Paper 140-2010

Dear Miss SAS®Answers: A Guide to Sorting Your Data

Jane Stroupe, SAS Institute Inc., Chicago, IL
Linda Jolley, SAS Institute Inc., Kansas City, KS

ABSTRACT

In this session, the fictitious advice columnist Miss SASAnswers tackles SAS sorting questions that were sent to her from instructors around the globe. Some of those questions include: "Is there some way to tell PROC SORT not to re-sort a previously sorted raw data file?", "What's with this NOEQUALS option that I've seen on someone else's program? Why did they use it? Should I?", and "What's the difference between PROC SORT and PROC SQL ORDER BY?" Many of Miss SASAnswers' replies contain new SAS 9.2 enhancements to PROC SORT that everyone will find exciting.

QUESTIONS AND ANSWERS

Dear Miss SASAnswers,

I've got all these data sets and I keep sorting them, but the log says that no sorting was done. So I have two simple questions:

- Why are they sorted already?
- How can I tell that?

Signed,

Beginning Billy

Dear BB,

The answer to the second one is easy. Just use PROC CONTENTS and see if the sort flag has been set. And notice the validated flag. Because it is YES, when SAS uses the data, it does not have to do a sequence check first. SAS knows that SAS has done the sort or that it has previously validated the sort order.

```
                          The CONTENTS Procedure
Data Set Name        SASHELP.CARS                     Observations          428
Member Type          DATA                             Variables             15
Engine               V9                               Indexes               0
Created              Wed, Jan 16, 2008 09:50:42 AM    Observation Length    152
Last Modified        Wed, Jan 16, 2008 09:50:42 AM    Deleted Observations  0
Protection                                            Compressed            NO
Data Set Type                                         Sorted                YES
Label                2004 Car Data
Data Representation  WINDOWS_32
Encoding             us-ascii  ASCII (ANSI)

< lines removed >

                                Sort Information
                              Sortedby       Make Type
                              Validated      YES
                              Character Set  ANSI
```

The first question is a little harder to answer because Miss SASAnswers doesn't really know what motivated the programmer who stored the data. But here are several reasons:

- Programmers often subset the data using a WHERE statement but don't want to index the data. A WHERE statement stops reading sorted data once the WHERE criteria is satisfied.
- They can avoid sorting every time they need to do BY-group processing. For example, they might be using the data in a DATA step merge, or using a BY statement in a procedure to report on groups of data. In addition, PROC SQL can optimize a join if the data is sorted.
- They need to create reports with the data in sorted order.

Happy Sorting,

Miss SASAnswers

Dear Miss SASAnswers,

I have a huge raw data file that is already in order by a product identifier number. I need to merge it with another SAS data set. I know I have to create a SAS data set from the raw data file first. But do I have to sort that data set?

Signed,

Sorting Sue

Dear Sue,

No, you don't have to sort the resulting SAS data set. Here are a couple of suggestions to help you tell SAS that the data is sorted.

- Assert a sort order with the SORTEDBY= data set option. However, this technique does not validate the sort. SAS still has to do a sequence check every time the data is used.

  Here's an example using a raw data file named **`detail.csv`**:

```
1,.5,Biscuit
1,.75,Toy
1,1,Food
.
.
.
20,5,Seed
```

```
data detail(sortedby=id);
   infile 'detail.csv' dsd;
   input ID Amount Purchase $;
run;
```

- Sort the resulting data set using the PRESORTED option in the PROC SORT statement. This option validates the sort and *allows* subsequent sequence checks to be bypassed.

```
data detail;
   infile 'detail.csv' dsd;
   input ID Amount Purchase $;
run;

proc sort data=detail presorted;
   by id;
run;
```

Happy Sorting,

Miss SASAnswers

PS Some parts of the system might choose to implement a sequence check regardless of the strength of the assertion (strong/validated or weak/not-validated).

Dear Miss SASAnswers,

It's me again. That PRESORTED option is really cool! When did that happen? Do you have to use it when you've created a data set from raw data? And what else can you do with it?

