CHAPTER

6

Accessing External Files

Referring to External Files

After allocating an external file, you can use the fileref or DDname of the file as a convenient way of referring to that file in any subsequent SAS language statement or command. (Note: The first time the DDname of an external file is used in a SAS statement or procedure, SAS assigns it as a fileref for the external file. Therefore, any information provided here about filerefs also applies to the DDnames of external files.) In the following example, the FILENAME statement associates the fileref REPORT with the sequential data set MYID.NEWDATA. The FILE statement later uses the fileref rather than the data set name to refer to the data set.

```
filename report 'myid.newdata' disp=old;
data _null_;
   file report;
   put ...;
run;
```

Here is a similar example in which a JCL DD statement associates the DDname IN with a member of a partitioned data set. The INFILE statement later uses the DDname rather than the data set name and member name to refer to the PDS member.

```
//IN DD DSN=MYID.NEWDATA(TRIAL1),DISP=SHR
//SYSIN DD *
data out;
   infile in;
   input ...;
run;
```

When referring to a member of a PDS or a PDSE, you also have the option of specifying only the data set name in the FILENAME statement (or FILENAME function) or in the DD statement. Then, in subsequent references, you specify the member name with the fileref. For example:

```
//IN DD DSN=MYID.NEWDATA,DISP=SHR
//SYSIN DD *
data out;
   infile in(trial1);
   input ...;
run;
```

If an external data set is not cataloged, you must also provide the volume serial number. See “FILENAME” on page 294 for more information about other options that you can specify.

How SAS Determines Device Types

A fileref has a device type of either MVS or HFS, where HFS identifies files that are stored in the hierarchical file system of UNIX System Services. The device type determines how SAS will access the file. This device type is somtimes referred to as an access method.

If a physical name does not contain a slash (/) or a tilde (~) character to identify it as an HFS filename, SAS uses the following algorithm to determine device type:

1. Use the HFS access method from the allocation statement, if provided, as in:

   ```
   FILE 'example' HFS;
   ```

   or

   ```
   FILENAME XXX HFS 'example';
   ```

   If the access method is not specified, use the MVS access method.
2. Use the access method specified by the MVS: or HFS: prefix in the physical filename, if one is provided, as in:

   ```
   FILENAME XXX 'HFS:first';
   ```
3. Use the HFS access method if a slash character (/)or tilde character (~) appears in the physical filename, as in:

   ```
   FILENAME XXX '~/first';
   ```
4. Use the access method specified by the FILESYSTEM= system option. See "FILESYSTEM=" on page 351.

Writing to External Files

After allocating an external file, you can use the FILE statement, FILE command, or FOPEN function to write to the file. This section describes the FILE statement. For information about the FILE command, see *SAS Language Reference: Dictionary*.

Note: You can also use FOPEN, FWRITE, FPUT, FNOTE, FPOINT, and FCLOSE to access external files. See *SAS Language Reference: Dictionary* for details. △

FILE Statement

The FILE statement specifies the current output file for PUT statements in the DATA step. (See *SAS Language Reference: Dictionary* for a complete description of the PUT statement.)

When multiple FILE statements are present, the PUT statement builds and writes output lines to the file that was specified in the most recent FILE statement. If no FILE statement was specified, the PUT statement writes to the SAS log.

The specified output file must be an external file, not a SAS data library, and it must be a valid access type.

The FILE statement is executable; therefore, you can use it in conditional processing (in an IF/THEN statement, for example).

As with INFILE, it is possible to alternately access multiple external files. See the example in "Reading from Multiple External Files" on page 90. You cannot write to

multiple members of a single PDS at the same time. However, you can write to multiple members of a PDSE at one time.

Under OS/390, SAS uses the IBM ENQUEUE/DEQUEUE facility to prevent multiple users from writing to the same physical file simultaneously. This facility also prevents the SAS System and ISPF from overwriting each other.

FILE Statement Syntax

This section provides a brief overview of FILE statement syntax. For complete information about the FILE statement, see "FILE" on page 288.

