Using INFILE and INPUT Statements to Introduce External Data into the SAS® System

– the interactive session –

Andrew T. Kuligowski

Using INFILE and INPUT …

Introduction

Introduction

Basic INFILE / INPUT

Conditional INPUT

Column and Line Pointers

Informats

Using INFILE and INPUT …

Introduction

Variable Length Files

Comma Separated Value Files

Dynamic Data Exchange

HTML

INPUT Function

Conclusion

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Francisco  CA
PHARMASUG 2006 Bonita Springs FL
```

<… Some DATALINES removed to fit example on screen …>

```
MWSUG      2006 Dearborn        MI
PNWSUG     2006 Seaside         OR
SUGI       2007 Orlando         FL
;
```

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Francisco  CA
PHARMASUG  2006 Bonita Springs FL
<… Some DATALINES removed to fit example on screen …>
MWSUG      2006 Dearborn       MI
PNWSUG     2006 Seaside        OR
SUGI       2007 Orlando        FL
;
```

OPEN UP `CONFYR1.TXT`
(directory to be announced)
(This is your input data.)
Add code around it, save as TC1.sas

DO NOT RUN IT YET!!

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

INFILE

Define the source of the external data to the DATA step. Usually followed by a file name or a *file reference* (a "nickname" for an external source).

May also include optional parameters describing the data source.

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

INPUT

Define the format of the data to be processed.

This is the command which actually causes the data to be transferred from the external source into the DATA step.

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

List Input

Input statement contains the name of each variable to be read in.

Fields separated on external file by one or more blanks (or other delimiter).

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

DATALINES

A special *file reference* that tells SAS there will be instream data following the conclusion of the DATA Step.

Terminated with a ; (semicolon)

Also known as CARDS

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES4;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES4;
SUGI       2006 San Fra
PHARMASUG  2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;;;;
```

(A quick aside …)

DATALINES4 can be used when the instream data could contain a semicolon in the first position.

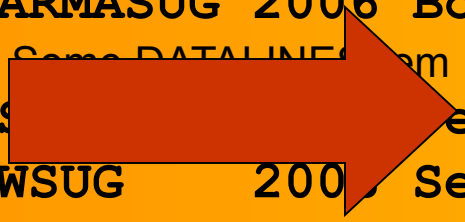
Terminated with ;;;; (4 consect. semicolons)

CARDS4 also works.

Using INFILE and INPUT …

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName
        ConfYear
        ConfCity
        ConfST  ;
DATALINES;
SUGI       2006 San
PHARMASUG  2006 Boni
<… Some DATALINES om …>
MWS                ear
PNWSUG     200  Seas
SUGI       2007 Orla
;
```



```
SUGI       2006 San Francisco  CA
PHARMASUG  2006 Bonita Springs FL
NESUG      2006 Philadelphia   PA
WUSS       2006 Irvine         CA
SESUG      2006 Atlanta        GA
SCSUG      2006 Irving         TX
MWSUG      2006 Dearborn       MI
PNWSUG     2006 Seaside        OR
SUGI       2007 Orlando        FL
```

<This is the entire instream dataset.>

NOW RUN IT.

Using INFILE and INPUT …

Basic INFILE / INPUT

```
NOTE: Invalid data for ConfName in line 9 1-9.
NOTE: Invalid data for ConfCity in line 9 16-18.
NOTE: Invalid data for ConfST in line 9 20-28.
RULE: ----+----1----+----2----+----3---
9     SUGI       2006 San Francisco  CA
ConfName=. ConfYear=2006 ConfCity=. ConfST=.
_ERROR_=1 _N_=1
NOTE: Invalid data for ConfName in line 10 1-9.
NOTE: Invalid data for ConfCity in line 9 16-18.
NOTE: Invalid data for ConfST in line 9 20-28.
10    PHARMASUG 2006 Bonita Springs FL
ConfName=. ConfYear=2006 ConfCity=. ConfST=.
_ERROR_=1 _N_=2
```

<… messages repeated for each line of input data …>

Using INFILE and INPUT …

2nd iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName $
        ConfYear
        ConfCity $
        ConfST   $ ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

Modified List Input

Input statement contains the name of each variable to be read in PLUS format modifiers that specify some additional piece of information about fields to be processed.

Using INFILE and INPUT …

2nd iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  INFILE DATALINES;
  INPUT ConfName $
        ConfYear
        ConfCity $
        ConfST   $ ;
DATALINES;
SUGI       2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG      2006 Dearbor
PNWSUG     2006 Seaside
SUGI       2007 Orlando
;
```

$ Format Modifier

Additional information about field to be read – Field is to be treated as character rather than numeric.

NOW RUN IT.

Using INFILE and INPUT …

2nd iteration

Basic INFILE / INPUT

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San	Francisc
2	PHARMASU	2006	Bonita	Springs
3	NESUG	2006	Philadel	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SUGI	2007	Orlando	FL

Using INFILE and INPUT …

3rd iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
         ConfCity $ 14.;
  INFILE DATALINES;
  INPUT ConfName   $
        ConfYear
        ConfCity & $
        ConfST     $;
DATALINES;
SUGI        2006 San Fra
PHARMASUG 2006 Bonita
<… Some DATALINES removed …>
MWSUG       2006 Dearbor
PNWSUG      2006 Seaside
SUGI        2007 Orlando
;
```

Using INFILE and INPUT Statements to Introduce Ex

& Format Modifier

Additional information about field to be read – Character field can have single embedded blanks.

