

How Fast Can You Type **or** **Go Ahead and Get Snippety**

John Charles Gober
U.S. Department of Commerce, Bureau of the Census

Introduction

When this paper was first envisioned, a number of titles were considered: Creating a Better Toolbox, Its OK to Plagiarize, Programming for Power, etc. However after discussing the concept with my fellow technical experts they all said the same thing. It all comes down to one thing. Time is money, and how fast can your people come up with the necessary code and type in those needed statements is the critical factor. Can you do the application in three weeks, three days, or three hours?

Whether the office you work in employs a single programmer or hundreds of programmers containing the complete spectrum of experience levels it always helps to have pre written code you can draw upon. Code a junior programmer can learn from. Code the experience programmer can easily adapt to suit their needs. Little pieces of code or snippets that can be copied, modified, or used as a learning tool. Code already located in a common area so you don't have to reinvent the wheel.

Scenario

It is 9:00 Friday morning on a Holiday weekend. You have been tasked with the following assignment which needs to be completed by Monday morning: Create a series data steps or SAS® procedures that will read a control file containing Fips State and Fips County codes for the entire United States. Using this control file, do a check for each State to see if that State has a full

complement of County files with a name pattern of ssu{ssccc} where ss represents the state and ccc represents the county. If yes, rollup the County files to a State file called ssu_{ss} and upload the original County files and the rolled up State level files to a remote platform. If not, e-mail your supervisor that there are some States with missing county files and that those State files containing the missing Counties could not be created. All files are SAS Version 8. The control file will have two variables call FIPST and FCNTY and the program will execute individually for each State.

None of this code is very difficult. Read a file, read a directory, a few if then else statements, and email, and an upload program. However, if you have never done an application similar to this it may take a few minutes to just come up with the ideas let alone the code. Fortunately examples of almost all of the code you need for this task has already been placed in a toolbox. You will be done by noon. Details to follow.

Input Snippets

Here are examples of several input snippets that might be useful for our scenario. The first one reads the files in a directory using Unix commands invoked by shelling out to operating system by use of the SAS 'X' command. This is used more for reading the names of non-SAS files but I thought I would throw it in. The second snippet is very similar to the first but uses the SAS filename statement and the pipe engine. The third snippet uses PROC SQL to create a view table

containing the SAS dataset names. All SAS dataset are assumed to be in the pre assigned library mylib. All data is stored in the directory /ssu/.

```
*Getting file list using x command;
*In Unix, 7th word from ls command;
*contains filename.;
*Best for non-SAS files.;
  x 'ls -lR /ssu/*.sas7bdat >
flist';
  data read1;
    infile 'flist' lrecl=100 pad
      missover;
    input (v1 - v7) ($);
    fipst = substr(v7,1,2);
    fcnty = substr(v7,3,3);
  run;

*Getting file list using pipe;
*In Unix, 7th word from ls command;
*contains the filename.;
*However may be flavor dependent.;
*Best for non-SAS files.;
  filename flist pipe
    'ls -lR /mylib/*.sas7bdat';
  data read1;
    length v1 - v7 $20;
    infile flist lrecl=100 pad
      missover;
    input (v1 - v7) ($);
    fipst = substr(v7,1,2);
    fcnty = substr(v7,3,5);
  run;

*Getting filelist using the;
*SASHELP.VTABLE;
  data read1;
    set sashelp.vtable;
    if libname='mylib' and
      substr(memname,1,4) = 'SSU' and
      memname ^= 'SSU_04';*arizona;
    fipst = substr(v7,1,2);
    fcnty = substr(v7,3,3);
  run;
```

The last method is my preferred method since you are using one of SAS's many behind the scene administrative files. It also gives a chance for less experienced programmers to see what SAS has hidden away in other administrative that they might not have found out otherwise.