The syntax of the FILE statement is

FILE *file-specification* < *type*> <*options* > <*host-options*>;

file-specification
identifies the file. It can be in the following forms:

Table 6.1 File Specification Examples for the FILE Statement

Form	Example
fileref	**report**
fileref(*member*)	**report(feb)**
'physical-filename'	**'library.daily.report'**
'physical-filename(member)'	**'library.daily.output(report1)'**
reserved filerefs	**LOG** or **PRINT**
HFS file	**'/u/userid/file'** **'HFS:myfile'**

See "Specifying Physical Files" on page 13 for details about different ways of specifying *physical-filename*.

type
specifies the type of file. Nonstandard (host-specific) file types that you can specify for OS/390 are

DLI	for IMS-DL/I databases (see "Accessing IMS-DL/I and CA-IDMS Databases" on page 91)
HFS and PIPE	for files in UNIX System Services (see "Accessing UNIX System Services Files" on page 93)
VSAM	for VSAM files (see "Accessing VSAM Data Sets" on page 92).

options
describe the output file's characteristics and specify how it is to be written with a PUT statement. Many of these options are not host-dependent and are documented in *SAS Language: Reference*. For information about options that are specific to OS/390, see "FILE" on page 288. You can use these options to do the following:

- define variables that will contain information about the external file
- specify special open and close processing
- specify file characteristics.

FILE Statement Examples

Table 6.2 File Specification Examples for the FILE Statement

Type of Data Set	Example
sequential	`file 'my.new.dataset';`
member of a PDS or PDSE	`file out(newdata);` `or file 'my.new.dataset(newdata)';`
sequential or member of a PDS or PDSE*	`file myfilerf;`
HFS	`file '/usr/tmp/newdata';`
HFS	`file 'newmem.dat' hfs;`
HFS	`file 'HFS:raw';`
VSAM	`file payroll` `vsam;`
IMS	`file psb dli;`
SAS log	`file log;`

* depending on what the fileref is associated with

Writing to Sequential Data Sets

The disposition of a sequential data set can be OLD, MOD, or SHR. Using OLD eliminates the possibility of another job writing to the data set at the same time your job is writing to it.

If you specify OLD or SHR, SAS begins writing at the beginning of the data set, replacing existing information. To append new information to the existing information, specify the MOD option in the FILE statement.

The following example assigns the fileref RAW to the data set MYID.RAW.DATAX and uses the fileref in a simple DATA step:

```
filename raw 'myid.raw.datax' disp=old;
data _null_;
   file raw;
   msgline='write this line';
   put msgline;
run;
```

Writing to Members of PDS or PDSE Data Sets

To write to a member of a PDS, include the member name along with the data set name in the FILE statement, the FILENAME statement, the FILENAME function, the TSO ALLOCATE command, or the JCL DD statement. Omitting the member name causes an error message because SAS tries to treat the PDS as a sequential data set.

The disposition of the PDS member can be OLD or SHR; you cannot use a disposition of MOD for a member of a PDS. In both cases, SAS begins writing at the beginning of the member, replacing existing information. Using OLD eliminates the possibility of another job writing into the member at the same time your job is writing into it.

In a single DATA step you can write to only one member of a particular PDS; however, you can write to members of separate PDSs. To write to more than one member of a given PDS, you must use a separate DATA step for each member. In a single DATA step, you can write to multiple members of a PDSE.

The following example assigns the fileref RAW to the PDS member MEM1 and then uses the fileref in a simple DATA step:

```
/* PDS Example */
filename raw 'myid.raw.data(mem1)' disp=old;
data _null_;
   file raw;
   put 'write this line';
run;
```

This next example assigned the fileref MYPDSE to the PDSE member REG3 and then uses the fileref in a simple DATA step:

```
/* PDSE Example */
filename mypdse 'sales.div1.reg3' disp=shr;
data a;
   x=1;
   file mypdse(june97);
   put x;
   file mypdse(jul97);
   put x;
run;
```

Writing to a Printer

This example uses the FILENAME and FILE statements to route output to a printer.

```
filename prnt printer sysout=a;
data _null_;
   file prnt;
   put 'text to write';
run;
```

Writing to the Internal Reader

This example uses the FILENAME and FILE statements to write to an internal reader.

```
filename injcl '.misc.jcl' disp=shr;
filename outrdr sysout=a pgm=intrdr
      recfm=fb lrecl=80;
data _null_;
  infile injcl(myjcl);
  file outrdr noprint notitles;
  input;
  put _infile_;
run;
```

Writing to a Temporary Data Set

The following examples use the FILENAME and FILE statements to write to a temporary data set.

- This example shows how to use default attributes to define a temporary file:

```
filename tempfile '&mytemp' ;
data out;
   file tempfile;
   put ...;
run;
```

- The next example defines a temporary file and specifies some of its attributes:

```
filename nextone '&mytemp' disp=new
   lrecl=80 blksize=320 space=(trk,(3));
data out;
   file nextone;
   put ...;
run;
```

Using the FILE Statement to Specify Data Set Attributes

You can specify data set attributes in the FILE statement as well as in the FILENAME statement or FILENAME function. SAS supplies default values for any attributes that you do not specify. (For information about default values, see "Overview of DCB Attributes" on page 302 and "DCB Option Descriptions" on page 300.)

