SAS Hints

Delete temporary files
Determine if a file exists
Direct output to different directory
Errors (specify # of errors for SAS to put into log)
Execute Unix command from SAS
Generate delimited file with no spaces between columns
Input – Group variables and informats
Noterminal option for PROC EXPORT or PROC IMPORT
Provide timing and memory usage information
Read in compressed data file using zcat
Read in space/tab delimited files
Return ID of a user in Unix environment
Return value of a specified operating environment variable
Rounding in SAS (ceil and floor functions)
Specify a character string to pass to SAS programs using sysparm
Use SAS with pipes or as a filter under Unix
Using X commands in Solaris vs. Linux

* Delete temporary files

Use proc datasets to delete temporary files created by SAS.

```
data name;
        set name;
run;
        SAS code ...
proc datasets nolist;
        delete name'
run;
```

* Determine if a file exists

Use %sysexec to spawn a shell

```
%sysexec /bin/ksh -c "if [ ! -a /etc/passwd ]; then exit 99; else exit 0; fi;";
%put &sysrc;
```

or

Use filename pipe

```
%let thefile=/etc/passwd;
filename testit pipe "if [ ! -f &thefile ]; then echo 'no'; else echo 'yes'; fi";

data _null_;
        infile testit pad missover lrecl=3;
        input answer $3.;
        put answer=;
run;
```

* Direct output to different directory

Direct SAS output to a directory (other than the one where SAS is being run)

sas –work /different_directory program-name

* Errors (specify # of errors for SAS to put into log)

Specify the number of errors for SAS to list in the log file in the options line (useful to obtain list of errors, i.e., missing values for markers)

options ls=65 ps=55 pageno=1 errors=40

* Execute Unix command from SAS

Single Unix commands can be executed by

```
X command;
        or
call system ('command');      (call system can only be used inside a data step)
```

Multiple Unix commands can be executed by

```
X 'command 1; command 2; … command n';
        or
call system ('command1; command2; … command n');
```

%sysexec macro can also be used (macro test will execute Unix commands 'pwd' and 'ls –l'):

```
%macro test
        %sysexec %str(pwd; ls –l);
%mend test;
%test;
```

X; (no Unix commands) starts a shell. The user can run programs, check output, etc., then type 'exit' to return to SAS.

* Generate delimited file with no spaces between columns

```
data _NULL_;
        set sasdataset;
        file "output.txt";      (or "output.csv")
        newvar=compress(var1||DELIM||var2);
        put newvar;
run;
        DELIM is comma , or tab '09'x
```

* Input – Group variables and informats

Informat lists can be grouped when input values are arranged in a pattern. A group informat list consists of 2 lists – the names of the variables to read enclosed in parentheses and the corresponding informats separated by either blanks or commas and enclosed in parentheses.

If values for the 5 variables SCORE1 through SCORE5 are stored as four columns per value without intervening blanks the input statement is

```
input (score1-score5) (4.);
```

The +1 column pointer moves the pointer forward one column after X is read. X is read with the 2. informat and the pointer moves to Y, which is read with the 2. informat. The pointer then moves to Z, which is read with the 2. informat.

```
data test;
    input (x y z) (2.,+1);
    datalines;
    2 24 36
    0 20 30;
```

The n* modifier can be used to specify the number of times to repeat the next informat.

```
input (name score1-score5) ($10. 5*4.);        score will be read in 5 times
```

* Noterminal option for PROC EXPORT or PROC IMPORT

The message "Error: Cannot open X display" may be received when using PROC IMPORT or PROC EXPORT in batch mode on Unix systems without a terminal present. This can be avoided by using the –noterminal option.

sas program-name -noterminal

* Provide timing and memory usage information

fullstimer collects performance statistcs on each SAS job step and for the job as a whole and puts them in the log file. Note: measures are a snapshot view of performance at step and job level; each SAS port yields different fullstimer statistics based on the host operating system.

sas –fullstimer program-name

Sample result of SAS data step

NOTE: DATA statement used:

real time	0.06 seconds
user cpu time	0.02 seconds
system cpu time	0.00 seconds
Memory	88k
Page Faults	10
Page Reclaims	0
Page Swaps	0
Voluntary Context Switches	22
Involuntary Context Switches	0
Block Input Operations	10
Block Output Operations	12

* Read in compressed data file using zcat

```
filename thedata pipe 'zcat datafile.Z';
        OR filename thedata pipe ‘gunzip –c datafile.gz’ for gzipped files
data _null_;
        infile thedata pad lrecl=40;
        input theline $80.;
        put theline=;
        run;
run;
```

* Read in space/tab delimited files

```
data name;
        infile use dsd delimiter=" " firstobs=2 truncover;
        input heading1 :$10 heading2 $ heading 3 ……;
run;
```

dlm can be used in place of delimiter
Tab delimited files: use dlm=’09’x
Comma delimited files: use dlm=”,”
firstobs=2 tells SAS to start reading at line 2 (skips header line)
:$10 – colon tells SAS to read up to the number of characters specified or to the next delimiter, whichever comes first

* Return the ID of a user in Unix environment

```
%let person=%sysget(USER);
%put User is &person;
```

SAS log: User is uniqname

* Return the value of a specified operating environment variable

```
%let homedir=%sysget(HOME);       (HOME must be in capital letters)
data _null_;
        put "The value of my HOME environment variable is: &homedir";
run;
```

SAS log: The value of my HOME environment variable is: /afs/sph.umich.edu/user/uniqname

* Rounding in SAS

CEIL (ceiling) function (rounds up)

Returns the smallest integer greater than or equal to the argument

```
data a;
x=2.102;
y=ceil(x);
put x= y=;
run;
```

Results in: x=2.102 y=3

Floor function (rounds down)

Returns the smallest integer less than or equal to the argument

```
data a;
x=2.102;
y=floor(x);
put x= y=;
run;
```

Results in: x=2.102 y=2

* Specify a character string to pass to SAS programs using sysparm

sas program-name -sysparm dept.projects (sysparm supplies name of data set for proc report)

```
proc report data=&sysparm report=test.resorces.priority.rept;
title "%sysfunc(date(),worddate.)";
title2 'Active Projects By Priority';
run;
```

SAS sees:

```
proc report data=dept.projects report=test.resorces.priority.rept;
title “Today’s Date";
title2 'Active Projects By Priority';
run;
```

or

```
sas height.sas –sysparm cm
data height;
    set height;
    if sysparm() = ‘cm’ then do;
        height=height*2.54;
        unit=‘centimeters’;
    end;
run;
```

SAS output (data height contains name, height, and inches as unit)

NAME	HEIGHT	UNIT
John	175.26	centimeters
Sally	162.56	centimeters
Peter	190.50	centimeters

* Use SAS with pipes or as a filter under Unix (writing stdout to stdin without using an intermediate file. See http://support.sas.com/sassamples/quicktips/pipes_0702.html.

* Using X commands in Solaris vs. Linux

Solaris – run X program, redirect STDOUT to file, and read file into SAS dataset:

```
%let xcomm="getrand -n &numobs > rand.dat";
        X &xcomm;

data new;
        infile "rand.dat";
        input ranu;
run;
```

Linux – this will only work with the pipe option

```
%let xcomm="getrand -n &numobs";

filename fromunix pipe &xcomm;

data new;
        infile fromunix;
        input rannum;
run;
```