



## CHAPTER

## 6

## Allocating External Files

---

<i>Introduction</i>	57
<i>Identifying an External File</i>	57
<i>Aggregate External Files</i>	58
<i>Concatenating External Files</i>	58
<i>Aggregate Files vs. Concatenated Files</i>	58
<i>Accessing an External File</i>	59
<i>Using the FILENAME Statement to Reference External Files</i>	59
<i>Advantages of Using the FILENAME Statement</i>	59
<i>FILENAME Statement Syntax</i>	60
<i>FILENAME Statement Examples</i>	60
<i>Using CMS Pipelines</i>	61
<i>Listing Current Filerefs</i>	63
<i>Specifying the Physical Attributes of a File</i>	63
<i>Order of Precedence for Input</i>	64
<i>Order of Precedence for Output</i>	64
<i>Options</i>	64

---

### Introduction

An external file is any file that is not maintained or structured by SAS and that you use during your SAS session. In a DATA step, you use INFILE and INPUT statements to read data from external files, and you use FILE and PUT statements to write to external files. Some SAS procedures, such as the CPORT and FSLIST procedures, operate directly on individual external files. You can also include external files of SAS program statements in your SAS session by using the INCLUDE command or the %INCLUDE statement.

As with SAS files in CMS, external files are stored within a Shared File System (SFS) file space or on minidisks. Refer to *VM/ESA CMS User's Guide* for information about SFS.

---

### Identifying an External File

To identify the external file to SAS, use a *file-specification* argument in a SAS statement. The *file-specification* argument takes one of the following forms:

*external-file*

specifies the CMS fileid of the external file and has the following form:

'filename filetype <filemode | SFS-directory | \*>'

This syntax can also be used to specify a MACLIB as an aggregate external file, as follows:

```
'filename MACLIB'
```

To specify a CMS minidisk or SFS directory as an aggregate file, use the following:

```
'filemode | SFS-directory | *'
```

Omitting the filename and filetype specifies an aggregate file, which consists of an SFS directory or minidisk. Files in the directory or minidisk are accessed as independent members of the aggregate file.

If you omit *filemode* or *SFS-directory*, or if you specify an asterisk (\*) when referencing the file for input, accessed disks are searched in the standard CMS search order, and the first occurrence of a file with a matching filename and filetype is read. If you omit *filemode* or *SFS-directory*, or if you specify an asterisk (\*) when referencing the file for output, the file is written on the first R/W disk.

You cannot use an asterisk (\*) for either *filename* or *filetype*.

#### *fileref*

specifies a logical name associated with an external file. This name contains 1–8 characters and is not enclosed in quotes. The first character must be a letter (A-Z or a-z) or an underscore (\_); the remaining characters can be any combination of letter, numbers, or underscores. If a SAS FILENAME statement or FILENAME function has assigned this logical name to an external file, then the device or file identified by that FILENAME statement is used. If a FILENAME statement has not been issued, then SAS creates an assignment for the fileref by creating a CMS file.

#### *fileref(member)*

specifies an independent member of an aggregate file previously associated with the fileref.

## Aggregate External Files

An aggregate external file is processed as a single file that contains multiple independent members. For input only, CMS MACLIBs can be specified as aggregate files. For input, update, and adding new members, an aggregate file can be an SFS directory or a CMS minidisk.

The FILENAME statement is used to specify an aggregate external file:

```
FILENAME fileref'filemode | SFS-directory | *';
```

## Concatenating External Files

To concatenate external files or directories, use the FILENAME statement. To concatenate files that contain multiple SAS programs in order to invoke the programs in a single invocation, use the SYSIN= system option.

## Aggregate Files vs. Concatenated Files

Aggregate files contain independent files that are processed as individual members. Concatenated files appear to SAS as a single file that contains the data of all of the files listed in concatenation. In other words, concatenated files are not independent of each

other. The order of the concatenation means that files listed first in the FILENAME concatenation take precedence over files that appear later in the list.