LENGTH

In this example, override default length of 8 for character variable.

NOW RUN IT.

Using INFILE and INPUT …

3rd iteration

Basic INFILE / INPUT

```
NOTE: SAS went to a new line when INPUT
      statement reached past the end of a line.
NOTE: The data set WORK.SASCONF has 8
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	ConfCity	Conf Year	Conf ST
1	SUGI	San Francisco	2006	CA
2	PHARMASUG	Bonita Springs	2006	NESUG
3	WUSS	Irvine	2006	CA
4	SESUG	Atlanta	2006	GA
5	SCSUG	Irving	2006	TX
6	MWSUG	Dearborn	2006	MI
7	PNWSUG	Seaside	2006	OR
8	SUGI	Orlando	2007	FL

Using INFILE and INPUT …

3rd iteration

Basic INFILE / INPUT

What happened?

Behind the scenes, reading the 2nd line …

```
INPUT ConfName    $     PHARMASUG    OK
      ConfYear    2006  OK
      ConfCity & $      Bonita Springs FL
```

Read “Bonita” + blank + “Springs” + blank + “FL”.
End of line, skip to next line.

```
      ConfST      $ ;   NESUG
```

(Ignore rest of this input line.)

So what happened to the “FL” in our output?

```
LENGTH ConfName $ 9.   ConfCity $ 14.;
```

Bonita Springs FL (“ FL” in position 15-17)

Using INFILE and INPUT …

4th iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
         ConfCity $ 17.;
  INFILE DATALINES
         TRUNCOVER ;
  INPUT ConfName   $
        ConfYear
        ConfCity & $
        ConfST     $;
  IF  ConfST = " "  TH
    ConfST = SUBSTR(Co
    ConfCity = SUBSTR(
  END;
DATALINES;
<… All DATALINES removed …>
;
```

TRUNCOVER

INFILE option. Prevents SAS from moving to the next input line if End of Line encountered in middle of an INPUT.

Default is FLOWOVER.

Using INFILE and INPUT …

4th iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
       ConfCity $ 17.;
  INFILE DATALINES
         TRUNCOVER ;
  INPUT ConfName    $
        ConfYear
        ConfCity & $
        ConfST      $;
  IF  ConfST = " "  TH
    ConfST = SUBSTR(Co
    ConfCity = SUBSTR(
  END;
DATALINES;
<… All DATALINES removed …>
;
```

(A quick aside …)

MISSOVER would also work. The difference – when INPUT reaches end of line in the middle of a variable, MISSOVER sets that value to missing, while TRUNCOVER keeps the partial value.

Using INFILE and INPUT …

4th iteration

Basic INFILE / INPUT

```
/* INTRO EXAMPLE */
DATA SasConf;
  LENGTH ConfName $ 9.
         ConfCity $ 17.;
  INFILE DATALINES
         TRUNCOVER ;
  INPUT ConfName   $
        ConfYear
        ConfCity & $
        ConfST     $;
  IF  ConfST = " "  THEN  DO;
    ConfST = SUBSTR(ConfCity,16);
    ConfCity = SUBSTR(ConfCity,1,14);
  END;
DATALINES;
<… All DATALINES removed to fit example on screen …>
;
```

(Additional coding logic added to deal with special case of field reaching maximum length.)

NOW RUN IT.

Using INFILE and INPUT …

4th Iteration & final

Basic INFILE / INPUT

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San Francisco	CA
2	PHARMASUG	2006	Bonita Springs	FL
3	NESUG	2006	Philadelphia	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SUGI	2007	Orlando	FL

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMPLE */
FILENAME SASCONF '<directory>.CONFYR3.DATA';
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType $CHAR1. @ ;
  IF  RecordType = 'R'  THEN DO;
     INPUT @  3     ConfName $CHAR9.
           @ 13     ConfYear      4.
           @ "LOC=" ConfCity $CHAR14.
           @ 36     ConfST   $CHAR2. ;
     OUTPUT;
  END;
  ELSE  INPUT;  /* optional */
RUN;
```

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMPLE */
FILENAME SASCONF '<directory>.CONFYR3.DATA';
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType $CHAR1. @ ;
  IF  RecordType = 'R'  THEN DO;
     INPUT @  3     ConfName $CHAR9.
           @ 13     ConfYear      4.
           @ "LOC=" ConfCity $CHAR14.
           @ 36     ConfST   $CHAR2. ;
     OUTPUT;
  END;
  ELSE  INPUT;  /* optional */
RUN;
```

Modify existing code, save as TC2.sas

DO NOT RUN IT YET!!

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EX
FILENAME SASCONF 'U
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordT
  IF  RecordType =
     INPUT @  3
           @ 13
           @ "LOC="
           @ 36
     OUTPUT;
  END;
  ELSE  INPUT;  /*
RUN;
```

'SASCONF3.DATA'

```
I SUGI       2006 LOC=San Francisco
S PHARMASUG  2006 LOC=Bonita Springs
R NESUG      2006 LOC=Philadelphia
R WUSS       2006 LOC=Irvine
R SESUG      2006 LOC=Atlanta
R SCSUG      2006 LOC=Irving
R MWSUG      2006 LOC=Dearborn
R PNWSUG     2006 LOC=Seaside
I SASGBLFRM  2007 LOC=Orlando
```

This is the entire dataset. It is basically the same data that we used in our previous example, except:

- A record type has been added.
- LOC= has been inserted.

