STA 311 Statistical Computing & Data Management

Instructor: Cheng Peng
Department of Mathematics
West Chester University
West Chester, PA 19383

Office: 25 University Avenue, RM 111
Phone: 610.436.2369
Email: cpeng@wcupa.edu

Topic 3. SAS Data Input and Output

WCU WEST CHESTER UNIVERSITY

Topics

1. Three styles of manual input
2. Length of SAS variables
3. A first look at SAS dates
4. Reading external data files
5. Data file is not aligned
6. Dealing with “messy” data file
7. Writing a nice SAS program.

SAS Data Sets: A Brief Review

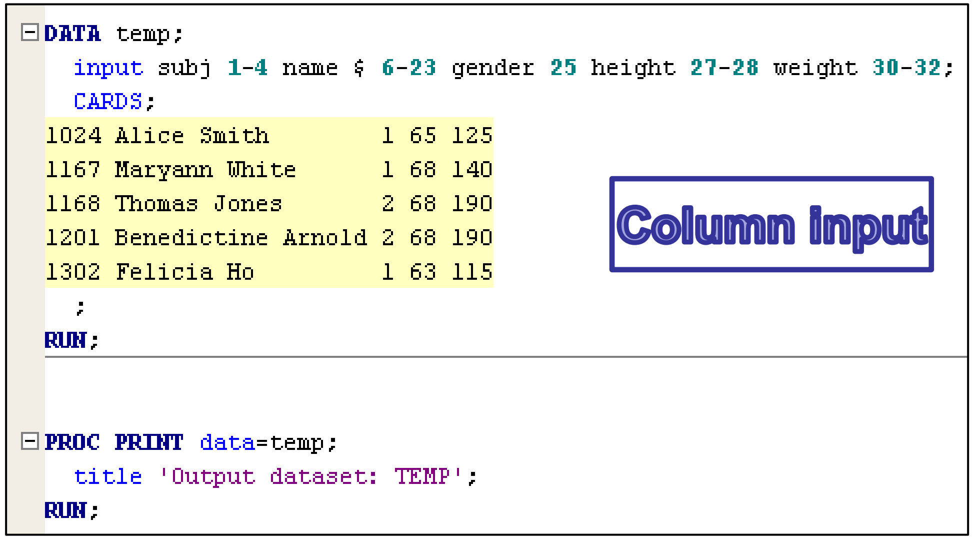
SAS Data Set - a binary formatted representation of the input data set stored in such a way that future SAS programs do not need to input the data in again.

Temporary SAS Data Sets - created and remain in working memory for the SAS session, but disappear when the SAS session ends. Fine for small to moderate size, simple input programs.

Permanent SAS Data Sets - created in one SAS session but stored on disk for later reuse. Convenient for large or complex input data sets that may require multiple analysis steps. Reduces time and computer resources.

LIBNAME statement - identifies to the SAS program where the previously created SAS data set is located.

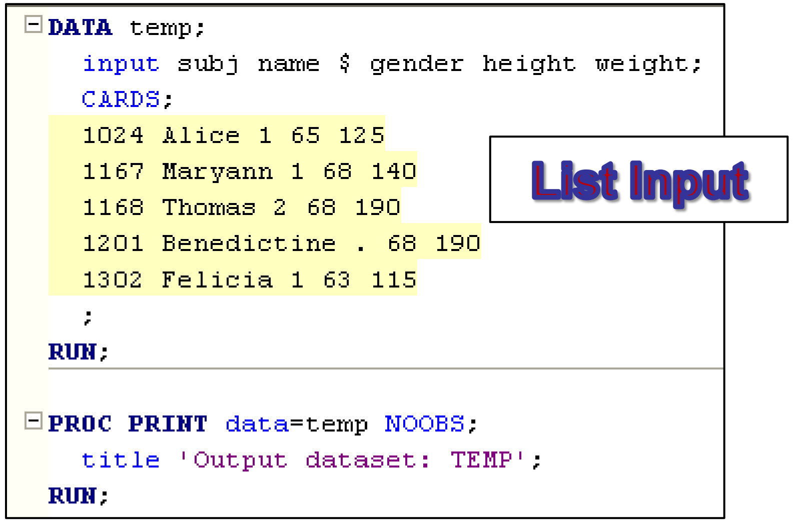
Three Basic Input Styles: An Overview



```
DATA temp;
   input subj 1-4 name $ 6-23 gender 25 height 27-28 weight 30-32;
   CARDS;
1024 Alice Smith        1 65 125
1167 Maryann White      1 68 140
1168 Thomas Jones       2 68 190
1201 Benedictine Arnold 2 68 190
1302 Felicia Ho         1 63 115
   ;
RUN;


PROC PRINT data=temp;
   title 'Output dataset: TEMP';
RUN;
```

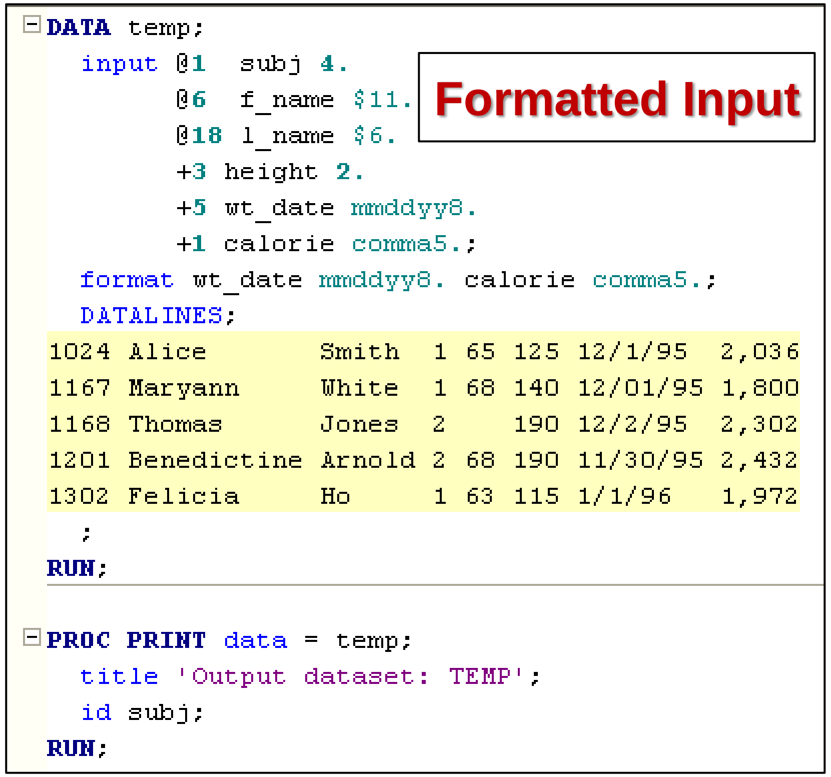
Column input



```
DATA temp;
   input subj name $ gender height weight;
   CARDS;
   1024 Alice 1 65 125
   1167 Maryann 1 68 140
   1168 Thomas 2 68 190
   1201 Benedictine . 68 190
   1302 Felicia 1 63 115
   ;
RUN;

PROC PRINT data=temp NOOBS;
   title 'Output dataset: TEMP';
RUN;
```

