TEMPLE UNIVERSITY
Computer Services

we bring IT.

SAS on the LINUX Platform
(BATCH MODE on compute.temple.edu)

Computer Services
Client Services
215-204-8000
tuhelp.temple.edu

Document Version 6.1

Table of Contents

I. Introduction

A. Objectives

This document will serve three functions:

1. It will introduce users to the LINUX version of SAS in non-interactive mode
2. It will show sample SAS programs
3. It will describe differences between the Windows and LINUX versions of SAS.

B. Prerequisites

1. Must acquire or have knowledge of basic LINUX commands and syntax.
2. Knowledge of SAS on other computer platforms is helpful.
3. Knowledge of any statistical package is helpful.

II. Overview of Capabilities

A. SAS DATA Step Statements

Data set creation, manipulation, and transformation is performed using DATA step statements, which do the following:

1. Create variables through assignment statements.

   Example: avgtest = MEAN (test1, test2, test3);

2. DROP variables that are not wanted or KEEP only the variables that are wanted

   Example: DROP age test3;
   KEEP avgtest;

3. DELETE unwanted observations.

   Example: IF lab = 3 THEN DELETE;

4. Set conditions through IF statements

   Example: IF computer = 1 THEN price = 1195;
   ELSE price = 1795;

5. Identify variables through LABEL statements.

   Example: LABEL test1 = "Score in fall"
   test2 = "Score in spring"
   test3 = "Score in summer";

6. Define character-missing codes through MISSING statements.

   Example: MISSING X;

7. Define numeric missing codes through IF statements

   Example: IF age = 99 THEN age = . ;

8. Use various DO commands to execute statements multiple times.

   Example: DO Year = 1990 to 2010;
   Salary = salary+salary*0.10;
   END;

9. Combine data sets through SET statement

   Example: DATA new;
   SET one two;
   BY id;

10. Allow the use of mathematical and logical operators (See III.E)

B. SAS PROC Step Statements

Data display, data rearrangement, and statistical analysis are performed by PROC STEP statements, which include the following:

1. Basic Statistical Procedures:

   a. PROC MEANS -- provides means and other univariate descriptive statistics.
   b. PROC UNIVARIATE -- provides descriptive statistics, with detail on the distribution of the variable.
   c. PROC FREQ -- provides one-way, two-way, and *n*-way cross tabulations.
   d. PROC CORR -- provides bivariate correlations and other measures of association for quantitative variables.
   e. PROC CHART -- draws bar, block, pie, and star charts of counts, mean, and sums.
   f. PROC PRINT -- prints all or part of the data in a tabular report format.

2. Advanced Statistical Procedures:

   a. PROC ANOVA -- for balanced analysis of variance designs.
   b. PROC GLM -- for general linear models (including nested, multivariate and repeated measures designs).
   c. PROC CATMOD for contingency table analysis as well as logistic regression.
   d. PROC LOGISTIC, PROC NLIN, and PROC REG-- for various types of regression: LOGISTIC is for logistic regression, PROC NLIN is for non-linear regression, and PROC REG is for linear regression (including all subsets regression.)
   e. PROC CLUSTER, PROC DISCRIM, PROC FACTOR -- provide cluster, discriminant, and factor analysis respectively.

C. LINUX SAS Processing

1. Types of processing modes

   On the LINUX platform (i.e., compute.temple.edu), SAS can be run in either the non-interactive mode (also called Batch mode) or interactive mode. This document will describe non-interactive processing. For users who have worked with SAS on a mainframe, the non-interactive mode should be familiar.

When SAS is run under LINUX using the interactive mode, an X-Windows emulation product (e.g., X-win32) is also required. With the use of X-Windows emulation software, SAS under LINUX will resemble SAS for Windows.

2. Types of statements

A SAS program includes two types of statements: Data Steps and Procedure Steps, which describe the data and analyze the data, respectively. The data are either added to the syntax file (recommended for less than 50 observations) or placed separately in an external file.

