DESY DV 90-01
January 1990

FIRST EXPERIENCE WITH PL/I IN A UNIX ENVIRONMENT

SEAS AM89 (PL/I Project - Group)

26th September 1989

E. Bassler

DESY - R02 -
Notkestr. 85
2000 Hamburg 52

ABSTRACT

An interactive graphic program for the analysis of high energy physics data, originally written in PL/I for an IBM, has been ported to the VAX and APOLLO workstation. This paper describes the experiences made when transferring a program of 100000 lines of code, as well as some differences in the computers of IBM , DEC/VAX and APOLLO workstations and especially in their PL/I-compilers.

CONTENTS

INTRODUCTION

DESY is an institute for research in high energy physics with about 800 scientists and students from about 80 German and foreign laboratories. At present a new 6.3 km long starge ring, called HERA, is being constructed, as well as 2 huge detectors having a volume of about 2000 m**3 and weighting 3000 tons each. These detectors will record more than 3 Tbytes of data each year, which have to be analysed using histogram and graphic program packages. One of these packages is GEP (Graphical Editor Program), which was written at DESY on the IBM-computer. GEP was used in 1988 at DESY on the IBM computer in more than 165000 batch jobs and 81000 interactive sessions. For HERA-times it is planned that the large amounts of data are to be analysed at the DESY computer center and in the home institutes. Therefore there is a strong requirement for common portable software. When I started to write the GEP program in 1980, portablity was not so important. We had smaller experiments and collaborations, 1 IBM 3033 and 2 IBM 168 computers for the data analysis and some mini-computers as DEC/VAX-computers. These were used as online computers or as so called 'number crunchers' to reduce the data rate to be transferred via a fast online link (50 - 100 kbytes/sec) from the experiments to the DESY computer center. On the DESY site at present, we have 1 IBM 3090/180E computer and 1 IBM 3084Q computer with the MVS/XA operating system, these will be upgraded before HERA starts running. We also have more than 500 alphanumeric and 100 graphical terminals online and remote terminals at other institutes, in addition to roughly 40 IBM or IBM compatible PCs, 40 DEC/VAXes from 8700 to VAX workstations with VMS, 15 APOLLO and 10 SUN workstations with UNIX. Thus there is interest to get the GEP-package, which was written on and for the IBM-computer, running on DEC/VAX-computers and UNIX workstations.

GEP-PACKAGE

The GEP-package has two parts, a batch part and an interactive part. In a batch job the analysed data are printed as line printer graphics and optionally a data base is generated for further interactive data manipulation. The batch part is written in FORTRAN (30000 lines) and the interactive part in PL/I (100000 lines). Both parts have already been transferred to VAX-computers in 1985/1986.

The interactive GEP part offers many options.

- various graphic presentations
- 3-dim. presentation with hidden line technique
- synopsis of complete pictures
- hardcopies of the produced pictures
- complete pictures can be embedded into documents generated by SCRIPT/VS (Document Composition Facility) or TeX
- graphical input (e.g. for text offsets)
- user defined line types or symbols
- interactive fitting by user supplied functions
- other numerical manipulations, such as arithmetics with histograms bin by bin
- affine transformations of polygons and texts
- interactive HELP option

There are two modes for the interactive GEP module: a 1-terminal-mode using the graphical terminal for both alphanumeric and graphic I/O, and a so-called 2-terminal-mode with an alphanumeric terminal for full screen dialog and a graphic terminal for graphic I/O, or 2 different windows on a workstation.

The total interactive module has a size of 1.8 Mbytes (GKS-Version 2.5 Mbytes) with about 200 kbytes of additional workspace. It is organized into about 350 subroutines and consists of 100000 lines of program code. 2000 lines commonly used as structures or texts are stored in 120 text-macros, which are inserted into the program before compilation by the PL/I-preprocessor INCLUDE-statements.

E.g.: On the IBM the text-macro for the compiler option is 'REORDER;'. On the VAX and on the APOLLO this option is not available and the text-macro is replaced by ';'.

