PRECHECKING SAS[1] SYNTAX WITH A MICROCOMPUTER

Jeff Bass
BASS Institute, Inc.[2]

INTRODUCTION.

Microcomputers are increasingly being used as terminals and workstations connected to large mainframe computers. The microcomputer, running an appropriate program, can act as a full screen editor or a very smart terminal with local storage. Mainframe computers can prepare and edit programs offline, saving connect charges and moving their mainframe usage to off peak times. After uploaded programs are run, the printouts can be downloaded and incorporated into reports using the microcomputer's word-processing abilities.

The usefulness of microcomputers could be further enhanced if the programs to be uploaded to the mainframe could be checked for syntax errors on the microcomputer. This could reduce or eliminate the syntax checking runs made on the mainframe and increase programmer productivity. We have implemented a SAS syntax prechecker that runs on an IBM[3] Personal Computer[3] using Pascal. This paper discusses the issues involved in developing such a program as well as some of the program's strengths and weaknesses.

There are five issues that must be addressed in checking the syntax of a SAS program: (1) Statement Syntax, (2) Statement Order, (3) Variable names, (4) PROC Syntax, and (5) Initialization of Datasets and Variables. The first three issues must be addressed by the syntax checker or parser of any computer language, while the last two are more specific to SAS programs. Each of these issues will be addressed in turn.

STATEMENT SYNTAX.

Each SAS statement must conform to a general syntax. For example, a SAS statement must begin with a SAS keyword (or a variable name if it is an assignment statement), and end with a semicolon. It may span multiple records. Left parentheses must occur in equal numbers with right parentheses. Arithmetic and logical operators must (usually) be separated from one another by variable names or keywords. Statement labels must occur at the beginning of a statement. Quote marks must be balanced. The program checks for all of these items for each statement individually.

STATEMENT ORDER.

Certain statements must occur in order if they are to be meaningful. For example, assignment statements must occur after the DATA statement to which they refer. END statements must occur after their associated DO statement, etc. Statement order is more difficult to check than statement syntax, because of the fact that a notion of position must be maintained and updated with each statement. This includes checking for the existence of statement labels and correct nesting of DO's and IF's.

VARIABLE NAMES.

Variable names must be checked for correct spelling and length. This is especially difficult in SAS programs because variables can be referred to in some statements by the minimum number of letters necessary to make the reference unique. For example, AGE can refer to AGEYRS after AGEYRS has been defined. Also, variables can be referred to indirectly in lists of variables (e.g., ABC1-ABC3 also refers to variable ABC2). Both of these cases make syntax checking more difficult than in a more typical language. We discourage the use of variable abbreviations, so we dealt with the variable name abbreviation issue by flagging such uses as initialization errors. For the second case, the syntax checker internally expands the variable lists and checks the spelling and initialization of each list member.

PROC SYNTAX.

Each SAS Proc has its own rules for what keywords may be used in the proc statement itself and what statements can be used to modify each proc. The syntax checker deals with this issue by accessing a reference dataset whenever a proc statement (or its subsequent statements) is being checked. The reference dataset contains the possible proc statement keywords and allowable proc statements for each procedure. Updating this dataset with user written procedures is relatively simple, so that syntax checking can be extended to the custom procedures that exist at a given site.

INITIALIZATION OF DATASETS AND VARIABLES.

When SAS is run with OBS=0 for syntax checking, it has to have all its associated datasets available (including offline disks and tapes). While this gives the most complete checking, it is often inconvenient to mount a tape just so SAS can check its variable list. Of course, when syntax checking with a microcomputer that is not running SAS, no mainframe datasets can be accessed. This means that initialization of variables and existence of datasets cannot be explicitly checked. The program deals with this issue by keeping lists of all datasets and variables that are referred to in a SAS program. For datasets and variables that are created previously in the same program, the existence of datasets and variables is explicitly checked and appropriate "not initialized" or "does not exist" messages are printed as necessary. When reference is made to a permanent SAS dataset, a list is made of all those variables that will need to be on that dataset in order for the program to work without error. At the end of the program, a message will appear that describes the expected input, for example:

```
PRECHECK: 'TEST1.ADULTS', A SAS DATASET, MUST
          ALREADY EXIST.  IT MUST BE DEFINED IN
          JCL.
PRECHECK: 'TEST1.ADULTS' MUST CONTAIN THESE
          VARIABLES: AGE,HEIGHT,ID,SEX,WEIGHT
```

This helps the programmer who is checking his or her program in identifying obvious input file errors.

OTHER ISSUES.

The present precheck program addresses all of the above issues, and has been found to be very effective in locating simple syntax errors. However, it has several shortcomings: (1) it does not check macros or macro expansions, (2) it does not check JCL (but it does allow JCL statements to be in the file--it skips over them to the SAS source statements), and (3) it does not check the PARMCARDS used by some procedures. These could be future enhancements. An example printout from the program appears below.

```
PRECHECK OF PROGRAM: PCHKTEST ON 9-2-83 at 15:33.

1   DATA TEST1(LABEL=SIMPLE PRECHECK TEST);
2     INFILE TAPE3;
3     INPUT PERSONID HOUSEHOLD AGEYRS SEX HEIGHT WEIGHT;
PRECHECK: THE NAME 'HOUSEHOLD' IS TOO LONG.
4     HT_WT=HEIGHT/WEIGHT
5     LABEL HT_WT = RATIO OF HEIGHT FOR WEIGHT;
PRECHECK: SYNTAX ERROR, PROBABLE MISSING ';'.
PRECHECK: 'TAPE3' MUST BE DEFINED IN JCL.
6   DATA TEST 2;
7     SET TEST1;
8     IF AGE GT 65;
PRECHECK: VARIABLE 'AGE' IS NOT DEFINED ON DATASET TEST1.
9   PROC FERQ DATA=TEST2;
PRECHECK: PROC 'FERQ' UNKNOWN. PROC STATEMENTS WILL NOT BE CHECKED.
10    TABLES HT_WT*SEX;

PRECHECK: END OF SOURCE LINES.
PRECHECK: 5 JCL LINES READ. SYNTAX NOT CHECKED.
PRECHECK: 10 SOURCE LINES READ. SYNTAX CHECKED.
PRECHECK: 4 ERRORS FOUND.
PRECHECK: NO PERMANENT SAS DATASETS WRITTEN.
PRECHECK: NO PERMANENT SAS DATASETS READ.
PRECHECK: 1 PERMANENT RAW DATASET READ:
          DDNAME(S): TAPE3
PRECHECK: NO PERMANENT RAW DATASETS WRITTEN.

PRECHECK: BASS INSTITUTE, INC.
          P. O. BOX 349
          CHAPEL HILL, NC  27514
```

NOTES

1. SAS is a registered trademark of SAS Institute, Inc.

2. For copies of this paper or copies of the program, the author may be contacted at:

   BASS Institute, Inc.
   P. O. Box 349
   Chapel Hill, NC 27514

3. IBM and IBM Personal Computer are registered trademarks of International Business Machines, INC.