List Input



```
DATA temp;
   input @1  subj 4.
         @6  f_name $11.
         @18 l_name $6.
         +3 height 2.
         +5 wt_date mmddyy8.
         +1 calorie comma5.;
   format wt_date mmddyy8. calorie comma5.;
   DATALINES;
1024 Alice       Smith  1 65 125 12/1/95  2,036
1167 Maryann     White  1 68 140 12/01/95 1,800
1168 Thomas      Jones  2    190 12/2/95  2,302
1201 Benedictine Arnold 2 68 190 11/30/95 2,432
1302 Felicia     Ho     1 63 115 1/1/96   1,972
  ;
RUN;

PROC PRINT data = temp;
   title 'Output dataset: TEMP';
   id subj;
RUN;
```

Formatted Input

We will see some hybrid input styles later!

Setting the Length of a Variable

- ❑ **The default length for character and numeric variables is 8 bytes in SAS.**

- ❑ **SAS uses exactly one byte for one character! This means that if the value of a character variable has more than 8 characters, SAS only keeps the first 8 characters (including the white space if any) and truncate the rest.**

- ❑ However, for a **numeric variable SAS variable, 8 bytes can store a number with up to 16 digits**. In other words, the default length of numeric variable is 16 digits.

- ❑ It is important to note that the **minimum length of a numeric is 3 bytes**. It does not mean it cannot store a numeric value lower than 3 digits. It can store values of 1 or 2 digits.

- ❑ The **maximum length** of any **character** value in **SAS** is 32,767 bytes!

Setting the Length of a Variable

Significant Digits and Largest Integer by Length for SAS Variables under Windows

Length in Bytes	Largest Integer 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}

Tip: Please note you cannot store accurately more than about 16 digits in a SAS numeric variable, so you need to import your column as a character variable and then do string manipulation and data type conversion.

Summary of Setting the Length of a Variable

- ❑ In SAS, both numeric and character variables have length of 8 bytes by default.

- ❑ Using the default length of **character variable** usually causes truncation issue.

- ❑ You can change the **length** of the **variable** by using a subsequent Length statement.

- ❑ The maximum **length** of any **character variable** in the **SAS** System is 32,767 bytes

Setting the Length of a Variable: Example

General form of a LENGTH statement:

LENGTH *variable-name <$> length-specification (bytes) ...*;

```
data airplanes;
   length ID $ 5;
   infile 'raw-data-file';
   input ID $
         InService : date9.
         PassCap CargoCap;
run;
```

Reads external files using INFILE statement!

The colon modifier tells SAS when it reads in **InService** to do it until there is a break in the character and then stop **Omitting this colon may cause error**

The value of a SAS date = # of days from January 1, 1960 to the given date! ➔ see more detail on next few slides!

Setting the Length of a Variable

Caution!

LENGTH *variable-name* <**$**> *length-specification* (*bytes*);

Example: **LENGTH** *state* **$ 20**;

- ❑ **Variable lengths specified in a LENGTH statement affect the length of numeric variables only in the output data set; during processing, all numeric variables have a length of 8 bytes.**
- ❑ **Lengths of character variables specified in a LENGTH statement affect both the length during processing and the length in the output data set.**

Different ways to read data into SAS

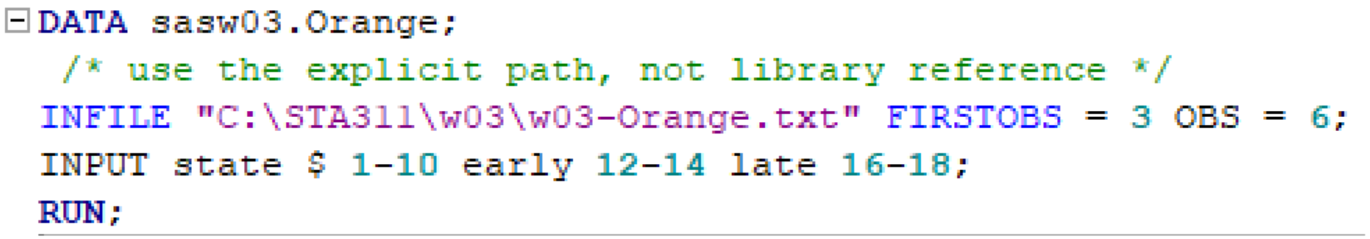
Reading Data from an External Text File

C:\STA311\w03\w03-Orange.txt ← external text file

```
Projected Orange Yields in October 1997
State    Early Late
Florida    130  90
California  37  26
Texas      1.3 .15
Arizona    .65 .85
Based on information obtained from the
Florida Agricultural Statistics Service
```