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF '<dir
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  3      Co
           @ 13      Co
           @ "LOC="  Co
           @ 36      Co
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

FILENAME

Assign a *file reference* to an external data source (which is an MVS dataset in this example).

This file ref. will be used by the INFILE statement.

This statement occurs outside of the DATA step.

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF '<dir
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  3      Co
           @ 13      Co
           @ "LOC="  Co
           @ 36      Co
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

Formatted Input

Formats and column pointers work together to perform the task of reading data based on a specified list of variables.

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF '<dir
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  3      Co
           @ 13      Co
           @ "LOC="  Co
           @ 36      Co
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

@ "At sign" Column Pointer Control

Move the column pointer to an absolute position on the input dataset.

Continue reading data from that point.

ALSO NOTE STRING!!

Using INFILE and INPUT …

Conditional INPUT

Formats

$CHAR*nn*. **Read character value, *nn* positions long, keeping leading blanks.**

($*nn*. would read character value, *nn* positions long, dropping leading blanks.)

***nn*.** **Read numeric value, *nn* positions long, store in numeric SAS variable.**

```
E */
ctory>.CONFYR3.DATA';


$CHAR1. @ ;
 THEN DO;
fName $CHAR9.
fYear      4.
fCity $CHAR14.
fST   $CHAR2. ;


onal */
```

Using INFILE and INPUT …

Conditional INPUT

@ "Trailing At sign"

Hold the line pointer on the current input line. Do not advance to the next line unless triggered to do so by another INPUT statement or by the RUN statement.

```
E */
ctory>.CONFYR3.DATA';


$CHAR1. @ ;
 THEN DO;
fName $CHAR9.
fYear      4.
fCity $CHAR14.
fST   $CHAR2. ;


onal */
```

Using INFILE and INPUT …

Conditional INPUT

(A quick aside)

@@ "Double Trailing At sign"

Hold the line pointer on the current input line. Do not advance to the next line unless triggered to do so by another INPUT statement.~~or by the RUN statement.~~

```
E */
ctory>.CONFYR3.DATA';


$CHAR1. @ ;
 THEN DO;
fName $CHAR9.
fYear      4.
fCity $CHAR14.
fST   $CHAR2. ;


onal */
```

Using INFILE and INPUT …

Conditional INPUT

```
/* "RECORD TYPE" EXAMP
FILENAME SASCONF '<dir
DATA SasConf;
  INFILE SASCONF ;
  INPUT @ 1 RecordType
  IF  RecordType = 'R'
     INPUT @  3      Co
           @ 13      Co
           @ "LOC="  Co
           @ 36      Co
     OUTPUT;
  END;
  ELSE  INPUT;  /* opt
RUN;
```

Null Input

Processes no data, but advances the internal line pointer to the next line (releasing any hold on that line, if applicable – as in this example).

NOW RUN IT.

Using INFILE and INPUT …

Conditional INPUT

```
NOTE: The infile SASCONF is:
      File Name=USERID.SASCONF.DATA,
      Lrecl=80,Recfm=FB,Blksize=960
NOTE: 9 records were read from the infile SASCONF.
NOTE: The data set WORK.SASCONF has
      6 observations and 5 variables.
```

< Results of a PROC PRINT >

Obs	Record Type	Conf Name	Conf Year	ConfCity	Conf ST
1	R	NESUG	2006	Philadelphia	PA
2	R	WUSS	2006	Irvine	CA
3	R	SESUG	2006	Atlanta	GA
4	R	SCSUG	2006	Irving	TX
5	R	MWSUG	2006	Dearborn	MI
6	R	PNWSUG	2006	Seaside	OR

Using INFILE and INPUT …

Column and Line Pointers

```
/* "Pointers" EXAMPLE */
FILENAME SASCONF2 'USERID.SASCONF2.DATA';
DATA  UGLYINPUT;
  LENGTH  ConfName $ 9. ;
  Pos1 =  9;
  Pos2 =  3;
  Str1 = "LOC=";
  INFILE SASCONF2 ;
  INPUT #1 ConfCity   $ 22-35 +1
           ConfST     $       +(-39)
           RecordType $       +Pos1
           ConfYear           +(-1*(Pos1+6))
           ConfName   $
```

OPEN UP `CONFYR3.sas`

Do not even try to type this one in!

DO NOT RUN IT YET!!

Using INFILE and INPUT …

Column and Line Pointers

```
    / @ "20"          Year2Digit2
      @    1          RecordType2 $
      @ Pos2          ConfName2   $  9.
      @ Str1          ConfCity2   $ 14.
      @ (Pos2*12+1) ConfST2       $ ;
  ConfYear2 = 2000+Year2Digit2 ;
RUN;
```

Using INFILE and INPUT …

Column and Line Pointers

```
/* "Pointers
FILENAME SAS
DATA  UGLYIN
  LENGTH  Co
  Pos1 =  9;
  Pos2 =  3;
  Str1 = "LO
  INFILE SAS
  INPUT #1 C
           C
           R
           C
           C
```

<continued …>

'USERID.SASCONF2.DATA'

```
I SUGI       2006 LOC=San Francisco  CA
S PHARMASUG  2006 LOC=Bonita Springs FL
R NESUG      2006 LOC=Philadelphia   PA
R WUSS       2006 LOC=Irvine         CA
R SESUG      2006 LOC=Atlanta        GA
R SCSUG      2006 LOC=Irving         TX
R MWSUG      2006 LOC=Dearborn       MI
R PNWSUG     2006 LOC=Seaside        OR
I SUGI       2007 LOC=Orlando        FL
R SESUG      2007 LOC=Hilton Head    SC
```

This is the entire dataset. It is basically the same data that we used in our previous example, except:

- **LOC=** has been added before City Name.
- Blank Padding has been restored.

