

UNIX X Command Tips and Tricks

David B. Horvath, CCP, MS
PhilaSUG Winter 2019
TD Bank, Wilmington DE
March 19, 2019

UNIX X Command Tips and Tricks

Copyright © 2019 David B. Horvath, CCP — All Rights Reserved

The Author can be contacted at:

504 Longbotham Drive, Aston PA 19014-2502, USA

Phone: 1-610-859-8826

Email: dhorvath@cobs.com

Web: <http://www.cobs.com/>

All trademarks and servicemarks are the property of their respective owners.

Abstract

SAS provides the ability to execute operating system level commands from within your SAS code – generically known as the “X Command”. This session explores the various commands, the advantages and disadvantages of each, and their alternatives. The focus is on UNIX/Linux but much of the same applies to Windows as well. Under SAS EG, any issued commands execute on the SAS engine, not necessarily on the PC.

- X
- %sysexec
- Call system
- Systask command
- Filename pipe
- &SYSRC
- Waitfor

Alternatives will also be addressed – how to handle when NOXCMD is the default for your installation, saving results, and error checking.

My Background

- David is an IT Professional who has worked with various platforms since the 1980's with a variety of development and analysis tools.
- He has presented at PhilaSUG, SESUG, and SGF previously and has presented workshops and seminars in Australia, France, the US, Canada, and Oxford England (about the British Author Nevil Shute) for various organizations.
- He holds an undergraduate degree in Computer and Information Sciences from Temple University and a Masters in Organizational Dynamics from UPENN. He achieved the Certified Computing Professional designation with honors.
- Most of his career has been in consulting (although recently he has been in-house) in the Philadelphia PA area. He is currently in Data Analytics "Engineering" at a Regional Bank.
- He has several books to his credit (none SAS related) and is an Adjunct Instructor covering IT topics for University of Phoenix.

X

- The basic X Command
 - Runs independent of data steps and macros
 - The SAS Engine will interpret some commands
 - Does not spawn off sub-process
 - This fact is important because information does not persist between sub-processes
 - Handling within the log is annoying

X

- X Command Examples:

- pwd and cd under UNIX:

NOTE: Current working directory is
'/this/is/the/sas/install/directory'.

```
26          x "pwd"  ; /* works within SAS */
27          x "cd /my/directory/is/here"
27          !                ; /* works within SAS */
```

NOTE: Current working directory is '/my/directory/is/here'.

```
28          x "pwd"  ; /* works within SAS */
```

- echo and combined commands under UNIX:

```
29          x "echo $HOME"
29          !                ; /* works but no output */
30          x "pwd; cd $HOME; pwd"
30          !                ; /* no output, works? */
```

X

- X Command Examples:

- We can combine commands on one line under UNIX:

```
31          x 'echo start; echo mid; echo end>temp2.txt'  
31          !                               ; /* output to file, works */
```

- > sends STDOUT to a file
- I know this works because I can look at the output file (temp2.txt):

end

- Why didn't "start" and "mid" appear?
- Because of the way I wrote the statement!

X

- X Command Examples:

- We can combine commands on one line under UNIX:

```
37      x '(echo start; echo mid; echo end)>temp1.txt'  
37      ! /* output to file, all 3 statements to file */
```

- I know this works because I can look at the output file (temp1.txt):

```
start  
mid  
end
```

- The difference is the parenthesis which combines the output in UNIX
- Because of the way I wrote the statement!

X and %SYSRC

- How do I know a command worked?

- %SYSRC will tell me – returns the UNIX error code

```
37          x '(echo start; echo mid; echo end)>templ.txt'  
37          ! ; /* output to file, all 3 statements to file */  
SYMBOLGEN: Macro variable SYSRC resolves to 0  
38          %put &SYSRC;  
0
```

- Zero is success in UNIX

- If the command does not exist, we get 127:

```
39          x 'this_command_doesnot_exist'  
39          ! ; /* non-zero RC */  
SYMBOLGEN: Macro variable SYSRC resolves to 127  
40          %put &SYSRC;  
127
```

X and %SYSRC

- How do I know a command worked?

- If the command does not exist, we get 127:

```
41          x 'this_command_doesnot_exist 2>test5.txt'
41          !                               ; /* non-zero RC */
SYMBOLGEN: Macro variable SYSRC resolves to 127
42          %put &SYSRC;
127
```

- And we can save error output (2> sends STDERR to a file):

```
/bin/bash: this_command_doesnot_exist: command not found
```

```
43          x 'ls -al no_such_file'
43          !                               ; /* non-zero RC */
SYMBOLGEN: Macro variable SYSRC resolves to 2
44          %put &SYSRC;
2
```

X and %SYSRC

- How do I know a command worked?