Line 3

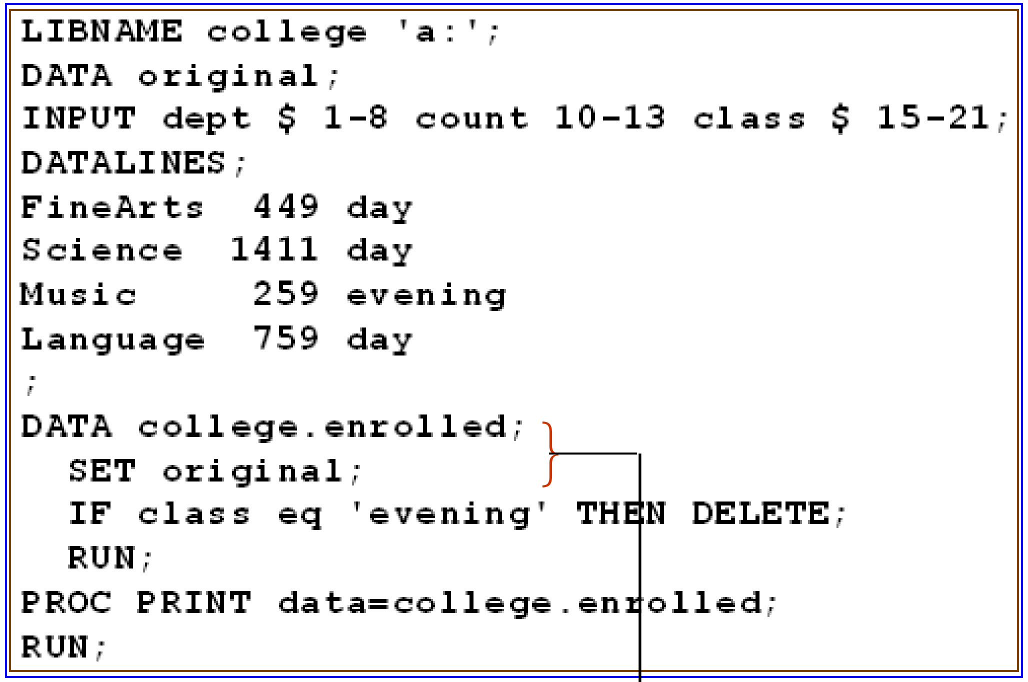
Line 6



```
DATA sasw03.Orange;
 /* use the explicit path, not library reference */
INFILE "C:\STA311\w03\w03-Orange.txt" FIRSTOBS = 3 OBS = 6;
INPUT state $ 1-10 early 12-14 late 16-18;
RUN;
```

Create Permanent SAS Data Sets

If a permanent data set is created, store it here.



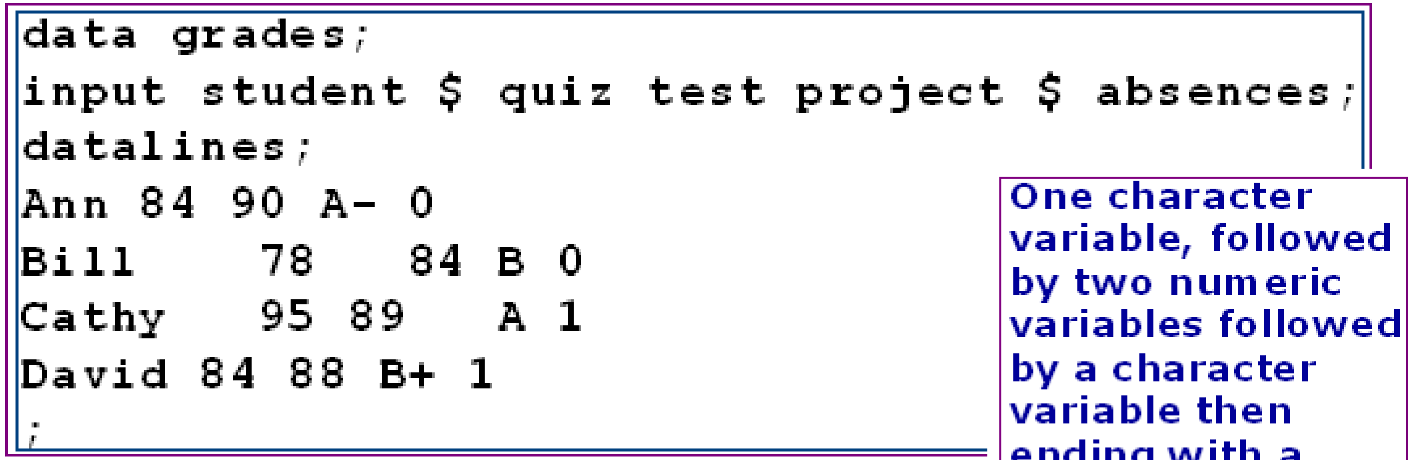
```
LIBNAME college 'a:';
DATA original;
INPUT dept $ 1-8 count 10-13 class $ 15-21;
DATALINES;
FineArts  449 day
Science  1411 day
Music      259 evening
Language  759 day
;
DATA college.enrolled;
  SET original;
  IF class eq 'evening' THEN DELETE;
  RUN;
PROC PRINT data=college.enrolled;
RUN;
```

Make a permanent data set called ENROLLED and place it in the COLLEGE archive.

ENROLLED.sas7bdat stored on the A drive

Create data set *name_1* from data set *name_2*

Input Methods: List Input



```
data grades;
input student $ quiz test project $ absences;
datalines;
Ann 84 90 A- 0
Bill    78   84 B 0
Cathy   95 89   A 1
David 84 88 B+ 1
;
```

One character variable, followed by two numeric variables followed by a character variable then ending with a numeric variable.

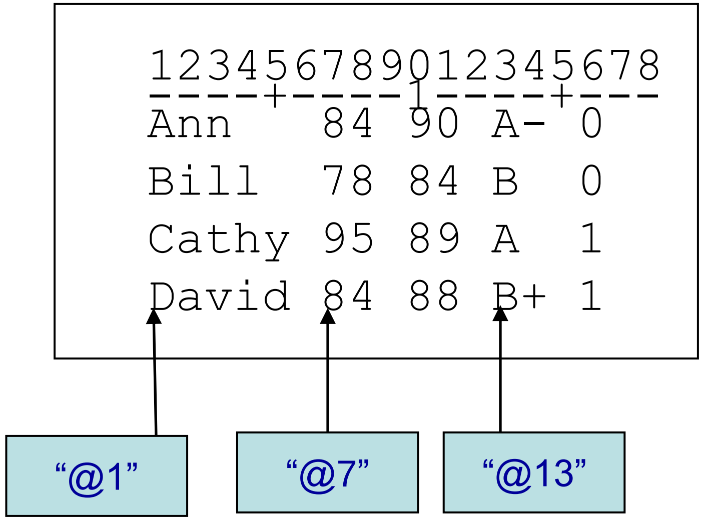
As simple as it gets.

Data columns are not lined up, but SAS doesn't care.

- At least one blank between variable data.
- No spaces in character data.
- Character data with less than or equal to 8 characters.

