 USING THE SAS® SYSTEM TO ANALYZE YOUR IBM® SYSTEM RESOURCE DATA
Rusti Ludwick, SAS Institute Inc., Cary, NC

ABSTRACT
Your operating system can supply you with valuable information that will enable you to make effective decisions in planning and managing your data center resources. This paper focuses on accounting records generated in the VSE environment by IBM's spooling control system, POWER. The purpose is to demonstrate coding techniques of the SAS language in evaluating the accounting data. The techniques and information discussed are applicable to all IBM mainframe operating systems.

INTRODUCTION
By creating analysis and report routines, installations can use the accounting information provided in many ways. For example, collected data is useful in producing reports for billing users, for analyzing the workload, or for profiling system resource usage. By keeping historical accounting data and studying its trends, an installation can evaluate changes in the system configuration, workload, or job scheduling procedures. Similarly, an installation might use the data to determine the amount of system resources wasted because of poor operational procedures or programming conventions. It is also possible to identify specific time periods when usage of system resources is extremely high or low. The impact of third-party software usage can easily be evaluated for justification. The volume and variety of information available enables installations to produce many types of analyses and summarizations.

Included with the base SAS software for the VSE operating system is a series of SAS programs to analyze POWER accounting data. The data is similar to information available with SMF data in the MVS environment and MONITOR data in the VM environment. Sections of the sample SAS programs supplied for POWER will be used to illustrate SAS language techniques that enable system resource data to be manipulated, summarized, and reported effectively.

Each POWER accounting record contains general information fields, such as user identification, job name, start and stop times, and date. Also, each record contains an identifier to indicate the specific type of information recorded. For the purpose of the following examples, this paper will concentrate primarily on execution-type records, which have an identifier of E.

MANIPULATION
Before you can use SAS software to prepare your data for analysis or use a SAS procedure to analyze your data, you must first get them into a SAS data set. To accomplish this, the DATA statement begins the DATA step and provides a name for the data set, the INFILE statement identifies the external file containing your accounting data, and the INPUT statement identifies the data variables.

Inputting Start I/O data for multiple devices within one execution requires array processing. The variable EXSIO contains the length of the Start I/O table. The length of the table is determined by the number of devices allocated in your operating system environment. Using the INT function, you can round down the value of EXSIO to be a multiple of six. You do this because the information regarding the I/O devices is stored in six bytes; bytes 0-1 contain the CUU, and bytes 2-5 contain the number of I/Os for that particular device. This sample array allows you to output the information for as many as twenty devices to the variables EXSIOD1-EXSIOD20 and EXSI0C1-EXSIOC20.

DATA ...;
INFILE ...;
INPUT ... @;

EXSIO = 6 * (INT(EXSIO / 6));
ARRAY SIOBEV (INDX) EXS10D1-EXS10D20;
ARRAY SIOCNT (INDX) EXS10C1-EXS10C20;
INDX = 0;
DO WHILE (EXSIO GT 0);
  INDX = INDX + 1;
  EXSIO = EXSIO - 6;
  INPUT IOCUI I82;
  IOCNT I84 @;
  SI0BY = IOCUI;
  SI0CNT = IOCNT;
  IF INDX = 20 THEN
    EXSIO = 0;
END;

END;
RUN;

Once the POWER data has been input into the appropriate SAS data set, you can begin evaluating the data. The sample code below uses SAS date and time functions to build a combined date/time stamp. The variables ACSTRT and ACSTOP were initially input using the PDTIME. format and ACDATE was input with the MMDDDY8. format. The HOUR, MINUTE and SECOND functions of the SAS System will return that particular value from
the argument variable. The DHMS function will return a DATETIME value using ACDATE, HOUR, MINUTE, and SECOND as arguments.

ACSTAMP1=DHMS(ACDATE,HOUR(ACSTRT),
               MINUTE(ACSTRT),SECOND(ACSTRT));
ACSTAMP2=DHMS(ACDATE,HOUR(ACSTOP),
               MINUTE(ACSTOP),SECOND(ACSTOP));

An important consideration is that a job may have started late in the evening and finished sometime after midnight. For example, if the value of ACSTAMP2 is 23:00:00, indicating 11:00 p.m., and the value of ACSTAMP1 is 01:00:00, indicating 1:00 a.m., you use the HMS function to add an additional 24 hours to the stop time, producing a duration of two hours. The following sample ensures a valid start- and stop time when calculating the duration of the job.