3. SAS execution (non-interactive mode)

To run a SAS program under LINUX in the non-interactive mode, type: sas *filename*. The file extension .sas is assumed but not included when SAS is told to execute a file. The file named filename.sas, refers to the program file containing the SAS commands.

4. Multiple programs

SAS provides a multiple job structure, allowing many procedures to be performed in a single run (See Example 1). The user must assess the amount of output which will result from the processing of multiple procedures to determine when to divide the SAS program into separate and smaller jobs.

D. SAS HELP

1. Obtain online assistance from the SAS Institute's support web site: http://support.sas.com

III. LINUX SAS Processing Components

A. SAS Modules

The SAS system under LINUX contains many modules such as BASE SAS, SAS/STAT, SAS/GRAPH, SAS/IML, SAS/INSIGHT, SAS/LAB, and more. However, some of these modules cannot be run in non-interactive mode (e.g., SAS/INSIGHT, SAS/LAB, and SAS/ASSIST) since the screens required for these particular modules cannot be displayed in non-interactive mode.

B. Description of Non-interactive Mode

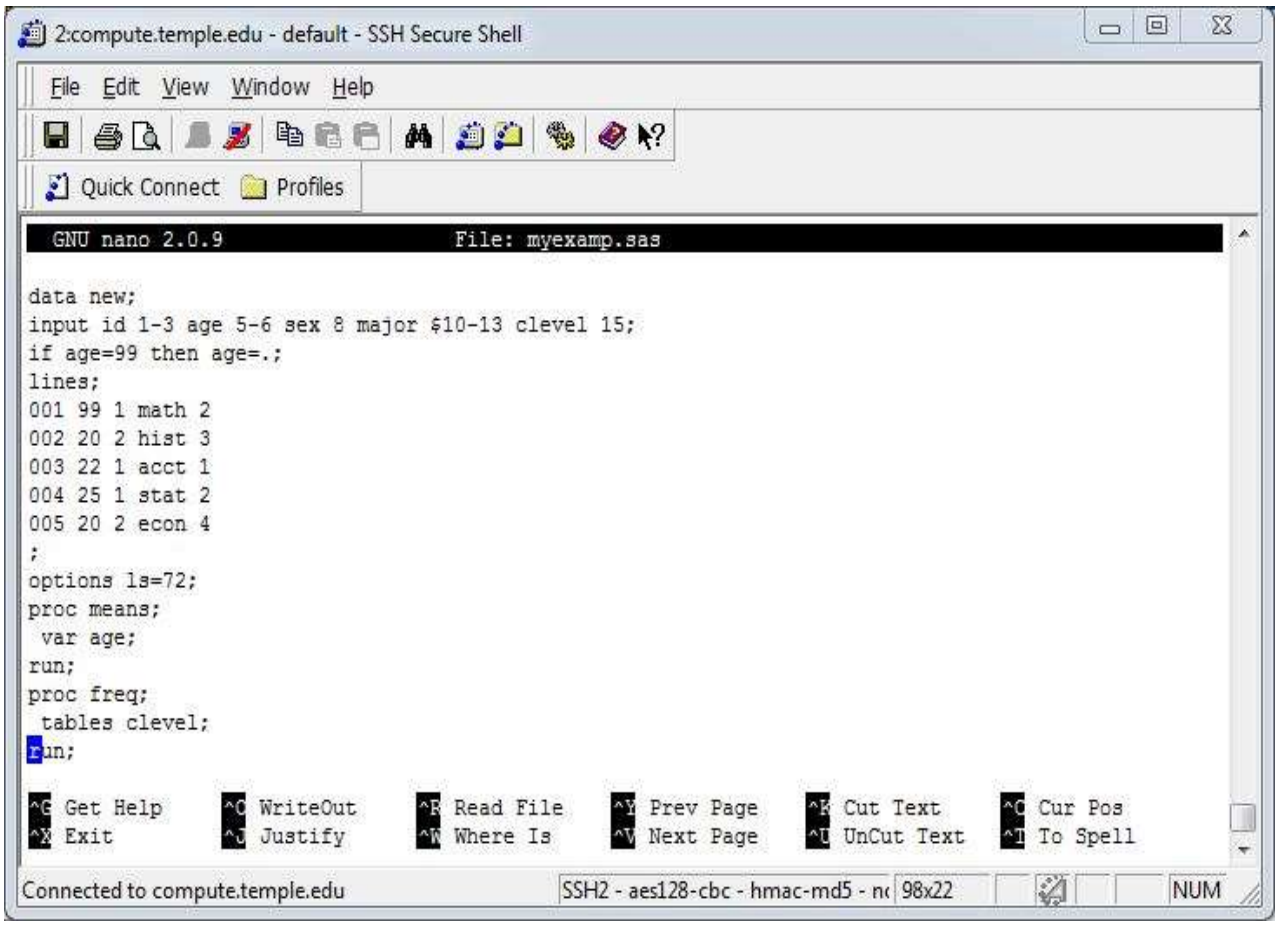
The non-interactive mode of SAS is used when an X-Windows client is not available or not being accessed. The execution of SAS under non-interactive mode requires the creation of a program file which is then executed. The following steps should be followed:

1. Use a word-processing program or a LINUX editor to create a SAS syntax file (containing SAS commands or SAS commands plus data). Use the extension *.sas* for this file. It must be in ASCII or text format.

2. Upload the file to the LINUX account (e.g., use SSH Secure Shell) if the file is not created on the LINUX system.

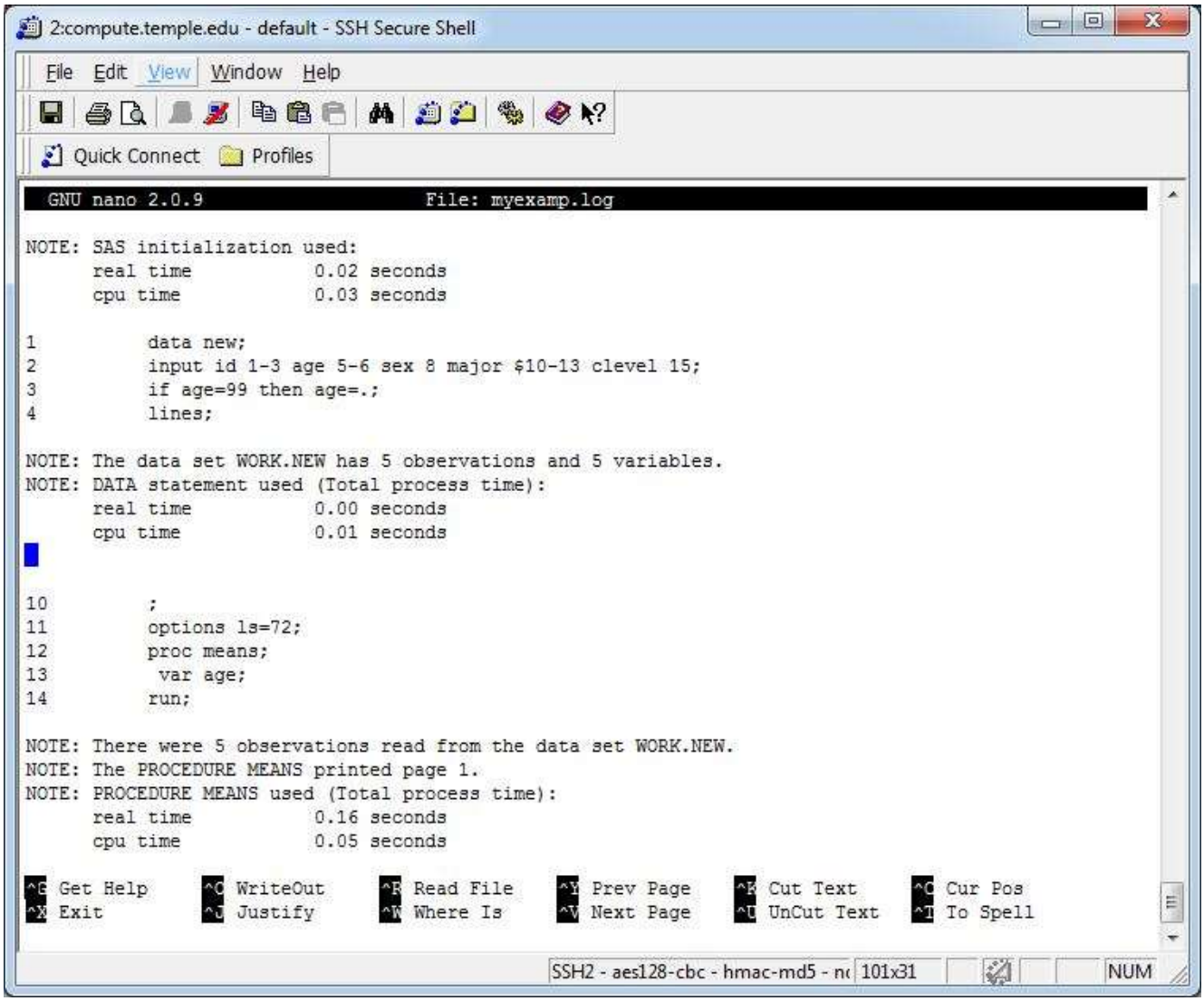
3. Figure 1 shows the lines of a sample SAS program named *myexamp.sas*. The program lines become visible, after the Nano editor is initiated, by typing the words: *nano myexamp.sas*

Figure 1: Sample SAS program file



4. Execute the program in Figure 1 above by using SAS in non-interactive mode and typing: *sas myexamp*

Figure 2: Sample SAS log output

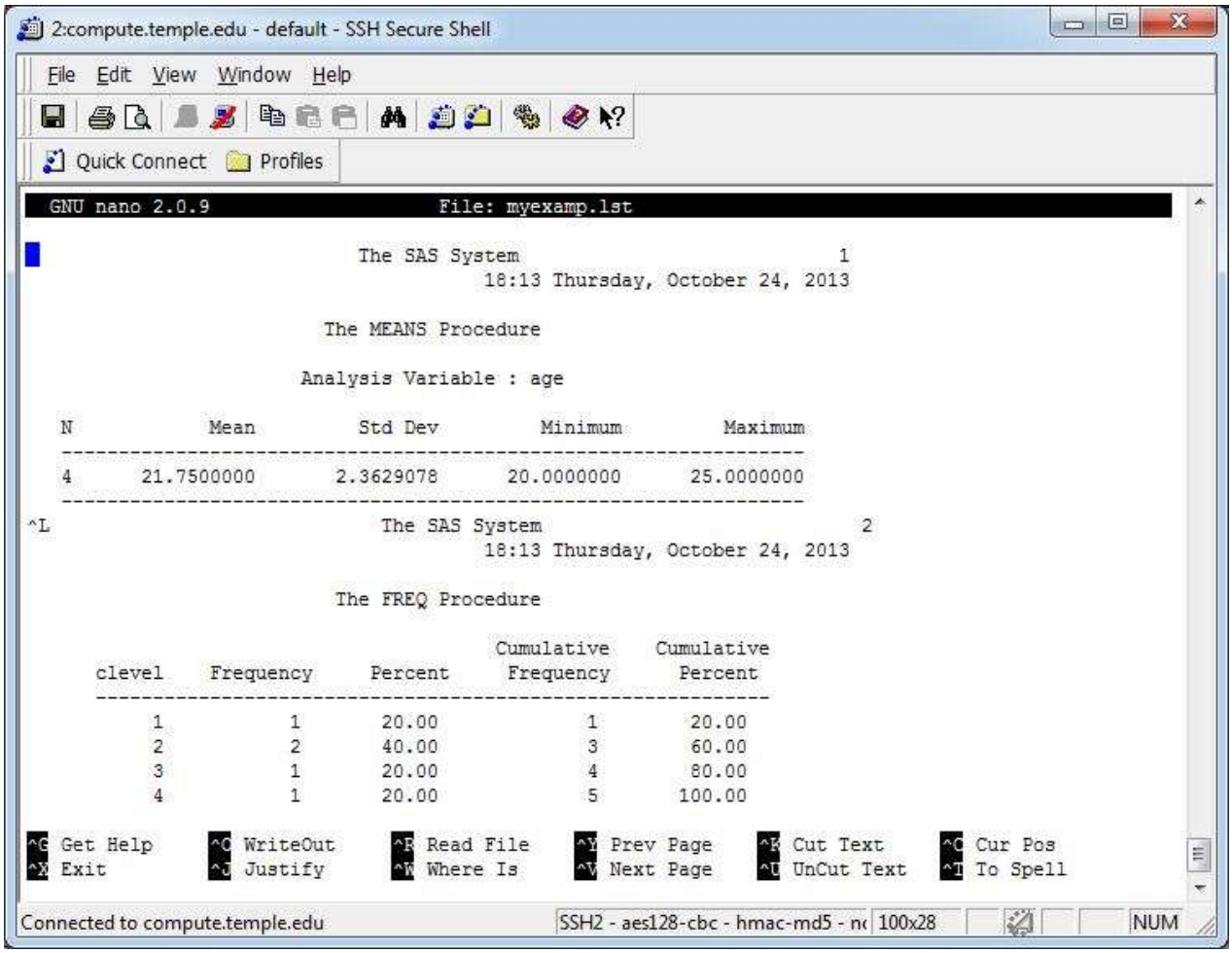


5. Two output files (i.e., the log and listing files) should appear in the directory used, and they should have the same filename as the SAS program file. If program errors occur, only the log file will be present.

   The program commands plus error messages will be seen in the log file. (See Figure 2 above). The results will be in the listing file. (See Figure 3 below) The log file will have the extension *.log* and the listing file will have the extension *.lst*. The complete names of the output files are: *myexamp.log* and *myexamp.lst* for this example since the program name that was executed was *myexamp.sas*.

6. Non-interactive execution of SAS does not involve the use of SAS windows (e.g., LOG, OUTPUT, PROGRAM EDITOR, etc.). These windows can only be used (and viewed) under LINUX when SAS is accessed through X-Windows emulation software, such as X-win32.

Figure 3: Sample SAS listing output file



Note:

1. When you first login to a LINUX account, you are automatically placed in your home directory. If you create the SAS program in your home directory, then both the log and listing files will also be stored in the home directory.

Note (Continued)

2. If you want to use a different directory (e.g., sasprog), then you must either use the ("cd") command to go to an existing directory, or use the ("mkdir") command to create a new directory.

3. To change your current directory to the sasprog directory, type this command at the LINUX prompt: **cd sasprog**.

4. To create the new directory name of sasprog, type this command at the LINUX prompt: **mkdir sasprog**. After you create the directory, you must use the "cd" command to go to the sasprog directory.

C. SAS Statements

The first word of a SAS statement (also called the keyword) tells you what kind of activity you want to perform (e.g., create a data set, run a statistical procedure, print lines of data, etc.). The remainder of the SAS statement provides more information about how you want the activity to be done (e.g., what to name the data set, which procedure to run, how to arrange the lines to be printed, etc.)

As seen on other platforms, LINUX SAS statements are divided into categories:

1. DATA step statements create SAS data sets. These statements are used to enter, describe, and modify the data; write reports; manage files; and retrieve information.

2. PROC step statements process or analyze SAS data sets. These statements identify which procedures are to be used and which variables in the data set are to be analyzed.

3. Other SAS statements can be used anywhere (e.g., OPTIONS statement). A SAS job consists of combinations of DATA steps and PROC steps. There is no special order to the statements except for those which relate to file usage, variable creation, and a logical sequence of steps. There are no special columns to be read, but the first word is usually placed in the first column and continuation lines for the same procedure are indented.

D. Data

Data may be included with SAS statements when the data size is small, or they may be stored in a separate file.