What if student is **Elizabeth** or **Mary Beth**?
What if **B+** is entered as **B Plus** or **B_Plus?**

Input Methods: Column Pointers



```
123456789012345678
----+----1----+---
Ann   84 90 A- 0
Bill  78 84 B  0
Cathy 95 89 A  1
David 84 88 B+ 1
```

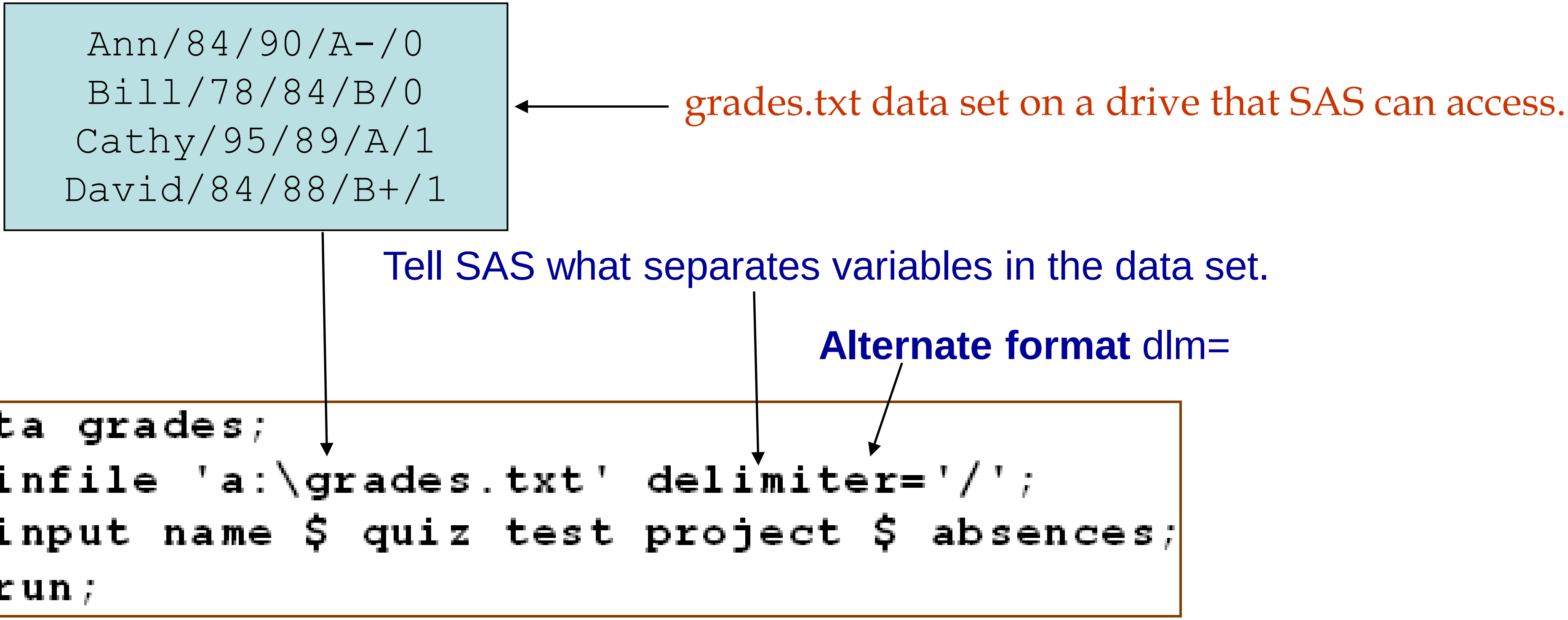
```
data grades;
infile ‘C:\grades.txt’ firstobs=2;
input @1 name $
      @7 q1
      @10 q2
      @13 project $
      @16 att;
run;
```

With column pointers you can tell SAS directly which column to begin reading a variable from.

- Go directly to the information you really need.
- Skip unnecessary information.
- Efficient data entry.

Input Methods: Delimited (Formatted)

DELIMITER - character used to separate items.



```
Ann/84/90/A-/0
Bill/78/84/B/0
Cathy/95/89/A/1
David/84/88/B+/1
```

grades.txt data set on a drive that SAS can access.

Tell SAS what separates variables in the data set.

Alternate format dlm=

```
data grades;
  infile 'a:\grades.txt' delimiter='/';
  input name $ quiz test project $ absences;
  run;
```

If the file delimiter is a tab - e.g. from EXCEL or LOTUS Tab-delimited file.

```
infile ‘C:\grades.txt' expandtabs;
```

Mixed Input

Methods of Input

Column Input

input state $ 1-10 early 12-14 late 16-18;

List Input

input student $ quiz test project $ absences ;

Column Pointers

input @1 name $ @13 project $;

```
12345678901234567890
Ann       84 90 A- 0
Bill      78 84 B  0
Catherine 95 89 A  1
David     84 88 B+ 1
```

```
data grades;
  infile 'a:\grades.txt’ firstobs=2;
  input name $ 1-9 quiz test project $ absences;
  run;
```

OR

```
data grades;
  infile 'a:\grades.txt’ firstobs=2;
  input name $ 1-9 @17 project $ absences;
  run;
```

SAS allows you to mix input types.

Line Pointers- Multiple Lines

```
Ann
84 90 A- 0
Bill
78 84 B 0
Cathy
95 89 A 1
David
84 88 B+ 1
```

Data may come to us with each observation recorded on more than one line. We need to be able to tell SAS to go to the next line.

- The slash (/) says skip to next line.
- The code #n says go to that line of the observations data to resume reading data.

```
data grades;
infile 'a:\grades.txt';
input name $ / quiz test project $ absences;
```

```
data grades;
infile 'a:\grades.txt';
input name $ #2 quiz test project $ absences;
```

```
data grades;
infile 'a:\grades.txt';
input name $ quiz test project $ absences;
```

SAS automatically continues reading on next line.

Line Pointers multiple Observations per Line

```
data grades;
  input name $ quiz test project $ absences @@;
datalines;
Ann 84 90 A- 0 Bill 78 84 B 0 Cathy 95 89 A 1
David 84 88 B+ 1
;
```

Trailing “at” symbols (@@) tells SAS to hold the current data line for further information.

- Read in *name--absences* then hold the current position on the data line.
- Read in another set of *name--absences* then hold position.
- Keep doing this until an end-of-line marker is reached.

Input Statement Specifics

```
Data new;
  infile  myfile “c:\name\project\mydir\mydata.dat”  missover;
  input gamedate mmddyy20. @22 name $ 20.;
```

Once you use @nn SAS uses a different mode to input data.

