# Obtaining A List of Files In A Directory Using SAS® Functions

Jack Hamilton, Kaiser Permanente Division of Research, Oakland, California

## ABSTRACT

This presentation describes how to use the SAS data information functions to obtain a list of the files in a directory under Windows or Unix. This method has the advantages of being written entirely in SAS and mostly platform independent. A common alternative method, "shelling out" to the host operating system, is also discussed.

The data information functions are available in all currently supported versions of SAS.

This paper is in two parts. The first describes obtaining the names of files in a single directory; the second part discusses how to obtain a recursive directory listing, i.e., the files in a directory and the directories underneath it.

## INTRODUCTION

It is often desirable to obtain a list of the files in a directory. You may, for example, want to process all files matching a particular complicated pattern that can't be expressed with wildcards. Or you might want to put them all into a ZIP file using the ODS PACKAGE facility. Or you might just want to list them in a report.

In the past, before the data information functions became available, the standard practice was to shell out to the operating system, issue a directory listing command, and capture and parse the output. This approach had two drawbacks: output from directory commands varies by system and can be hard to parse, and it is not always possible to run an operating system command.

The data information functions, while complicated, avoid these problems.

## THE DATA INFORMATION FUNCTIONS

| Function Name | Function Purpose |
|---|---|
| Filename | Assigns or deassigns a fileref to an external file, directory, or output device |
| DOpen | Opens a directory and returns a directory identifier value |
| DNum | Returns the number of members in a directory |
| DRead | Returns the name of a directory member |
| MOpen | Opens a file by directory id and member name, and returns the file identifier or a 0 |
| DClose | Closes a directory that was opened by the DOPEN function |

**Table 1. Data Information Functions**
**Source: http://support.sas.com/onlinedoc/913/getDoc/en/lrdict.hlp/a000245852.htm**

## READING DIRECTORY ENTRY NAMES

The basic algorithm is this:

1. Open the directory for processing using the Filename and DOpen functions.
2. Count the number of files in the directory using the DNum function
3. Iterate through the files (1 to number-of-files) and get the name of each entry using the DRead function.
4. Attempt to open each entry as a directory using the MOpen function. If it can't be opened as a directory, it's a file. If it's a file, output the name.
5. After looking at all entries, close the directory using the DClose function.

Here's an example that reads the names of the entries in Y:\WUSS2012:

```
data yfiles;

    keep filename;

    length fref $8  filename $80;
    rc = filename(fref, 'Y:\wuss2012');
    if rc = 0 then
        do;
        did = dopen(fref);
        rc = filename(fref);
        end;
    else
        do;
        length msg $200.;
        msg = sysmsg();
        put msg=;
        did = .;
        end;

    if did <= 0
    then
        putlog 'ERR' 'OR: Unable to open directory.';

    dnum = dnum(did);

    do i = 1 to dnum;
        filename = dread(did, i);
        /* If this entry is a file, then output. */
        fid = mopen(did, filename);
        if fid > 0
        then
            output;
    end;

    rc = dclose(did);

run;

proc print data=yfiles;
run;
```

On my laptop, this prints:

```
Obs    filename
1     CGF_55.pdf
2     CGF_57.pdf
3     filenames.sas
4     filenames_recurse.sas
5     FP_57.docx
6     FP_57.pdf
7     pipe.sas
8     WUSS2012.zip
```

## READING DIRECTORIES RECURSIVELY

Reading directories and their subdirectories can be tricky. The example SAS Institute includes with PROC FCMP uses true recursion - it's a routine that calls itself multiple times. But recursion can be hard to understand, and can be slow.

In general, anything that can be done with recursion can be done in another way, and this case it's possible to use a SAS data set as a stack. The dirs._found data set contains a list of the directories to search. It starts out containing only the name of the top level directory. As new directories are encountered, they are added to dirs._found, and processed as the data step steps though the dirs_found data set.

```
/* Data set dirs_found starts out with the names of the root folders    */
/* you want to analyze.  After the second data step has finished, it    */
/* will contain the names of all the directories that were found.       */
/* The first root name must contain a slash or backslash.               */
/* Make sure all directories exist and are readable.  Use complete      */
/* path names.                                                          */
data dirs_found (compress=no);
    length Root $120.;
    root = "y:\wuss2012";
    output;
run;

data
    dirs_found                 /* Updated list of directories searched */
    files_found (compress=no); /* Names of files found.                */

    keep Path FileName FileType;

    length fref $8  Filename $120 FileType $16;

    /* Read the name of a directory to search.             */
    modify dirs_found;

    /* Make a copy of the name, because we might reset root.  */
    Path = root;

    /* For the use and meaning of the FILENAME, DOPEN, DREAD, MOPEN, and  */
    /* DCLOSE functions, see the SAS OnlineDocs.                          */

    rc = filename(fref, path);

    if rc = 0 then
        do;
        did = dopen(fref);
        rc = filename(fref);
        end;
    else
        do;
        length msg $200.;
        msg = sysmsg();
        putlog msg=;
        did = .;
        end;
    if did <= 0
    then
        do;
        putlog 'ERR' 'OR: Unable to open ' Path=;
        return;
        end;

    dnum = dnum(did);

    do i = 1 to dnum;
        filename = dread(did, i);
        fid = mopen(did, filename);
        /* It's not explicitly documented, but the SAS online  */
        /* examples show that a return value of 0 from mopen   */
        /* means a directory name, and anything else means     */
        /* a file name.                                        */
        if fid > 0
        then
            do;
            /* FileType is everything after the last dot.  If */
            /* no dot, then no extension.                     */
            FileType = prxchange('s/.*\.{1,1}(.*)/$1/', 1, filename);
            if filename = filetype then filetype = ' ';
            output files_found;
            end;
        else
            do;
            /* A directory name was found; calculate the complete  */
            /* path, and add it to the dirs_found data set,        */
            /* where it will be read in the next iteration of this */
            /* data step.                                          */
            root = catt(path, "\", filename);
            output dirs_found;
            end;
    end;

    rc = dclose(did);

run;

proc print data=dirs_found;
run;

proc print data=files_found;
run;
```

On my laptop, this prints:

```
Obs    Root

 1     y:\wuss2012
 2     y:\wuss2012\fwdwuss2012acceptanceletter
 3     y:\wuss2012\wuss2012
```

```
Obs   Filename                                    Type   Path

  1   CGF_55.pdf                                  pdf    y:\wuss2012
  2   CGF_57.pdf                                  pdf    y:\wuss2012
  3   filenames.sas                               sas    y:\wuss2012
  4   filenames_recurse.sas                       sas    y:\wuss2012
  5   FP_55.docx                                  docx   y:\wuss2012
  6   FP_57.docx                                  docx   y:\wuss2012
  7   FP_57.pdf                                   pdf    y:\wuss2012
  8   pipe.sas                                    sas    y:\wuss2012
  9   WUSS2012.zip                                zip    y:\wuss2012
 10   WUSS2012.zip                                zip
y:\wuss2012\fwdwuss2012acceptanceletter
 11   WUSS2012_AcceptanceLetter 48.pdf            pdf
y:\wuss2012\fwdwuss2012acceptanceletter
 12   WUSS2012_AcceptanceLetter_Hamilton_DM.pdf   pdf
y:\wuss2012\fwdwuss2012acceptanceletter
 13   WUSS2012_PresentersCopyrightFormDir.pdf     pdf
y:\wuss2012\fwdwuss2012acceptanceletter
 14   WUSS2012_PresentersCopyrightFormZIP.pdf     pdf
y:\wuss2012\fwdwuss2012acceptanceletter
 15   WritersGuidelines2012.pdf                   pdf
y:\wuss2012\wuss2012
 16   WUSS2012_PaperTemplate.doc                  doc
y:\wuss2012\wuss2012
 17   WUSS2012_PresentersCopyrightForm.pdf        pdf
y:\wuss2012\wuss2012
 18   WUSS2012_PresentersFAQs.pdf                 pdf
y:\wuss2012\wuss2012
```

## REFERENCES

"Functions and Call Routines”
Available at < http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245852.htm>.

## ACKNOWLEDGMENTS

Jamila Gul, Kaiser Permanente Division of Research

## UPDATES

**Please check www.sascommunity.org for an updated version of this paper after the conference.**

## CONTACT INFORMATION

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

Jack Hamilton
jfh@acm.org