There are some subroutines of the IBM-version written in ASSEMBLER, namely the subroutines for the dynamic allocation, the full screen I/O and the I/O of the data base, which is partitioned organized. Unfortunately these subroutines are highly coupled with the GEP program and we had to build up the calling sequences to VAX and UNIX service routines.

A further problem to be solved was the interface to the graphic terminal. On the IBM we use the AGF-Plotfile, which is a pre-GKS standard of the Arbeitsgemeinschaft der Großforschungsanstalten (AGF). For the VAX we originally used a simple interface from the AGF-Plotfile to the PLOT10 package of Textronix (written in FORTRAN). One AGF-Plotfile-routine, which dynamically loads the calligraphic fonts, was written in ASSEMBLER and we had to replace it by a PL/I-procedure, which reads the font coordinates.

FIRST GENERAL VIEW

I do not know whether the LPI-PL/I compiler has a common root with the VAX-PL/I-compiler, which is based orginally on the PL/I-compiler of R. Freiburghouse. His PL/I compiler has also been successfully adapted by Wang, Data General, Prime, Control Data Corporation, Honeywell-Bull, CII, and Stratus Computer Inc.. Some of these companies have disappeared, but PL/I has survived.

For porting to APOLLO workstations, it was helpful to already have the PL/I code portable for VAXes, and so I shall also report on the portation to the VAX. To be portable to APOLLO workstations, it is reasonable to use a subset of the IBM and VAX PL/I and it is, of course, better to know this subset before. It is easier to write with the restrictions than to remove them. But this was not our situation.

Before we transferred the program in 1985 to the VAX, we studied two old papers of Walter Mueller from the Gesellschaft für Schwerionenforschung (GSI), 'PL/I comparison: VAX-V2.0 <--> IBM-Release 3.1' and 'Common DEC - IBM PL/I Standard', to estimate the extent of the necessary modifications (now VAX V3.2 - IBM V2R2). We found many discrepancies between the compilers, but only a few were important for our program, which unfortunately happens frequently beginning with different default values. So we decided to initially use separate program source management, keeping the source easily transferable. Later we decided for for a common source management using preprocessor macros.

For the APOLLO workstation we used the PL/I compiler from the Language Processors, Inc., LPI, in Framingham MA (USA). This company offers compilers for BASIC, C, COBOL, FORTRAN, PASCAL, PL/I, and RPG II and a debugger named CodeWatch, for microprocessors such as M680X0, Intel 80386, and WE32XXX under versions of UNIX(tm), XENIX(tm), and DOS. LPI products will also be available for M88000 and Sun SPARC(tm)-RISC-based processors. Their PL/I is a full implementation of ANSI PL/I X3.74-1981 General Purpose Subset with extentions with 3 optimizations levels. One of their customers is a division of General Electric, GE Solid State, who transferred more than 200000 lines of PL/I code of their chip testing program MIMIC (Module Imitating Modern Integrated Circuits) and 25000 lines PL/I

code of their R-CAP program (RCA Circuit Analysis Program) to run on SUN-3 and APOLLO DN3000 workstations under UNIX. - By the way, the company had a similar portability problem. They decided against using a language translater to convert PL/I source into C source code because of the risk in reprogramming an existing application and also because of the large required time factor.

For DESY we bought the Version 03.02.00 of LPI-PL/I for APOLLO AEGIS/UNIX System V Rel.9.5 in Sept. 1987, but the plan of porting to UNIX workstations has been frozen for some time. Meanwhile there have been 2 or 3 updates for the compiler available and the latest release is ordered. So it is possible that there are improvements in the LPI-PL/I compiler, which I can not report on. E.g. the removal of the following wicked compiler error: we have to compile PROCs with further ENTRYs with optimization level 2 instead of 3 (dead code elimination) to get the ENTRY-externals resolved.