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
   LEN
   Pos
   Pos
   Str
   INF
   INPUT #1 ConfCity    $ 22-35 +1
            ConfST      $       +(-39)
            RecordType  $       +Pos1
            ConfYear            +(-1*(Pos1+6))
            ConfName    $
```

 # Absolute Line Pointer Control

Move the line pointer to the xth line of the current input buffer. The size of the input buffer can be overridden by the N= option on the INFILE statement.

<continued>

Using INFILE and INPUT …

Column and Line Pointers

Column Specification

Read the current variable between positions x and y of the current input line – in this example, columns 22 & 35.

```
/* "P
FILEN
DATA
   LEN
   Pos
   Pos
   Str
   INF
   INPUT #1 ConfCity    $ 22-35 +1
            ConfST      $       +(-39)
            RecordType  $       +Pos1
            ConfYear            +(-1*(Pos1+6))
            ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
  LEN
  Pos
  Pos
  Str
  INF
  INP
```

Column Input

Start and end positions are specified for each input variable. Character data can contain embedded blanks and can exceed 8 characters in length, but it must be aligned consistently throughout the file.

```
RecordType $        +Pos1
ConfYear            +(-1*(Pos1+6))
ConfName   $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

+ Relative Column Pointer Control

Move the column pointer X positions – in this example, 1 position to the right.

```
/* "P
FILEN
DATA
  LEN
  Pos
  Pos
  Str
  INFILE SASCONF2 ;
  INPUT #1 ConfCity    $ 22-35 +1
           ConfST      $       +(-39)
           RecordType  $       +Pos1
           ConfYear            +(-1*(Pos1+6))
           ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
  LEN
  Pos
  Pos
  Str
  INFILE SASCONF2 ;
  INPUT #1 ConfCity    $ 22-35 +1
           ConfST      $       +(-39)
```

+ Relative Column Pointer Control

Move the column pointer X positions – in this example, 1 position to the right.

And in this example, 39 positions to the left. The + denotes movement – positive number is default, but not mandatory.

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
   LEN
   Pos1 =  9;
   Pos
   Str
   INF
   INP
```

+ Relative Column Pointer Control

And in this example, 9 positions to the right. (The variable `Pos1` was assigned the numeric value 9 and is substituted.)

```
RecordType $        +Pos1
ConfYear            +(-1*(Pos1+6))
ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
/* "P
FILEN
DATA
   LEN
   Pos1 =  9;
   Pos
   Str
   INF
   INP
```

+ Relative Column Pointer Control

And in this example, 15 positions to the left. (The variable `Pos1` was assigned the numeric value `9`, add `6`, and multiply by `–1` to get `–15`, which is substituted.)

```
        ConfYear              +(-1*(Pos1+6))
        ConfName    $
```

<continued>

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

/ (Implied) Relative Line Pointer Control

Move the line pointer one line forward. No flexibility – use 3 slashes to move 3 lines forward, cannot move backwards.

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

@ Absolute Column Pointer Control

Move the column pointer to the specified position – in this example, 1 position to the right of the next occurrence of the text string "20" – NOT the 20th position.

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

@ Absolute Column Pointer Control

And then back to the 1st position of the current line.

Using INFILE and INPUT …

Column and Line Pointers

```
    Pos2 =  3;
    Str1 = "LOC=";
< … first part of input line removed for space limitations … >
        / @ "20"          Year2Digit2
          @    1          RecordType2 $
          @ Pos2          ConfName2   $  9.
          @ Str1          ConfCity2   $ 14.

    Con
RUN;
```

@ Absolute Column Pointer Control

Then move the column pointer to position 3 on the current line. (The variable `Pos2` was assigned the numeric value 3 and is substituted.)

Using INFILE and INPUT …

Column and Line Pointers

```
Pos2 =  3;
Str1 = "LOC=";
```

< … first part of input line removed for space limitations … >

```
      / @ "20"          Year2Digit2
        @    1          RecordType2 $
        @ Pos2          ConfName2   $  9.
        @ Str1          ConfCity2   $ 14.

Con
RUN;
```

@ Absolute Column Pointer Control

Next, move the column pointer to the position immediately after the value "LOC=" on the current line. (The variable `Str1` was assigned the string value "LOC=" and is substituted.)

Using INFILE and INPUT …

Column and Line Pointers

```
   Pos2 =  3;
   Str1 = "LOC=";
< … first part of input line
      / @ "20"          Year2Digit2
        @    1          RecordType2 $
        @ Pos2          ConfName2   $  9.
        @ Str1          ConfCity2   $ 14.
        @ (Pos2*12+1)   ConfST2     $ ;
   ConfYear2 = 2000+Year2Digit2 ;
RUN;
```

@ Absolute Column Pointer Control

Finally, move the column pointer to position 37 of the current line. (The variable `Pos2` was assigned the numeric value 3, multiply by `12`, and add `1` to get 37, which is substituted.)

Using INFILE and INPUT …

Column and Line Pointers

```
   Pos2 =  3;
   Str1 = "LOC=";
< … first part of input line
      / @ "20"          Year2Digit2
        @    1          RecordType2 $
        @ Pos2          ConfName2   $  9.
        @ Str1          ConfCity2   $ 14.
        @ (Pos2*12+1)   ConfST2     $ ;
   ConfYear2 = 2000+Year2Digit2 ;
RUN;
```

… And one more thing …

We need to adjust the value for the year to include the century. (Remember, we used the "20" to find the year – we never actually *read* those 2 digits!)

NOW RUN IT.

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
           @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Tricks with text column pointers

OPEN UP `CONFYR4.sas`

Don't try to type this one in, either.

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
           @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Lower case, no trailing blank.

NEXT8_1 = "me of th"

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
           @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Lower case, trailing blank.

