AAEC/E251

AAEC/E251

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

AESYNTAX - A FORTRAN SYNTAX ANALYSIS SYSTEM FOR THE PDP9L

by

J.M. BARRY

December 1972

ISBN 0 642 99522 2

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AESYNTAX

A FORTRAN SYNTAX ANALYSIS SYSTEM FOR THE PDP9L

by

J. M. BARRY

ABSTRACT

A facility is presented to assist Fortran programmers in the development of new programs by the immediate listing and syntax analysis of Fortran coded source decks independently of the central computer.

National Library of Australia card number and ISBN 0 642 99522 2

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA–INIS–12 (INIS: Manual for Indexing) and IAEA–INIS–13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

A CODES; FORTRAN; IBM COMPUTERS; PDP COMPUTERS

CONTENTS

1. INTRODUCTION

Users of Australian Atomic Energy Commission computing systems have in the past been assisted in the preparation and correction of programs and data by the ready availability of off-line equipment capable of producing printed listings from card decks. The introduction of the IBM360 computer saw a new character set (EBCDIC) which differed from the BCD system used by the IBM704 listing facility. To restore this most useful service an additional card reader and line printer have been connected to the 8192 word PDP9L (DEC 1968) whose prime function is the control of communications of the AAEC computer network with the IBM360 (Richardson 1968). With the reader and printer under computer control it is now possible to offer additional options including Fortran syntax analysis as well as an extended listing service.

Besides the control of network communications with the IBM360 the PDP9L controls one teletype, two seven track magnetic tape units, a Calcomp incremental plotter, and a fast paper tape punch. The syntax analyser must share the PDP9L with the link monitor, the network handler, and with programs that control the paper tape punch and plotter. The communication between the link monitor and each program that may be in core at the same time is described by Richardson (1971). As a result of core sharing the syntax analyser is currently limited to 3584 words of storage. When work commenced on the analyser no specified amount of core was set aside for its use and future expansion of the network may further limit the core available to the analyser. In keeping with this dynamic storage constraint the analyser was developed in a modular fashion, and future expansion to include more features or enlarge table storage space is possible, as well as the deletion of existing sections should the network expansion require it.

The limited core available meant that a full analysis of IBM specified Fortran IV (IBM 1971) would not be possible. The approach followed has been to try to test thoroughly those areas of Fortran language usage that are prone to errors, and in frequent use by the majority of AAEC Fortran users, while ignoring those sections that are not commonly employed (some analysis is necessary therefore on each statement even if only to decide that it falls into a class not to be analysed further). Between the two extremes there are a number of statements on which a partial analysis is performed. For such statements the occurrence of a definite error will be indicated as such while in general a doubtful piece of code will be permitted to pass. Experimentation while the modules were being developed showed this to be preferable in general (the one exception is the arithmetic statement function 4.4) to the

generation of messages warning of possible errors or that a section of the language is not checked.

Errors detected result in one of 63 causes being isolated. The line in error is printed with a $ sign indicating the column where the syntax violation is thought to occur, and a numbered message is printed to the right of column 73.

2. CONSTRUCTION OF THE ANALYSER

The construction and testing of the analyser on a modular basis by the use of randomly selected Fortran users' checks also provided the opportunity to isolate those areas of the language that AAEC users most frequently employed. It was immediately clear that it was more beneficial to the majority of users to test for a language based on IBM Fortran IV rather than USA standard Fortran. This language has been easier to implement on the restricted environment of the PDP9L since mixed mode arithmetic is not tested for and storage is not required for a symbol table to distinguish between non-subscripted integer and real variables (Restriction 4.5). The survey of users' programs also revealed that most programmers wrote very simple code, avoiding many features available in the Fortran language. LOGICAL and COMPLEX variables were used so infrequently that neither are considered by the analyser.

Each program is processed in a one pass fashion, since no secondary storage is available for the analyser to store partially processed statements (the two magnetic tape units are reserved as a buffered well for the holding of plotting and punching information transmitted by the IBM360). Within each statement processing is also sequential with only one card being processed before it is despatched to a printing routine, and a look-ahead procedure is used to determine the presence of a continuation indicator in the next card image in the input buffer. Input/output transfers take place through buffers where the characters are packed three to each 18 bit word. The analyser initiates the reading and writing of cards leaving the interrupt handler to complete the data transfers.

The syntax analyser allows the user to list ordinary data through the selection of an appropriate control card. There are 6 control cards to which the analyser responds:

(i) $COMPILE

(ii) $LIST1

(iii) $LIST2

(iv) $STOP

(v) $CALL&XXXXXXXX

(vi) $CALL@XXXX.....XXXX

The first card initiates the analysis of syntax, clears the storage areas and pointers, and causes the printer to commence on a new page with the control card being printed. The $LIST1 and $LIST2 stop the analysis of syntax causing a skip to a new page. The following cards are then listed with single (60 lines per page) or double (30 lines per page) spacing.