This example specifies values for LRECL= and RECFM= in the FILE statement and allows SAS to use the default value for BLKSIZE=:

```
filename x 'userid.newdata' disp=new
       space=(trk,(5,1)) volume=xyz111;
data out;
   file x lrecl=80 recfm=fb;
   put ... ;
run;
```

Using the Data Set Attributes of an Input File

In this example, data is read from the input file; then the data is written to an output file, using the same file characteristics. The DCB option in the FILE statement tells SAS to use the same data set attributes for the output file as were used for the input file.

```
filename in  'userid.input';
filename out 'userid.output';
data;
   infile in;
   input;
   file out dcb=in;
   put _infile_;
run;
```

Using the FILE Statement to Specify Data Set Disposition

Appending Data with the MOD Option

In this example, the MOD option is used to append data to the end of an external file.

```
filename out 'user.output';
data _null_;
 /* New data is written to 'user.output'  */
   file out;
   put ... ;
run;

data _null_;
 /* data is appended to 'user.output'     */
   file out mod;
   put ... ;
run;
```

Appending Data with the MOD Disposition

This example is similar to the previous one except that instead of using the MOD option, the DISP= option is used. The OLD option is then used to overwrite the data.

```
filename out 'user.output' disp=mod;
data _null_;
 /* data is appended to 'user.output'    */
   file out;
   put ... ;
run;

data _null_;
 /* data is written at the beginning of  */
 /* 'user.output'                         */
   file out old;
   put ... ;
run;

data _null_;
 /* data is written at the beginning of  */
 /* 'user.output'                         */
   file out;
   put ... ;
run;

data _null_;
 /* data is appended to 'user.output'    */
   file out mod;
   put ... ;
run;
```

Writing to Print Data Sets

A print data set contains carriage-control information (also called ASA control characters) in column 1 of each line. These characters (blank, 0, –, +, and 1) control the operation of a printer, causing it to skip lines, to begin a new page, and so on. They do not normally appear on a printout. A nonprint data set does not contain any carriage-control characters.

When you write to a print data set, SAS shifts all column specifications in the PUT statement one column to the right in order to accommodate the carriage-control characters in column 1. Therefore, if you expect to print an external file, you should designate the file as a print data set either when you allocate it or when you write to it.

- The preferred method for designating a data set as a print data set is when you allocate it, using the RECFM= option in the FILENAME statement, the FILENAME function, the JCL DD statement, or the TSO ALLOCATE command. Adding the letter A to the end of the value for the RECFM= option (RECFM=FBA or RECFM=VBA, for example) causes SAS to include carriage control characters in the data set that is being created. See "FILENAME" on page 294 for complete information about the RECFM= option.
- When you write to a data set that was not designated as a print data set when it was allocated, you can designate it as a print data set in several ways, depending on what you plan to do with the data set. Here are some examples:
  - Use the PRINT option in the FILE statement, as in:

    ```
    file saveit print;
    ```

    SAVEIT is the fileref of the data set. The PRINT type in the FILE statement includes a page number, date, and title; this is the simplest way to create a print data set.
  - Use PRINT as the fileref in the FILE statement (different from the PRINT option above), as in:

    ```
    file print;
    ```

    The PRINT fileref in the FILE statement causes SAS to write the information either to the standard SAS procedure output file (PRINT=SASLIST), or to another output file if you have used a PROC PRINTTO statement to redirect your output. (See "PRINTTO" on page 259 and "Using the PRINTTO Procedure" on page 107 for information about PROC PRINTTO.) In either case, this file contains carriage-control characters by default. You can suppress the carriage-control characters by specifying the NOPRINT option in the FILE statement (see "Writing to External Files" on page 79).
  - Use the letter A as part of the value in the RECFM= option in the FILE statement:

    ```
    file saveit recfm=vba;
    ```

    As in the FILENAME statement or FILENAME function, the letter A in the RECFM= option of the SAS FILE statement causes SAS to include carriage-control characters in the data set that is being created. SAS also changes the record format of the target data set.

For information about how to process print files as input, see "Reading from Print Data Sets" on page 91.

Designating a Print Data Set as a Nonprint Data Set

The NOPRINT option is useful when you use a DATA step to copy a data set that already contains carriage-control information. In this case, use NOPRINT to prevent SAS from adding an additional column of carriage-control information.