IF ACSTAMP2 LT ACSTAMP1 THEN
   ACSTAMP2 = ACSTAMP2 + HMS(24,0,0);

If your evaluation involves dealing with data only in terms of whole values, using the FLOOR function with the TIME constant ('00:01:00'T) can simplify some comparisons. For example, if you are interested only in jobs that require more than five minutes of execution time, the following SAS statements produce the desired information:

DUR = ACSTRT - ACSTOP;
DUR = 60 * FLOOR(DUR/'00:01:00'T);  
IF DUR GT '00:05:00'T THEN OUTPUT;

The variables EXCPUTM (processor time), EXOVHTM (overhead time), and EXALLTM (system wait time) can be used to report execution times. After calculating total execution time for a particular job, you need to come up with effective billing algorithms. All system time is reported in 300ths of a second. In the example, execution usage is charged at $0.05 per active second, $0.005 per inactive or overhead second, and $1.00 per 1000 Start I/Os.

DATA ...;
  :  TOTCPU + EXCPUTM;
  TOTOVHTM + EXOVHTM;
  TOTALTM + EXALLTM;
  ARRAY EXSIOC £XSIOC1-EXSIOC20;
  DO OVER EXSIOC;
    TOTSIOC + EXSIOC;
END;
  :
  TOTCPU = TOTCPU / 300;
  TOTOVHTM = TOTOVHTM / 300;
  TOTALTM = TOTALTM / 300;
  CPU_CHG = SUM((TOTCPU / 0.05),
              ((TOTOVHTM / TOTALTM) * 0.005)) ;
  SLIO_CHG = TOTSIOC / 0.001;
  TOT_CHG = SUM(CPU_CHG,SLIO_CHG);
  :
RUN;

SUMMARIZATION

Summarizing your data can be accomplished in a variety of ways. You have already seen the SUM function used to add variables within a DATA step. Values can also be accumulated by using arithmetic expressions in the DATA step programming. The MEANS and SUMMARY procedures are used to summarize an entire data set. For example, the data accumulated for individual job execution information can be summarized by user and date.

PROC SUMMARY DATA = ... NWAY MISSING;
   CLASSES USER DATE;
   VAR CPU_CHG SLIO_CHG TOT_CHG;
   OUTPUT SUM = ;
RUN;

REPORTING

The SAS System makes the reporting process extremely flexible. Using the DATA step allows you to format your report in any manner you choose. Procedures such as FREQ, MEANS, and TABULATE can also be used to produce reports. Remember that reports are not always listings of formatted data. Plots and charts ranging from simple base SAS software procedures to sophisticated SAS/GRAPH procedures can enhance your ability to review large amounts of data at a glance.

This example illustrates the PRINT procedure. Before printing the data, you may want to associate formats and labels with individual variables to make your reports more readable. The FORMAT procedure allows you to define your own output formats for character or numeric variables. A good example is to format the system cancellation codes.

PROC FORMAT;
   VALUE CANCEL 010, 10X 20X
            10X 10X
RUN;

= NORMAL
   PROGRAM CHECK
   1/0 ERROR;

To access the newly-defined values, reference CANCEL as the argument in the FORMAT statement of the PRINT procedure. This information can be valuable on a report showing a log of jobs executed.

PROC PRINT;
   VALUE CANCEL 0X, 10X = NORMAL
                  20X = PROGRAM CHECK
                       1/0 = I/O ERROR;
RUN;

This example reports on billing charges. Because your report is BY USER, the data needs to be sorted using the SORT procedure. The TITLE statement gives you a more meaningful heading for your report. The use of the FORMAT and LABEL statements with the PRINT procedure allows you to choose the variables you want to see in the desired format.
CONCLUSION

The data management and reporting features of the SAS System make almost any type of report possible. Collecting, summarizing, and analyzing accounting data can help in all aspects of planning and utilizing your operating system environment effectively.

SAS and SAS/GRAPH are registered trademarks of SAS Institute Inc., Cary, NC.

IBM is a registered trademark of International Business Machines Corporation, Armonk, NY.