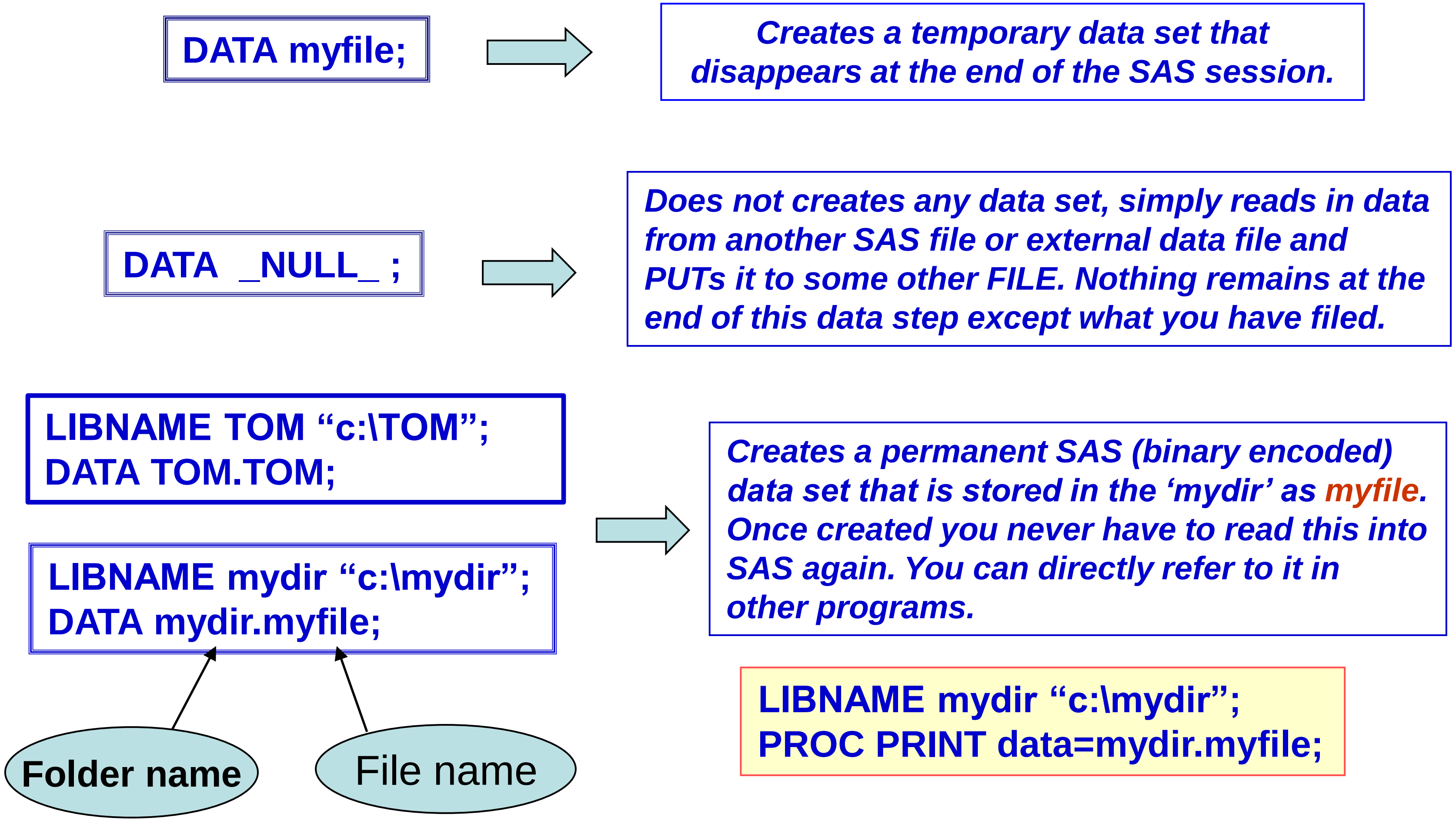
Names with spaces in it will not be read in correctly.

```
Data new;
  infile  myfile “c:\name\project\mydir\mydata.dat”  missover;
  input month 1-2 day 9-10 year 17-20 name $ 22-42;
  gamedate = mdy(month,day,year);
```

A SAS function is used to create a SAS date variable after separate month, day and year values read in.

List input for character data assures that even spaces in the name will be read in and included in the database.

Temporary and Permanent SAS Data Sets



```
DATA myfile;
```

Creates a temporary data set that disappears at the end of the SAS session.

```
DATA  _NULL_ ;
```

Does not creates any data set, simply reads in data from another SAS file or external data file and PUTs it to some other FILE. Nothing remains at the end of this data step except what you have filed.

```
LIBNAME TOM "c:\TOM";
DATA TOM.TOM;
```

```
LIBNAME mydir "c:\mydir";
DATA mydir.myfile;
```

Creates a permanent SAS (binary encoded) data set that is stored in the 'mydir' as myfile. Once created you never have to read this into SAS again. You can directly refer to it in other programs.

```
LIBNAME mydir "c:\mydir";
PROC PRINT data=mydir.myfile;
```

Writing nice programs

```
/* STA 311 Lecture 1 */
Data students;
 input name $ age degree $;
 datalines;
Mary     21 MA
John     22 MS
Alice    22 MBA
Joe      25 PhD
;
proc print data=students;
  title “Student Data”;
run;
```

```
DATA Students;
INPUT Name $ Age Degree $; datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
PROC PRINT Data=Students; RRUN;
```

Rule: Try to keep at most one SAS statement to a line.

Rule: Indent subcommands within DATA and PROC statements.

Rule: Use comments.

Rule: Use titles.

Comments

```
*This comment statement starts with an asterick;


/* This is also a comment statement. */

/*
  This is a comment block. It can be used to
  provide more detailed comments. Also blocks of
  code can be commented out when you are done
  running that part of the program.

  Note, anything, including special characters
  !@#$%^&() except asterisk and forward slash can
  be used.
*/
```

Using Block Comments

Block out some old code but keep it in the program as a record of what you have attempted.

```
Data students;
 input name $ age degree $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
/*
proc print data=students;
  var name age;
  run;
*/
proc sort data=students;
  by degree name;
  run;
proc print data=students;
  by degree;
  run;
```

Uses of Comments

1. To note the history of your analysis.
    - Keep track of the steps you took to create the data set.
    - Keep track of data modifications.
    - Keep track of all statistical analyses attempted.