```
NEXT8_2 = "Thebes t"
```

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Upper case, no trailing blank.

`NEXT8_3 = " theme o"`

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
           @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <etc.>
;
```

Upper case, trailing blank.

NEXT8_4 = "theme of"

Using INFILE and INPUT …

Column and Line Pointers

```
/* Column pointer controls w/character values. */
DATA  THE_data;
   INFILE CARDS;
   INPUT # 1 @"the"  NEXT8_1 $CHAR8.
         # 1 @"the " NEXT8_2 $CHAR8.
         # 1 @"The"  NEXT8_3 $CHAR8.
         # 1 @"The " NEXT8_4 $CHAR8.
             @"The"  NEXT8_5 $CHAR8.;
CARDS;
The theme of the Thebes theater's <e
;
```

Upper case, trailing blank, continue on same input line.

NEXT8_5 = “bes thea”

NOW RUN IT.

Using INFILE and INPUT …

Informats

RELAX FOR A COUPLE OF MINUTES!

Informats are used to interpret incoming data into a form that SAS can understand.

They specify whether an input value is character or numeric, its length, the number of decimal places (if applicable), and other special conditions

Using INFILE and INPUT …

Informats

Two methods to assign an informat to a SAS variable.

1) Temporary – Specify the informat on the INPUT statement.

```
INPUT # 1 @"the"  NEXT8_1 $CHAR8.
```

(This method has already been demonstrated in this presentation.)

Using INFILE and INPUT …

Informats

Two methods to assign an informat to a SAS variable.

2) Permanent – Specify the informat with a separate statement.

```
INFORMAT NEXT8_1 $CHAR8. DEFAULT=$CHAR20.;

ATTRIB variables INFORMAT=informat;
```

Using INFILE and INPUT …

Informats

Five types of SAS Informats:

1) Numeric
2) Character
3) Date / Time
4) Column-binary — not covered
5) User-Defined — `PROC FORMAT`

Using INFILE and INPUT …

Informats (Numeric)

A few examples of SAS Informats

`w.d` **Read numeric value of width *w* with *d* decimal places.**

`COMMAw.d` **Read numeric value of width *w* with *d* decimal places. Ignore embedded commas (and other accounting symbols, such as "$", percent signs, etc.). A left parenthesis at the beginning of the value causes the subsequent numeric value to be treated as a negative.**

Using INFILE and INPUT …

Informats (Character)

A few examples of SAS Informats

`$ w.` Read character value of width *w*, ignore leading blanks (if any).

`$CHARw.` Read character value of width *w*, include leading blanks (if any).

`$ASCIIw.` On IBM Mainframe, convert from ASCII to EBCDIC. On other systems, treat as `$CHAR`.

`$EBCDICw.` On IBM Mainframe, treat as `$CHAR`. On other systems, convert from EBCDIC to ASCII.

Using INFILE and INPUT …

Informats (Date / Time)

A few examples of SAS Informats

`DATEw.` **Read date (*ddMMMyy* or *ddMMMyyyy*) into numeric field.**

`JULIANw.` **Read Julian date (*yyddd* or *yyyyddd*) into numeric field.**

`TIMEw.` **Read time (*hh:mm:ss.ss*) into numeric field.**

Using INFILE and INPUT …

Variable Length Files

```
DATA SasConf;
/*INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr1.txt' TRUNCOVER;*/
  LENGTH ConfName $ 14.  ConfST $ 2. ;
  INFILE DATALINES TRUNCOVER;
/*INPUT ConfName $CHAR14.
        ConfYear       4.
        ConfST   $CHAR2. ;  */
  INPUT ConfName
        ConfYear
        ConfST  ;
DATALINES;
…
```

OPEN UP `CONFYR5a.TXT`
(directory to be announced)

DO NOT RUN IT.

Using INFILE and INPUT …

Variable Length Files

```
DATA SasConf;
/*INFILE 'C:\Docume
Documents\SASHOW07\
  LENGTH ConfName $
  INFILE DATALINES
/*INPUT ConfName $C
       ConfYear
       ConfST   $C
  INPUT ConfName
       ConfYear
       ConfST  ;
DATALINES;
…
```

Input data

```
SUGI 2006 CA
PHARMASUG 2006 FL
NESUG 2006 PA
WUSS 2006 CA
SESUG 2006 GA
SCSUG 2006 TX
MWSUG 2006 MI
PNWSUG 2006 OR
SASGLOBALFORUM 2007 FL
…
```

No longer column-driven.

Using INFILE and INPUT …

Variable Length Files

```
DATA SasConf;
/*INFILE 'C:\Documents
 Documents\SASHOW07\Co
  LENGTH ConfName $ 14
  INFILE DATALINES
/*INPUT ConfName $CHAR
       ConfYear
       ConfST   $CHAR
  INPUT ConfName
       ConfYear
       ConfST  ;
DATALINES;
…
```

FORMATS and POINTERS

Get rid of them!

Why?

