Published: 02/21/69

Identification

The pl1 Command
James D. Mills

Purpose

The pl1 command invokes the pl1 compiler to translate a segment containing pl1 source code into text, link, and symbol segments. A listing segment may also be produced. These results are placed in the user's current working directory.

References

The compiler is described in MSPM BZ.8. The language is defined in Form Y33-6003-0, PL/1 Language Specifications, IBM Corporation, March, 1968. Deviations from the IBM specifications are given below under "Language".

Usage

The command

```
pl1  pathname  -opt1- . . . -optn-
```

invokes the pl1 compiler to translate a pl1 source segment identified by pathname. (The typed command consists of the letters "pl" and the numeral "1".) A directory path name and on entry name, segname, are derived from pathname by calling entryarg (Ref. BY.2.04) and the compiler takes its input from segname.pl1. Opt1, . . ., optn are optional arguments to the compiler whose interpretation is defined below under "Options".

A normal compilation will produce these segments and leave them in the user's working directory with ring brackets 1, 48, 48.

segname

- The compiled text segment. Its access modes will be set to read and execute.

segname.link

- The link segment corresponding to segname. Its access modes will be read, write, and execute.

segname.symbol

- The symbol segment for segname. This will consist of a segment symbol table header and binding information for segname, segname.link, and segname.symbol. The access mode of segname.symbol will be read.

The user's options will control the absence or presence of the listing segment for segname.pl1 and the contents of that listing. If created, the listing segment is named segname.list and has the read access mode.

Provided the compilation is successful, previous copies of segname, segname.link, segname.symbol, and (if the list option is on) segname.list are replaced by the new segments created by the compilation.

Note that because of the Multics standard which restricts segment names to be not greater than 32 characters in length, a pl1 source segment name may not be longer than 25 characters. Otherwise, truncation would occur when concatenating segname with ".symbol".

The pl1 command will look for the presence of "%;" as the first two characters in segname.pl1. The presence of such characters implies that segname.pl1 contains "% include" compile-time statements. Pl1, therefore, calls expand_seg to create a new source segment with all "% include" statements expanded. The compiler then takes its input from the expanded results, and the segment segname.expand will be left in the working directory.

Command Options

In the absence of the full Multics option machinery, character string arguments to the command provide the user with a certain amount of control over the output from pl1. The options are summarized here. Further information is contained under "Error Diagnostics" and "Listing".

Option	Result
"list"	Pl1 produces a printable ascii listing. The default is no listing.
"symbols"	If a listing is created it will contain a list of all the variables declared in the program with their attributes. The default is no symbols.
"assembly_list"	If a listing is produced it will contain an assembly-like listing of the text, link, and symbol segments that were compiled. The default is no assembly listing.
"brief"	Error messages written into the stream "user_output" will contain only an error number, statement identification, and when appropriate the identifier or constant in error. In the normal, non-brief mode an explanatory message of one or more sentences will also be written.
"severityi"	Error messages whose severity is less than i (where i is 1, 2, 3, or 4, eg. severity3) will not be written into "user_output" although all errors will be written into the listing. The default is 1.

Error Diagnostics

The pl1 compiler can diagnose and issue messages for about 350 different errors. These messages are graded in severity as follows:

Severity Level	Meaning
1	Warning only - compilation continues without ill effect.

Severity Level	Meaning
2	Correctable error - the compiler remedies the situation and continues probably without ill effect. For example, too few end statements can be and is corrected by simulating the appending of a sufficient number of strings ";end;" to the source to complete the program. This does not guarantee the right results however.
3	An uncorrectable but recoverable error. That is, the program is definitely in error and cannot be corrected but the compiler can and does continue executing up to the point just before code is generated. Thus, any further errors will be diagnosed.
4	An unrecoverable error. The compiler cannot continue beyond this error. The message is printed and then control is returned to the pl1 command unwinding the compiler. The command writes an abort message into "user_output" and returns to its caller.

Error messages are written into the stream "user_output" as they occur. Thus, a user at his console can quit the compilation process immediately when he sees something is amiss. As indicated above, the user can set the severity level so that he is not bothered by minor error messages. He can also set the brief option so that the message is shorter.

An example of an error message in its long from is:

ERROR 281.1 IN STATEMENT 1 ENDING ON LINE 17

The entry name `zilch' has been declared internal but has not been defined within the block of declaration.