3. Assist others in understanding your analysis.

4. To block out sections of the program (/* and */) allowing you to run partial analyses but return to previous analyses if you change your approach.

5. To refresh your memory about the project when you have to return to the analysis to answer questions from Journal reviewers, academic advisors, colleagues, months or years after the data have been analyzed.

Variable Names

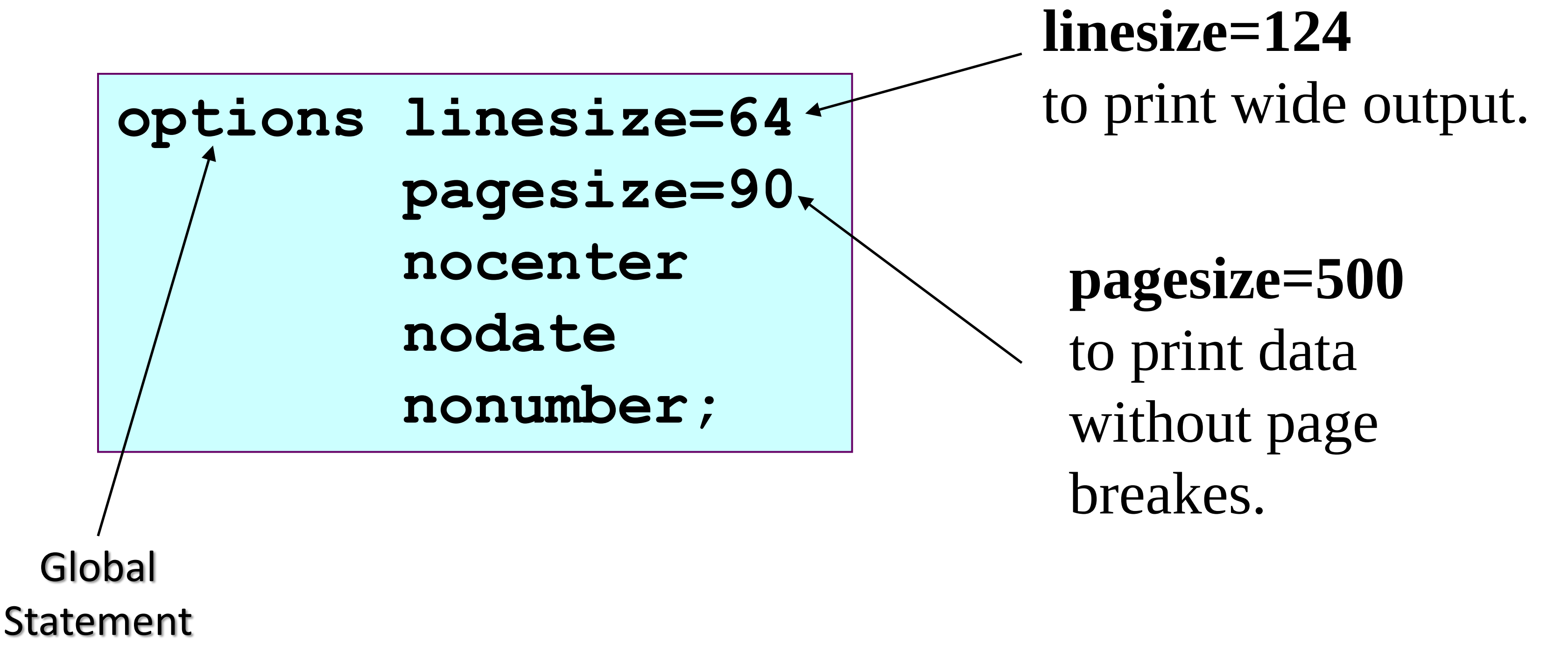
```
Data students;
 input A $ B C $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
proc print data=students;
  title ‘Student file’;
run;
```

```
Data students;
 input first_name $ age degree $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
proc print data=students;
  title ‘Student File’;
run;
```

You have up to 32 characters for each variable name, with up to 37 possible characters in each position except the first which has 27. Use this flexibility to give variables understandable names.

The OPTIONS statement

Place at the beginning of your program to control output options.



```
options linesize=64
        pagesize=90
        nocenter
        nodate
        nonumber;
```

linesize=124
to print wide output.

pagesize=500
to print data
without page
breakes.

Global
Statement

The TITLE and FOOTNOTE statements

Use a TITLE and/or FOOTNOTE statement to place comment information at the top (TITLE) or bottom (FOOTNOTE) of each output page.

Once set, TITLE and FOOTNOTE information will be printed for each procedure output unless a new TITLE or FOOTNOTE statement is encountered.