An Email Snippet

In a production environment, in my opinion, there is not a better way of notifying people of a job status. This is especially true when you have hundreds of jobs a day executing. Messages that range from *'MR. Smith, Your weekly status reports are ready.'* to *'Warning!!! Job xyx123 has possible data anomalies and has been halted'*. Below is an email snippet from our system, which is easily modifiable to suit our needs. Just break out the code you need and insert.

```
*Email to two recipients;
*with attachment;
filename pgmmail email
  "recipient.one@yourworkplace
  recipient.two@yourworkplace"
  subject="SAS email test"
  attach="/ssu/attach.txt";
data _null_;
  file pgmmail;
  put "this is a test.";
  put "more lines, etc.";
run;

*email info can also be placed on;
*the file statement.;
file pgmmail
to=("recipient.one@your.workplace"
  "recipient.two@your.workplace")
subject="SAS email test"
attach="/ssu/attach.txt";
```

An item to note that with mail, depending on your mail application, messages can also be sent to group addresses. This is a very nice feature for contacting support teams and help desks.

A Snippet to Create a String of Datasets

According to the SAS Map viewtable there are 3,143 Counties in the United States. Since these Counties at to be 'rolled up' to the State level this means the possibility that someone will have to manually type in those thousands of County datasets into their programs. Two ideas on how to accomplish this task come to

mind. The first idea uses a data step routine to read a view containing the State and County variables for all the Counties in a State. These two variables are concatenated together along with a prefix representing a library name. This new variable is then retained and added together with the same variable from the next dataset read. This creates the extremely long variable representing all of the state county combinations found in the input view. Note the length of the variable 'filestring' which means that this snippet can only be executed in SAS Version 8.

```
data temp;
  length filestring $32000;
  retain filestring ' ';
  set sashelp.vtable end=last;
  where lowercase(libname) = 'mylib'
    and substr(memname,1,4) = "ssu_"
    and memname ^= 'ssu_us';
  x='mylib.ssu' || put(memname,$8.);
  filestring = x||' ' ||
    left(filestring);
  if last then call
  symput('filestring',filestring);
run;
```

The second snippet does the exact same thing as the snippet above except that it uses PROC SQL and the ability to automatically create a macro variable containing all of the state/county combinations in a view.

```
proc sql noprint;
  select distinct "mylib." ||
    trim(memname)
  into: filestring separated by ' '
  from sashelp.vtable
  where lowercase(libname) = 'mylib'
    substr(memname,1,4) = "ssu_"
    and memname ^= 'ssu_us';
quit;
```

Notice the use of the colon to create a macro string in the PROC SQL example. Both sets of codes above should also have their records subset by State to make the scenario run correctly.

An Upload Snippet

Using proc upload is relatively easy to use. You can upload text files, SAS datasets, or even entire SAS libraries. I am including it here just to show a relatively unknown feature of PROC UPLOAD and PROC DOWNLOAD. The ability to group and upload/download SAS datasets by prefix. The following example will upload all datasets with the prefixes of 'ssu_'. This is a great advantage for uploading multiple files however one must be very conscious of dataset naming patterns.

```
*libname for local;
libname one '/mylib';
signon;
rsubmit;
  *libname for
  libname two '/mylib/';
  proc upload in=one out=two;
    select temp: txp: ;
  run;
endrsubmit;
signoff;
```

A Snippet to Subset a SAS Dataset

This snippet is not needed for our scenario but might be useful for other applications so I thought I would also include it in this paper. It is not very efficient and can definitely be improved but the general concept is one I thought might be useful. Actually it is more of an entire program than a snippet but still warrants some attention. However since it is also rather long I will include the code at the end of the paper after we piece together all the other snippets.

Conclusion

It does not matter if you are an experienced programmer or one who just dabbles. It also does not matter if you are the only SAS programmer in your office or one of a hundred. Having a centralized location where you can access ideas, techniques and code has

great benefits. Foremost it saves time and effort when coding. Secondly it is a good learning tool. A toolbox can contain small pieces of adaptable code, macro routines, example of functions, and redundant code. It is something every office needs to create and to serve its full potential it must be dynamic.

Credits and Acknowledgments

This paper contains programming code and techniques other than those developed by the author. Many people from the U.S. Bureau of the Census have contributed and acknowledgement is duly given to all.

