CHAPTER 8 Data Representation

Introduction

This chapter provides guidelines for deciding which formats and informats to use with binary data and describes how character and numeric variables and missing values are represented on UNIX platforms.

Reading and Writing Binary Data

Different computers store numeric binary data in different forms. Hewlett-Packard, SUN, and RS/6000 store bytes in one order. IBM-compatible microcomputers and Compaq Tru64 UNIX (formerly Compaq's DIGITAL UNIX), which are all Intel-based UNIX systems, store bytes in a different order, called byte-reversed.

Binary data stored in one order cannot be read by a computer that stores binary data in the other order. When you are designing SAS applications, try to anticipate how your data will be read and choose your formats and informats accordingly.

The SAS System provides two sets of informats and formats for handling binary data:

- The IB*w.d*, PD*w.d*, PIB*w.d*, and RB*w.d*. informats and formats read and write data in native mode, that is, using the byte-ordering system that is standard for the machine.
- The S370FIB*w.d*, S370FPD*w.d*, S370FPIB*w.d*, and S370FRB*w.d* informats and formats force the data to be read and written according to the IBM 370 standard, regardless of the native mode of the machine. These informats and formats allow you to write SAS programs that can be run in any SAS environment, regardless of how numeric data is stored.

If a SAS program that reads and writes binary data runs on only one type of machine, you can use the native mode informats and formats. However, if you want to write SAS programs that can be run on multiple machines using different storage systems for numeric data, use the S370 formats and informats, which exist for that purpose.

For example, suppose you have a program that writes data with the PIB*w.d* format. You execute the program on a microcomputer so the data is stored in byte-reversed mode. Then you run another SAS program on the microcomputer that uses the PIB*w.d* informat to read the data. The data is read correctly because both the programs are run on the microcomputer, using byte-reversed mode. However, you cannot upload the data to a Hewlett-Packard 9000-series machine and read it correctly, because it is stored in a form native to the microcomputer but foreign to the Hewlett-Packard 9000. To avoid this problem, use the S370FPIB*w.d* format to write the data; even on the microcomputer, this causes the data to be stored in IBM 370 mode. Then read the data using the S370FPIB*w.d* informat. Regardless of which type of machine you use when reading the data, it is read correctly.

For more information on all of the informats and formats, refer to *SAS Language Reference: Dictionary*.

Character Variables

In the SAS System under UNIX, character values are sorted by using the ASCII collating sequence.

The following character informats and formats produce different results in different operating environments, depending on which character-encoding system the operating system uses. Because UNIX uses the ASCII character-encoding system, the following informats and formats convert data to or from ASCII.

$ASCII*w*.
: Since ASCII is the native format under UNIX, this informat performs no conversion. The $ASCII format behaves exactly like $CHAR (see *SAS Language Reference: Dictionary*).

$BINARY*w*.
: This informat converts binary values to ASCII character data, and this format converts ASCII character data to binary values.

$CHARZB*w*.
: This informat reads character data and converts any byte that contains a binary zero to a blank.

$EBCDIC*w*.
: This informat converts EBCDIC character data to ASCII, and this format converts ASCII character data to EBCDIC.

$HEX*w*.
: This informat converts hexadecimal data to ASCII character data, and this format converts ASCII character data to hexadecimal data.

$OCTAL*w*.
: This informat converts octal data to ASCII character data, and this format converts ASCII character data to octal data.

$PHEX*w*.
: This informat converts packed hexadecimal data to ASCII character data.

All of the information that you need to use these informats and formats under UNIX is in *SAS Language Reference: Dictionary*.

Character dummy variables (those whose only purpose is to hold 0 or 1) can be stored in a variable whose length is 1 byte.

Numeric Variable Length and Precision

The default length of numeric variables in SAS data sets is 8 bytes. (You can control the length of SAS numeric variables with the LENGTH statement in the DATA step.)

The issue of numeric precision affects the return values of almost all SAS System math functions and many numeric values returned from SAS System procedures. Numeric values in the SAS System for UNIX are represented as IEEE double-precision floating-point numbers. The decimal precision of a full 8-byte number is approximately 16 decimal digits.

Table 8.1 on page 161 specifies the significant digits and largest integer that can be stored exactly in SAS numeric variables.

Table 8.1 Significant Digits and Largest Integer by Length for SAS Variables under UNIX

Length in Bytes	Largest Interger Represented Exactly	Exponential Notation
3	8,192	2^{13}
4	2,097,152	2^{21}
5	536,870,912	2^{29}
6	137,438,953,472	2^{37}
7	35,184,372,088,832	2^{45}
8	9,007,199,254,740,992	2^{53}

When you are specifying variable lengths, keep in mind that variable length affects both the amount of memory used and the time required for I/O and arithmetic operations. See *SAS Language Reference: Dictionary* for more information on specifying variable lengths.

If you know that a numeric variable will be between 0 and 100, you can use a length of 3 to store the number and thus save space in your data set. For example:

```
data mydata;
   length num 3;
   ...more SAS statements...
run;
```

Numeric *dummy variables* (those whose only purpose is to hold 0 or 1) can be stored in a variable whose length is 3 bytes.

CAUTION:

Use 3 bytes only for those variables with small values, preferably integers. If the value of a variable becomes large or has many significant digits, you may lose precision in arithmetic calculations when the length of a variable is less than 8 bytes. △

For more information on specifying variable lengths and optimizing system performance, refer to *SAS Language Reference: Concepts*.

Missing Values

In the SAS System on UNIX, missing values are represented by IEEE Not-a-Number values. An IEEE Not-a-Number value is an IEEE floating-point bit pattern that represents something other than a valid numeric value. These numbers are not computationally derivable.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for UNIX Environments, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS® Companion for UNIX Environments, Version 8

ISBN 1–58025–502–7

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999