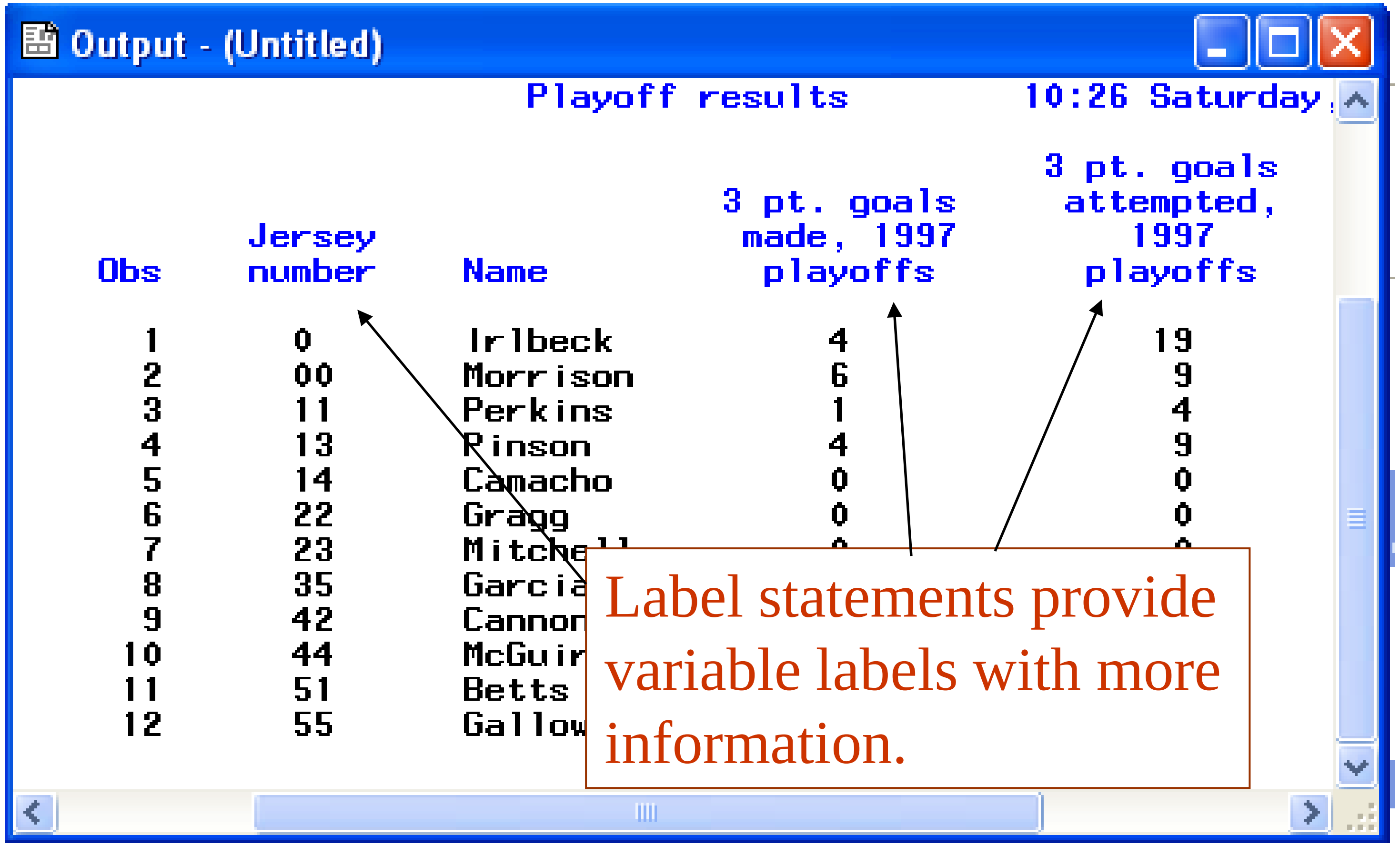
```
title1  ‘First line’;
title2  ‘Second line’;
title3  ‘Third line’;
footnote1  ‘Line one’;
footnote2  ‘Line two’;
```

```
Data students;
 input name $ age degree $;
 datalines;
Mary    21 MA
John    22 MS
Alice   22 MBA
Joe     25 PhD
;
proc print data=students;
title ‘Four New Post-Bacc Students’;
footnote ‘Ages as of last birthday’;
run;
```

Labels

```
data basket;
  input jerseyno $ name $ _3pmpo97 _3papo97 @@;
  label jerseyno='Jersey number'
        name='Name'
        _3pmpo97='3 pt. goals made, 1997 playoffs'
        _3papo97='3 pt. goals attempted, 1997 playoffs';
  datalines;
0  Irlbeck 4 19  00 Morrison 6 9   11 Perkins 1 4
13 Pinson 4 9    14 Camacho 0 0    22 Gragg 0 0
23 Mitchell 0 0  35 Garcia 0 1     42 Cannon 3 3
44 McGuire 0 0   51 Betts 2 4      55 Galloway 2 4
;
  run;
proc print data=basket label;
  title 'Playoff results';
  run;
```

Labels in Output



```
                         Playoff results          10:26 Saturday,

                                                             3 pt. goals
                                          3 pt. goals        attempted,
                Jersey                    made, 1997           1997
     Obs        number     Name            playoffs           playoffs

       1           0       Irlbeck              4                19
       2          00       Morrison             6                 9
       3          11       Perkins              1                 4
       4          13       Pinson               4                 9
       5          14       Camacho              0                 0
       6          22       Gragg                0                 0
       7          23       Mitchell
       8          35       Garcia
       9          42       Cannon
      10          44       McGuir
      11          51       Betts
      12          55       Gallow
```

Label statements provide variable labels with more information.

PROC PRINT options

DATA =*name* - specifies data set name
DOUBLE - double-spaces output
NOOBS - suppresses the observation number in the first column
HEADING=H - column headings should be printed (h)orizontally
HEADING=V - column headings should be printed (v)ertically
UNIFORM - asks for all pages of output to have the same appearance

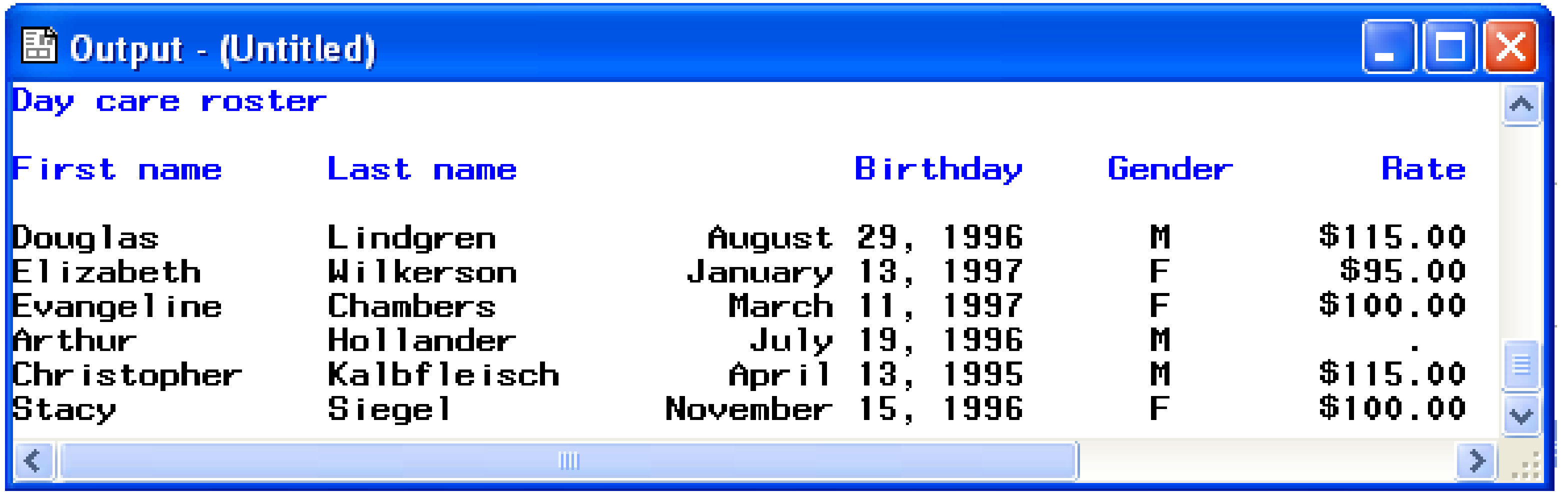
```
proc print data=basket(obs=5);
  var jerseyno;
  id name;
  run;
```

Print only the first 5 observations for variable **jerseyno** but add the identification data in variable **name**.