If the brief option had been set the user would see instead:

```
ERROR 281.1 IN STATEMENT 1 ENDING ON LINE 17
zilch
```

In the second case the user could look up error number 281 in his handy error dictionary and get the full message. The digit after the decimal point in the error number is the severity of the message. Thus, if the user had set his severity level to 2 he would have seen no message at all.

If the listing option is on, the error messages are also written into the listing segment. They appear, sorted by line number, after the listing of the source program. Because of an implementation restriction no more than 100 error messages will be printed in the listing.

Listing

The listing created by pl1 is a line numbered image of the possibly expanded source segment. This is followed by a table of all of the variables declared within the program. The variables are categorized by declaration type which are:

1. Declare Statement
2. Explicit Context (labels, entries, and parameters)
3. Implicit and Context

Within these categories the symbols are sorted alphabetically and then listed with their location; storage class; data type; size, precision, or level; and attributes such as "initial", "array", "abnormal", "internal", "external", "aligned", "unaligned", and "irreducible". The symbols are followed by the error messages.

Finally, the listing contains the assembly-like listing of the text, link, and symbol segments produced. The executable instructions are grouped under an identifying header indicating the source statement which produced the instructions. Opcode, base-register, and modifier mnemonics are printed along-side the octal instruction. The addresses are numerical but if an identifier or constant corresponds to the address it is printed in the "remarks"

field of the line. Constants and links are printed with symbolic interpretation also.

In general, the assembly listing resembles the format of that produced by EPLBSA with the exception that the addresses of the generated machine words are ordered sequentially from location 0.

Language

The basic language specification is Y33-6003-0 with certain restrictions, a few extensions and the definitions of areas of the language which PL/1 leaves to the implementer.

Language Restrictions

The following features of the PL/1 language are not implemented in the first version of the Multics pl1 compiler.

1. All input/output features including all related statements, declarations, on conditions and built-in functions.
2. Sterling data and pictured data.
3. Tasking - all related options, declarations, on conditions and built-in functions.
4. Scaled fixed point arithmetic.
5. Complex arithmetic.
6. Precision controlled aritmetic.
7. Decimal arithmetic is implemented as binary with the appropriate conversion of the declared precision.
8. Controlled storage class.
9. The attributes: defined, position, like, cell, generic.
10. Conversion between character string and arithmetic and between arithmetic and character string.
11. Aggregate expressions and array cross sections.
12. Check and size condition prefix.
13. Some built-in functions are omitted but the EPL subset is available.

14. Division of fixed point values must be done using the divide function.

15. Prologue dependencies are not resolved. Values available upon entry to a block do not include values declared as automatic in that block.

16. All compile-time statements except % include.

Language Extensions

The following features of the Multics pl1 compiler are, with respect to the IBM PL/1 language definition, extralegal.

1. The built-in functions inherited from EPL:

   STAC REL BASENO

   PTR ADDREL BASEPTR

2. The built-in functions NULL and NULLO may be used as values for the initial attribute.

3. The built-in function STRING may be used as a pseudo variable.

4. Asterisk (*) may be used as an extent for return attributes written on procedure or entry statements.

Implementation Definitions

The following features are aspects of PL/1 left to the implementer to define.

1. The RENAME and VALIDATE options are implemented as options in a procedure statement.

2. The ASCII character set is used. All keywords and special letters use lower-case letters. Otherwise, upper and lower-case both may be used and no mapping is done.

3. The dollar-sign convention of EPL is observed. That is, a reference to an external variable of the form a$b implies location b within segment a.

An external entry a is understood to be a$a but an external static variable v is understood to be stat_$v (unless stat_has been renamed by the rename option). Finally, seg$ implies location 0 in segment seg.

4. The ASCII collating sequence is used.

5. The double quote (") is used for string constants.

6. The defaults and maxima are:

Type	Defaults	Maximum
Float Binary	27	63
Float Decimal*	8	18
Fixed Binary	17	71
Fixed Decimal*	5	21
Bit String	--	36 * 2 ** 16
Character String	--	4 * 2 ** 16

*Decimal is implemented as binary. Declared precisions are converted to binary precisions according to the rules of the language.

Language Compatibility

The issue of compatibility between this implementation of PL/1 and the EPL implementation is too extensive to be covered in this document. It is considered in a repository document by R. Freiburghouse, Compatibility Considerations of the PL/1 Implementation, G0081, 20 January 1969.