Chapters 1, 2, 5, and 7 MISCELLANEOUS

Chapter 1. DESCRIPTOR AND DATA PORTIONS OF SAS DATA FILES

Conceptually, a SAS data set is a file that consists of two parts: a **descriptor portion** and a **data portion**.

* The **descriptor portion** of a SAS data set contains information about the data set, including
* the name of the data set
* the date and time when the data set was created
* the number of observations
* the number of variables
* The **data portion** of a SAS data set is a collection of data values that are arranged in a rectangular table. Observations for a particular individual are stored in **rows** and variables are stored in **columns**.

In addition to general information about the data set, the descriptor portion contains information about the attributes of each variable in the data set. The attribute information includes the variable’s name, type (numeric or character), length, format, informat, and label.

In SAS, the **length** of a variable is the number of bytes SAS allocates for storing the variable. It is not necessarily the same as the number of characters or digits in the variable.

* Character variables can have values up to 32, 767 bytes long.
* Numeric variables have a default length of 8 bytes. Variables containing integer values can be stored using less than 8 bytes.

|  |  |
| --- | --- |
| Largest integer represented exactly  by length for SAS variables | |
| Length in bytes | Largest integer represented exactly |
| 3 | 8,192 |
| 4 | 2,097,152 |
| 5 | 536,870,912 |
| 6 | 137,438,953,472 |
| 7 | 35,184,372,088,832 |
| 8 | 9,007,199,254,740,990 |

Chapter 2. SAS LIBRARY ENGINES

Data formatted by other software products may be read by SAS without using Import/Export Wizard. It suffices to specify the engine type in the LIBRARYNAME statement. The syntax is

LIBNAME libref engine 'path\filename';

Note that the **actual filename** is specified, not just the path. This is an exception to the rules of libref’s.

~~Example. libname mylib excel 'C:\Users\mstudent\Desktop\exercise.xls';~~

~~options validmemname=extend;~~

~~libname mylib1 xlsx 'C:\Users\mstudent\Desktop\exercise.xlsx';~~

~~options validmemname=extend;~~

~~libname mylib2 csv 'C:\Users\mstudent\Desktop\exercise.csv';~~

~~options validmemname=extend;~~

~~SAS stores the datafile Sheet1. To print the file, for example, type~~

~~proc print data=libref.sheet1~~

~~run;~~

THE CONTENTS PROCEDURE

* **PROC CONTENTS** may be used to view the contents of a library. The syntax is

proc contents data=libref.\_all\_ nods;

run;

where the filename \_all\_ requests a listing of **all** files in the library, and nodetails (or nods) suppresses the printing of detailed information about each file. Nodetails option may be used only if the filename \_all\_ is specified.

Example. To view without any details the contents of the library MyLib that contains a single file exercise, type

proc contents data=MyLib.\_all\_ nods;

run;

The output is

The SAS System 19:04 Monday, January 29, 2018 1

The CONTENTS Procedure

Directory

Libref MyLib

Engine V9

Physical Name I:\

Filename I:\

Member File

# Name Type Size Last Modified

1 EXERCISE DATA 5120 28Jan10:19:05:30

Note that the output includes the list of filenames, type of files, file sizes, dates and times of last modification.

* **PROC CONTENTS** may be used to view the descriptor information for a particular SAS data set. The syntax is

proc contents data=libref.*filename*;

run;

Example. To view the descriptor information for the file exercise, type

proc contents data=MyLib.exercise;

run;

The output is

The SAS System 19:04 Monday, January 29, 2018 3

The CONTENTS Procedure

Data Set Name MYLIB.EXERCISE Observations 6

Member Type DATA Variables 4

Engine V9 Indexes 0

Created Tuesday, July 20, 2010 07:05:29 PM Observation Length 24

Last Modified Tuesday, July 20, 2010 07:05:29 PM Deleted Observations 0

Protection Compressed NO

Data Set Type Sorted NO

Label

Data Representation WINDOWS\_32

Encoding wlatin1 Western (Windows)