The remaining control cards serve special functions and they will not be listed on the printer when used. The $STOP causes the syntax analyser to cancel itself by removing the subroutine jump instructions from the API trap addresses and severing communication between it and the monitor program. The $CALL can be used in two modes. When followed by a blank character it causes the syntax analyser to communicate a name of up to 8 characters of the program to be summoned from the IBM360 to the monitor, and the cancellation of the analyser when the request can be satisfied. The second mode distinguished by the @ enables commands with a length of up to 24 characters to be relayed through the monitor to the IBM360 where they appear on the operator's console and are treated as normal operator commands.

The processing of each statement involves the following steps:

(i) Each statement is examined as a possible control card.

(ii) Statement numbers are detected, processed and stored.

(iii) Each statement is examined for the presence of a Fortran key word. All the standard key words are allowed although with some there is no further processing.

(iv) If a Fortran key word is not found the statement is then assumed to be an assignment statement and is processed accordingly.

For each new statement, with the exception of the logical IF, the process is repeated, starting at step (i). The logical IF in Fortran is one statement that may be accompanied by another selected Fortran statement i.e.

IF (logical expression) restricted executable statement.

The restricted executable statement may be any executable statement other than a DO, or logical IF. Repetition for the logical IF is through step (iii) where the Fortran key words are ordered so that the two restricted executable statements and non-executable statements are not considered on the second pass.

Because Fortran key words are considered before assignment statements the use of a key word as part of a variable name is not permitted (this is in keeping with standard Fortran).

3. PROGRAM INTERFACE

A Burroughs card reader (200 cards/minute) and an Anelex printer (300

lines/minute) are connected to the automatic priority interrupt (API) system of the PDP9L on levels 2 and 3 respectively. Devices attached to the API system have their own trap addresses in the PDP9L so that I/O interrupts transfer control directly to an appropriate subroutine (through a subroutine jump instruction placed at the trap address), without the need for device polling by the monitor. When the analyser is loaded it contains the routines necessary to handle I/O interrupts and is responsible for placing subroutine jump instructions at the appropriate addresses. Should these not be free the loading process is terminated. After a successful loading the section of code used to enable the hook-up with the monitor is freed for table storage.

The cards pass the read station laterally and it is essential that each column be read within 2.0 milliseconds of the column flag being raised, otherwise that column is lost. The connection of the card reader at a higher API level than all time dependent devices enables this to be accomplished in a most efficient manner. The cards are read in alphanumeric mode with the EBCDIC punched characters having a 6 bit internal representation. The interrupt handler translates these to the line printer code and packs them 3 to a word in the input buffer.

The link monitor passes control to the syntax analyser and to other programs in core in a round robin fashion. For any one period of control the analyser may only retain control for 1 millisecond as this is the maximum time delay allowed for servicing requests from the network. The return mechanism (Richardson 1971) consists of jumps to a small subroutine which merely saves the return address and accumulator and returns to the monitor program. It may be called at any point in the analysis subject only to the timing constraint. The operating speed of the analyser is thus very dependent upon other activity taking place in the PDP9L. In the absence of the monitor the analyser is capable of working near maximum card reader speed.

Apart from initiating card reading on the first entry the analyser leaves the conversion of input characters and the filling of the input buffer to the interrupt handler which sets a flag to indicate the presence of a card image for analysis. When control is passed to the analyser on each time slice this flag is tested giving a quick return to the monitor should the analyser be in its normal quiescent state.

4. RESTRICTIONS

All the features of IBM specified Fortran IV are tested for, with the following restrictions:

(i) Variable names must not commence with Fortran key words, nor should they contain blank characters.

(ii) Due to the one pass nature of the system COMMON, DIMENSION, and type specifications should precede all executable statements.

(iii) EQUIVALENCE, IMPLICIT, ENTRY, EXTERNAL, and DATA statements are not analysed. The initialisation of a variable in a type declaration is not tested for, but analysis of such a statement continues at the conclusion of the initialising section.

(iv) Arithmetic statement functions are not implemented and will be indicated as errors. This is unfortunate but results from the need to test the incorrect use of scalar variables with subscripts when they are the target of an arithmetic assignment statement.

(v) While the type declaration of variables is tested no attempt is made to add such variables to a symbol table, unless the variable is dimensioned. This coupled with the infrequent use of logical and complex variables by the great majority of users results in all variables being treated as having the combined properties of integer and real with no specified precision. When manufacturer supplied functions are used no checking can be performed on the type of argument supplied. The use of a logical expression will be treated as an error should a logical operator be employed. When the logical IF is used a logical expression consisting of one primary term will be incorrectly interpreted as an arithmetic IF. For a logical IF statement a logical expression involving relational operators is correctly handled.

(vi) ASSIGN, assigned GO TO, direct access, Fortran II type READ, PUNCH and PRINT statements, and Fortran IV (G) debug statements are not permitted and will be indicated as errors.

(vii) The following areas of the language are only partially treated and syntax errors in them may not be flagged as errors:

- Object time dimension declarations,
- PAUSE,
- Argument lists in subroutine and function definitions,
- Argument list in a subroutine CALL statement,
- NAME LIST I/O declaration.