Aggregate files can be concatenated. SAS treats an aggregate concatenation as it does any other, by searching each aggregate in turn, in order of appearance in the initial concatenation specification.

---

## Accessing an External File

To access an external file, you specify its filename in a SAS statement or command. For example, this display manager INCLUDE command accesses a file that contains SAS statements and includes it into the Program Editor window:

```
include 'mycode sas a'
```

These statements access an external file of data that is used to create a SAS data set:

```
data mydata;
  infile 'rawdata data b';
  ...
```

If you plan to use the same external file several times in your SAS program, it is more efficient to use the FILENAME statement to establish a fileref to the file. (See “Advantages of Using the FILENAME Statement” on page 59.) You can subsequently use the fileref to refer to the file instead of specifying the filename again.

*Note:* Use the CMS FILEDEF command to assign a DDname, which is also a logical name, only when reading OS/390-sequential or partitioned data sets on OS/390 disks accessed by shared DASD, or when reading OS/390-simulated CMS files identified by filemode 4. △

---

## Using the FILENAME Statement to Reference External Files

A fileref established by a FILENAME statement or FILENAME function remains in effect until the SAS session ends, or until it is changed or deleted by a FILENAME statement that specifies the same fileref. A FILENAME statement for disk always overrides a CMS FILEDEF command for disk. When you use a FILENAME statement to assign a fileref to a disk file, the native CMS interface is used for I/O. If you use a CMS FILEDEF command to assign a fileref to a disk file, OS/390 Simulation Services (provided by CMS) are used.

### **CAUTION:**

**Do not assign different filerefs to the same physical file or use the same fileref for concurrent access.** For example, if you assign a fileref to a file, then browse the file through the FSLIST window, do not attempt to go to the Program Editor window and submit a DATA step to write to the file while it is still displayed in the FSLIST window. △

---

## Advantages of Using the FILENAME Statement

There are several advantages to using the FILENAME statement to identify external files.

- It is portable. The FILENAME statement is recognized by SAS under all operating environments. You can develop a SAS program under CMS and port it

to another operating environment with fewer changes to your program statements than if you used the CMS FILEDEF command.

- It is easier for novice CMS users to understand than the CMS FILEDEF command.
- It enables you to list the filerefs that you have assigned.
- It reduces processing time.
  - Filerefs assigned by a FILENAME statement are found before filerefs assigned by a FILEDEF command.
  - Filerefs assigned by the FILENAME statement use OS/390 Simulation only for nondisk files. Filerefs assigned by a FILEDEF command always use OS/390 Simulation, even for disk files.

## FILENAME Statement Syntax

This section provides a brief overview of FILENAME statement syntax. For complete information about the FILENAME statement, see “FILENAME” on page 209. The general form of the FILENAME statement is

**FILENAME** *fileref* | *\_ALL\_* *device-type* <'external-file'> <*options*>;

The FILENAME statement takes the following options:

*fileref*

is a logical name by which the external file is referenced. The fileref must begin with a letter or underscore and must contain 1-8 characters consisting of letters, numbers or underscores.

*\_ALL\_*

is a reserved fileref that is used only to list or clear filerefs.

*device-type*

specifies the type of output or input device for the file. If not specified, SAS assumes that the file is on disk.

*'external-file'*

identifies the physical file to be associated with the fileref.

*options*

is a list of options that control how the file is read or written. When specifying more than one option, use a blank space to separate each option. All options use a *keyword=value* format.

## FILENAME Statement Examples

- The two FILENAME statements below associate the fileref MYDATA with the external file HOUSES DATA A on disk. Note that the second example is equivalent to the first, but the disk option is unnecessary.

```
filename mydata 'houses data a';
```

```
filename mydata disk 'houses data a';
```

- The following FILENAME statement associates the fileref OUTDATA with the external file SOME FILE:
 

```
'some file fpool:user3.data97';
```
- The following FILENAME statement associates the fileref INDATA with the external file SURVEY MARCH on disk. If this fileref is used for output, it is

written to the first R/W disk. If this fileref is used for input, the standard CMS minidisk search order is followed.

```
filename indata 'survey march';
```

- The following FILENAME statement associates the fileref OUT with the virtual PUNCH. Any output to OUT is directed to the virtual PUNCH.

```
filename mydata punch;
```

- The following FILENAME statement associates the fileref MYLIB with the tape device defined as 182.

```
filename mylib tape 'tap2' ;
```

- The following FILENAME statement associates the fileref MYDIR with the SFS directory FPOOL:MYUSER.MYDIR as an aggregate external file:

```
filename mydir 'fpool:myuser.mydir';
```

- The following FILENAME statement clears the association between the fileref MYDATA and any device or file, provided the fileref was originally assigned by a FILENAME statement. CLEAR is assumed by default. However, you may state it explicitly.

```
filename mydata clear;
```

- The following FILENAME statement concatenates three files. When the fileref corresponding to a list of files to be concatenated occurs in a later open-for-read operation, the data is read sequentially. When the fileref occurs in a later open-for-write operation, output is written to the first file in the list.

```
filename all
('mar file' 'apr file' 'may file');
```

- The following FILENAME statement specifies multiple members in multiple MACLIBs. You cannot specify that output be sent to a MACLIB file.

```
filename a
('one maclib' 'two maclib' 'three maclib');
```

---

## Using CMS Pipelines

You can use a standard CMS pipeline specification in place of the external file specification in the FILE, FILENAME, and INFILE statements, and in the FILENAME function. This enables you to receive input from any CP or CMS command or pipeline device driver, or to route output to any pipeline device driver.

Use of pipelines requires, as a minimum, CMS Release 12 or level 1.1.9 of the CMS Pipelines Runtime Library.

For the FILENAME statement and FILENAME function, specify PIPE as the device type. For the FILE and INFILE statements, specify PIPE as an option. If you specify PIPE as an option in a FILE or INFILE statement, you cannot specify any other options in that statement.

SAS supports inline comments as an extension of the standard CMS pipeline interface, as shown:

```
filename adata pipe
'(name adata) /* name pipeline */
< archive data a /* read in file */
| unpack /* unpack */
```

```
';
```

Data transfer between SAS and the CMS pipeline takes place in the pipeline's FITTING stage. By default, SAS inserts a FITTING stage at the beginning of pipeline specifications in FILE and INFILE statements, and at the end of FILENAME statements or functions. For example, the FILENAME statement above is internally coded in SAS as:

```
filename adata pipe
  '(name adata)      /* name pipeline */
  < archive data a /* read in file  */
  |  unpack          /* unpack      */
  |  fitting sasio  /* added by SAS */
  ';
```

You can override the default insertion of the FITTING stage by explicitly inserting the stage into your pipeline specification. This is allowed as long as the FITTING stage is always either the first or last stage of a stream. For example, the FANIN stage in the middle of this pipeline delivers data to the FITTING stage:

```
filename twostrm pipe
  '(endchar %) < jan data a
  | a: fanin
  |   fitting sasio
  %
  < feb data a
  | a:
  ';
```

You may prefer to write complex pipelines as REXX pipeline stages, then to specify the REXX stage name in your SAS program. For the unpack example above, you could write a file called UNPKDATA REXX:

```
/* UNPKDATA REXX */
parse arg fileid
'callpipe <' fileid,
  '|  unpack',
  '|  *:'
exit
```

Then you could write your FILENAME statement as

```
filename mydata pipe
  '(name mydata)      /* name pipeline */
  rexx unpkdata archive data a
  /* read and unpack a file      */
  ';
```

Macro variables are resolved within a pipeline specification that is enclosed in double quotes.