SOME GENERAL DIFFERENCES

BYTE SWAPPING

On the IBM and APOLLO the bytes are counted from left to right, on the VAX from right to left. However the bits are counted from left to right on the IBM and from right to left on the VAX and APOLLO. Thus there is a difference in the half word addressing, if e.g. full words and half words are overlaid. This is valid for FORTRAN, too. The first half word on the IBM is the first on the APOLLO, but the second on the VAX. In our program there are only a few of these overlays and we have solved this problem of storage and read-out by using a special (machine dependent) subroutine.

In a similar way it is valid for the PL/I-function UNSPEC with bit strings or integer numbers. The result is different.

```
DCL B_PAT BIT(16) ALIGNED;
DCL I_PAT BIN FIXED(15);
I_PAT=1;B_PAT=UNSPEC(I_PAT);
VAX-result         : '1000000000000000'B
IBM-,APOLLO-result: '0000000000000001'B
```

FLOATING POINT NUMBERS

The internal representation of floating point numbers is different. If you copy data sets from one computer to another one, one has to do more than copy the bit-stream. One has to transform each floating point number into the right floating point representation for the target machine. It is similar for the character representation.

EBCDIC - ASCII

On the IBM main frame, EBCDIC is used for characters, while on the VAX and APOLLO ASCII is used. The collating-sequence is different.

```
EBCDIC-sequence: $ , a - z, A - Z, 0 - 9
ASCII-sequence:  $ , 0 - 9, A - Z, a - z
```

In the old version of our program, we had introduced a hidden EBCDIC-dependence. We had overlaid characters and positive integer in the same stored word. Using EBCDIC, there is a simple test for the program logic. If the integer is negative, it is interpreted as a character string. This is no longer valid in ASCII. Fortunately we could circumvent this problem in most of the cases by using other program dependent tests. The reminder was reduced to 10 special character strings, which have to be tested.

We use the EBCDI-Code of characters as an index into vector tables for our calligraphic fonts (Hershey - fonts). To keep these tables and the definitions for special symbols unique, we have to translate each character on the VAX and APOLLO from ASCII to EBCDIC before it can be graphically drawn. We put this translation into the special symbol translation procedure, which is always called before drawing.

SYSLIB CONCATENATION

For the linkage of the executable module, there is a similar concatenation of libraries on the VAX, but the search algorithm on the VAX is different. On the IBM the search for unresolved references always begins with the first SYSLIB-library, while the VAX starts the search with the library where the unresolved references were found, and continues all the following libraries. A reference in the first library, which is referred in the second one, will not be resolved. If several libraries are used, the library organisation on the IBM has to be changed to keep a one to one correspondence with the modules in the libraries of the VAX. Under UNIX the linkage is again controlled in a different way.

DYNAMIC ALLOCATION AND LIBRARIES

On the VAX and APOLLO there are different dataset managements. So the subroutines for the dynamic allocation and the partitioned organized I/O had to be written using DEC/VAX or UNIX system subroutines. The VAX-expert decided for user-specified libraries. The UNIX-expert decided against the archive-file organization. He has built up the library-structure on the hierarchical file system.

Most of the PL/I-ENVIROMENT attributes are different on the VAX. For the APOLLO there are no ENVIRONMENT options for the OPEN statement.

It is important that on VAX and APOLLO there are common run-time libraries for all languages. That means an executable module can be executed on VAXes with VMS and APOLLOs of the M680X0 series without a PL/I compiler.

FULL SCREEN I/O

We use full screen I/O for the alphanumeric I/O dialog and have designed our own full-screen package, which is highly coupled with the GEP program. We can interface the calling sequences to the alphanumeric functions of GDDM by a small package of about 500 lines of code. For the VAX we use DEC/VAX system subroutines (Screen Management Facility SMF of VAX/VMS). For UNIX the problems are under study.

PL/I DIFFERENCES IBM - NONIBM

CHARACTER SET

There are some restrictions in the character set. '@' and '#' are not allowed in any names of PL/I on the VAX and APOLLO. The '$' in external references is reserved for VAX system references.

