A Method for Sorting a Large Data Set with Limited Memory
Leslie LaBrecque, Health Care Access and Cost Commission,
Maryland Department of Health & Mental Hygiene

ABSTRACT
You’re asked to do the impossible: perform a sort on a large data set that resides on a server or PC with limited memory. In the case where a more practical approach, such as indexing, is not feasible because data retrieval patterns have not been established, or perhaps a quick look at the data is needed, it is possible to break the sort down into chunks that your system can handle. Writing a small reusable macro which sorts the data chunks and merges the chunks can accomplish this task.

INTRODUCTION
Sorting data is a commonplace task that we are sometimes required to perform on large data sets. In the LAN environment, whether you are running a sort at the server or at the PC client, a sort on a large dataset can fail due to memory limitations at either place. Many more efficient methods of sorting or achieving a sort without actually sorting are available and each has its advantages. This method is best suited for ad hoc data processing in situations where indexing is not efficient and non-duplicate records are needed.

METHOD
One way to accomplish a sort when a straight PROC SORT on the data set fails with a memory error, is to subset the data set into multiple smaller data sets that have a manageable sort size for your system. The number of observations for each subset would be a function of the range of values of the variables being sorted, and the number of variables being sorted.

The following macro allows the user to specify the maximum number of observations to sort in each subset, as well as the data set name and the sort variables. This macro is written so that for any given input data set, the user needs only to change the LIBNAME statement, data set name (infile), maximum number of observations to sort (incremnt), sort variables (byvars), and variables to keep in the output sorted data set (keepvar).

```sas
libname lib1 'd:..library location..';
%let infile = filenm1;
%let incremnt = 999999;
%let byvars = ssn dob;
%let keepvar = ssn dob;
%macro qdsort;
  * get number of records in dataset *;
  data _null_
  retain p 1;
  set lib1.&infile nobs=nobs point=p;
  call symput('nobs',nobs);
  stop;
  run;
  
  * initialize macro counters and sort *;
  * 'incremnt' size chunks of data and *;
  * write to temporary files *;
  %let process = 0;
  %do %while(&process<&nobs);
    %let process = %eval(&process + 1);
    proc sort data=lib1.&infile
    (firstobs=&process obs=%eval(&process + &incremnt)
      keep=&keepvar)
    out=temp&process nodupkey;
    by &byvars;
    run;
    %let obs1=
    %eval(&obs1+(&incremnt+1));
  %end;
  * merge sorted temporary files *
  data lib1.&infile;
  merge
  %do i=1 %to &process;
    temp&i;
  %end;
  by &byvar;
  run;
%mend;
%qdsort;
```

HELPFUL HINTS
It will take some experimentation to determine an increment that allows the sort to complete without a memory error. If possible, try to increase the size of the work space and try specifying a drive where more space is available. You can also modify the code to write the subsets to permanent data sets and then delete them when done.

CONCLUSION
The macro determines the number of records in the data set and reads the data set incremnt observations at a time and outputs a sorted subset. When all subsets of the data set have been sorted, the subsets are then merged together with a BY statement which preserves the sorted records across all subsets. The result is a sorted and unduplicated version of the original dataset.

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries.

Leslie M. LaBrecque
4201 Patterson Avenue, 5th Floor
Baltimore, MD 21215
Phone: (410) 764-3583
E-mail: LLABREQ@EROLS.COM