Signed,

Sorting Sue

Dear Sue,

Well, this did open a flood gate of questions, didn't it? You are right; it is really cool.

To answer your first two questions:

- The PRESORTED option was introduced in SAS 9.2.
- You can use it on any SAS data set. If there is no sort flag on the data, the PRESORTED option performs a sequence check first. If the data set is in sorted order by the BY variables, then PROC SORT doesn't do a sort. It sets the sort flag and the validated flag to YES. If it isn't in the correct order, PROC SORT does a sort, and sets the sort flag and the validated flag to YES. Here's a SAS log snippet to show what happens:

```
104  proc sort data=test presorted;
105     by id;
106  run;
NOTE: Sort order of input data set has been verified.
NOTE: There were 8 observations read from the data set WORK.TEST.
```

And here's what PROC CONTENTS shows:

```
Sort Information
Sortedby       id
Validated      YES
Character Set  ANSI
```

As far as other uses of the PRESORTED option, it really has only one use: to validate sort order. But look at the attached example to see how powerful that is!

Happy Sorting!

Miss SASAnswers

Attachment: PreSortedEx.sas

```
data jan_wk1;
   infile datalines;
   input date:date9. amount;
datalines;
05jan2009 500
06jan2009 400
07jan2009 600
08jan2009 100
09jan2009 400
;

data jan_wk2;
   infile datalines;
   input date:date9. amount;
```

```
datalines;
12jan2009 300
13jan2009 700
14jan2009 400
15jan2009 600
16jan2009 300
;


data jan_wk3;
   infile datalines;
   input date:date9. amount;
datalines;
19jan2009 400
20jan2009 200
21jan2009 700
22jan2009 200
23jan2009 400
;

data jan_wk4;
   infile datalines;
   input date:date9. amount;
datalines;
26jan2009 100
27jan2009 300
28jan2009 600
29jan2009 700
30jan2009 200
;


data jan_v/view=jan_v;
   set jan_wk1-jan_wk4;
run;

proc sort data=jan_v out=jan presorted;
   by date;
run;
```

Dear MissSASAnswers,

What's the difference between the "dup" options for PROC SORT?

Signed,

Duplicated

Dear Duplicated,

I'm assuming you mean the NODUPKEY, NODUPRECS, and DUPOUT options for the PROC SORT statement.

Here are the options and what each of them does:

Option	Description
NODUPKEY	Checks for and eliminates observations with duplicate BY values.
NODUPRECS	Checks for and eliminates consecutive observations with duplicate BY values.
DUPOUT=	Specifies the output data set to which duplicate observations are written.

Use NODUPKEY to create a data set containing the unique values of the BY variables.

```
proc sort data=numbers nodupkey
          out=unique;
   by Num1;
run;
```

Data Set Numbers

Num1	Num2	Num3
1	2	3
2	3	8
1	2	3
1	4	5
1	2	3
2	3	8
2	4	6
1	4	5
2	5	4

Data Set Unique

Num1	Num2	Num3
1	2	3
2	3	8

Use DUPOUT= with either of the other options to create a data set containing the duplicate observations of the BY variables.

```
proc sort data=numbers nodupkey
          out=unique
          dupout=extras;
  by Num1;
run;
```

Data Set Numbers

Num1	Num2	Num3
1	2	3
2	3	8
1	2	3
1	4	5
1	2	3
2	3	8
2	4	6
1	4	5
2	5	4

Data Set Unique

Num1	Num2	Num3
1	2	3
2	3	8

.

