Paper 034-30

A Better SYSIN Than SYSIN: Instream Files on Any Platform

Ted Conway, Ted Conway Consulting, Inc., Chicago, IL

ABSTRACT

Surprising as it may seem, those coming from the mainframe world may find themselves sorely missing the instream file features provided by the **SYSIN** JCL statement as they move applications to Windows and Unix. Without these features, one is often forced to create many small external files to support an application, making it impossible to see at-a-glance what the processing is doing. This paper presents a simple macro that not only provides **SYSIN**-like functionality on the Windows and Unix platforms, but also goes beyond some of **SYSIN**'s limitations, allowing one to effect macro-like substitutions on instream data for use by SAS or non-SAS applications. It may be of interest to anyone who uses Base SAS on the PC, Unix, or mainframe platforms.

INTRODUCTION

Gather round kiddies, Grandpa Ted's got a story to tell you.

Once upon a time, SAS programmers used to work on IBM mainframes, where a JCL feature called the **SYSIN** statement could be used to provide all of the data and code required by an application in a single file, making it very easy to see at-a-glance what the processing was doing.

But as processing was moved off of the mainframes to Windows and Unix, **SYSIN** was no longer available, and the data and code tended to find its way into many small, but separate, external files.

No longer able to see the forest for the trees, what was Grandpa Ted to do?

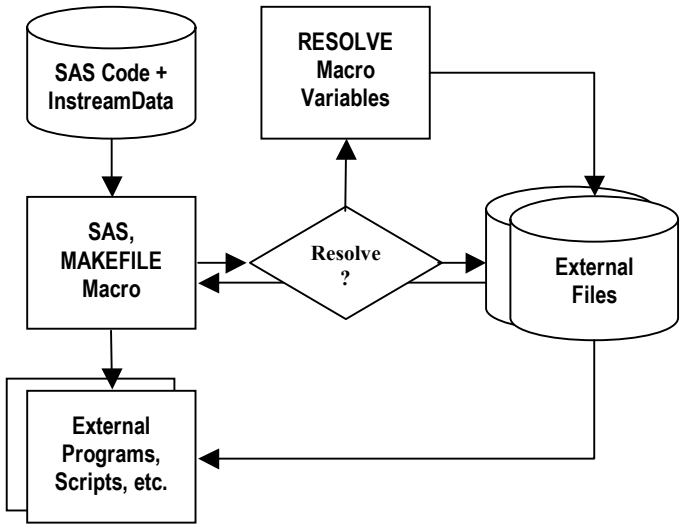
He first considered the platform-independent **CARDS** statement (**DATALINES** for the under-50 set!), but that really only addressed SAS input. Furthermore, each SAS step would still only be able to process input from one **CARDS** statement, and the data associated with a **CARDS** statement could only be used by one SAS step. In addition, this instream data would only be available to SAS processing.

A second possibility he considered was the Unix **here document**, but since this approach was platform-specific and forced one to drop SAS code and data down to the shell script level and employ user-defined functions, this Unix-tail-wagging-the-SAS-dog approach was also deemed less than desirable.

It was starting to look like Grandpa Ted might not be able to recapture his beloved **SYSIN** functionality.

Or would he?

MAKEFILE MACRO – OVERVIEW



1. Prepare a single file with your ***SAS Code*** and any ***Instream Data*** required by the SAS code or any other ***External Applications*** (e.g., FTP, DB2, Unix Shell) that may be invoked.
2. When ***SAS*** is run, the ***MAKEFILE Macro*** creates ***External Files*** from the ***Instream Data***, optionally resolving any references to SAS macro variables by using the ***RESOLVE*** function.
3. The generated ***External Files*** are used by your ***SAS Code*** and/or any ***External Programs, Scripts, etc.*** invoked by SAS.

MAKEFILE MACRO – VERSION 1.0

The first step in providing **SYSIN**-like functionality, Grandpa Ted realized, was to make it easy for both the SAS and non-SAS data to coexist with the SAS code.

So what was the best way to go about this?

A reusable macro might do the trick, he thought, and so the first incarnation of the **MakeFile** macro was born:

```
%macro MakeFile(filename);
options nocardimage;
data _null_;
input;
file "&filename" lrecl=255;
put _infile_;
%mend;
```

