect, that only the two records (events) with run# 3033, event# 17, and run# 3035, event# 113 are read. A very important option is the selection of events from one or more physics classes, stored within the record key (see fig. 2). For example, the statement

```
        RSELECT  INPUT  CLASS = 'RUNEVENT' 2, 8:10  !  D* candidates
```

means selection of all events from physics classes 2, 8, 9 or 10. Whenever meaningful, different selection criteria can be combined, and are possible for input *and* output. For almost all keywords the selection can also be inverted, for example the keyword NOTRECORD indicates records to be skipped. Another option allows to select data blocks within records, by specifying the names of data blocks to be accepted; the use of this option requires of course, that the receiving program (BOS in the H1 case) allows such a suppression of data blocks. Further keywords are CLOSE, REWIND and FILECOPY. The statement

```
        FILECOPY  INPUT TO OUTPUT
```

copies the records from one file to another, eventually with selections, if these are defined for the files by RSELECT statements.

References

[1] V. Blobel, The **BOS** System, *Dynamic Memory Management*, 2. updated printing, Report DESY R1-88-01 (1988)

[2] V. Blobel, The **F-package** for Input/Output, H1 Software Note 1990; P. Binko, V. Blobel, Z. Szkutnik, H1 Software Note 1992.

[3] M. Delfino et al., *Rapid access to Event Subsamples in Large Disk Files Through Random-Access Techniques*, Computing in High Energy Physics '91, Proceedings, Universal Academy Press, Tokyo (1991) p. 353-357

[4] V. Blobel, **LOOK**, *a system for Data Analysis*, H1 Software Note (1990, 1991, 1992)

[5] L. Criegee, **MDB**, *Bank handling in the H1 data base*, H1 Software Note (1992)