FORTRAN Check System Using a Small Scale Computer

Koji TOCHINAI*, Shizuo NAKAMURA** and Yuko WADA***

Abstract

In this paper, a program for pre-checking FORTRAN programs using a small scale computer system is described. With the check program, the user is capable of removing syntactic errors from his program before processing it on the main computer, and hence the increase in the job processing efficiency and the reduction of job turn around time are expected. The check program has been developed and utilized at the Hokkaido University Computing Center, and apparent improvements on the system performance are observed.

1. Introduction

In university computing centers in Japan, many FORTRAN programs are processed in every day. The considerable amount of them contain several syntactic errors and are rejected from the computing system at the compilation step, and hence they decrease the system performance.

If an appropriate sub computer system is available, the method of pre-checking syntactic errors by using it before processing on the main computer system is useful. The major benefits of this method are:

1) to increase the job processing efficiency of the main computer system,

2) to perform syntax checking more precisely than the compiler built-in checking facilities, and

3) to present kind information of errors to the user.

Then the increase in the system performance and the reduction of the period for debugging are expected.

Based on the point of view as mentioned above, the check program system has been developed. The program has been utilized in the Hokkaido University Computing

This paper first appeared in Japanese in Joho－Shori (Journal of the Information Processing Society of Japan), Vol. 18, No. 2 (1977), pp. 135～141.

* Faculty of Engineering, Hokkaido University
** Research Institute of Applied Electricity, Hokkaido University
*** Hokkaido University Computing Center

Center since November 1973, and apparent improvements on the system performance in job processing are observed.

2. Main functions of the check program

The check program is designed to check the card punched FORTRAN program, and implemented on a small scale sub computer system with 24KB of core memory, a 512KB drum and two 5MB discpacks. It can check the syntax of each statement, the control flow of the program, the definition and usage of statement numbers and names, and so

a) Source program listing.

```
      C     CHECK PROGRAM
1           DIMENSION A(100,100),B(30,30)
2           DO 400 I=1,20
3           LAMDA=((IALPHA/IBETA)**2-256*IGAMMA(I)))/180
4           IERR(I+2)=LAMDA+I
5       300 GO TO I,(301,302,303,304,305)
6           IC=(2.0,5.0)+IALPHA
7       400 CONTINUE
8           STOP
9           END
```

Underlines show the place where the error exists.

b) Error messages.

ISN	1	**	ERROR	NO.2007	Symbol error in array declaration.
ISN	3	**	ERROR	NO.2702	Incoincidence of the number of parentheses.
ISN	4	**	ERROR	NO.2605	Undeclared array name.
ISN	6	**	ERROR	NO.2806	Illegal combination of types.

c) List of statement numbers and names.

* STATEMENT NUMBERS *

NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER
400	300 *	301 N	302 N	303 N	304 N
305 N					

* NAMES *

NAME	TYPE	KIND	NOTE	NAME	TYPE	KIND	NOTE	NAME	TYPE	KIND	NOTE
LAMDA	* I	VAR.		IALPHA	I	VAR.		IBETA	N I	VAR.	
I	N I	VAR.		IC	* I	VAR.					

Fig. 1 An example of the checking result.

on. As any error is found, an error message corresponding with it is printed. An example of the checking result is shown in Fig.1.

3. Program design

The check program consists of three phases as shown in Fig.2. Various tables are used in the program, and major three of them are:

1) the name table,

2) the statement number table, and

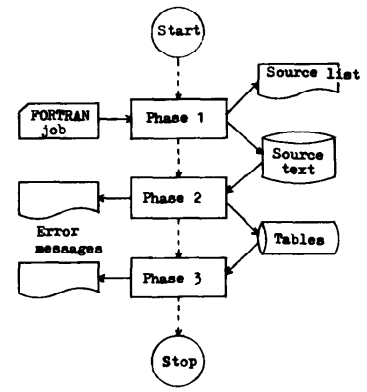


Fig. 2 General flow of the check program.

3) the control table.

They are located in the drum and divided into pages. A page consists of 512 16-bit words, and a software paging technique is used.

The name and statement number table are combined together based on the investigation of user programs. Fig.3 shows the result of it and the correlation

$$N_T \sim 0.65n \qquad (1)$$

is recognized.

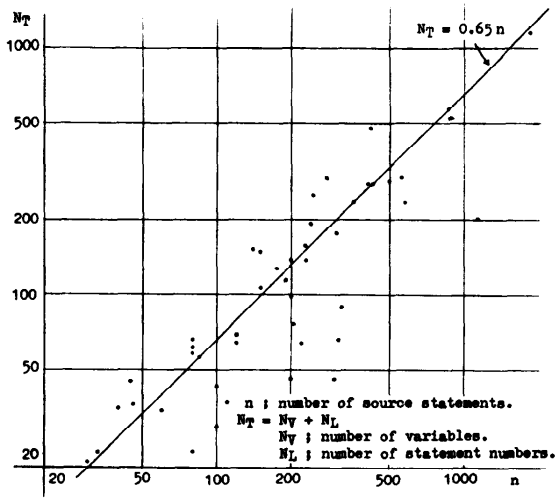


Fig. 3 Relation between n and N_T.

According to the eq.(1), the name and statement number table has a capacity of 1100 entries corresponding with the program up to 1600 lines. To determine the location of an entry in the table, the hash technique is used. The structure of an element of the table is shown in Fig.4 together with the paging and hashing system.

The control table is used for checking the control flow of the program. The structure of an element is shown in Fig.5. The location of an entry is determined sequentially.

As any syntactic error is found during the processing on the check program, corresponding error message is printed. The message consists of the location,level and kind. (see Fig.1) The kind of the error is expressed by a 4-digit integer, because the memory capacity is rather small to store precise explanations of errors in the memory.

