Paper 312-2013

Some Useful Utilities on Unix Platform

Kevin Chung, Fannie Mae, Washington DC

ABSTRACT

While using SAS® on Unix platform, one might want to quickly browse data; obtain contents; or perform a frequency on various data fields within a SAS data set. You can always write a specific one use SAS program and submit the program to get the results you need. However we are able to obtain this information in more efficient and effective manner by using the Unix shell script along with SAS codes. This paper demonstrates some useful utilities in Unix. Because of the flexibility of this Unix shelling structure, a user will both save time and increase productivity.

INTRODUCTION

All the SAS programs demonstrated in this paper are triggered by an associated Unix Korn shell script. By combining the Unix shell scripts and SAS codes, the user can execute each utility and acquire the results either to be displayed on the screen or sent out to the user via email. The following seven Korn shell scripts are used to trigger the same-named SAS programs.

- con – display the contents of a SAS data set
  con2excel – write the contents of a SAS data set to an Excel file and sent out to the user via email
- list – print the data portion of a SAS data set on the screen
  list2excel – write the data portion of a SAS data set to an Excel file and sent out to the user via email.
- freq – list the frequency table on the screen
- comp_var – compare the variables, case-insensitive, of two SAS data sets
- find_var – List all names of SAS data sets that contain a particular variable name

The SAS procedures used by each program are fundatemtal and simple. Each job is executed by the following program flow:

- ❖ User invokes a shell script and provides arguments at the command line of the shell script
- ❖ Shell script performs basic validation and creates Unix environment variables
- ❖ SAS program is triggered by shell script
- ❖ SAS reads the Unix environment variables by **%sysget** and creates macro variables
- ❖ Macro variables are placed in appropriate location to form a customized procedure
- ❖ Procedure is executed and results are displayed either on the screen or sent to user via email

These utilities are handy tools and you can save many keystrokes typing and obtain the results quickly. This paper is intended for any SAS user who is currently using SAS on Unix platform and is willing to use any tools to improve his/her productivity.

The SAS codes and Korn shell scripts discussed in this paper have been tested thoroughly using SAS 9.2 on IBM AIX 6.1 platform.

ENVIRONMENT SETUP

Assume the utilities are only used by a single user; therefore, you can save all programs and output files in your home directory. The diagram below shows the directory structure for the utilities. You can change the alias you like but remember to point to the right script.

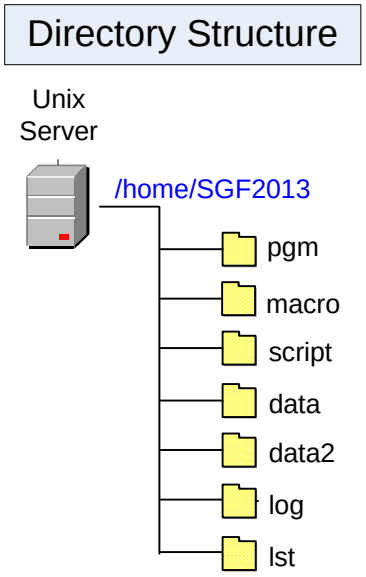


Figure 1

The aliases below must be sourced first. The part you choose is based on the Unix shell you logon. You can save these in either .cshrc for C shell or .profile for Korn shell.

```
# for ksh
export UTIL=/home/SGF2013/script
alias con='$UTIL/con'
alias con2excel='$UTIL/con2excel'
alias list='$UTIL/list'
alias list2excel='$UTIL/list2excel'
alias freq='$UTIL/freq'
alias find_var='$UTIL/find_var'
alias comp_var='$UTIL/comp_var'

# for csh
setenv UTIL /home/SGF2013/script
alias con $UTIL/con
alias con2excel $UTIL/con2excel
alias list $UTIL/list
alias list2excel $UTIL/list2excel
alias freq $UTIL/freq
alias find_var $UTIL/find_var
alias comp_var $UTIL/comp_var
```

The descriptions of each directory are listed below.