Data Set Extras

Num1	Num2	Num3
1	2	3
1	4	5
1	2	3
1	4	5
2	3	8
2	4	6
2	5	4

Use NODUPRECS to create a data set containing the *consecutive* duplicate observations.

```
proc sort data=numbers noduprecs
          out=unique
          dupout=extras;
   by Num1;
run;
```

Data Set Numbers

Num1	Num2	Num3
1	2	3
2	3	8
1	2	3
1	4	5
1	2	3
2	3	8
2	4	6
1	4	5
2	5	4

Data Set Unique

Num1	Num2	Num3
1	2	3
1	4	5
1	2	3
1	4	5
2	3	8
2	4	6
2	5	4

Data Set Extras

Num1	Num2	Num3
1	2	3
2	3	8

Remember that the operative word in the definition of NODUPRECS is “consecutive”.

Use NODUPRECSS with all of the variables in the BY statement to create a data set containing the duplicate observations.

```
proc sort data=numbers noduprecss
          out=unique
          dupout=extras;
   by Num1 Num2 Num3;
run;
```

Data Set Numbers

Num1	Num2	Num3
1	2	3
2	3	8
1	2	3
1	4	5
1	2	3
2	3	8
2	4	6
1	4	5
2	5	4

Data Set Unique

Num1	Num2	Num3
1	2	3
1	4	5
2	3	8
2	4	6
2	5	4

Data Set Extras

Num1	Num2	Num3
1	2	3
1	2	3
1	4	5
2	3	8

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

What's with this NOEQUALS option that I've seen on someone else's program? Why did they use it? Should I?

Signed,

Equally Confused

Dear Equally Confused,

The NOEQUALS option is turning off the default sort option of EQUALS. With EQUALS turned on, SAS is maintaining sort stability. What that means is that when SAS sorts the data, it maintains the original order of the data within each BY group.

Here's an example. Notice the order of the highlighted rows of data.

```
proc sort data=Feb;
   by amount;
run;
```

before the sort

date	amount
03FEB2010	100
06FEB2010	200
14FEB2010	300
07FEB2010	300
01FEB2010	100
16FEB2010	200
05FEB2010	300
04FEB2010	200

after the sort

date	amount
03FEB2010	100
01FEB2010	100
06FEB2010	200
16FEB2010	200
04FEB2010	200
14FEB2010	300
07FEB2010	300
05FEB2010	300

Notice that in the results, the variable amount is in sorted order and that within each value of amount the value of data is in the original order. For example, for the amount 100, the dates are 03FEB2010 and 01FEB2010, in that order.

Hope this helps!

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

That makes sense, but how does the EQUALS option work? And what happens if you use the NOEQUALS option?

Signed,

Equally Confused

Dear Equally Confused,

With the EQUALS option, a sequence number is appended to the key created from the BY variables. Then the data is sorted by the BY values, keeping the sequence numbers in order.

With the NOEQUALS option, there is no sequence number used, so PROC SORT does not necessarily preserve the original order in the output data set.

Using NOEQUALS can conserve memory and CPU, if you do not care that the relative order of the observations within the input data set is maintained in the output data set.

Note that sort stability is separate from the issue of non-deterministic ordering of the input. Sort stability only preserves the relative order of the input within the output BY groups. If the input ordering isn't consistent from one read of the data to the next, then, even with EQUALS, SORT won't produce the same output from one run to the next. Because Base SAS data sets have a physical order (a sequence) to them, the order in which observations are (sequentially) read from them is consistent from one read to the next. But data arriving from other engines (SAS/ACCESS engines, for instance) might not be consistent from one read to the next. Because of multi-threading, partitioning, and other reasons, DBMSs may deliver rows to SAS in a non-deterministic order.

There are a few concerns about EQUALS|NOEQUALS:

- When NODUPRECS or NODUPKEY is used to remove observations in the output data set, the choice of EQUALS or NOEQUALS can affect which observations are removed.
- I/O performance might be reduced when using the EQUALS option with the multi-threaded sort because partitioned data sets will be processed as if they consist of a single partition.
- The order of observations within BY groups that are returned by the multi-threaded sort might not be consistent between runs if the NOEQUALS option is used.

Happy Sorting!

Miss SASAnswers

PS The observations that are eliminated and the one that is kept for NODUPKEY and NODUPRECS can't be guaranteed for SAS/ACCESS engines. It is guaranteed that PROC SORT will keep the first observation it encounters for each BY group when reading the input data set, but which observation ends up being first is dependent upon the order in which they are delivered to SAS. In other words, if the delivery order is non-deterministic, then so is the behavior of NODUPKEY and NODUPRECS.

Dear Miss SASAnswers,

I can use either a sorted or an indexed data set for my ordered processing. Is it better to index than to sort?

Signed,

Too Many Choices

Dear Choices,

Well, that depends on your data and what you mean by “better”.

You have to answer the following questions.

1. Do many users need this data set in this order?

   If many users need the data in this order, then sorting is probably your best bet. So store the data in sorted order.

```
proc sort data=detail;
   by amount;
run;
```

2. Are you combining this data set with other data sets?

   If you are combining this data with other data sets, you can use either an index or sorted data. The sparseness of the match and the match technique determine which one you choose.

   For sparse matches with either an SQL inner join or the DATA step SET/SET KEY technique, use an index.

```
proc sql;
  create table match as
    select *
      from detail as d,
           customers as c
        where d.id=c.id;
quit;

/*********************************************************/
/* The following method can use lots of I/O if the data is */
/* not sorted prior to indexing. But once that is          */
/* done, this method is really quick.                      */
/*********************************************************/

data match;
  set customers;
  set detail key=id;
run;
```

   For dense matches with either an SQL inner join or the DATA step MERGE technique, sort the data.

```
proc sql;
  create table match as
    select *
      from detail as d,
           customers as c
        where d.id=c.id;
quit;

data match;
  merge customers detail;
  by id;
run;
```

3. Is the data already indexed on the variables you need?

   If the data is already indexed, then the BY statement can take advantage of the index. However, if the data is not ordered by the indexed variable, then there might be a large I/O cost for using the index.

4. Are you using PROC APPEND to update your data?

   PROC APPEND and the PROC SQL INSERT INTO statement will update the index once the append has been completed. In this case, if there is an index, you do not have to re-sort the data as you would if you were relying on sorted data for BY-group processing.

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

My company has Syncsort licensed. Can I use Syncsort for my SAS sorts?

Signed,

Syncsort Savvy

Dear Savvy,

Sure! There are three SAS system options that work together to let you use a host sort. Remember, the Syncsort product itself needs to be installed, and then SAS needs to be told how to find and use it.

The values for these options are operating system specific.

Option	Description
SORTPGM=	Specifies whether SAS sorts using the SAS sort utility or the host sort utility.
SORTCUTP=	Specifies the number of bytes up to which SAS sorts. If the number of bytes in the data set is greater than the specified number, the host sort program sorts the entire data set.
SORTNAME=	Specifies the name of the host sort utility.

Here's an example:

```
options sortpgm=host
        sortcutp=1M
        sortname=syncsort;
proc sort data=detail;
   by ID;
run;
```

Consult the Operating System Companion for your platform to see the specifics on using host sorts in your environment.

Happy Sorting!

Miss SASAnswers

Dear MissSASAnswers,

We have a big machine with lots of CPUs available. Is the SAS sort taking advantage of all these processors?

Signed,

CPU Happy

Dear Happy,

That depends on how many of those CPUs have been allocated for SAS use with the CPUCOUNT SAS system option and whether threading has been enabled. You can determine how many CPUs you have by running a PROC OPTIONS step.

```
proc options option=cpucount;
run;
```

Then use the CPUCOUNT= system option to set the number of CPUs.

```
options cpucount=actual;
proc sort data=detail nothreads;
   by ID;
run;
```

Threading is actually the default when you submit a PROC SORT step. To disable threading, you need to use the NOTHREADS option in the PROC SORT statement or the NOTHREADS SAS system option.

The scalability of the multi-threaded sort is limited. Processor utilization is affected by I/O speeds and data characteristics (for example, data dimension – wide versus narrow observations, etc.). For compute bound jobs, it can effectively use more than one processor – usually two, but often no more than four.

Threading takes advantage of multiple CPUs by dividing processing among the available CPUs, but some types of threading can be performed using a single CPU. The operating system of a symmetric multiprocessing (SMP) machine is capable of scheduling code segments so that they execute in parallel.

Hope this explanation is helpful.

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

I analyze data for several large apartment complexes. When I try to list the tenants in order based on their unit number within a building, the sequence is off. For example, units 10A and 11C appear before units 2B and 3D. How can I sort the apartments so that the unit numbers are in the correct sequence?

Signed,

Untenantable

Dear Untenantable,

There is a new SAS 9.2 LINGUISTIC option value for the SORTSEQ= option in the PROC SORT statement that will make your life much easier. The LINGUISTIC option allows several different suboptions and values for the collating rule modifier. Here's an example:

```
proc sort data=apartments
          sortseq =
          linguistic(numeric_collation=on);
   by AptNum;
run;
```

Here's the result, in case you are curious.

Data Set: Apartments

AptNum
22B
22C
12A
12B
3C
3A
10A
11D

Default Sort

AptNum
10A
11D
12A
12B
22B
22C
2B
3A

Numeric Collation

AptNum
2B
3A
3C
10A
11D
12A
12B
22B

There is a lot more you can do with the SORTSEQ= option.

Here are some additional suboptions for the LINGUISTIC option:

Suboption	**Values**	**Purpose**
CASE_FIRST=	UPPER | LOWER	Either value can sort data so that like letters are sorted correctly irrespective of case.
LOCALE	*locale_name*	Sorts data according to culturally sensitive rules.
STRENGTH=	PRIMARY | SECONDARY | TERTIARY | QUATERNARY | IDENTICAL	The PRIMARY value supports sorting that is not case sensitive. The other values handle diacritical and punctuation differences.

The UPPER | LOWER values for the CASE_FIRST= argument specify which comes first in the list, uppercase or lowercase letters.

Here's an example of CASE_FIRST=:

```
proc sort data = names
          sortseq = linguistic(case_first = upper|lower);
  by value;
run;
```

Data Set Names

Name
Linda
JANE
linda
LINDA
Jane
jane
LINda
jaNe

Default Sort

Name
JANE
Jane
LINDA
LINda
Linda
jaNe
jane
linda

CASE_FIRST=UPPER

Name
JANE
Jane
jaNe
jane
LINDA
LINda
Linda
linda

CASE_FIRST=LOWER

Name
jane
jaNe
Jane
JANE
linda
Linda
LINda
LINDA

You can also use NODUPKEY with SORTSEQ=LINGUISTIC(STRENGTH=PRIMARY). That will eliminate duplicates even if they differ in order or case.

Here's an example of STRENGTH=:

```
proc sort data=names nodupkey
          sortseq=linguistic(strength=primary|secondary);
  by Name;
run;
```

Data Set Names

Name
Linda
JANE
linda
LINDA
Jane
jane
LINda
jaNe

Default Sort with NODUPKEY

Name
JANE
Jane
LINDA
LINda
Linda
jaNe
jane
linda

NODUPKEY STRENGTH=PRIMARY

Name
JANE
linda

NODUPKEY STRENGTH=SECONDARY

Name
Jane
jane
jaNe
linda
LINDA
LINda

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

I can use techniques other than PROC SORT to order my data: for example, the CLASS statement in PROC MEANS or the ORDER BY clause in an SQL SELECT statement. How does the data get ordered? By PROC SORT? Some kind of "hocus pocus" magic?

Signed,

Getting It Done

Dear Getting It Done,

The ORDER BY clause in the SQL procedure calls the SAS sort routine behind the scenes. The CLASS statement uses another technique called an AVL tree to group the data. One advantage to you, of course, is that you do not have to code a PROC SORT step before using either one.

Happy Sorting!

Miss SASAnswers

PS Of course, you need to benchmark all of these. PROC MEANS with a CLASS statement might take more memory than PROC MEANS with a BY statement.

And because PROC SQL is using ANSI standards, an ORDER BY clause does not maintain sort stability like PROC SORT with the EQUALS option does; it might take less CPU.

But I'm not guaranteeing any of this, so make sure you benchmark correctly. And if you want to know how to do that, just send me another e-mail!

Happy Sorting!

Dear Miss SASAnswers,

Our organization determines the age of a data set by the last modified date shown by PROC CONTENTS. Unfortunately, when I sort a data set, the date changes even though none of the underlying data values have changed. This can be misleading for our research analysts. Is there a way to have PROC SORT leave the date alone?

Signed,

Leave My Dates Alone

Dear LMD,

The DATECOPY option in the PROC SORT statement is designed to do just that. It copies the SAS creation date and time and the SAS modification date and time to the resulting sorted data set.

```
proc sort data=detail datecopy;
   by id;
run;
```

Happy Sorting!

Miss SASAnswers

PS The dates copied are *not* the date/time stamps on the *files* associated with the operating system but the dates stored in the data set descriptor portion. The option has an effect only when replacing the input data set (either implicitly as in `proc sort data=x;` or explicitly as in `proc sort data=x out=x;`).

Dear Miss SASAnswers,

Every time I sort a data set "in place", I lose the indexes associated with the data set. (I've learned about using the FORCE option to make the sort happen despite the indexes, and then rebuilding the indexes with PROC DATASETS.) Can PROC SORT re-create those indexes automatically after the sort is complete?

Signed,

Tired of Sorting and Indexing

Dear Tired,

You can use the INDEX= data set option with the OUT= PROC SORT option to re-create your indexes. Just be prepared for it to take a while!

Here's an example:

```
proc sort data=detail
          out=detail (index=id);
   by id;
run;
```

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

Why does it take so loonngg to sort my data?

Signed,

Time on My Hands

Dear Time,

In order to answer your question, you need to understand how PROC SORT works.

The SAS sort utility has the following characteristics:

- is supplied by SAS for all operating environments
- executes in memory up to the limit imposed by the SORTSIZE= option
- minimizes its use of external storage

The first bullet point has a corollary: Because the sort utility must run on all platforms supported by SAS, it can't take advantage of any platform-specific sort enhancements. The SAS sort tries to complete entirely in memory, if possible.

And remember that using more threads will help reduce the real time for your sort to complete.

Happy Sorting!

Miss SASAnswers

Dear MissSASAnswers,

Can you please give me some guidance on how much memory it takes to sort a 2GB table?

Signed,

Memory Deprived

Dear Memory Deprived,

If you want the sort to complete entirely in memory, a simple rule of thumb is four times the size of the data set.

And I'm assuming the data set is not compressed or being subset with a DROP=/KEEP= data set option or a WHERE statement.

A better estimate would be to use this formula to predict the amount of memory:

```
((length of observation+sum of lengths of BY variables)*number of observations)* 1.10
```

The amount of space that the SAS sort needs depends on the following conditions:

- whether the sort can be done with threading
- the length of the observations
- the number of variables in the BY statement and their storage lengths
- the operating environment in which PROC SORT executes
- whether the LINGUISTIC= option is being used (that takes more memory)

Use the SORTSIZE= option in the PROC SORT statement to do the following:

- specify the amount of memory that is available to the SORT procedure
- improve the sort performance by restricting the swapping of memory to disk that is controlled by the operating system

SORTSIZE=*n* | *n*K | *n*M | *n*G | MIN | MAX | *hex*X | SIZE

The syntax is very dependent on your operating environment.

From a general application perspective, all memory is "virtual memory". That is, applications simply request memory from the operating system without reference to and without knowing whether it will be swapped out to external storage. When the memory is referenced, it might either already be in RAM, or it might have been swapped out to disk (in which case there is a "page fault", and the memory is read back in from disk before it's actually used). This all happens transparently.