Information has also been collected and modified from SAS OnlineDoc®, Version 8.

SAS Institute Inc., SAS OnlineDoc®, Version 8, Cary, NC: SAS Institute Inc., 1999.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

```
*PROGRAM TO DEMONSTRATE SNIPPETS;
*THAT CAN BE USED TO ROLL UP    ;
*DATASETS.;
*ARIZONA IS THE TEST STATE. BE  ;
*CAREFULL ABOUT LEADING ZEROS  ;
*AND NAMING CONVENTIONS WHEN   ;
*WRITING YOUR OWN.;
```

```
*GET STATE NUMBER FROM SCRIPT FILE;
%let state = %sysget(STATE);
```

```
*OR HARDCODE THE VALUE;
*%let state = 4;
```

```
*ADD LEADING ZERO TO STATE;
*WOULD NOT NEED IF BETTER NAMEING;
```

```
*CONVENTION IS USED;
data null;
  call
  symput('ST',put("&STATE",z2.));
  stop;
run;
```

```
*GET LIST OF ALL COUNTIES IN STATE;
data allctys(keep=state county);
  set
  maps.cntyname(where=(state=&STATE));
run;
```

```
*GET LIST OF ALL FILE IN DIRECTORY;
data allfiles(keep=state county);
  set sashelp.vtable;
  if upcase(libname) = 'MYLIB' and
  upcase(substr(memname,1,4)) =
  'SSU_' and
  upcase(memname) ^= "SSU_&ST";
  state =
  input(substr(memname,5,2),2.);
  county =
  input(substr(memname,7,3),3.);
run;
```

```
*DETERMINE IF ALL FILES ARE PRESENT
IN DIRECTORY;
data _null_;
  length found $2;
  retain found ' ';
  merge allctys(in=base)
        allfiles(in=trans)
        end=last;
  by state county;
  if base and not trans then found =
  'no';
  *if last and found = 'no' then
  abort abend 123;
run;
```

```
*CREATE STRING OF ALL FILE IN
DIRECTORY;
proc sql noprint;
  select distinct "mylib.ssu_&ST" ||
  trim(put(county,z3.))
  into : filelist separated by ' '
  from maps.cntyname
  where state = &STATE;
quit;
*THIS SQL WILL CREATE THE MACRO
VARIABLE '&FILELIST';
*WHICH CONTAINS THE VALUE
'mylib.ssu_04001 mylib.ssu_04003
mylib.ssu_04005 mylib.ssu_04007
mylib.ssu_04009 mylib.ssu_04011
mylib.ssu_04012 mylib.ssu_04013
mylib.ssu_04015 mylib.ssu_04017
mylib.ssu_04019 mylib.ssu_04021
mylib.ssu_04023 mylib.ssu_04025
mylib.ssu_04027';
```

```

*PERFORM ROLLUP IF ABEND DID NOT
OCCURR;
data mylib.ssu_&STATE;
    set &FILELIST;
run;

***

*PROGRAM TO SEPARATE DATASET INTO ;
*COMPONENTS;

*GET STRING OF ALL FILE IN DIRECTORY;
libname in1 '/acsw';
libname out1 '/hm/gober001';
*MASTER FILE IS ON THE US LEVEL;
*MUST SUBSET INTO SEPARATE STATE
FILES;

*MASTER FILE IS CALLED CO2000 AND ;
*HAS VARIABLE CALLED Fipst. ;
*STATE LEVEL FILE WILL BE CALLED
ST_{STATE};

*create list of output datasets;
proc sql noprint;

```

```

    select distinct "out1.ssu_" ||
trim(fipst)
    into : filelist separated by ' '
    from in1.co2000;
quit;
%put &filelist;

*CREATE CODE TO OUTPUT RECORDS;
proc sql noprint;
    select distinct "if fipst = " ||
trim(fipst) || " then output
out1.ssu_" || trim(fipst) || " ;else"
    into : pgmcode separated by ' '
    from in1.co2000;
quit;

data &filelist;
    set in1.co2000;
    &pgmcode;;
run;

```