ON-LINE DOCUMENTATION AND TUTORIAL FOR SAS

Jan M. Hanley, The RAND Corporation
Franklin Young, United Airlines

I - INTRODUCTION

Quality training and software support ultimately determines the successful use of the software itself. As an alternative or a complement to the traditional lecture style class, we have explored a computer assisted instructional system (CAI) for SAS. The CAI system discussed in this paper has been developed under TSO. A natural companion to this system is an on-line documentation capability (i.e. HELP). Our goal is to demonstrate that a CAI system and on-line documentation for SAS can be implemented with readily available building blocks. The benefits and need for such tools will be presented in section II. Section III will outline the guidelines relied upon to develop the prototypes. Section IV will discuss the design considerations and the implementation problems encountered.

II - BENEFITS OF THE TSO TUTORIAL

2.1 Tutorial

The value of an on-line tutorial can best be illustrated by contrasting this mode of instruction to a lecture style class.

An on-line tutorial offers a more individualized approach in terms of the:

- Availability of instruction.
- Pace at which material is presented.
- Selection of topics covered.

However, the price for this flexibility is the lack of direct interaction between the instructor and the student. In addition, a minimal knowledge of the operations of a terminal and TSO is required.

An on-line tutorial is probably most useful for the occasional user who needs frequent refreshers. It would also be beneficial at an installation with a continuous stream of new SAS users.

2.2 HELP

SAS in its current state is not a truly interactive system. Hence, the user can expect little help (in the form of prompts) from the system. This deficiency can be partly removed by an on-line documentation capability patterned after the TSO HELP feature. It acts as a memory aid and is especially convenient for the user who has limited requirements.

III - GUIDELINES

The tutorial and the HELP facility are meant to be complementary. The former emphasizes SAS programming techniques whereas the latter describes the usage of selected features.

3.1 Tutorial

The following guidelines were used in the development of the tutorial:

- The purpose is to teach SAS as a data processing utility, not to teach statistics.
- The student is assumed to have no previous experience with SAS and only a limited knowledge of data processing, statistics, and TSO.
- Only items of general interest are covered.
- The tutorial is modular in design. It allows the student to select the topics to be presented.
- Active participation from the student is required.

3.2 HELP

The HELP commands were designed with different guidelines:

- The purpose is to document, not teach, SAS procedures.
- Only the use of keywords are documented, not the use and interpretation of the SAS procedures.
- Only options of general interest are

The authors would like to thank Mitchell Tuller, The RAND Corporation, for his consulting expertise on PL/I and TSO, and Joyce Baptista, The RAND Corporation, for her valuable assistance in reviewing and editing this paper.
presented. For example, options such as LEAVE and MESSAGE for PROC SORT are omitted.

IV - DESIGN AND IMPLEMENTATION

4.1 Tutorial Design

Our lessons offer a combination of theory and practice. The former is given in a narrative. The latter takes the form of exercises followed by an on-line 'drill.'

During these drills the student is asked to code exercises under the guidance and supervision of a CAI program. It prompts the student for each line of code and can provide, upon request, additional hints (or HELP). As they enter each line, the students receive instant feedback and their errors are clearly identified. Several attempts are permitted. If the number of attempts exceeds 3 (or 5 for the more difficult exercises), the program will provide the correct answer for that line.

To facilitate the implementation, we have imposed a number of restrictions:

- Variable names in the exercises are pre-assigned.
- Only one statement per line is allowed.
- Only one line per statement is allowed.
- Certain statements must appear in a designated sequence even though in reality SAS itself allows considerably more flexibility.

These restrictions were imposed primarily to reduce the quantity of programming required to implement the tutorial.

4.2 Tutorial Implementation

At the present, only two lessons have been implemented. SAS, PL/I and the TSO CLIST command language have been used in their development. The lessons themselves are CLIST. From the CLIST a parsing routine is executed. The cornerstone of a CAI program is the parser which breaks each of the input lines into tokens. We have explored two approaches to implement such a parser:

- Line specific parser written in SAS.
- Generalized parser written in PL/I.

4.2.1 CLIST

The tutorial was developed using the TSO CLIST language which is fully interactive and similar to PL/I. The various lessons are dispatched interactively from a master CLIST. The master CLIST provides an index of the subjects presented in the lessons and allows the student to select the lesson desired. At the completion of the lesson, control is returned to the master CLIST at which time the student has the option of selecting another lesson or exiting. The interaction between the master CLIST and the lesson CLIST is summarized in the figure below.