Using INFILE and INPUT …

Variable Length Files

```
DATA SasConf;
/*INFILE 'C:\Documents
 Documents\SASHOW07\Co
  LENGTH ConfName $ 14
  INFILE DATALINES
/*INPUT ConfName $CHAR
       ConfYear
       ConfST   $CHAR
  INPUT ConfName
       ConfYear
       ConfST  ;
DATALINES;
…
```

LENGTH statement

Why bring this back now?

HINT: It's tied to format change on last slide.

Using INFILE and INPUT …

Variable Length Files

OPEN UP `CONFYR5b.TXT` (directory to be announced)

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5b.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
  INPUT ConfName @ ;
  STpos = LENGTH( ConfName ) + 2;
  index = ((linelen - STpos + 2) / 3);
  DO i = 1 to index;
     INPUT @ STpos  ConfST $CHAR2. @;
 output;
 STpos = STpos + 3;
  END;
DATALINES;
…
```

DO NOT RUN IT.

Using INFILE and INPUT …

Variable Length Files

```
DATA SasConf;
  INFILE 'C:\Docume
 Documents\SASHOW07
  LENGTH ConfName $
  INPUT ConfName @
  STpos = LENGTH( C
  index = ((linelen
  DO i = 1 to index
     INPUT @ STpos
 output;
 STpos = stop
  END;
DATALINES;
…
```

Input data

```
SUGI CA
PHARMASUG FL CO
NESUG PA MD
WUSS CA
SESUG GA SC
SCSUG TX
MWSUG MI IA
PNWSUG OR WA
SASGLOBALFORUM FL TX
…
```

Multiple states listed on each record.

Using INFILE and INPUT …

Variable Length Files

LENGTH= Option

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5b.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
  INPUT ConfName @ ;
  STpos = LENGTH( ConfName ) + 2;
  index = ((linelen - STpos + 2) / 3);
  DO i = 1 to index;
     INPUT @ STpos  ConfST $CHAR2. @;
      output;
      STpos = STpos + 3;
  END;
DATALINES;
…
```

Sets a temporary SAS variable to the length of the current input line.

Using INFILE and INPUT …

Variable Length Files

```
DATA SasConf;
  INFILE 'C:\Documen
 Documents\SASHOW07\
  LENGTH ConfName $
  INPUT ConfName @ ;
  STpos = LENGTH( Co
  index = ((linelen
  DO i = 1 to index;
     INPUT @ STpos  ConfST $CHAR2. @;
 output;
 STpos = STpos + 3;
  END;
DATALINES;
…
```

The processing logic …

Use a loop to read in each state abbreviation on each line, however many there happen to be!

Using INFILE and INPUT …

Variable Length Files

```
DATA SasConf;
  INFILE 'C:\Documen
 Documents\SASHOW07\
  LENGTH ConfName $
  INPUT ConfName @ ;
  STpos = LENGTH( Co
  index = ((linelen
  DO i = 1 to index;
     INPUT @ STpos  ConfST $CHAR2. @;
 output;
 STpos = STpos + 3;
  END;
DATALINES;
…
```

The processing logic …

Use a loop to read in each state abbreviation on each line, however many there happen to be!

NOW RUN IT.

Using INFILE and INPUT …

Variable Length Files

Insert saslog here

may be multiple slides

also results of PROC PRINT

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5a.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
**INFILE DATALINES LENGTH=linelen;
  INPUT @ ;
  varlen = linelen ;
  INPUT ConfInfo  $VARYING40.  linelen;
  ConfName = SUBSTR( ConfInfo, 1, linelen - 8 );
  ConfYear = SUBSTR( ConfInfo, linelen - 6, 4 );
  ConfST   = SUBSTR( ConfInfo, linelen - 1 );
/*
```

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

```
DATA SasConf;
  INFILE 'C:\Docume
Documents\SASHOW07\
  LENGTH ConfName $
**INFILE DATALINES
  INPUT @ ;
  varlen = linelen
  INPUT ConfInfo  $
  ConfName = SUBSTR
  ConfYear = SUBSTR
  ConfST   = SUBSTR
/*
```

instream data

```
DATALINES;
SUGI 2006 CA
PHARMASUG 2006 FL
NESUG 2006 PA
WUSS 2006 CA
SESUG 2006 GA
SCSUG 2006 TX
MWSUG 2006 MI
PNWSUG 2006 OR
```

Etc. etc. etc.

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5a.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
**INFILE DATALINES LENGTH=linelen;
  INPUT @ ;
  varlen = linelen ;
  INPUT ConfInfo  $VARYING40.  linelen;
  ConfName = SUBSTR( ConfInfo, 1, linelen - 8 );
  Con
  Con
/*
```

Reads a character value of varying length, based on the specified length (obtained from `LENGTH=` option).

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5a.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
**INFILE DATALINES LENGTH=linelen;
  INPUT @ ;
  varlen = linelen ;
  INPUT ConfInfo  $VARYING40.  linelen;
  ConfName = SUBSTR( ConfInfo, 1, linelen - 8 );
  Con
  Con
/*
```

Another way to avoid the infamous SASLOG message:

```
NOTE: INPUT statement reached
      past the end of a line.
```

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5a.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
**INFILE DATALINES LENGTH=linelen;
  INPUT @ ;
  varlen = linelen ;
  INPUT ConfInfo  $VARYING40.  linelen;
  ConfName = SUBSTR( ConfInfo, 1, linelen - 8 );
  Con
  Con
/*
```

Need to invoke INPUT twice. The 1st time will assign the value of the LENGTH= variable, the 2nd time will use it. (Remember to use the trailing @.)

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5a.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
**INFILE DATALINES LENGTH=linelen;
  INPUT @ ;
  varlen = linelen ;
  INPUT ConfInfo  $VARYING40.  linelen;
  ConfName = SUBSTR( ConfInfo, 1, linelen - 8 );
  Con
  Con
/*
```

LENGTH= produces a temporary SAS variable. We need to assign that value to a permanent SAS value if we want to look at it after the DATA step is done.

Using INFILE and INPUT …

Variable Length Files

Alternate Approach

$VARYING. Format

```
DATA SasConf;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr5a.txt' LENGTH=linelen;
  LENGTH ConfName $ 14.  ConfST $ 2. ;
**INFILE DATALINES LENGTH=linelen;
  INPUT @ ;
  varlen = linelen ;
  INPUT ConfInfo  $VARYING40.  linelen;
  ConfName = SUBSTR( ConfInfo, 1, linelen - 8 );
  Con
  Con
/*
```

NOW RUN IT.

LENGTH= produces a temporary SAS variable. We need to assign that value to a permanent SAS value if we want to look at it after the DATA step is done.

Using INFILE and INPUT …

Variable Length Files

Insert saslog here

may be multiple slides

also results of PROC PRINT.

Using INFILE and INPUT …

Comma Separated Value Files

```
/DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
/*** INPUT statement with Formats & Lengths
specified is bad. Left in code, commented out, as
a warning to others .      ***/
/*INPUT ConfName $CHAR15     */
/*      ConfYear       4.    */
/*      ConfCity $CHAR17.    */
/*      ConfST    $CHAR2. ; */
  INPUT ConfName  $
        ConfYear
        ConfCity  $
        ConfST    $ ;
```

OPEN UP CONFYR6a.sas (directory to be announced)

DO NOT RUN IT YET!!

Using INFILE and INPUT …

Comma Separated Value Files

```
/* CSV EXAMP
DATA SasConf
  LENGTH Con
         Con
  INFILE 'C:
  INPUT Conf
        Conf
        Conf
        Conf
RUN;
```

C:\<directory>\ConfYr6a.CSV

```
SUGI,2006,San Francisco,CA
PHARMASUG,2006,Bonita Springs,FL
NESUG,2006,Philadelphia,PA
WUSS,2006,Irvine,CA
SESUG,2006,Atlanta,GA
SCSUG,2006,Irving,TX
MWSUG,2006,Dearborn,MI
PNWSUG,2006,Seaside,OR
SASGLOBALFORUM,2007,Orlando,FL
```

This is basically the same data that we used in our previous example, except that:

- Blank Padding has been removed.
- Character strings now in quotes.
- Fields separated by commas, not blanks.

Using INFILE and INPUT …

Comma Separated Value Files

```
/DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
/*** INPUT statem
specified is bad.
a warning to othe
/*INPUT ConfName
/*      ConfYear        4.   */
/*
/*
  INP
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **The default delimiter is changed from blank to comma. This default, whether blank or comma, can always be overridden with the DELIMITER= option.**

Using INFILE and INPUT …

Comma Separated Value Files

```
/DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
/*** INPUT statem
specified is bad.
a warning to othe
/*INPUT ConfName
/*      ConfYear        4.   */
/*
/*
  INP
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **Multiple consecutive delimiters (commas) are assumed to represent missing values, instead of being treated as redundant and ignored.**

Using INFILE and INPUT …

Comma Separated Value Files

```
/DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
/*** INPUT statem
specified is bad.
a warning to othe
/*INPUT ConfName
/*      ConfYear        4.   */
/*
/*
  INP
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **Character values are assumed to be enclosed in double quotes; these quotation marks are stripped off the input before storing the value.**

Using INFILE and INPUT …

Comma Separated Value Files

```
/DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
/*** INPUT statem
specified is bad.
a warning to othe
/*INPUT ConfName
/*      ConfYear        4.   */
/*
/*
  INP
```

DSD parameter

***D*elimiter *S*eparated *D*ata**

- **FYI – You can retain the quotation marks in the variable, if desired, by using the tilde format modifier (~).**

Using INFILE and INPUT …

Comma Separated Value Files

```
NOTE: Invalid data for ConfYear in line 11 10-13.
RULE: ----+----1----+----2----+----3---
11    "SUGI",2006,"San Francisco","CA"
ConfName="SUGI",20 ConfCity=San Francisco"
ConfYear=. ConfST=," _ERROR_=1 _N_=1
NOTE: Invalid data for ConfYear in line 12 10-13.
12    "PHARMASUG",2006,"Bonita Springs","FL"
ConfName="PHARMASU ConfCity=006,"Bonita Sp
ConfYear=. ConfST=ri _ERROR_=1 _N_=2
NOTE: Invalid data for ConfYear in line 13 10-13.
13    "NESUG",2006,"Philadelphia","PA"
ConfName="NESUG",2 ConfCity="Philadelphia"
ConfYear=. ConfST=," _ERROR_=1 _N_=3
```

Etcetera

Using INFILE and INPUT …

Comma Separated Value Files

```
/DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
/*** INPUT statement with Formats & Lengths
specified is bad. Left in code, commented out, as
a warning to others .      ***/
/*INPUT ConfName $CHAR15     */
/*      ConfYear       4.    */
/*      ConfCity $CHAR17.    */
/*      ConfST    $CHAR2. ; */
  INPUT ConfName  $
        ConfYear
        ConfCity  $
        ConfST    $ ;
```

NOW RUN IT.