1. Data Included with SAS program:

The LINES statement must precede the data lines when data are in the same file as other SAS statements. (See Example 1)

2. Data in an external file.

   The INFILE statement is to be used when data are read from an external file. It works with the INPUT statement. (See Example 2)

3. Writing Data to an External file:

   The FILE statement is required when data are written to an output file. It works with the PUT statement. (See Example 5)

E. Data Sets

SAS system files are referred to as DATA SETS and are created by SAS in binary form. The form of the SAS DATA statement determines whether the data set is temporary (See Example 1) or permanent (see Example 3). The name of the data set cannot be more than 8 characters and must begin with a letter.

F. SAS Operators

SAS Operators are symbols (used within an IF statement, or an assignment statement) that request either a comparison, an arithmetic calculation, or a logical operation. See the charts below.

COMPARISON OPERATORS

Symbol	Mnemonic Equivalent	Definition
=	EQ	equal to
^= (or ~=)	NE	not equal to
>	GT	greater than
<	LT	less than
>=	GE	greater than or equal to
<=	LE	less than or equal to

F. SAS Operators (Continued)

ARITHMETIC OPERATORS

Symbol	Definition
**	exponentiation
*	multiplication
/	division
+	addition
-	subtraction

LOGICAL OPERATORS

Symbol	Definition
&	AND
|	OR
^ (or ~)	NOT

IV Terminology

Some of the statements and terms typically used with SAS are defined below and then used in the EXAMPLES Section on the following pages. The use of the word "statement" denotes an instruction or command that is included as a line within the SAS program. Statement names are shown in upper case for emphasis here, but can be typed in lower case within the SAS program.

DATA statement -- tells SAS to create a data set. It is typically the first line in a SAS program. (See Examples 1-3) When only one name is shown after the word DATA, a temporary data set is created.

INFILE statement -- used to identify the input data file. This statement must designate the filename and its location (e.g., directory, subdirectory). It precedes the INPUT statement. (See Example 2)

INPUT statement -- describes the variable names, type of variables in the data set (i.e., numeric or character) and the column location, if present. It is typically placed after the DATA statement. (See Examples 1-3)

FILE statement -- identifies the output file. It should be used in association with a LIBNAME statement to designate the name and location of the output file. It precedes the PUT statement. (See Example 5)

LIBNAME statement -- used to designate a *libref* with the directory where you want to do one of the following:

1. Store a permanent data set (Refer to Example 3)
2. Access a permanent data set (Refer to Example 4)
3. Store an output data file (Refer to Example 5)

LINES statement – precedes data lines when present in the Program file

PROC statement – identifies the procedure that will be used to read the data set, perform computations, and print results. (See Examples 1-4)

BY statement – used with a PROC statement to process the data set by groups. (See Example 1)

NULL – a keyword used in a DATA statement which allows the use of an existing data set without the creation of another one, thereby saving computer time. _NULL_ can also be used to create an output text file. (See Example 5)

LOG file – one of the two output files from a SAS job; it contains a listing of the SAS statements, processing notes, and error messages (if present). It is stored with the extension of LOG (can be upper or lower case).

LST file–one of the two output files from a SAS job; it contains the results of any procedure executed in the job. It is stored with the extension of LST (can be upper or lower case).

PROGRAM file – the file that contains SAS commands or SAS commands plus data lines. This type of file is stored with the extension of SAS (can be upper or lower case). Each of the five sample programs shown below is a complete SAS program that can be stored and executed. (See Examples 1-5)

V. Sample Programs

Example 1: SAS Program with Data Included

```
DATA sales;
INPUT salesrep $ sales region $ pctype $;
/* 1st 15 lines of data are shown */
LINES;
Stafer    20664  east    DELL
Young     22969  east    GW2K
Stride    27253  east    MAC
Topin     86432  east    HP
Spark     99210  east    DELL
Vetter    38928  west    DELL
Curci     21531  west    MAC
Marco     79345  west    GW2K
Greco     18523  west    IBM
Ryan      32915  west    CMPQ
Tomas     42109  west    CMPQ
Thal      94320  south   IBM
Moore     25718  south   HP
Allen     64700  south   IBM
Stelam    27634  south   GW2K
;
PROC PRINT;
RUN;
PROC FREQ;
    TABLES region region*pctype;
RUN;
PROC SORT;
    BY pctype;
RUN;
PROC MEANS;
    BY pctype;
RUN;
PROC ANOVA;
CLASS pctype;
MODEL sales=pctype;
MEANS pctype / TUKEY ALPHA = .05;
RUN;
```

Notes for Example 1:

1. This example creates a temporary data set named **work.sales** which contains four variables: salesrep, sales, region, and pctype.