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

That helps, but I still don't understand the SORTSIZE= option. What should I set it to?

Signed,

Memory Deprived, but hunting for more

Dear Memory Deprived,

That question is difficult to answer! There's no single SORTSIZE value that provides optimal performance for all possible sorting jobs.

So don't look for an optimal setting. You'll just have to benchmark to find a SORTSIZE value that is reasonable and good enough for all your jobs.

Using a SORTSIZE value that is either too small or too large can result in poor performance. So rule those out!

Here's a hint, however. For optimal performance, set the SORTSIZE= option to a value less than the available physical memory. This enables the programs and the operating environment to stay resident in memory.

To benchmark, just run a program like this with different values of the SORTSIZE= option.

```
options fullstimer;
proc sort data=SAS_dset sortsize=??????;
   by Var;
run;
options nofullstimer;
```

Another hint is to try an undocumented PROC SORT option, DETAILS, that gives you information about the memory used.

```
proc sort data=orders details
   by Order_ID;
run;
```

Here is the log from a multi-threaded sort:

```
NOTE: There were 747953 observations read from the data set WORK.ORDERS.
NOTE: Sort completed in memory.
NOTE: The data set WORK.ORDERS has 747953 observations and 6 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time           2.23 seconds
      cpu time            0.57 seconds
```

Here is another log from a multi-threaded sort where the data set was much larger:

```
NOTE: UTILITY FILE REQUIRED.
NOTE: UTILITY FILE 1 PAGE SIZE IS 65536 BYTES.
NOTE: THERE WERE 2243859 OBSERVATIONS READ FROM THE DATA SET WORK.ORDERS.
NOTE: UTILITY FILE 1 CONTAINS 2243859 RECORDS AND 3 SORTED RUNS.
```

```
NOTE: UTILITY FILE 1 CONTAINS 1372 PAGES FOR A TOTAL OF 87808.00 KB.
NOTE: THE DATA SET WORK.ORDERS HAS 2243859 OBSERVATIONS AND 6 VARIABLES.
NOTE: PROCEDURE SORT USED (TOTAL PROCESS TIME):
      REAL TIME           22.71 SECONDS
      CPU TIME            2.98 SECONDS
```

Here is another log when the data set was much smaller and the sort was not multi-threaded:

```
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
mempage=16384 alocsize=1184 isa=16384 osa=16384 xmisa=0
holds=292 nway=1 sortsize=67108864 memoryuse=17568.00
keylen=16 reclen=40 dkin=0 inrec=19 outrec=19 yieldobs=0
nruns=1 xcbpage=16384 npages=0 diskuse=0.00
```

There is a lot of information in the log. I'll describe some of it later, but take note of the SORTSIZE and the MEMORYUSE options. They are telling you how much memory is available (SORTSIZE) and the amount used (MEMORYUSE). Some of the other information is for the developers, so don't worry about that!

Option	Description
MRGCOUNT	The number of passes over the utility file that are required to merge the sorted runs. If this number is greater than 1 (a multi-pass merge) then the amount of memory provided to sort (SORTSIZE) should be increased. Multi-pass merging requires a doubling of the utility file storage space and requires proportionally more time.
SORTSIZE	Amount of memory that PROC SORT believes is available for allocation.
MEMORYUSE	Approximately, the amount of memory that the sort used for in-core (in-memory) sorting.
KEYLEN	Length of the composite sort key. Generally, the sum of the lengths of the BY variables plus some additional amount for other purposes.
RECLEN	Length of the sort record. Generally, length of the observation from the data set being sorted. If variables are dropped (DROP/KEEP), then this number can be less than the length of the observation. For a TAGSORT, this will be considerably less than the length of the observation (it will be the length of the record identifier or RID plus some additional amount).
INREC	The number of records (observations) read by the sorting routine.
OUTREC	The number of records (observations) written by the sort. The NODUPKEY and NODUPRECS options can cause the number of output records to be less than the number of input records.
NRUNS	The number of sorted runs formed within the utility file if the sort has gone external. This indicates that the sort required external storage because it cannot be completed within the confines of memory. (This number will be 1 if the sort completed in memory.)
XCBPAGE	The size of the utility file page. This is the I/O unit used by PROC SORT when reading from and writing to the utility file.
NPAGES	The number of pages written to the utility file.
DISKUSE	The amount of utility file storage space required to perform the sort. This is equivalent to (xcbpage * npages).

Happy sorting!

Miss SASAnswers

Dear Miss SASAnswers,

Well, that is interesting. And while you were typing all that, I found two options that might help. One of them is the OVERWRITE option and the other is the TAGSORT option. But I don't understand either of them very well. Can you explain what they do, please?

Signed,

Memory Deprived, and still looking

Dear Memory Deprived,

One of those options might help your memory problem, but the other one won't. I'll explain what they do.

The OVERWRITE option can help with disk space issues, but not memory. It will delete the original data set before it writes out the new one. It doesn't do anything if you use the OUT= option.

Just please, please, do not use the OVERWRITE option unless you have backed up the original data!

The TAGSORT option in the PROC SORT statement potentially could save memory. The TAGSORT option creates a temporary file containing only tags that contain the BY variable values and the observation numbers. PROC SORT then sorts the tags. When the sort is finished, the tags and the original data are combined to create the sorted data.

The TAGSORT option can save disk space when the length of the observations is large and the length of the BY variables are short in comparison to total observation length. On the other hand, it can use more computer resources. In addition, it turns multi-threaded sorts into single-thread sorts.

Keep all of the caveats for the OVERWRITE and the TAGSORT options in mind when you contemplate using them. Remember that you can use both of them in the same PROC SORT statement.

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

Is it better to sort two data sets individually and then interleave them, or to concatenate them and then sort the result?

Signed,

Putting Them Together

Dear Putting Them Together,

I'd say that if you have only two data sets, then it probably doesn't matter which technique you use. However, if you are interleaving a lot of data sets, then it is probably better to concatenate them and then do the sort. Keep in mind that when interleaving, all the data sets have to be open in memory at one time.

As with any question about which is better, you'll have to test this with your data to determine which technique handles your data better.

Happy Sorting!

Miss SASAnswers

Dear Miss SASAnswers,

Which should I use to sort the data: PROC SORT or PROC SQL with an ORDER BY clause? My co-workers say PROC SORT, but I'd prefer PROC SQL!

Signed,

Love that SQL

Dear Love that SQL,

SQL is wonderful, but if all you are doing is sorting and your data sets are large, then PROC SORT is your choice. For small data sets, either is fine.

What's the difference? Well, SQL sends the results of the sorting to one of its own utility files before copying them to the output table, and that increases your I/O considerably.

Of course, one PROC SQL query can do much more than just sort, so keep that in mind.

Happy Sorting!

Miss SASAnswers

CONCLUSION

Miss SASAnswers hopes she has answered some of your sorting questions. Just remember, if you understand how the SORT procedure works and how to use all of the great options available for sorting, you can control the resources that are used. In addition, there are many sort options that provide functionality beyond just sorting the data, options such as DUPOUT= and DETAILS.

Miss SASAnswers might not exist in real life, but there are many places that you can find assistance. When you have questions, help is just a phone call, e-mail, or Web search away. Contact SAS Technical Support with questions, or check out SAS training that is offered on the Web or in the classroom. Happy sorting!

ACKNOWLEDGMENTS

The authors appreciate the help of Scott Mebust in reviewing this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Linda Jolley, Technical Training Specialist
SAS Institute Inc.
Kansas City Regional Office
Phone: (913) 491-1166
E-mail: Linda.Jolley@sas.com

Jane Stroupe, Technical Training Specialist
SAS Institute Inc.
Chicago Regional Office
Phone: (919) 531-9049
E-mail: Jane.Stroupe@sas.com