MAKEFILE MACRO – VERSION 1.0 – EXAMPLES

And using the MAKEFILE macro to create a file was quite simple–just invoke the macro and follow it with a **CARDS** statement and your instream data:

```
%MakeFile(c:\students.txt);
cards;
123456787,Student One
123456788,Student Two
123456789,Student Three
;
%MakeFile(c:\grades.txt);
cards;
123456787,English,B
123456787,Computer Science,B
123456788,Political Science,C
123456789,Economic,A
;
Data students;
Infile "c:\students.txt" dsd dlm=",";
Input ssn student : $20.;

Data grades;
Infile "c:\grades.txt" dlm=",";
Input ssn subject : $20. grade : $1.;

proc sql;
select s.ssn format=z9., s.student, g.subject, g.grade from students s, grades g
where s.ssn=g.ssn order by ssn;

Output:

      ssn  student               subject               grade

123456787  Student One           English               B
123456787  Student One           Computer Science      B
123456788  Student Two           Political Science     C
123456789  Student Three         Economic              A
```

And Grandpa Ted really liked that the MAKEFILE macro could also be used to allow non-SAS processing, like FTP, to co-exist with his SAS code:

```
%MakeFile(c:\testftp.bat);
cards;
ftp -nv -s:c:\testftp.txt members.aol.com
;
%MakeFile(c:\testftp.txt);
cards;
user anonymous pass tedconway@aol.com
cd tedconway
pwd
dir
quit
;
x c:\testftp.bat;
```

MAKEFILE MACRO – VERSION 2.0

As much as he liked the first version of the **MakeFile** macro, Grandpa Ted realized it could even be more useful if it supported macro-like substitutions (which wasn't even supported by his beloved **SYSIN**!).

And so the second incarnation of the **MakeFile** macro was born, with an optional parameter to allow macro-like substitutions via the **RESOLVE** function:

```
%macro MakeFile(filename,resolve=n);
options nocardimage;
data _null_;
input;
file "&filename" lrecl=255;
%if "&resolve"="y" %then
  _infile_=resolve(_infile_);;
put _infile_;
%mend;
```

MAKEFILE MACRO – VERSION 2.0 – EXAMPLE

And using the revised **MAKEFILE** macro to create a file was also quite simple–Grandpa Ted just had to set any macro variables before issuing the **MAKEFILE** macro, and specify macro variables as needed in the instream data:

```
%let site=ftp.sas.com;
%let dir=pub;

%MakeFile(c:\testftp.bat,resolve=y);
cards;
ftp -nv -s:c:\testftp.txt &site
;
%MakeFile(c:\testftp.txt,resolve=y);
cards;
user anonymous pass tedconway@aol.com
cd &dir
pwd
dir
quit
;
x c:\testftp.bat;
```

MAKEFILE MACRO – VERSION 3.0

Knowing that he would sometimes like to use the same file over and over again while just changing a parameter or two, Grandpa Ted came up with yet a third incarnation of the **MakeFile** macro, this time allowing macro-like substitutions to be made to data in an instream **or** external file:

```
%macro MakeFile(in=cards,out=,resolve=n);
options nocardimage;
data _null_;
%if  "&in"^="cards" %then
  Infile "&in" lrecl=255;;
Input;
file "&out" lrecl=255;
%if "&resolve"="y" %then
  _infile_=resolve(_infile_);;
put _infile_;
%mend;
```

MAKEFILE MACRO – VERSION 3.0 – EXAMPLE

Using the final version of the **MakeFile** macro, Grandpa Ted was able to reuse the instream files to his heart's content, in this example invoking FTP repeatedly to list the contents of the sugi24-29 directories in the public folder on ftp.sas.com:

```
%MakeFile(out=c:\testftp.bat,resolve=n);
cards;
ftp -nv -s:c:\testftp.txt ftp.sas.com
;
%MakeFile(out=c:\templateftp.txt,resolve=n);
cards;
user anonymous pass tedconway@aol.com
cd &dir
pwd
dir
quit
;
%macro chkdirs;
%do sugi=24 %to 29;
  %let dir=pub/sugi&sugi;
  %makefile(in=c:\templateftp.txt,out=c:\testftp.txt,resolve=y);
  x c:\testftp.bat;
%end;
%mend;
%chkdirs;
```

CONCLUSION

The **MAKEFILE** macro not only replaces sorely-missed **SYSIN** functionality, it can also be used to effectively construct rudimentary macro processors for use with applications other than SAS that lack macro features.

ACKNOWLEDGMENTS

While uses for the **RESOLVE** function have been discussed over the years by many on SAS-L, it was a paper by Ian Whitlock that first caught my attention and provided the inspiration for most of what's described in this paper.

REFERENCES

Ian's paper – **The RESOLVE Function** - What Is It Good For? – can be found in the NESUG '98 proceedings at http://www.nesug.org/html/Proceedings/nesug98/code/p088.pdf.

CONTACT INFORMATION

Ted Conway currently works for Ted Conway Consulting, Inc. (guess how he got that job!) in Chicago, Illinois. He can be reached at tedconway@aol.com. With his oldest child still in college, he is thankfully not yet a Grandpa!

TRADEMARKS