Engine/Host Dependent Information

Data Set Page Size 4096

Number of Data Set Pages 1

First Data Page 1

Max Obs per Page 168

Obs in First Data Page 6

Number of Data Set Repairs 0

Filename I:\exercise.sas7bdat

Release Created 9.0202M2

Host Created XP\_PRO

Alphabetic List of Variables and Attributes

# Variable Type Len Format Informat Label

3 actlevel Char 4 $4. $4. actlevel

2 age Num 8 age

1 id Num 8 id

4 gender Char 1 $1. $1. gender

Note that the output includes the list of variables in alphabetical order, types of variables, lengths, formats, informats, and labels. The list of variables in alphabetical order comes in handy when Excel data file is imported into SAS and it has a very large number of variables, so that a particular variable name may be looked up.

THE DATASETS PROCEDURE

Alternatively to PROC CONTENTS step, **PROC DATASETS** may be used. For viewing the contents of a library, the syntax is

proc datasets;

contents data=libref.\_all\_ nods;

quit;

For viewing the descriptor information of a data file, the syntax is

proc datasets;

contents data=libref.*filename*;

quit;

Note that the DATASETS procedure requires a **QUIT statement** rather than a RUN statement.

THE VARNUM OPTION

By default, PROC CONTENTS and PROC DATASETS list variables in alphabetical order. To list variable names in their **creation order** (as they appear in columns in the dataset), use the **VARNUM option**.

Example. To view the descriptor information for the file exercise, type

proc contents data=MyLib.exercise varnum;

run;

or

proc datasets;

contents data=MyLib.exercise varnum;

quit;

The list of the variable names in the output is

Variables in Creation Order

# Variable Type Len Format Informat Label

1 id Num 8 id

2 age Num 8 age

3 actlevel Char 4 $4. $4. actlevel

4 gender Char 1 $1. $1. gender

Chapter 5. THE FILENAME STATEMENT

To reference a data file on external drive, the following syntax in the DATA step was used

data *dataname*;

infile 'path\filename';

input *variable1* *variable2* …;

run;

Alternatively, the **FILENAME statement** may be used. The syntax is

filename *fileref* 'path\filename';

data *dataname*;

infile *fileref*;

input *variable1* *variable2* …;

run;

Example. Both programs below read the data file correctly.

data exercise;

infile ' C:\Users\mstudent\Desktop\exercise.dat';

input id $ age actlevel $ gender $;

run;

or

filename test 'C:\Users\mstudent\Desktop\exercise.dat';

data exercise;

infile test;

input id $ age actlevel $ gender $;

run;

DATE, TIME AND DATETIME CONSTANTS

* Suppose it is necessary to assign to all individuals in a data set a variable that has a constant date value. A **constant date** must have the format

**'**ddmmm<yy>yy**'**D or "ddmmm<yy>yy"D

where dd is a one- or two-digit value for the day, mmm is a three-letter abbreviation for the month (JAN, FEB, etc.), and yy or yyyy is a two- or four-digit value for the year.

Example. The data below are surgery dates for several individuals. The current date is assigned as a constant date, and the duration of survival is computed.

data survival;

input surgdate date9.;

currentdate='30jan2018'd;

duration=(currentdate-surgdate)/365.25;

cards;

15dec2005

4apr1999

11jul2010

;

proc print;

format surgdate date9. currentdate date9.;

run;

The output is

Obs surgdate currentdate duration

1 15DEC2005 30JAN2018 12.1259

2 04APR1999 30JAN2018 18.8255

3 11JUL2010 30JAN2018 7.5565

* SAS **time constants** have the format **'**hh:mm<:ss.s>**'**T or "hh:m<:ss.s>"T . For example, time='9:25't or time="9:25:19pm"t.
* SAS **datetime contants** have the format 'ddmmm<yy>yy:hh:mm<:ss.s>'DT or "ddmmm<yy>yy:hh:mm<:ss.s>"DT. For example, dtime='01may04:9:30:00'dt or dtime=**"**18jan2003:9:27:05am**"**dt.