2. Variables that have character values (e.g.,salesrep) must include the $ symbol after their names on the INPUT statement.

Notes for Example 1 (Continued)

3. No columns need to be specified on the INPUT statement when data is separated by spaces, has no missing data, and is lined up. However, do not use tabs to provide spaces between columns.

4. Multiple procedures are processed in the same job: Proc Print, Proc Freq, etc.

Example 2: Reading Data from an External Data File

```
DATA sales2;
INFILE 'onlydat.dat';
INPUT salesrep $ 1-7 sales 10-14 region $ 18-22 pctype $ 25-28;
PROC PLOT;
PLOT sales*region = pctype;
RUN;
```

Notes for Example 2:

1. The data are accessed from the file named **onlydat.dat**, which is in the current directory.

2. When there is a large data file being accessed that has missing data, especially at the end of a line, add the MISSOVER parameter on the INFILE statement.
(e.g., **INFILE 'onlydat.dat' MISSOVER;**)

3. If the data file were located in the home directory, while the SAS command file was in a different directory, the INFILE statement would be: **INFILE '~/onlydat.dat';**

Example 3: Creating a Permanent SAS Data Set

```
LIBNAME sales3 '~/sasfiles';
DATA sales3.perm;
INFILE 'onlydat.dat';
INPUT salesrep $ sales region $ pctype $ ;
RUN;
PROC UNIVARIATE;
VAR sales;
RUN;
```

Notes for Example 3:

1. At the system prompt, type **mkdir sasfiles** to create a directory with the name **sasfiles** before running this program.

2. On the LINUX account, the data set will be named **perm.sas7bdat.** It will be located in the directory named on the Libname statement (e.g., ~/sasfiles).

Example 4: Accessing a SAS Data Set

```
LIBNAME sales3 '~/sasfiles';
PROC CONTENTS DATA = sales3.perm;
RUN;
```

Notes for Example 4:

1. The internal SAS data set name (e.g., **sales3.perm**) is provided after the DATA= parameter on the PROC statement.

2. A LIBNAME statement is required to identify the location (e.g., the directory **~/sasfiles**) of the data set being accessed.

3. If the internal SAS data set name is not known, the user can supply an arbitrary name (8 characters maximum) that appears to the left of the decimal after the DATA = parameter and also appears after the word LIBNAME on the LIBNAME statement.

4. PROC CONTENTS will describe the variables within the data set.

Example 5: Creating an Output Text File

```
LIBNAME sales3 '~/sasfiles';
LIBNAME salesout '~/';
DATA _NULL_;
SET   sales3.perm;
IF region = 'east';
FILE salesout;
PUT salesrep $ 1-10 sales 12-16;
RUN;
```

Notes for Example 5:

1. The SET command accesses the data set **sales3.perm.** The first LIBNAME command indicates the directory (e.g., **sasfiles)** in which the data set being accessed is located.

2. The output data file **salesout.dat** is stored in the directory in which the program file was executed, no matter where the program file is stored. In this example, the program file was stored in the **sasfiles** directory but was executed from the home directory, using the command: *sas ~/sasfiles/myfile* (where *myfile.sas* is the name of the program file and the symbol „~/" indicates the home directory). Thus, the output data file was stored in the home directory.

3. A subset of the data is retrieved (e.g., region = "east") by the IF statement. By means of the FILE and PUT statements, two variables, salesrep and sales, were written at specified column locations to the output file, which now contains a subset of the data. The new output text file is called **salesout.dat**, where SAS added the .dat extension.

Notes for Example 5 (Continued)

4. An alternate method of running this program is to remove line: LIBNAME salesout '~/' and replace: FILE salesout by **FILE 'salesout.dat'** if you want to place the output text file in the home directory. Use **FILE '~/sasfiles/salesout.dat'** if you want to place the output text file in the sasfiles directory. By means of this alternate approach, the output text file is placed in the specified directory no matter where the program file is executed.

VI Differences Between SAS for LINUX and SAS for Windows

SAS for LINUX (Non-Interactive mode)

1. Non-interactive mode, which does not have menus or windows, is used when X-Windows emulation software is not accessed or available.
2. Stored output files are given the extension .log for the LOG file and .lst for the LISTING file.
3. To print program or output files:
   Use SSH Secure Shell to link to LINUX; use Nano to open the file; highlight lines of inter- est; go to File, then Print, and specify Selec- tion on the Print screen before clicking OK.
4. No function keys are available
5. None of the windows (e.g., LOG, OUTPUT, PROGRAM EDITOR, etc.) are available after a program is executed.
6. No command box, menu bar, or menu options are available.
7. Do not have interrupt capability; can execute LINUX commands from within a SAS program.
8. SAS automatically terminates after it executes a SAS program.
9. SAS/ASSIST, SAS/INSIGHT, and SAS/ LAB cannot be used in non-interactive mode.
10. SAS Help is not available unless X-windows software (e.g., X-win32) is used

SAS for Windows

1. Interactive mode, which involves the use of menus and windows, is more likely to be used
2. Stored output files are given the extension .log for the LOG file and .lst for the LISTING file.
3. To print program or output files, make the appropriate window active, use the File menu and select Print.
4. Function keys can be used with the Program Editor screen
5. Multiple windows are displayed together under interactive mode (e.g., ENHANCED EDITOR, LOG, RESULTS)
6. The command box, menu bar, and menu options are available
7. Interrupt capability is available to allow exit to Windows and then return to SAS.
8. To exit from interactive mode, type BYE or ENDSAS on the command line of any screen or use the command box
9. SAS/ASSIST, SAS/INSIGHT, and SAS/LAB are available.
10. SAS Help is available

VII. Exercises

1. Do Example 1

2. Place the data in a separate file using the Nano Editor (i.e., Follow Step 1 under Sequence of Commands as described below)

3. Do Examples 2-5.

VIII. Sequence of Commands

Step 1: Using the Nano Editor

Type the following: *nano filename.ext*

The above command creates a file called "filename.ext". Filenames are arbitrary. However, extensions should relate to the type of file involved. Files that have the .sas extension should contain SAS commands, while files that contain data should have .dat or .data as their extension.

Step 2: Entering Commands

SAS program commands can be typed at the cursor position after Nano has been initiated. Use the Enter key to advance to the next line. Specific Nano commands are shown at the bottom of the screen. For example, to move forward, type the "control" and "v" keys together. To search for text within the document, use the “control” and “w” keys and then enter the text for which you are searching. Make any changes to the program as needed.

Step 3: Save changes and exit Nano

To save changes and exit Nano, type the Ctrl and O keys together. Next, at the “File Name to write:” prompt, provide a new filename or accept the current file name and hit the Enter key. The changes to the file will now be stored. Type the Ctrl and X keys together to exit from Nano.

Step 4: Execute the SAS syntax file

At the LINUX prompt, type: *sas filename*

This executes the SAS program file (e.g., filename.sas) and creates two output files which are called "filename.log” and “filename.lst”.

Step 5: Printing Files

There are various ways to print the SAS program and output files that are stored in a LINUX account. The methods described below imply the use of SSH Secure Shell software to link the PC to the Compute LINUX Server. The SSH software can be downloaded from the download.temple.edu website.

1. Use SSH Secure Shell and Nano Editor
   After linking to LINUX by means of SSH Secure Shell software, use the Nano editor to open the file. Then highlight the lines of the file that you want printed, go to the File menu, and click Print. When the Print dialog box appears, click on the Selection option prior to clicking OK. The selected lines of the file will then be printed by your local printer.

2. Use SSH Secure Shell and Word Processing software
   Download the file to be printed using SSH Secure Shell software. Then access the file with a word processing package (e.g.,Word), from which the file can be printed.