This example illustrates the use of pipelines with INFILE, FILENAME, and FILE statements, and with macro variables.

```
/* Data step to get spoolid. Only accept files      */
/* sent by a specified userid                       */
data _NULL_;
  infile                                          /* define the pipeline */
  'cp query rdr                                /* query all rdr files */
  |strfind /USERABC / /* from this userid only */
```

```

        |take 1          /* only process one file */
        |specs word 2 1' /* keep only the spoolid */
    PIPE;
input spid $;          /* read spoolid as char variable */
                        /* save spoolid as macro variable */
call symput('SPOOLID',spid);
run;

/* define fileref for reading netdata rdr file */
filename nd pipe
"reader file &spoolid /* read the file */
|drop 1             /* skip TAG */
|specs 2-* 1       /* skip CC byte */
|deblock netdata   /* interpret netdata format */
|strfind xC0       /* keep only data records */
|specs 2-* 1       /* skip control byte */
";

/* datastep to read input from rdr file */
/* write data to USERABC DATA in packed format */
data abc;
file          /* pipe for PUT processing */
' strip      /* strip blanks */
| pack V 80   /* write packed format */
| > userabc data a fixed' pipe; /* this file */
infile nd;    /* read from ND fileref */
input x;      /* read var x from each rec */
put x;        /* write to USERABC DATA in packed format */
run;

```

---

## Listing Current Filerefs

In a windowing SAS session, you can issue the FILENAME command from the command line to display the ACTIVE FILE SHORTCUTS window. This window provides a listing of all current filerefs assigned by SAS FILENAME statements and the complete CMS fileid to which each fileref is assigned.

You can use the following form of the FILENAME statement in any SAS job to obtain the same information that is presented in the FILENAME window:

```
filename _all_ list;
```

---

## Specifying the Physical Attributes of a File

You can specify values for file characteristics in the *options* argument in the FILE, FILENAME, and INFILE statements. Note that you do not need to specify file characteristics when you are concatenating files, since the existing characteristics of the files are used.

The following section explains which values are used if you specify file characteristics for the same file in multiple places.

---

## Order of Precedence for Input

If you use a fileref to reference a file for input, any values that you specify for file characteristics in the INFILE statement take precedence over any that you have specified in a FILENAME statement or in a CMS FILEDEF command. And any values that you specify for file characteristics in a FILENAME statement or CMS FILEDEF command take precedence over the file characteristics of the existing file. If you do not specify any values for file characteristics, then the values for the existing file are used.

---

## Order of Precedence for Output

If you use a fileref to reference a file for output, any values that you specify for file characteristics in the FILE statement take precedence over any that you have specified in a FILENAME statement or in a CMS FILEDEF command. If you do not specify any values for file characteristics when you are appending to an existing file, the values for the existing file are used. If you do not specify any values for file characteristics when you are writing to a new file, the defaults are used. Default file characteristics are shown in “FILENAME” on page 209.

---

## Options

The following file characteristics can be specified in the *options* argument in a FILE, FILENAME, or INFILE statement. All options use a *keyword=value* format, and multiple options are separated by blank spaces. For the FILE statement, note that options cannot be specified if you use FILE LOG, FILE PRINT, or FILE PIPE. For the FILENAME statement, note that if file characteristics are given in a statement that specifies several files to be concatenated, then all files must have the same attributes.

### BLKSIZE=*value*

specifies the buffer size that is allocated to contain records. Valid values are 1 through 65535.

Records do not have to be blocked. However, because disk or tape activity is reduced when records are blocked, it is more efficient to use blocks. Blocking records is different when you use CMS Native I/O Services than it is when you use OS/390 Simulation Services.

For CMS Native I/O Services, the following apply:

- No blocking of records is performed in variable-length files. The BLKSIZE= option is ignored, and RECFM=VB is treated the same as RECFM=V.
- For fixed-length files, BLKSIZE= specifies the maximum buffer size to contain records. The default block size is the value of LRECL= times the number of logical-record-length records that fit into 1,024 bytes. For example, a fixed-length file with LRECL=80 by default has BLKSIZE=960. Twelve 80-byte records can fit into 1,024 bytes, taking up 960 bytes.
- CMS does not support blocking for a *device-type* of TERMINAL.
- Any time the RECFM= option includes an A (specifying a PRINT file), add 1 to the value of the LRECL= option in case the column contains carriage-control characters.