- The command itself may return an error as well:

```
43          x 'ls -al no_such_file'
43          !                               ; /* non-zero RC */
SYMBOLGEN:  Macro variable SYSRC resolves to 2
44          %put &SYSRC;
2
```

- In this case, "no_such_file" does not exist

- 'man ls' executed at the command line will provide this information

Exit status:

```
0          if OK,
1          if minor problems (e.g., cannot access subdirectory),
2          if serious trouble (e.g., cannot access command-line argument).
```

%sysexec – Macros

- We can execute system commands within a macro:

```
72      %macro commands;  
73          %sysexec %str(ls -al > test6.txt);  
74          %put &SYSRC;  
75          %sysexec %str(command_doesnot_exist 2>  
test7.txt);  
76          %put &SYSRC;  
77      %mend commands;  
78  
79      %commands;
```

- Note the use of %SYSRC

%sysexec – Macros

- And get the following results:

```
79          %commands;
MLOGIC(COMMANDS):  Beginning execution.
MLOGIC(COMMANDS):  %SYSEXEC  ls al  test6.txt
MLOGIC(COMMANDS):  %PUT &SYSRC
SYMBOLGEN:  Macro variable SYSRC resolves to 0
0
MLOGIC(COMMANDS):  %SYSEXEC  command_doesnot_exist 2
test7.txt
MLOGIC(COMMANDS):  %PUT &SYSRC
SYMBOLGEN:  Macro variable SYSRC resolves to 127
127
MLOGIC(COMMANDS):  Ending execution.
```

Call System – Data Step

- Use Call System within a data step:

```
85         data dir;
86             filename commands PIPE "ls | head -2";
87             infile commands trunccover;
88             input result $char60.;
89
90             string="echo " || result || " >> test4.txt";
91             call system(string); /* no output - but it executes
multiple times */
92             system_rc=symget("SYSRC");
93             call system("this_command_doesnot_exist");
94             system_rc2=symget("SYSRC");
95
96             systask command "pwd" wait shell; /* runs once */
NOTE: LOG/Output from task "task59"
> /my/directory/is/here
NOTE: End of LOG/Output from task "task59"
97             output;
98         run;
```

Call System – Data Step

- Call System results:

NOTE: The infile COMMANDS is:

 Pipe command="ls | head -2"

NOTE: 2 records were read from the infile COMMANDS.

 The minimum record length was 22.

 The maximum record length was 24.

NOTE: The data set WORK.DIR has 2 observations and 4 variables.

NOTE: Compressing data set WORK.DIR increased size by 100.00 percent.

 Compressed is 2 pages; un-compressed would require 1 pages.

NOTE: DATA statement used (Total process time):

 real time 0.02 seconds

 cpu time 0.00 seconds

Call System – Data Step

- Call System output:

- Results:

Obs	result	string	system_rc	system_rc2
1	FILE1	echo FILE1 >> test4.txt	0	127
2	FILE2	echo FILE2 >> test4.txt	0	127

- Test4.txt:

```
FILE1  
FILE2
```

- Two records were read from infile
- Two records were written to work.dir
- Two records were written to test4.txt via the echo command
- SAS Engine does not interpret these commands

'Systask command'

- 'Systask command' operates two modes:
 - With "shell" modifier, SAS does not interpret the commands
 - Without "shell", it behaves like X
 - &SYSRC is set

```
54          systask command 'echo BOL $HOME EOL' wait;
```

```
NOTE: LOG/Output from task "task62"
```

```
> BOL $HOME EOL
```

```
NOTE: End of LOG/Output from task "task62"
```

```
54          !          /* $HOME not interpreted */
```

```
55          systask command "echo BOL $HOME EOL" wait shell;
```

```
NOTE: LOG/Output from task "task63"
```

```
> BOL /my/directory/is/here EOL
```

```
NOTE: End of LOG/Output from task "task63"
```

```
55          !          /* $HOME interpreted */
```

'Systask command' and 'Systask list'

- Other Options:
 - Wait: wait for this command to execute before starting next
 - Cleanup: wait for this command and any nowait before starting next
 - Shell: Can also specify the shell to use: shell="/usr/bin/ksh"
 - Status: Can specify a status variable to check later (rather than &SYSRC)
 - Taskname: Can specify task name for later use in Waitfor
- Systask list will provide status of any nowait systasks

```
"task150" -----  
    Type: Task  
    State: COMPLETE  
    Status Macro Variable: Unspecified
```

73

74 systask list;

'Systask command' and 'Waitfor'

- The waitfor command is used to wait for systask command nowait to complete
 - Can wait for `_ANY_` of listed tasknames to complete (default)
 - Can wait for `_ALL_` of listed tasknames to complete
 - Can specify length of time to wait: Timeout=number-of-seconds
- General form is

```
waitfor _ALL_ task1 task2 task3;
```

Filename pipe

- Acts like normal filename statement
 - Accepts data written from SAS (File/Put)
 - Provides data for SAS to read (Infile/Input)
- Allows for execution of UNIX command
- Is more efficient than running command separately (parallelization)
- Handy when you have version limitations:

```
filename gzipit zip '/my/output/directory/file.txt.gz' gzip;  
* requires 9.4M5;
```

```
filename gzipit pipe 'gzip -c >  
< /my/output/directory/file.txt.gz'; * run the UNIX gzip;
```