SUBSETTING DATA USING IF STATEMENT

To process only those observations that meet a specified condition, the **IF statement** may be used in any DATA step. The syntax is

IF *expression*;

Example. In the exercise example, to print the data for females only, type

data exerciseF;

set exercise;

if gender='F';

run;

proc print;

run;

The output is

Obs id age actlevel gender

1 2810 61 MOD F

2 2804 38 HIGH F

3 2833 32 MOD F

THE \_NULL\_ DATA STEP

The **\_NULL\_ DATA step** is a specialized case of the DATA step. When the \_NULL\_ keyword is specified, all statements within the DATA step are processed without a data set being created. The following statements should be specified.

data \_null\_;

set libref.*filename*;

file **'**path\*filename1.dat***'**;

put *variable1* startcol-endcol *variable2* startcol-endcol … ;

run;

Here the data are read from the data file libref.*filename*, and written to the file *filename1.dat*. The PUT statement lists the variable names, and starting and ending columns for each variable. Because the output file is a raw data file with extension .dat, there is no need to follow character variables with a dollar sign ($).

Example. In the exercise example, after the DATA step that creates the exercise data file, type

data \_null\_;

set exercise;

file ' C:\Users\mstudent\Desktop\exerciseout.dat';

put id 1-4 actlevel 9-12;

run;

The contents of the file exerciseout.dat is

2810 MOD

2804 HIGH

2807 LOW

2816 HIGH

2833 MOD

2823 HIGH

Chapter 7 PERMANENTLY STORING VARIABLE FORMATS

When PROC FORMAT statement is used to create a format, the format is stored in a **format catalog**. A default format catalog is **Work.Formats**. The formats that are stored in this catalog are **temporary** and exist only until the SAS session ends.

To assign a **permanent** **format catalog**, the library has to be specified. The syntax is

libname *libref* 'path';

proc format library=*libref*;

…

run;

Now the formats that are created in this PROC FORMAT statement are stored in a permanent format catalog called **Libref.Formats**.

* An abbreviation lib may be used instead of library:

libname *libref* 'path';

proc format lib=*libref*;

…

run;

DISPLAYING A LIST OF VARIABLE FORMATS

When a format catalog is created for a large number of variables, it is easy to forget the exact labels or range of values. The keyword **FMTLIB** added to the PROC FORMAT statement displays a list of all the formats in the format catalog, along with descriptions of their values. The syntax is

libname *libref* 'path';

proc format library=*libref* fmtlib;

…

run;

Example. The following SAS program reads instream data on student id, gender, and grade. The permanent format catalog is created called **Grades.Formats**. Its contents is displayed as output.

data grades;

input id $ gender $ grade $;

cards;

0041 F A

6686 M D

4859 M F

4448 M C

8154 F A

;

libname grades ' C:\Users\mstudent\Desktop\';

proc format lib=grades fmtlib;

value $genderfmt 'F'='Female'

'M'='Male'

;

value $gradefmt 'A'='Excellent'

'B'='Good'

'C'='Fair'

'D'-'F'='Poor'

;

run;

The output is

|  |
| --- |
| The SAS System |

|  |
| --- |
| „ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ†  ‚ FORMAT NAME: $GRADEFMT LENGTH: 9 ‚  ‚ MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 9 FUZZ: 0 ‚  ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ…ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ…ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‰  ‚START ‚END ‚LABEL (VER. 9.3 22JUL2014:14:19:21)‚  ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‰  ‚A ‚A ‚Excellent ‚  ‚B ‚B ‚Good ‚  ‚C ‚C ‚Fair ‚  ‚D ‚F ‚Poor ‚  Šƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‹ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‹ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒŒ      „ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ†  ‚ FORMAT NAME: $GENDERFMT LENGTH: 6 NUMBER OF VALUES: 2 ‚  ‚ MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 6 FUZZ: 0 ‚  ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ…ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ…ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‰  ‚START ‚END ‚LABEL (VER. V7|V8 22JUL2014:14:19:21)‚  ‡ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒˆƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‰  ‚F ‚F ‚Female ‚  ‚M ‚M ‚Male ‚  Šƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‹ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ‹ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒŒ |