

Music to My Ears: Using SAS® to Deal with External Files (and My iPod)

Sean Hunt, Quality Insights of Pennsylvania, Pittsburgh, PA

ABSTRACT

Working with multiple external data files usually presents a set of challenges (and “serendipitous” opportunities). The challenges lie in dealing with files having different formats, naming conventions, and incomplete data. Using SAS® allows a virtually unlimited set of solutions to automatically and conditionally process as well as manipulate the external files and the data they contain. This is true whether you are dealing with text files, spreadsheets, and even MP3 (audio) files. This paper will demonstrate the use of SAS to view, import, and manipulate entire directory structures full of external data files, using a combination of SAS “x” statements to invoke MS-DOS commands, SAS macros, and some SAS text functions. Two real world solutions will be shared: importing a set of monthly data files that seem to change every month; and using SAS to catalog, organize, and label MP3 files (using ID3 tags) for use with an iPod® or other MP3 player.

INTRODUCTION

It is safe to assume that the actual data in the files any SAS programmer must regularly process will change every day, week, month, etc. But what happens when the file names change? Or the total number of data files? Such seemingly minor inconveniences may only serve to break up the monotony of an otherwise routine task. But, as the volume of external files increases, using SAS to analyze and manipulate the files *before* processing the data they contain can be a worthwhile endeavor.

The first step entails automatically building, organizing, and editing directory structures with potentially thousands of files and sub-folders. The next step is to combine this metadata about the files (e.g., file name and location) with information from within the files themselves (e.g., primary/foreign keys). The end result is a SAS data set that can then be used to conditionally process these directories and their accompanying data files according to the data they contain.

The same logic that can be applied to the management of more typical data files (.txt, .csv, etc.) can also be applied to audio files (.mp3 format). This allows an automated approach to cataloging, organizing, and even editing music files. While there are dozens of available software applications with similar capabilities, the power of SAS becomes quite obvious when dealing with several gigabytes of music files.

This paper is intended for beginner to intermediate SAS programmers on a Windows platform with a working knowledge of SAS macros (and lots of external data files to work with).

USING SAS AND MS-DOS TO GET ACQUAINTED WITH YOUR DATA FILES

If the data files you regularly receive always arrive exactly the way you expect them to with a constant number of files and consistent naming conventions, then this section will likely be of no use. However, if you never know how many files to expect or if you do not know how they will be named, SAS and MS-DOS work quite well together. Using the SAS “x” command or the pipe option on the SAS filename statement allows the user to invoke MS-DOS commands in the middle of a SAS program. As many papers have been written on this subject, suffice to say that both options achieve the same end result (Yu and Huang, 2002). The “x” command will be used throughout this paper simply because it requires only one line of code per MS-DOS command, as opposed to a separate filename statement and dataset with the pipe option. But before jumping right into some SAS “x” statements, a few SAS options are necessary:

```
options noxsync noxwait noxmin;
```

Typically my goal when using MS-DOS commands via SAS is for SAS to pause processing, issue some commands via MS-DOS, wait for these commands to finish executing, and then continue with the rest of the program using the output of the MS-DOS commands. The first two SAS options listed above allow just that: “noxsync” tells SAS to let MS-DOS run independently, while “noxwait” tells SAS to not to wait for MS-DOS commands to finish processing. The “noxmin” option just ensures that you will see a new MS-DOS window appear for each command. While this is useful for testing and debugging purposes, it can easily be avoided by using the opposite option (“xmin”).

With our options in place and some sample files to work with, we can then use SAS to issue MS-DOS commands such as “DIR”, “COPY”, “MOVE”, etc. Note that a convenient source for all available MS-DOS commands as well as all available options can be found in the references section of this paper.

For a quick example assume you receive a few hundred data files that you need to import into SAS, and then you dump all of these files into your "c:\sesug" directory. Let's use a quick SAS macro to create ten sample folders in the "c:\sesug" directory, named "folder1" through "folder10". We will also create three sample data files in each folder, called "sample_data1" through "sample_data3".

```
%macro sample_data;

  %do i=1 %to 10;

    x "mkdir c:\sesug\folder&i.\";

    %do j=1 %to 3;

      proc export data=sashelp.class

        outfile="c:\sesug\folder&i.\sample_data&j..csv";

      run;

    %end;

  %end;

%mend sample_data;

%sample_data;
```

Now, assuming we do not already know how many folders or files exist, we can use the "DIR" command to build a list of files for SAS to import:

```
x "dir c:\sesug\";
```

This code will open a new MS-DOS window, successfully issue any command(s) between the double quotes, and then exit back to SAS. However the information we wish to gather from the "DIR" command will be displayed ever so briefly on your screen, but then it will be lost as soon as the MS-DOS window closes. To get this directory information into SAS, we just need to use the ">" option on the "DIR" command to take the output from the "DIR" command and send it to an external file (rather than simply displaying it on the screen). This text file can then be imported into SAS as follows:

```
x "dir c:\sesug\ > c:\sesug\all_files_and_folders_list.txt";

data directory;

  infile "c:\sesug\directory_list.txt" trunccover;

  input directory $;

run;
```

One quick glance at the "directory" data set will show that we captured a good bit of extraneous information along with the file and folder names. The "/b" option on the "DIR" statement allows us to eliminate everything except for the file names, the "/s" option yields the complete file path of all files in all subdirectories. Using "/ad" or "/a-d" yields the complete file path for only folders or files, respectively:

```
x "dir /b /s /ad c:\sesug\ > c:\sesug\folder_only_list.txt";

x "dir /b /s /a-d c:\sesug\ > c:\sesug\file_only_list.txt";
```

Once we have a listing of all desired data files we can use this data set to automatically import each data file. This can be accomplished by reading the MS-DOS output into a SAS dataset, limiting the dataset to include only .csv files, and extracting the filename to tell SAS what to call the new data set. Finally, we can use the SAS "call execute" statement to call a macro to actually perform the import:

```
%macro import(filepath,name);

proc import datafile="&filepath" out=&name;
```

```

run;
%mend import;
data files;
    infile "c:\sesug\file_only_list.txt" trunccover;
    input filepath $100.;
    if index(filepath, ".csv") > 0 then do;
        filename = reverse(scan(reverse(filepath), 1, "\"));
        foldername = reverse(scan(reverse(filepath), 2, "\"));
        datasetname = catt(foldername, substr(filename, 1, (index(filename, ".csv") - 1)));
        call execute ('%import(' || catx(", ", filepath, datasetname) || ')');
    end;
run;

```

This example uses several SAS text functions to define the name of the actual file itself, the folder it resides in, and the desired name for the SAS data set once the data has been imported. The filename is simply everything in the file path after the final "/". The folder name is defined as everything between the second to the last "/" and the last "/". Finally, the dataset name is defined as the folder name plus the file name, minus the file extension (".csv" in this case). The complete file path and the desired dataset name are then passed into a simple macro that uses "PROC IMPORT" to get the data via the "call execute" statement. Note that the "call execute" statement is identical to issuing the following command outside of a data step:

```
%import(filepath,datasetname);
```

Using "call execute" allows the user to issue a macro (or conditionally choose between a variety of macros) for each observation in a data set, while also passing any number of parameters from the dataset into a macro. Obviously, this example is not intended to be a very robust solution, but it hopefully illustrates the potential for using only file and folder names to automatically catalog and then conditionally import external data files.

COMBINING INTERNAL AND EXTERNAL FILE INFORMATION

At this point we have a list of data files that have been imported into SAS and the resulting dataset names. But what if the dataset names or the file names do not appear as we would like? We can use an additional SAS macro to extract some information from within each file, and then use this information to rename and move the original data files. This is often necessary for organizing and archiving purposes. For simplicity, assume that the information we need to rename the original data file is stored in the first row. We can then use the first observation from the imported data (from the column "Name" in our example) set to issue the MS-DOS "RENAME" command as follows:

```

%macro rename(filepath,newname);
x "rename "&filepath" "&newname..csv"";
%mend rename;

%macro getnewfilename(filepath,datasetname);
data _null_;
    set &datasetname;
    if _n_ eq 1 then
        call execute ('%rename(' || catx(", ", &filepath, name) || ')');
run;

```

```
%mend getnewfilename;  
data files;  
    set files;  
    call execute ('%getnewfilename(' || catx(", ", filepath, datasetname) || ')');  
run;
```

Basically, we use the macro “getnewfilename” to extract the desired file name and then pass this information on to the separate “rename” macro to issue the MS-DOS command. The same logic can be applied to move, delete, and even zip data files for archiving purposes.

OTHER APPLICATIONS

Now that we have found, imported, and organized our data files, we can move on to discussing the use of SAS to manage audio files (in MP3 format). A few preliminary details are in order, but we will essentially follow the same steps as outlined above to modify and catalog music files.

OVERVIEW OF MP3 FILE STRUCTURE

The simplest way to think of MP3 audio files is that they are just lengthy text files made up of very large observations. Each observation (which is actually referred to as a frame) can span multiple lines in the data file, with each line consisting of a maximum of 1024 bytes. Modifying any of the information within a frame boundary can result in a useless stream of unrecognizable characters. A more detailed discussion on the frame boundaries and the audio rendering are well beyond the scope of this paper (as well as this author's expertise).

Perhaps the most interesting piece of each MP3 file is the “tag”, which is an embedded string of bytes that allows most MP3 players to display information about the file during playback. Common pieces of information include song title, artist, album, genre, etc. This information adds some advanced functionality to MP3 players, such as the ability to play all of the songs on an album or all of the songs by a particular artist. Without MP3 tags, MP3 players will simply display the actual file name of the MP3 file, which can be quite confusing depending on how your files are labeled. For example, imagine having to choose between 20 files all labeled “Track01.mp3” to find the song you are looking for.

Two problems with MP3 tags, at least in my experience, are that they can often be omitted and in most cases must be entered manually for each audio file. Many software applications will allow you to, for example, add the artist and album information for several tracks at once. But, the individual track names must still be entered manually for each track. This task is obviously quite tedious and subject to data entry errors. Luckily, SAS can perform much of this dirty work for us.

EXTRACTING MP3 TAGS

MP3 tags are referred to as “ID3” tags, and there are several different versions of them. Detailed information on the file layout and evolution of these tags can be found at www.id3.org. The first version of ID3 tags (ID3v1) was the most straightforward format, although its simplicity comes with one glaring limitation: the available space. ID3v1 tags are located in the last 128 bytes of an MP3 file. They are denoted by the three characters “TAG”, followed by 125 bytes of information defined as follows:

- Song name (30 bytes; position 4 – 33)
- Artist (30 bytes; position 34 – 63)
- Album (30 bytes; position 64 – 93)
- Year (4 bytes; position 94 – 97)
- Comments (30 bytes; position 98 – 127)
- Genre (1 byte; position 128)

To get this information out of an MP3 file, we can read the MP3 file one byte at a time in binary format (“RECFM=N”), until we come across the first potential byte of a tag (“T”). If we find a “T” followed by an “A” and a “G”, then we will read in the next 125 bytes and call it our tag. The example below simply puts this text string into a new macro variable, but it could just as easily be sent to a new data set or appended to an existing data set:

```

data _null_;
  infile 'c:\sesug\any_mp3_file.mp3' recfm=n;
  retain t a g 0;
  if t+a+g=3 then do;
    input text $char128.;
    call symput('tag',text);
  end;
  else do;
    input x $char1.;
    if x="T" then do;
      t=1; a=0; g=0;
    end;
    else if x="A" and t=1 then do;
      a=1; g=0;
    end;
    else if x="G" and t=1 and a=1 then do;
      g=1;
    end;
    else do;
      t=0; a=0; g=0;
    end;
  end;
end;
run;

```

While other solutions are certainly possible (and probably more elegant), simply reading each MP3 file in binary format allows an easy way to extract only the information that is needed to tag the file, namely the last 128 bytes. Now that we have the ability to extract ID3 tags from MP3 files, we can use the code from the first section of this paper to automatically process entire directories of audio files, thus building a SAS dataset that can function as a music library. The tags can be parsed to obtain the artist and album information, directories can be created for each album (if they do not already exist), and files can be moved accordingly. Files from this master music library could also be chosen and copied at random, essentially creating playlists for an MP3 device.

CREATING AND MODIFYING MP3 TAGS

After extracting the tags from all of your MP3 files, you will likely find that many tags are either missing, incomplete, or simply not the way that you would like them to be. We can again use SAS to automatically add or edit these tags. A frequent scenario is that all files for a particular album will be stored using the track name for the file name, in a folder that contains the name of the album. This folder might be contained in another folder that contains the artist name. As this artist and album information can be extracted automatically using MS-DOS and SAS as described earlier, we can add these variables to our music library relatively easily, and then use them to create new MP3 tags if one does not already exist. The macro "tag_file" takes as input parameters the file path, song, artist, and album name. For completeness variables for the year, comments, and genre are included as well, although they are often blank. Assuming each of these variables has already been padded with the appropriate number of blank spaces to fill the

allotted number of bytes, adding a tag to an existing MP3 that does not have a tag is as simple as appending 128 bytes to the actual file:

```
%macro tag_file(filepath,song,artist,album,year,comments,genre);  
filename tag_file "&filepath." mod;  
data _null_;  
file tag_file recfm=F lrecl=128;  
put "TAG&song.&artist.&album.&year.&comments.&genre.";   
run;  
%mend;
```

By using the “mod” option on the filename statement, SAS will simply append the desired 128 characters to the end of the file. One other important note is the use of the fixed record format (“refcm=F”) and a record length of 128 (“lrecl=128”), which together instruct SAS to write exactly 128 characters to the MP3 file, but also to exclude any end of record markers.

Editing tags on the other hand is a bit trickier. Here we must edit only the last 128 bytes of a file, without any further modifications. One way to accomplish this is with the “sharebuffers” option on the SAS “filename” statement. By specifying the same file for both input and output, the “sharebuffers” option allows you to modify an external file in place, essentially overwriting any undesired characters. Please refer to the SAS technical note in the references for more information on this topic, as well as some sample code. This is perfect for changing existing MP3 ID3v1 tags, since we simply want to change the 125 bytes following the “TAG” identifier. The SAS macro below presents an example of editing a tag, again assuming that we have the file path of the MP3 file, along with the song, artist, and album information (plus any comments, the year, and genre if desired). We will use the same logic as presented above to find the “TAG”. But, rather than simply extracting it, we will instead overwrite the existing data with the desired text:

```
%macro tag_file(filepath,song,artist,album,year,comments,genre);  
data _null_;  
  infile "&filepath." recfm=n sharebuffers end=endof;  
  file "&filepath." recfm=n;  
  retain t a g 0;  
  if t+a+g=3 then do;  
    put +1 "&song.&artist.&album.&year.&comments.&genre.";   
    stop;  
  end;  
  else do;  
    input x $char1.;  
    if x="T" then do;  
      t=1; a=0; g=0;  
    end;  
    else if x="A" and t=1 then do;  
      a=1; g=0;  
    end;  
  end;  
end;
```

```
        else if x="G" and t=1 and a=1 then do;
            g=1;
        end;
        else do;
            t=0; a=0; g=0;
        end;
    end;
run;
%mend;
```

LIMITATIONS

All of the sample code presented for extracting, adding, and editing MP3 tags only works for ID3v1 tags. As mentioned previously, this format offers limited space for file name, artist, etc. Subsequent versions of ID3 tags (e.g., ID3v2) allow much more flexibility in these fields. However, rather than simply being appended to the end of MP3 files, ID3v2 tags do not have a pre-determined length and are included in the header information of audio files. Due to this, modifying MP3 files in place with the SAS "SHAREBUFFERS" method would likely corrupt the MP3 file. A more robust solution would be necessary for working with ID3v2 tags.

CONCLUSION

The ideas presented in this paper represent potential mechanisms for organizing, editing, and cataloging multiple external data files using a combination of SAS and MS-DOS commands. Combining these concepts with a few SAS macros allows great flexibility in automatically managing and archiving large volumes of data files. The same ideas that can be used to manipulate more traditional files (.txt, .csv, etc.) can also be extended to other file types, such as audio files. While the code presented here is far from a complete solution, the ideas can be extended to account for a variety of particular needs, such as a well organized and labeled digital music collection.

REFERENCES

ComputerHope.com. Microsoft DOS and command prompt. [Online] Available <http://www.computerhope.com/msdos.htm>, July 17, 2007.

SAS Institute, Inc. Remove carriage return and linefeed characters within quoted strings. [Online] Available <http://support.sas.com/ctx/samples/index.jsp?sid=1647>, July 17, 2007.

Yu, H., and Huang, G. "Create Directory on Windows Without the Fleeting DOS Window". Proceedings of the 2002 Pharmaceutical Industry SAS User's Group, paper CC14.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sean Hunt
Quality Insights of Pennsylvania
2 Penn Center Blvd., Suite 220
Pittsburgh, PA 15276
shunt@wwmi.org

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.