Filename Pipe – Earlier Code Example

- UNIX Commands are Executed prior to input statement:

```
85         data dir;
86             filename commands PIPE "ls | head -2";
87             infile commands trunccover;
88             input result $char60.;
89
90             string="echo " || result || " >> test4.txt";
91             call system(string); /* no output - but it executes
multiple times */
92             system_rc=symget("SYSRC");
93             call system("this_command_doesnot_exist");
94             system_rc2=symget("SYSRC");
95
96             systask command "pwd" wait shell; /* runs once */
NOTE: LOG/Output from task "task59"
> /my/directory/is/here
NOTE: End of LOG/Output from task "task59"
97             output;
98             run;
```

Filename Pipe

- Unfortunately, &SYSRC is not set by Filename Pipe:

```
113      data baddir;
114          filename commands PIPE "lx ; lx ; lx";
115          system_rc=symget("SYSRC");
116
117          infile commands trunccover;
118          input result $char60.;
119          output;
120      run;
```

NOTE: The infile COMMANDS is:
Pipe command="lx ; lx ; lx"

Obs	system_rc	result
1	0	/bin/bash: lx: command not found
2	0	/bin/bash: lx: command not found
3	0	/bin/bash: lx: command not found

NOTE: 3 records were read from the infile COMMANDS.
The minimum record length was 32.
The maximum record length was 32.

Shell Scripts

- When NOXCMD is set, none of these will work.
- You can move the commands into a shell script:

```
#!/bin/ksh
cd /desired/directory
# You can check return codes here with the $?
if [[ $? -gt 0 ]]; then
    echo "cd failed"
    exit 3
fi
# if we get here 'cd' succeeded
ls -al | head -2 > temp.txt
sas your_sas_part_1.sas
pwd
sas your_sas_part_2.sas
gzip /my/output/directory/file.txt
```


Final Thoughts

- It is all about choices
 - Sometimes it is better to execute UNIX commands in your program
 - Sometimes not

USER FRIENDLY by Illiad



?!
!
!
Questions
?
?
?
Answers
?!
?!
?!
?!
?!

Named Pipes under UNIX – Filename Pipe with NOXCMD

- I can use UNIX “Named Pipes” with files
- Datasets make use of the "Sequential Data Engine" ("TAPE" engine)
- External Command Example – Write:

- UNIX/Linux commands:

```
mknod mypipe p
gzip -c mypipe > input.gz & /* runs in background */
sas writepipe.sas
```

- writepipe.sas Program:

```
libname test "mypipe";

data test.test_no (compress=no drop=text1-text44) ;
  array text[44] $20 (/* list of 44 words or phrases */);

  format longstring $200. ;

  DO indexvariable=1 TO 20000000;
    /* create a bunch of random values */
    output test.test_no;
  END;
run;
```

Named Pipes under UNIX – Filename Pipe with NOXCMD

- External Command Example – Read:

- UNIX/Linux commands:

```
mknod mypipe p /* not needed if created before)
gzip --stdout input.gz > mypipe & /* runs in background/parallel */
sas readpipe.sas
```

- readpipe.sas Program:

```
libname test "mypipe";

data _null_;
set test.test_no;
    retain total 0;
    total=total+num1;
run;
```

UNIX X Command Tips and Tricks

The Author can be contacted at:

David B. Horvath, CCP
504 Longbotham Drive, Aston PA 19014-2502, USA
Phone: 1-610-859-8826
Email: dhorvath@cobs.com
Web: <http://www.cobs.com/>
LI: <http://www.linkedin.com/in/dbhorvath>

References 1

- x, sysexc, system
 - <https://v8doc.sas.com/sashtml/unixc/xcomm.htm>
 - <http://support.sas.com/documentation/cdl/en/hostunx/63053/HTML/default/viewer.htm#p0w085btd5r0a4n1km4bcdpgqibt.htm>
- %sysexec
 - <http://support.sas.com/documentation/cdl/en/mcrolref/62978/HTML/default/viewer.htm#n08ecabbpebv2xn13ieu8uylroh.htm>
- filename pipe
 - <http://support.sas.com/documentation/cdl/en/hostunx/63053/HTML/default/viewer.htm#n1ceb0xedanuj3n19l3g73awk1wf.htm>
 - <https://documentation.sas.com/?docsetId=hostunx&docsetTarget=n1ceb0xedanuj3n19l3g73awk1wf.htm&docsetVersion=9.4&locale=en>

References 2

- bash scripts
 - <https://www.taniarascia.com/how-to-create-and-use-bash-scripts/>
- systask:
 - <https://v8doc.sas.com/sashtml/unixc/z1215125.htm>
- xsync
 - <https://v8doc.sas.com/sashtml/win/zp-xsync.htm>
- xcmd
 - <http://support.sas.com/documentation/cdl/en/hostwin/69955/HTML/default/viewer.htm#p0xtd57b40ehdfn1jyk8yxemfrtv.htm>

References 3

- x command (x windows) options
 - <https://v8doc.sas.com/sashtml/unixc/xcom.htm>
- waitfor
 - <https://v8doc.sas.com/sashtml/unixc/z1215125.htm>

Wrap Up (for Real)

?! ! ! ?!

 ? ?

 and

 ?

?! ?! ! ! ?!