Using INFILE and INPUT …

Comma Separated Value Files

```
/DATA SasConf;
  LENGTH ConfName $ 15   ConfCity $ 17 ;
  INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
/*** INPUT statement with Formats & Lengths
specified is bad. Left in code, commented out, as
a warning to others .      ***/
/*INPUT ConfName $CHAR15    */
/*      ConfYear       4.   */
/*      ConfCity $CHAR17.   */
/*      ConfST    $CHAR2. ; */
  INPUT ConfName   $
        ConfYear
        ConfCity   $
        ConfST     $ ;
```

BEFORE YOU RUN IT …

Using INFILE and INPUT …

with fix

Comma Separated Value Files

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San Francisco	CA
2	PHARMASUG	2006	Bonita Springs	FL
3	NESUG	2006	Philadelphia	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SAS GLOBAL FORUM	2007	Orlando	FL

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
**INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
  INFILE DATALINES DSD DLM="~";
  INPUT ConfName  $
        ConfYear
        ConfCity  $
        ConfST    $ ;
DATALINES;
```

OPEN UP `CONFYR6b.TXT`
(directory to be announced)

DO NOT RUN IT YET!!

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
DATA SasConf
  LENGTH Con
**INFILE 'C:
Documents\SA
  INFILE DAT
  INPUT Conf
        Conf
        Conf
        Conf
DATALINES;
```

Instream data

```
SUGI~2006~San Francisco~CA
PHARMASUG~2006~Bonita Springs~FL
NESUG~2006~Philadelphia~PA
WUSS~2006~Irvine~CA
SESUG~2006~Atlanta~GA
SCSUG~2006~Irving~TX
MWSUG~2006~Dearborn~MI
PNWSUG~2006~Seaside~OR
SASGLOBALFORUM~2007~Orlando~FL
```

This is basically the same data that we used in our previous example, except that:

- Blank Padding has been removed.
- Character strings now in quotes.
- Fields separated by commas, not blanks.

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
/* CSV EXAMPLE */
DATA SasConf;
  LENGTH ConfName
         ConfCity $ 14.;
  INFILE 'C:\Papers\ConfListTilde.CSV'
         DSD DLM="~";
  INPUT ConfName $
        ConfYear
        ConfCity $

RUN;
```

DELIMITER= parameter
(DLM=) Delimiter override

- **Changes default delimiter from blank or comma (DSD only) to specified value. All other rules of delimiters unchanged.**

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
DATA SasConf;
  LENGTH ConfName $ 15.  ConfCity $ 17.;
**INFILE 'C:\Documents and Settings\KuligoAT\My
Documents\SASHOW07\ConfYr6a.csv' DSD;
  INFILE DATALINES DSD DLM="~";
  INPUT ConfName  $
        ConfYear
        ConfCity  $
        ConfST    $ ;
DATALINES;
```

DELIMITER= parameter
(DLM=) Delimiter override

- **Changes default delimiter from blank or comma (DSD only) to specified value. All other rules of delimiters unchanged.**

Using INFILE and INPUT …

Comma Separated Value Files

Alternate Approach

```
NOTE: The data set WORK.SASCONF has 9
      observations and 4 variables.
```

< Results of a PROC PRINT >

Obs	ConfName	Conf Year	ConfCity	ConfST
1	SUGI	2006	San Francisco	CA
2	PHARMASUG	2006	Bonita Springs	FL
3	NESUG	2006	Philadelphia	PA
4	WUSS	2006	Irvine	CA
5	SESUG	2006	Atlanta	GA
6	SCSUG	2006	Irving	TX
7	MWSUG	2006	Dearborn	MI
8	PNWSUG	2006	Seaside	OR
9	SAS GLOBAL FORUM	2007	Orlando	FL

Using INFILE and INPUT …

DDE

Work in Progress

See separate PowerPoint presentation for DDE information.

Using INFILE and INPUT …

HTML

Work in Progress

```
x1=htmldecode ('not a &lt;tag&gt;');      not a
<tag>
x2=htmldecode ('&amp;');      &
x3=htmldecode ('&#65;&#66;&#67;');  ABC
```

Using INFILE and INPUT …

INPUT Function

Converts SAS variable from one form to another.

Does not interface with external files.

As such, will not be discussed in this presentation. (A quick explanation is in the paper.)

Conclusion

Many ways to use INFILE and INPUT to process external data.

This is just a starting point and cannot hope to be all-inclusive.

You can learn a lot from my errors, but you will learn a lot more from making and correcting your own.

Conclusion

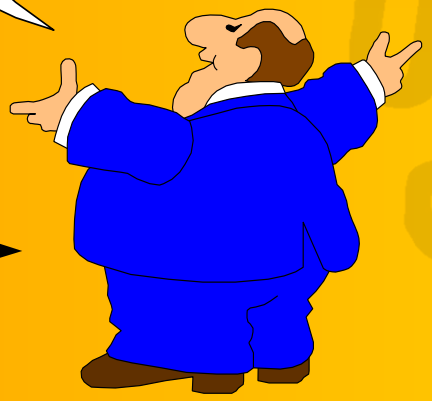
The author can be contacted at:

KuligowskiConference@gmail.com

Using INFILE and INPUT …

Conclusion

SAS is a registered trade-mark or trademark of SAS Institute, Inc. in the USA and other countries. (R) indicates USA registration.



My lawyer