- pgm – store all sas programs
- macro – store all supporting sas macros
- script – Korn shell scripts
- data – Test data copied from sashelp contain class.sas7bdat and shoes.ss7bdat
- data2 – Test data copied from sashelp contain prdsale.sas7bdat
- log – log files
- lst – output files

The directories data and data2 are not required and they are created for demo purpose.

The file site.info below should be defined manually based on the SAS installation and directory structure you use. This file is saved under script directory and is sourced by each shell script at the beginning.

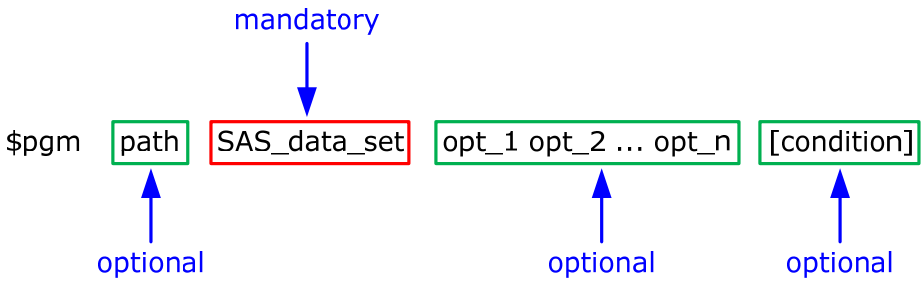
```
#
# site.info
#

export dir=/home/SGF2013          ← change this
export pgm_dir=$dir/pgm
export log=$dir/log/$pgm.log
export lst=$dir/lst/$pgm.lst
export mac=$dir/macro
export FPATH=$dir/script
export SASEXE=[fullpath]/sas      ← depends on your site
export SASTEMP=/sastemp           ← change this
export RECIPIENT=kevin_chung@fanniemae.com    ← email user

opt="-nonews -noautoexec -work $SASTEMP -log $log -print $lst"
autoload chk_input_data chk_where_clause disp_usage disp_result
```

SYNTAX AND USAGE

The utilities are triggered based on the syntax below. A Korn shell script is invoked and then followed by a SAS data set, with or without the full or relative path, as the first argument. Then one or more optional arguments can be placed after the SAS data set. Three scripts, **list**, **list2excel**, and **freq** also support the WHERE condition. The [and] are required when condition is applied.



For example,

I want to see how many branches got the shoes sales amount no less than $100,000 in each region. I can run the freq with the arguments as follows:

```
freq shoes region [sales ge 100000]
      OR
freq /home/SGF2013/data/shoes region [sales ge 100000]
```

Five scripts, **con**, **con2excel**, **list**, **list2excel**, and **freq** only accept command line arguments. The scripts **find_var** and **comp_var** are interactive. After invoked, user has to follow the instructions on the screen and enter the appropriate data to run the script.
The script **con** and **con2excel** work very similar. The only difference is **con** displays results on the screen and **con2excel** creates an XML-based file using ExcelXp tagset and send out to user via email. Same manner applied to **list** and **list2excel**.

If the script itself is submitted without any argument, several lines of text messages are displayed on the screen. The purpose is to help user understand the syntax and usage of the script. For example, if only con is executed, the following information displayed on the screen:

```
Usage:
  con <<SAS data set name>>

Example:
  con class
       OR
  con class.sas7bdat
```

These information are maintained in a file called usage.txt with the format below. This allows users to define their own help messages.

```
# con          ← must match the shell script name
Usage:
  con <<SAS data set name>>


Example:
  con class

       OR

  con class.sas7bdat
# EOF-con      ← use EOF- before shell script name
```

EXAMPLES

con
con2excel

con – display the contents of a SAS data set on the screen
con2excel – Write the contents of a SAS data set to an XML-based file using ExcelXp tagset and sent out to the user via SAS filename with email method.

How do you quickly view the descriptor portion or contents of a SAS data set on a Unix server? The first option is to write a small program and run it by either interactive or batch mode. The second option is to run SAS and invoke a windowing environment on Unix. In this case, you might have to run a third-party software that supports the X Windows first and then invoke the SAS interactive session. Once you are in the SAS interactive session, you always can use SAS Explorer to view the contents of a SAS data set.

The Korn shell script **con** and **con2excel** perform the validation of the input data as follows:
Check the existence of the directory that contains the SAS data set
Check the existence of the input SAS data set

Since con displays the contents on the screen, it can work with Unix pipe command to find a specific variable. For example, you can search the variable height in class SAS data set.

```
con class | grep -ni height
```

This is similar to the **find_var** script which is discussed later in this section. The find_var script displays all SAS data set names based on a specified variable entered by user.

An example of the use of the con is as follow:

```
con class
```

```
Location: /home/SGF2013/data

Data Set Name       LIB.CLASS                          Observations          19
Member Type         DATA                               Variables             5
Engine              V9                                 Indexes               0
Created             Wed, Nov 12, 2008 10:34:38 PM      Observation Length    40
Last Modified       Wed, Nov 12, 2008 10:34:38 PM      Deleted Observations  0
Protection                                             Compressed            NO
Data Set Type                                          Sorted                NO

#    Variable    Type    Len

1    Name        Char       8
2    Sex         Char       1
3    Age         Num        8
4    Height      Num        8
5    Weight      Num        8
```

If you have several hundreds of variables in a SAS data set, it might not be easy to view the output on the screen. The utility con2excel is the one you can use. For example, you can submit

```
con2excel  shoes
```

The result is sent via email. Open the email and double click on the attachment, you can see the output as Figure 1 on the right.

The result on the right is created based on the output of proc contents with noprint option.

	A	B	C	D
1	NAME	TYPE	LENGTH	FORMAT
2	Region	Char	25	
3	Product	Char	14	
4	Subsidiary	Char	12	
5	Stores	Numeric	8	
6	Sales	Numeric	8	DOLLAR12.
7	Inventory	Numeric	8	DOLLAR12.
8	Returns	Numeric	8	DOLLAR12.

Figure 2

list
list2excel

list – display the data portion of a SAS data set on the screen
list2excel – Write the data portion of a SAS data set to an XML-based file using ExcelXp tagset and sent out to the user via SAS filename with email method.

Sometimes you need to do a quick view about the data of a SAS data set you just created, you can use **list** to view the data of a given SAS data set. This utility supports the following options:

list prdsale ➔ display the first 100 rows with all variables
list prdsale.sas7bdat ➔ display the first 100 rows with all variables
list prdsale 10 ➔ display the first 10 rows with all variables
list prdsale -20 ➔ display the last 20 rows with all variables
list prdsale ACTUAL COUNTRY YEAR MONTH ➔ display the first 100 rows with four variables
list prdsale 15 _numeric_ ➔ display the first 15 rows with NUMERIC fields
list prdsale 30 pr: ➔ display the first 30 rows with variable name starting with PR
list prdsale -20 ACTUAL _char_ ➔ display the last 20 rows with variable ACTUAL plus all CHARACTER fields

The **list** script also supports the where statement as follow:

```
list shoes region product Subsidiary sales [sales ge 700000]

Location: /home/SGF2013/data
Data: shoes
sales ge 700000

Obs    Region         Product          Subsidiary          Sales

102    Canada         Men's Dress      Vancouver         $757,798
104    Canada         Slipper          Vancouver         $700,513
107    Canada         Women's Dress    Vancouver         $756,347
```

If the data set contains too many fields to view on the screen, you can run **list2excel** to create the output to an Excel file and send via email.

freq

freq – display the one-way to n-way frequency tables in list format

Sometimes you need to take a quick look at the frequency of a SAS data set, the **freq** script is the one for you.
The freq script supports the where condition. For example, I want to see the **freq**uency on SAS data set **class** for the variable **sex** where **age=12**, then you can run the freq script with the arguments below.

```
freq class sex [age=12]
```

You can see the output on the right displaying on the screen.

```
Data: class
age=12

                                        Cumulative    Cumulative
Sex    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------
F            2       40.00             2        40.00
M            3       60.00             5       100.00
```

If you need to specify the character type condition between [and], remember to put backslash, ****, immediately before the single or double quote. For example, the condition sex='M' is perfect for SAS syntax, but it can not be interpreted correctly by Unix.

```
freq class age [sex='M']
```

The correct syntax should be

```
freq class age [sex=\'M\']        ← backslash \ must be used
```

comp_var

comp_var – compare all variables in two SAS data sets and display the output in three parts:

1. variables in both data sets
2. variables in first data set only
3. variables in second data set only

You might modify a complicated data step to create a SAS data set. New fields might be added to the data set and some fields might be dropped from the data set. After the job is done, you want to compare the variables in both data sets to make sure the modification was made correctly. You can use **comp_var** to achieve this. This script is run interactively only.

In Unix prompt, invoke comp_var and hit enter. The following message displayed on the screen.

```
1st data set info:
       Name: class          ← 1st SAS data set name
  Full path: .              ← Enter full path or . for current directory

2nd data set info:
       Name: class2         ← 2nd SAS data set name
  Full path:                ← Enter full path or press enter to use the same
                              directory as 1st data set

Select
  1. Disply output on screen
  2. Send email with Excel file attachment
Enter choice (1 or 2): 1    ← output destination



1st data set
  Name: class
  Path: /home/SGF2013/data

2nd data set
  Name: class2
  Path: /home/SGF2013/data

Disply output on screen


Are the info above correct(Y/N): y      ← Enter y to confirm
                                          case-insensitive


1st - /home/SGF2013/data/class  NOBS: 19
2nd - /home/SGF2013/data/class2  NOBS: 19

    3 Variables             2 Variables             2 Variables
  In Both Data Set         In class Only          In class2 Only
====================    ====================    ====================

        AGE                     Height                  ht
        NAME                    Weight                  wt
        SEX
```

NOTE:
Two test data sets, class and class2, are created under /home/SGF2013/data directory.

find_var

find_var – display all SAS data set names that contain a variable that entered by user

Suppose you need to use a variable to do some query, but you don't know which data sets contain this variable? Or you might already know a variable is created in several data sets but you want to know which data set first creates this variable? By using the **find_var** script, you can easily identify the data set you need. You can invoke this script by two ways: command line argument or interactive.

By command line –

```
find_var name /home/SGF2013/data
```

By interactive –

Invoke **find_var**

The following message displayed on the screen. Enter data at the right of the prompt.

```
Input variable name to be searched:
  Variable Name: name
```

← This is the first argument in the command line

```
Input directory with full path:
  Directory Name: /home/SGF2013/data
```

← this is the second argument in the command line

Both ways produce the same result as follow:

Output –

```
Location: /home/SGF2013/data

Table Name                                  Date Created
-----------------------------------------------------
CLASS2                                 02/18/2013 20:41:55
CLASS                                  11/12/2008 22:34:38
```

INSTALL UTILITIES

All source codes can be downloaded from the web site provided in CONCLUSION section. Follow the instructions below to install the utilities.

1. Choose a location on Unix as the working directory
2. Copy the zip file, **312-2013.zip**, to the working directory.
3. Unzip this file. All directories are created with all source codes and data restored to each directory
4. Update **site.info** in script directory. Refer to ENVIRONMENT SETUP at page 2.
5. Depend on the Unix shell you use, modify the content of variable UTIL
6. You can save the aliases in .cshrc or .profile.

Appendix A – Korn shell script

```
#!/bin/ksh
#
# con
#

export pgm=con
. $UTIL/site.info

[[ $# != 1 ]] && disp_usage
chk_input_data $1

$SASEXE $pgm_dir/$pgm.sas $opt
disp_result Y
```

```
#!/bin/ksh
#
# list
#

export pgm=list
. $UTIL/site.info

[[ $# == 0 ]] && disp_usage
chk_input_data $1

[[ $2 = +([-0-9]) ]] && {
  export nobs=$2
  shift; shift
} || {
  export nobs=100
  shift
}

export var_list=$*
chk_where_clause
[[ "$var_list" == "" ]] && var_list="_all_"

$SASEXE $pgm_dir/$pgm.sas $opt
disp_result
```

```
#
# chk_input_data
#

chk_input_data () {
  export data=$1
  slash=`echo "$data"|awk '{print index($1,"/")}'`

  if [[ $slash == 0 ]] then
    path=$PWD
    ds=$data
  else
    path=`echo $(dirname $data)`
    [[ -d $path ]] && path=`echo $(cd $path; pwd)` || {
      echo "Directory $path does not exist..."
      exit
    }
    ds=`echo $(basename $data)`
  fi
  export path ds
  echo path=$path
  echo ds=$ds

  dot=`echo "$ds,."|awk -F, '{print index($1,$2)}'`
  (( $dot > 0 )) && export ds=`echo "$ds"|awk -F. '{print $1}'`

  if [[ ! -f $path/$ds.sas7bdat ]] then
    echo; echo
    echo "SAS data set $ds.sas7bdat does not exist"
    echo "in directory $path"
    echo; echo
    exit
  fi
}
```

```
    #
    # disp_usage
    #

    disp_usage () {
      clear
      echo; echo
      sed -n "/# $pgm/,/# EOF-$pgm/p" $FPATH/usage.txt|grep -v '#'
      echo; echo
      exit
    }
```

```
#
# disp_result
#

disp_result () {
  [[ $1 == N ]] && echo || {
    clear
    cd $dir/lst
    . $pgm.err

    echo; echo
    if [[ $syserr == 0 ]] then
      if [[ $1 == Y ]] then  # for con only
        head -9 $dir/lst/$pgm.lst
        echo
        sed -n '/ Variable /,$p' $dir/lst/$pgm.lst
      else
        cat $dir/lst/$pgm.lst
      fi
    else
      echo "$err_msg"
    fi
    echo; echo;
  }
}
```

```
#!/bin/ksh
#
# freq
#

export pgm=freq
. $UTIL/site.info

[[ $# == 0 ]] && disp_usage
chk_input_data $1

shift
export var_list=$*
chk_where_clause
[[ "$var_list" == "" ]] &&
var_list="_all_"

$SASEXE $pgm_dir/$pgm.sas $opt
disp_result
```

```
#
# chk_where_clause
#

chk_where_clause () {
  export condition=""

  left=`echo "$var_list"|awk '{print index($0,"[")}'`
  right=`echo "$var_list"|awk '{print index($0,"]")}'`

  if (( $left > 0 && $right > 0 )) then
    export condition=`echo "$var_list,$left,$right"|awk -F, '{print substr($1,$2+1,$3-
$2-1)}'`
    echo "condition=$condition"
    (( $left == 1 )) && var_list="" || var_list=`echo "$var_list,$left"|awk -F,
'{print substr($1,1,$2-1)}'`
  elif (( $left > 0 && $right == 0 )) then
    echo 'Missing "]"'
    exit
  elif (( $left == 0 && $right > 0 )) then
    echo 'Missing "["'
    exit
  fi
}
```

Appendix B – SAS program

```
/* con.sas */

option nocenter nodate nonumber
ps=max;
%let dir=%sysget(dir);
%let path=%sysget(path);
%let pgm=%sysget(pgm);
%let ds=%sysget(ds);

libname lib "&path";
filename mactools ("%sysget(mac)");
options sasautos=(mactools
sasautos);

ods noproctitle;
title "Location: &path";
proc contents data=lib.&ds varnum;
run;

%chk_err(&pgm)
```

```
/* list.sas */

option nocenter nodate nonumber nofmterr;

%let dir=%sysget(dir);
%let pgm=%sysget(pgm);
%let path=%sysget(path);
%let ds=%sysget(ds);
%let nobs=%sysget(nobs);
%let var_list=%sysget(var_list);
%let condition=%sysget(condition);

libname lib "&path";
filename mactools ("%sysget(mac)");
options sasautos=(mactools sasautos);

%let data=lib.&ds;
%let start=%sysfunc(ifc(&nobs >= 0,1,
%sysfunc(max(1,%eval(%nobs(&data)+&nobs+1)))));
%let nobs=%sysfunc(ifc(&nobs >= 0,&nobs,max));

title  "Location: &path";
title2 "Data: &ds";
title3 %sysfunc(ifc(%bquote(&condition)
ne ,&condition,));
%print
%chk_err(&pgm)
```

```
%macro print(dest);
  proc print data=&data(firstobs=&start obs=&nobs) %sysfunc(ifc(&dest=xls,noobs,));
    %if (%bquote(&condition) ne ) %then %do;
      where &condition;
    %end;
    var &var_list;
  run;
%mend print;
```

```
/* freq.sas */

option nocenter nodate nonumber nofmterr;

%let dir=%sysget(dir);
%let pgm=%sysget(pgm);
%let path=%sysget(path);
%let ds=%sysget(ds);
%let var_list=%sysget(var_list);
%let condition=%sysget(condition);

libname lib "&path";
filename mactools ("%sysget(mac)");
options sasautos=(mactools sasautos);

ods noproctitle;
title  "Location: &path";
title2 "Data: &ds";
title3 %sysfunc(ifc("&condition" ne "",&condition,));
%freq
%chk_err(&pgm)
```

Appendix C – usage.txt

Partial listing of usage.txt

```
#
# usage.txt
#

# con
Usage:
  con <<SAS data set name>>


Example:
  con class

       OR

  con class.sas7bdat
# EOF-con

# list
Usage:
   list <<SAS data set name>>  <<n>>  <<var list>>

Example:
   list prdsale           ==> display the first 100 rows with all variables
   list prdsale.sas7bdat  ==> display the first 100 rows with all variables
   list prdsale 10        ==> display the first 10 rows with all variables
   list prdsale -20       ==> display the last 20 rows with all variables

   list prdsale ACTUAL COUNTRY YEAR MONTH
                          ==> display the first 100 rows with four variables
   list prdsale 15 _numeric_
                          ==> display the first 15 rows with NUMERIC fields
   list prdsale 30 pr:
                          ==> display the first 30 rows with variable name starting
with PR
   list prdsale -20 ACTUAL _char_
                          ==> display the last 20 rows with variable ACTUAL & all CHAR
fields
# EOF-list
```

CONCLUSION

The examples of the utilities in this paper are meant to provide users a handy tool to improve the productivity in the daily routine job. You still have to write a program for a complex query.
All the source codes and presentation materials can be downloaded from www.kevin-chung.com

REFERENCES

[1] SAS OnlineDoc® 9.2, SAS Institute Inc. Cary, NC.
http://support.sas.com/documentation/cdl/en/lrdict/64316/PDF/default/lrdict.pdf

[2] SAS 9.2 Companion for UNIX Environment
http://support.sas.com/documentation/cdl/en/hostunx/61879/PDF/default/hostunx.pdf

[3] UNIX Shells by Example, 4th edition
By Ellie Quigley, October 2004

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Feel free to contact the author at:

Kevin Chung
Fannie Mae
4000 Wisconsin Ave., NW
Mail Stop: 2H-4S/07
Washington, DC 20016
Work Phone: 202-752-1568
E-mail: kevin_chung@fanniemae.com
kchung01@hotmail.com