For OS/390 Simulated I/O Services, the following apply:

- Blocking is not supported for a *device-type* of READER.
- Blocking sizes specified for *device-type* of TAPE, as well as OS/390-shared DASD files, must match the block size of the existing file.

- When RECFM=FB, the value of the LRECL= option is the length of the longest record, and BLKSIZE is an integer multiple of LRECL. For example, if the longest record is 70 bytes, then LRECL=70 and BLKSIZE=70 x *n*, where *n* is an integer.
- When RECFM=VB, the value of the LRECL= option is the length of the longest record plus 4 bytes that hold record descriptor information. The value of the BLKSIZE= option can be any value up to 32,760, as long as it is at least 4 bytes longer than the value of the LRECL= option. (These 4 bytes are the block descriptor information.)
- Any time the value of RECFM= option includes A (which specifies a PRINT file), add 1 to the value of the LRECL= option in case the column contains carriage-control characters.

**DENSITY=***value*

specifies tape density in bits per inch. Valid values include 200, 556, 800, 1600, 6250, and 38K.

**DISP=***value*

specifies the status (disposition) of the file. Acceptable values are

- |            |   |
|------------|---|
| <b>MOD</b> | specifies that output lines are to be written after any existing lines in the file.   |
| <b>OLD</b> | specifies that any output lines are to be written at the beginning of the file. This is the default. For CMS, this option is identical to the NEW option. |
| <b>NEW</b> | specifies that any output lines are to be written at the beginning of the file. For CMS, this option is identical to the OLD option.                      |

**FILEVAR=***variable*

enables you to dynamically change input and output files in the middle of a DATA step.

**LABEL=***value*

indicates the type of label processing for a tape file. LABOFF is the default. Valid values are

- BLP
- LABOFF
- NL
- SL

If SL, NL, or BLP is specified, then an additional label value *n* can be specified after the SL, NL, or BLP. The value of *n* indicates the file position in a multfile volume. The default value is 1.

**LEAVE=**YES

indicates that a multfile tape is not repositioned at open for LABOFF or BLP processing. For SL tapes, LEAVE=YES does not reposition before label processing. Omitting LEAVE or specifying LEAVE=NO causes a tape to be rewound and repositioned each time a file is opened. (See “Working with SAS Files on Tape” on page 36 for details on tape processing.)

**LRECL=***value*

specifies the logical record length in bytes. Valid values are 1 through 65,535.

**RECFM=***format*

specifies the format of records in the file.

You can choose one of the following record formats:

F	specifies fixed-length records, unblocked.
FB	specifies fixed-length records, blocked.
V	specifies variable-length records, unblocked.
VB	specifies variable-length records, blocked.
U	specifies undefined record format. For CMS disk files, this is the same as V.

You can use the following values in any of the previously listed formats except U.

A	specifies that the first byte of each record is an ANSI printer control character and that the file is to be handled as a print file.
S	specifies that the file contains spanned records (V), or that the file contains standard blocks (F). FS, FBS, VS, and VBS files must be assigned with a CMS FILEDEF command.

**SYSPARM=***value*

passes an option string to a tape management system for standard label tapes. Entering a question mark (?) causes the SAS System to prompt you for option settings.

**TRACK=***value*

specifies the tape setting. Valid values are

- 7TRACK
- 9TRACK
- 18TRACK
- 3489B
- 3490C
- 3590B
- 3590C

**VOLID=***value*

specifies the volume serial number to be verified in the tape. If the value contains any special characters, it must be enclosed in single quotes.

See the FILEDEF command entry in *VM/ESA CMS Command Reference* for more information about these options.

SAS does not interfere with FILEDEF options that are already specified. Therefore, TAPE options that are specified in the FILE, FILENAME, or INFILE statements that conflict with an existing FILEDEF to TAPE are ignored. See “Working with SAS Files on Tape” on page 36 for more information.

If a variable-length file is opened for update (possibly in the DATA step) and if the replacement line is not the same length as the existing line, then the standard CMS file system action truncates the file at that point. No message is given. The next read to that file returns EOF.