Output Formats

```
options nodate nonumber nocenter;
data kids;
  input @1 firstnam $11. @12 lastname $11.
        @23 birthday mmddyy10. @33 gender $1. @34 wklyrate 3.;
  label firstnam='First name' lastname='Last name'
        birthday='Birthday' gender='Gender'
        wklyrate='Rate';
datalines;
Douglas    Lindgren   08/29/1996M115
Elizabeth  Wilkerson  01/13/1997F95
Evangeline Chambers   03/11/1997F100
Arthur     Hollander  07/19/1996M.
ChristopherKalbfleisch04/13/1995M115
Stacy      Siegel     11/15/1996F100
;
run;
proc print data=kids noobs label;
  format birthday worddate18. wklyrate dollar7.2;
  title 'Day care roster';
  run;
```

Formatted Output



```
Day care roster

First name    Last name                 Birthday    Gender      Rate

Douglas       Lindgren            August 29, 1996      M     $115.00
Elizabeth     Wilkerson          January 13, 1997      F      $95.00
Evangeline    Chambers             March 11, 1997      F     $100.00
Arthur        Hollander             July 19, 1996      M           .
Christopher   Kalbfleisch          April 13, 1995      M     $115.00
Stacy         Siegel            November 15, 1996      F     $100.00
```

Input data
using column format

```
Douglas    Lindgren   08/29/1996M115
Elizabeth  Wilkerson  01/13/1997F95
Evangeline Chambers   03/11/1997F100
Arthur     Hollander  07/19/1996M.
ChristopherKalbfleisch04/13/1995M115
Stacy      Siegel     11/15/1996F100
```

Creating Your Own Formats - PROC FORMAT

```
proc format;
  value $sexfmt 'F'='Female' 'M'='Male';
  value ratefmt 0-<100='Low'
              100-HIGH='High'
                     .='Missing';
  run;
```

```
proc print data=kids noobs label;
  var firstnam lastname gender wklyrate;
  format gender $sexfmt. wklyrate ratefmt.;
  title 'Day care roster';
  run;
```

NOTE: Formats always end with a period.

Formatted Output

```
First name   Last name              Birthday   Gender       Rate
Douglas      Lindgren         August 29, 1996    M       $115.00
Elizabeth    Wilkerson       January 13, 1997    F        $90.00
Evangeline   Chambers          March 11, 1997    F       $100.00
Arthur       Hollander          July 19, 1996    M             .
Christopher  Kalbfleisch       April 13, 1995    M       $115.00
Stacy        Siegel         November 15, 1996    F       $100.00
```

```
Day care roster

First name     Last name      Gender    Rate

Douglas        Lindgren       Male      High
Elizabeth      Wilkerson      Female    Low
Evangeline     Chambers       Female    High
Arthur         Hollander      Male      Missing
Christopher    Kalbfleisch    Male      High
Stacy          Siegel         Female    High
```

Previous Output

Formats and PROC Format

There are formats that SAS has predefined for you.

```
INPUT gamedate mmddyy20. ;
```

```
PUT name $char22. cost dollar7. ;
```

When you need a format that SAS doesn’t provide, you use PROC FORMAT to create it. Your newly created format is used just like other (output) formats (remember the period at the end of a format statement).

```
PROC Format;
  value $gender ‘M’=‘Male’  ‘m”=‘Male’
                      ‘F’=‘Female’ ‘f’=‘Female’;
 run;

PROC Print data=mydata;
  var sex;
  format sex $gender. ;
  run;
```

Issue Review

TXT files:

when you create a TXT file(using FILE “ “;) you create a permanent file on your disk. To view/print the file use any application that can read a text (ascii) file.

FORMAT and INFORMAT

INFORMATs tell SAS that the variable you are reading in has something special about it.

FORMATs tell SAS that the variable you wish to print out has something special about it.

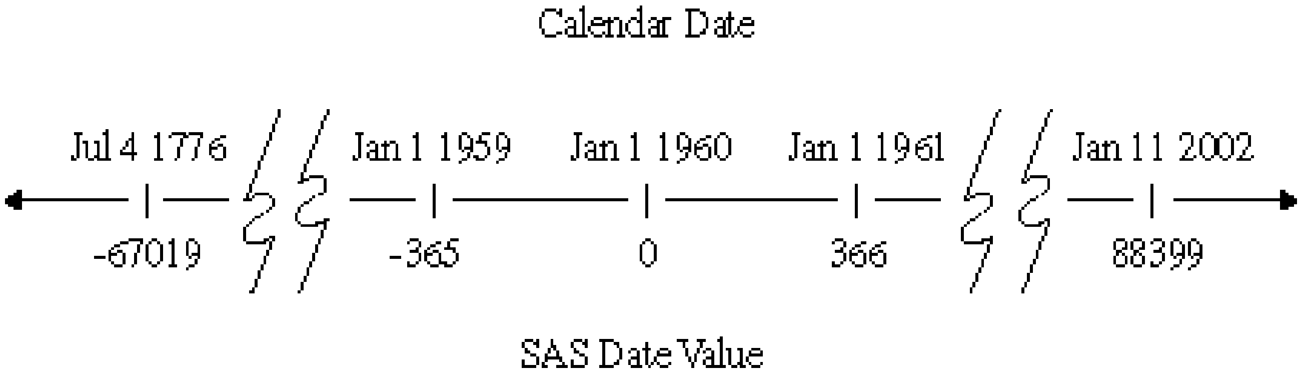
You can read in with one INFORMAT and write out with a different FORMAT

DATA and PROC statements

Be careful that you do not try to use DATA step statements (such as PUT or FILE) within a PROC step.

A First Glance of SAS Dates: Value and Formats

A SAS date is stored as a numerical value - 1/1/1960 00:00:00 as ZERO



A SAS Date has different formats, so does SAS time!