DEFAULT VALUES

The data-type defaults for the DECLARE statement are different:

IBM-default (FORTRAN-rule):

(A-H,O-Z)	DEC FLOAT(6)
(I-N)	BIN FIXED(15)

FIXED	-->	DECIMAL(5,0)
FLOAT	-->	DECIMAL(6)
BINARY	-->	FLOAT(21)
DECIMAL	-->	FLOAT(6)

VAX-default:
BIN FIXED(31)

FIXED	-->	BINARY(31,0)
FLOAT	-->	BINARY(24)
BINARY	-->	FIXED(31,0)
DECIMAL	-->	FIXED(10,0)

APOLLO-default:
BIN FIXED(15)

FIXED	-->	BINARY(15,0)
FLOAT	-->	BINARY(23)
BINARY	-->	FIXED(31,0)
DECIMAL	-->	FIXED(5,-1)

On the APOLLO the compiler option '-longint' changes the default precision for FIXED BINARY from (15) to (31).

To get a common standard, we had first declared all used variables and changed our FLOAT(6) declarations into DEC FLOAT (for the VAX in 1985). But then we found differences. DEC FLOAT and BIN FLOAT are equivalent on IBM and VAX, but not on the APOLLO. If DEC FLOAT is used on the APOLLO the mantisse of floating point number is 1 digit smaller and the exponent has a range up to E+308. The range of single floating point numbers is from 0.29*E-38 to 1.7*E+38 on the VAX and APOLLO, if FLOAT BINARY is used.

Different from the LPI-PL/I-manual, the size of DEC FLOAT is 12 bytes long instead of 4 bytes long. This is especially important if passing floating point numbers to routines of another language e.g. GKS-routines, which are written in FORTRAN. There is a second difference between BIN FLOAT and DEC FLOAT. With BIN FLOAT, up to 32k bytes can be dynamically allocated, with DEC FLOAT up to 32k 12byte-words. The double precision is different. We have to use BIN FLOAT(52). DEC FLOAT(16) is interpreted on the VAX as quadro precision and for LPI-PL/I BIN FLOAT(53) generates an error.

PASSING ARGUMENTS OF PROCEDURES

PL/I procedures

By default, on the VAX and APOLLO, PL/I passes all arguments, except character strings and arrays with non-constant extents, by reference. The attributes of the arguments always have to be specified in the parameter descriptor list of the ENTRY declaration and must not be omitted or indicated by an asterisk. On the IBM we had used this options for structures and for entries of ASSEMBLER subroutines.

Non-PL/I procedures

All arguments of the procedures can be passed on the VAX or APOLLO to non-PL/I procedures either by immediate value, by reference or by descriptor. There is no OPTIONS(ASM,INTER) and no OPTIONS(FORTRAN) but an OPTIONS(VARIABLE) for a variable number of arguments. We had to replace the corresponding text-macros in the ENTRY declarations.

COMPILER RESTRICTIONS ON THE VAX OR APOLLO

The following list is not complete, but gives the differences, which often occur in our program.

Multi label

Multi labels simply occur after removal of some program text.

```
A:
B:
```

They are not allowed and have to be separated by a semicolon.

Multi assignment

Multi assignments are not allowed.

```
A,B = 0;
```

On the IBM this gives a better code.

Non-connected arrays and cross-sections

Cross-sections are not allowed in assignment statements or in declarations.

```
A(*,1) = 0;
```

In the array assignment, the array has to be connected.

```
DCL 1 A(10),
      2 B DEC FLOAT,
      2 C DEC FLOAT;
C = 0.0; /* not allowed on the VAX or APOLLO,
            has to be replaced by a DO-loop  */
```

DO-lists

There are restrictions for DO-lists. The following form is not allowed.

```
DO I = 2 , 5 TO 7;
```

Parameterless functions

Parameterless functions require a pair of parenthesis on the VAX.

```
DCL P PTR;
    P = NULL();
```

The LPI-PL/I compiler offers a SETNULL option, where the return value of the NULL-pointer can be set.