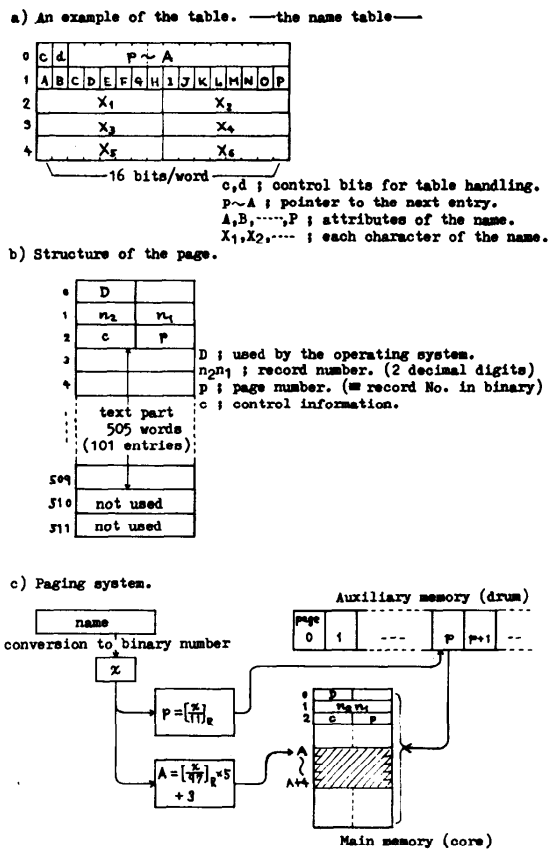
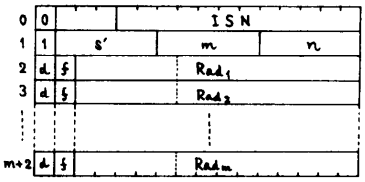


Fig. 4 Table construction.

Consequently, the user is necessary to refer the error message manual.



ISN ; internal statement number.
s',n ; attributes of the statement number.
d,f,m ; used in phase 2.
Rad ; pointer to the control table
or the statement number table.

Fig. 5 Control table.

4. Results

After the check program service has been available since November 1973, the increase in the job processing efficiency and the reduction of the job turn around time are observed.

4.1. Increase in the job processing efficiency

Fig.6 shows the variation in the ratio of the core time (uset) to the cpu time (cput) of the processed jobs on the main computer system for job class A: the smallest class and cput is limited shortly. The ratio clearly decreased after beginning of the check program service. The considerable amount of class-A jobs are under

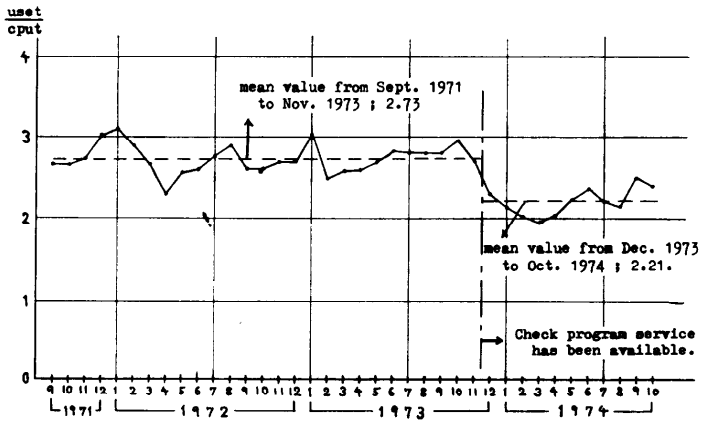


Fig. 6 Ratio of the use time to the cpu time for class A jobs.

debugging, and therefore this result can be interpreted that some of them has become to be processed by the check program, the number of errorneous jobs has reduced in the main system and then the cpu utilization has increased. Uset to cput ratio of other job classes have not varied during these period. As the most of them are considered the "debugged" jobs, the result supports the interpretation mentioned above.

4.2. Reduction of the job turn around time

As the job turn around time reduces, users are capable of processing more jobs in a given period. Therefore, suppose that the computer system has a sufficient job

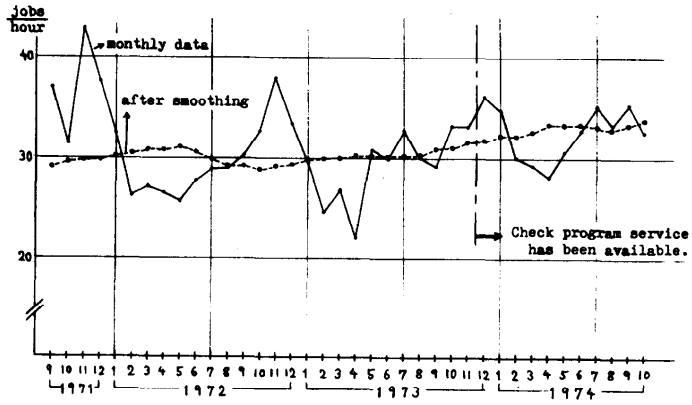


Fig. 7 Number of jobs processed in unit time.

processing ability, the increase in jobs processed in unit time is expected. Because of the concentration of job processing for graduation studies at the end of the semester, this value is varying periodically. However, by the smoothing operation over more than 12 months, the apparent increase is recognized as shown in Fig.7.

It is interpreted that users can perform syntactic debugging of their programs rapidly by using the check program system effectively, and then are capable of processing more jobs in a unit time.

5. Conclusion

The design objectives of the check program system are well accomplished except the processing speed is not so high: about 1 source statement in a second from card reading to the end of checking. It is well utilized in the computing center, and apparent improvements in the job processing efficiency are observed. Therfore, it is concluded that the check program system developed in this study is valuable for our computing center.