4.2.2 Lesson I: Line Specific SAS Parser

Lesson I uses a parser written in SAS. The CAI program consists of only a DATA step. The SAS codes entered by the student are read as data. In other words, the INFILE statement points to the terminal.

The parser follows a stepping-stone logic. Each input line is searched from left to right for the various delimiters expected in that statement. Each delimiter is used as a stepping-stone toward the next delimiter. In practice this search is performed with the SCAN function available in SAS 79.3.

Once a line is broken into elements, the elements are compared to the expected keywords (e.g. DATA, INPUT) or to the preassigned variable names. Certain comparisons (e.g., checking for the ending semicolons) have been coded as MACROS. This approach becomes quite tedious when a statement can have several correct forms. Therefore, as the exercises become more complex, the use of a generalized parser would greatly simplify their implementation.
4.2.5 Lesson II: Generalized PL/I Parser

As an alternative approach, the second CAI program has been written entirely in PL/I. PL/I was chosen for its extensive string handling capabilities and general flexibility. The parser is coded as a subroutine and is built around the SUBSTR function. The SAS statements entered by the student are examined character by character searching for operators and delimiters. The input lines are broken into variables, keywords, operators, and delimiters. An array consisting of the individual elements for the source input is then returned to the main program which performs a contextual analysis.

The CAI program increases in complexity as the students are allowed more freedom in writing their SAS programs (e.g. selecting the order the variables appear on the INPUT statement). Implementing such a program would be time consuming but not difficult. Nevertheless, with or without restrictions on the student input, extremely accurate error messages can be generated.

4.3 HELP Command Implementation

A HELP module has been implemented for selected PROCs. Each module has been coded as a self contained, executable SAS MACRO. Each MACRO consists of a DATA _NULL; step. The HELP information is coded with PUT statements. A RUN statement concludes each MACRO.

MACRO HEXAMPLE
DATA _NULL;
PUT 'SAMPLE HELP';
RUN; %

We have patterned our HELP commands after established TSO conventions. For example, the user enters HELP; to obtain HELP on PROC PRINT. The ground rules of the HELP commands are outlined in a "HELP." 

4.4 HELP Limitations

In its current implementation there are two major limitations:

• Loading the HELP library.
• Limited availability.

SAS lacks the facility of a true MACRO library. Moreover, standard TSO does not allow concatenation of terminal input to an OS data set. Hence, before a HELP command can be executed, it must first be loaded into SAS. In practice this detail is transparent to the student. A two step call is incorporated into a SAS CLIST. The CLIST first executes SAS with the HELP library as the input file. Then a second call to SAS is issued after resetting the input file to the terminal.

HELP can only be accessed at the end of a DATA step or between PROCs. Even with this restriction, care must still be exercised when accessing those commands. For example, PROC FREQ; HFREQ would first produce frequency distributions for all variables in the data set, and then display the HELP. A way to avoid this problem is to enter STOP, for example:

PROC FREQ; STOP; HFREQ

STOP would be treated as a misplaced statement by SAS, but prevents the execution of the PROC statement. This is merely a convenient method of avoiding the execution of unwanted procedures. A more elegant solution would be a cancel command, which at the present time is not within SAS's capabilities.

V - CONCLUSION

On-line tutorials are a valuable addition to the repertoire of SAS educational tools. Although our work can yield immediate benefits, it still remains exploratory in nature. A complete CAI package for SAS would be a major undertaking. In particular, as more complex exercises are used in the lessons, the implementation becomes increasingly more difficult. An alternative approach would be to access the SAS parser and interpreter itself. We believe that a CAI package should ultimately be part of SAS itself.
LINE SPECIFIC SAS PARSER

LESSON 1: INTRODUCTION TO SAS

ERROR MESSAGES

PROMPTS

USER INPUT

PARSER

SAS CAL PROGRAM (PARSER)

TERMINAL (USER)

MASTER CLIST

LESSON CLIST

OUTPUT

SAS

GENERALIZED PL/I PARSER

LESSON 2: SORTING AND READING DISK AND TAPE FILES

ERROR MESSAGES

PROMPTS

USER INPUT

PARSER

PL/I

TERMINAL (USER)

MASTER CLIST

LESSON CLIST

GENERATED SAS CODE

OUTPUT

SAS

339