Label-arrays

Label arrays have to be declared only on the IBM

```
 DCL L(2) LABEL;  /* must not be declared on the VAX and APOLLO */
     statements
L(1):
     statements
L(2):
     statements
```

Terminal I/O

We get our terminal input in the line mode by

```
DCL ICHAR CHAR(79);
        GET EDIT(ICHAR)(A(79));
```

For the APOLLO we had to change the format list of the EDIT statement into

```
DCL ICHAR CHAR(79);
        GET EDIT(ICHAR)(A);
```

Otherwise you have to enter 79 characters. Fortunately we control our terminal I/O in one single procedure and so we had to make this change only once.

On-conditions

For some ON-conditions the abbreviations as UFL are not supported and have to be replaced by the full specification as UNDERFLOW. The ON CONVERSION has to be replaced by ON ERROR on the APOLLO.

Further restrictions for the APOLLO.

For automatic variables there are further restrictions: e.g. The INITIAL attribute is only possible for STATIC variables. The range of automatic variables is restricted to 32K, for larger arrays STATIC variables have to be declared.

We had some trouble with PICTURE'ed variables, but I want to note that we do not have the latest version of the LPI-compiler.

The REFER option for self-defining structures is not supported by LPI. Fortunately, we can circumvent this problem in our program.

There is a compiler option '-lowcase', which converts all uppercase names of internal and external variables and constants to lowercase.

The filenames in the TITLE-option are case-sensitive.

Some nested statements cannot be compiled, if the length of a CHARACTER VARYING is unknown at compile-time.

```
    DCL (X,S) CHAR(20) VAR;
    X=TRIM(TRIM(S));
```

This has to be resolved into separate statements

```
    DCL (X,Y,S) CHAR(20) VAR;
    X=TRIM(S);
    Y=TRIM(X);
```

Because of the fact that IBM does not support the nice TRIM-function, we have no statements of this kind.

IMPLEMENTATION

For the implementation on the VAX, I was supported by a physicist, H. Zobernig, who is system manager of the DEC/VAX-computer. We started in April 1985 and had the 1-terminal-mode ready to test in October 1985. We did not work full-time, and estimate the consumed time to be 3 months. Much time was spent in programming the I/O package on the VAX. Much time was also spent in the replacement of the default declarations. All variables are now declared. Most of the multi-assignments, DO-lists and cross-sections could be found by a simple FIND-operation of '=' and ',' on the total source text, but hade to be individually replaced for any multi-assignment that was not context-free. (e.g. IF I = 0 THEN A,B = 0.0;). Most of the program source modifications were done on the IBM. The multi-label errors were removed on the VAX.

At first we decided to keep separate program source on the IBM and VAX, so it was easy to change. E.g. The final modifications of 30000 lines of code for the full-screen-mode for the VAX had taken two days on the IBM and one additional day on the VAX to get it free of PL/I-errors. The total implementation, including the full-screen-package, took 2 months in spring 1986. But then we ported the VAX source code back to the IBM for most of the routines controlled by INCLUDE preprocessor statements. This code was used for the APOLLO version. A summer-student untrained in UNIX started work on the APOLLO version in the middle of July with part-time support of a semi-experienced UNIX-expert for our special I/O-package. We underestimated the work on the I/O package with respect to the time-schedule. We do have a proto-type ready, which will be released for β-test when the new LPI-PL/I Release for APOLLO-UNIX Rel 10.1 becomes available.

For the future development we intend to migrate to keyed-access I/O using VSAM on the IBM, indexed-sequential files on the VAX, and C-ISAM(tm) of INFORMIX for UNIX. Personally I will continue in the SAA-PL/I-line.

CONCLUSION

There is a PL/I compiler available in the UNIX environment. It covers the subset G with extensions. This is a subset of the OS-PL/I of IBM and DEC/VAX PL/I. If one wants to use this compiler, it has to be studied first, whether the range of the language is large enough. This was the case in our application and we have a proto-type of our PL/I module under a UNIX-operating-system.