(vii) Subscript expressions are not permitted in lists of variables for the READ or WRITE commands.

5. CONCLUSIONS

The syntax analyser provides a very desirable service to users of batch

computing systems despite its limitations. Through interaction with the analyser, users of the AAEC computing system have a high chance that all Fortran syntax errors are removed before their programs are first run on the IBM360. Should more PDP9L core store be freed, the analyser is capable of being expanded and the present indication is that a full analysis should be possible with the present store dedicated to the analyser.

6. REFERENCES

IBM, (1971) - 'IBM System 360 and System 370 Fortran IV language', GC28-6515-8.

Richardson, D.J. (1968) - A Generalised Computer to Computer Link, AAEC/TM485.

Richardson, D.J. (1971) - Generalised Computer to Computer Communication. Ph.D. Thesis, U.N.S.W.

DEC, (1968)- 'PDP9L User Handbook', DEC-9L-GRVA-D.

APPENDIX A

ERROR MESSAGES

Error Message Number	Explanation
1	A non-numeric character is used in the Fortran statement number field.
2	All the available table space is exceeded. Syntax analysis continues but the diagnostics produced may be unreliable.
3	DO loops are incorrectly nested.
4	Two statements have identical statement numbers.
5	Incorrect use is made of the continuation indicator column.
6	A blank card is encountered.
7	An error is detected in DIMENSION or COMMON statement.
8	A variable is not correctly dimensioned.
9	A variable is dimensioned more than once.
10	The TYPE of a variable is incorrectly specified.
11	The statement number to end a DO range has previously been defined.
12	Incorrect syntax is used in a DO statement.
13	(a) an incorrect character follows a '(' in a FORMAT statement, or (b) a FORMAT field separator is missing.
14	(a) an illegal character follows a ')' in a FORMAT statement, or (b) an illegal character follows a '/' in a FORMAT statement.
15	(a) a comma is misplaced, or (b) an illegal character follows a comma in a FORMAT statement.
16	A FORMAT statement is not wholly contained in parentheses.
17	A FORMAT declaration is not followed by a '('.
18	Unbalanced parentheses are detected.
19	FORMAT groups may only be nested 2 deep.
20	Incorrect use is made of one of the FORMAT codes.
21	An illegal character is used in a Fortran I/O command.
22	An array is followed by an illegal character in a Fortran I/O list.
23	An illegal character is used as an array subscript.

Error Message Number	Explanation
24	An array is used as an array subscript in an I/O command.
25	An implied DO loop is not completed.
26	(a) an array subscript is followed by an illegal symbol, or (b) the syntax analyser does not support arithmetic operators or subscripted arrays in I/O array subscript expressions.
27	A variable has an incorrect number of subscripts.
28	A repeat counter is incorrectly used in a FORMAT statement.
29	A variable is incorrectly used as an array.
30	The use of an arithmetic operation in an arithmetic expression is incorrect.
31	A constant is incorrectly specified in an arithmetic expression.
32	An illegal character is used in an arithmetic expression.
33	(a) this statement has not been recognised (this may be due to limitations in the analyser), or (b) a reserved Fortran word is used to commence a variable name.
34	An array is not followed by a '('.
35	A comma is misplaced in an arithmetic expression.
36	Nesting within a subscript expression is deeper than that which can be handled by the analyser.
37	A pair of parentheses encloses a null expression.
38	The analyser does not support arithmetic statement functions.
39	A variable name is longer than 6 symbols or is padded with blanks.
40	An attempt is made to initialise a variable in a COMMON block.
41	(a) a function subprogram is specified without an argument list, or (b) arithmetic statement functions are at present not supported.
42	The type specification of a function subprogram is in error.
43	A function subprogram fails to return a value when invoked.
44	A recursive call to a function is not permitted.
45	A statement label has a length in excess of 5 digits.

Error Message Number	Explanation
46	A function subprogram name is not correctly used.
47	A function or subroutine statement is misplaced.
48	There is a syntax error in a SUBROUTINE statement.
49	An IF statement is not complete.
50	A relational operator is incorrectly used.
51	In a logical IF statement a logical operator is expected but is not present.
52	(a) The branching conventions of an arithmetic IF have been violated, or (b) limitations in the syntax checking have resulted in a logical variable not being recognised.
53	A FORMAT statement is used without a statement number.
54	Incorrect syntax has been detected in a GO TO statement.
55	An attempt has been made to branch to a FORMAT statement.
56	(a) the branch label in a GO TO statement is undefined, or (b) A DO loop is not closed.
57	A referenced FORMAT statement is not supplied.
58	In a Fortran program or subprogram there is no STOP or RETURN statement.
59	A syntax error is detected in a STOP or RETURN statement.
60	Illegal use is made of a call statement.
61	A subroutine name is incorrectly used.
62	There are unbalanced parentheses in a CALL statement.
63	A Fortran statement is not complete.