If a data set has been allocated as a print data set, you can use the NOPRINT option in the FILE statement to omit carriage-control information. For example, suppose you specified RECFM=VBA, indicating a print data set, when you allocated a file and that you assigned with the fileref OUTDD. The following SAS statement designates OUTDD as a nonprint data set:

```
file outdd noprint;
```

To write lines without carriage-control information to the SAS procedure output file, specify:

```
file print noprint;
```

Reading from External Files

After you allocate an external file, you can read from the file in a SAS DATA step by specifying it in the INFILE statement, the INCLUDE command, or the %INCLUDE statement.

This section describes the INFILE statement. For information about the INCLUDE command, the %INCLUDE statement, and the DATA step, see *SAS Language Reference: Dictionary*.

INFILE Statement

In a SAS DATA step, the INFILE statement specifies which external file is to be read by a subsequent INPUT statement. Every external file that you want to read must have a corresponding INFILE statement. The external file can be a sequential data set on disk or tape, a member of a partitioned data set (PDS or PDSE), or any of several nonstandard file types (see the description of the *type* argument in "INFILE Statement Syntax" on page 86). The file can also be entered from a terminal.

The INFILE statement is executable. Therefore, it can be used in conditional processing–in an IF/THEN statement, for example.

When multiple INFILE statements are present, the INPUT statement reads from the external file that was specified by the most recent INFILE statement. (See *SAS Language: Reference* for a complete description of the INPUT statement.)

INFILE Statement Syntax

This section provides a brief overview of INFILE statement syntax. For complete information about the INFILE statement, see "INFILE" on page 309.

The syntax of the INFILE statement is

INFILE *file-specification* <*type*> <*options*>;

file-specification
identifies the file. It may be in the following forms:

Table 6.3

Form	Example
fileref	`report`
fileref(*member*)	`report(feb)`
'physical-filename'	`'library.daily.report'`
'physical-filename(member)'	`'library.daily.source(report1)'`
reserved fileref	`DATALINES`

See "INFILE" on page 309 for information about partial physical filenames and wildcard member names.

type
specifies the type of file. When you omit *type*, the default is a standard external file. Nonstandard (host-specific) file types that you can specify for OS/390 are

DLI — for IMS-DL/I databases. For information about IMS-DL/I options for the FILE statement, see *SAS/ACCESS Interface to IMS-DL/I Software*.

HFS and PIPE — for files in UNIX System Services (see "Accessing UNIX System Services Files" on page 93). PIPE allows you to issue UNIX System Services commands from within the INFILE statement.

IDMS — specifies that the file is a CA-IDMS file. For information about CA-IDMS options for the INFILE statement, see *SAS/ACCESS Interface to CA-IDMS Software: Reference*.

ISAM — specifies that the file is an ISAM file. See "Accessing Nonstandard Files" on page 91.

VSAM — for VSAM files (see "Accessing VSAM Data Sets" on page 92).

VTOC — specifies that the Volume Table of Contents (VTOC) is to be accessed.

options
describe the input file's characteristics and specify how it is to be read with an INPUT statement. Many of these options are not host-dependent and are documented in *SAS Language Reference: Dictionary*. Those that are host-specific are documented in "INFILE" on page 309. You can use these options to do the following:

- define variables that will contain information about the external file
- specify special open and close processing
- specify file characteristics.

INFILE Statement Examples

Table 6.4 File Specification Examples for the INFILE Statement

Type of Data Set	Example
sequential	**infile 'library.daily.data';**
member of a PDS or PDSE	**infile report(feb);** or **infile 'lib.daily.src(rpt1)';**
sequential or member of a PDS or PDSE*	**infile data;**
IMS	**infile psb dli;**
in-stream	**infile datalines;**

* depending on what the fileref is associated with

Reading from a Sequential File

This example assigns the fileref RAW to the data set MYID.RAW.DATAX and uses the fileref in a simple DATA step:

```
filename raw 'myid.raw.datax' disp=shr;
data out;
   infile raw;
   input ... ;
run;
```

This example is similar to the previous one, except that it specifies a value for the SYSPREF= system option and then uses a partially qualified data set name in the FILENAME statement:

```
options syspref=sys2.sas7;
filename raw2 '.raw.datax' disp=shr;
data out;
   infile raw2;
   input ... ;
run;
```

See "Specifying Physical Files" on page 13 for information about using SYSPREF= and partially qualified data set names.

Reading from a Member of a PDS or PDSE

This example specifies the PDS name in the FILENAME statement and then specifies the member name in parentheses following the fileref in the INFILE statement:

```
filename mypds 'user.my.pds';
data out;
   infile mypds(mydata);
   input ... ;
run;
```

This example specifies both the PDS name and the member name in the FILENAME statement. Therefore, only the fileref is specified in the INFILE statement:

```
filename mymember 'user.my.pds(mydata)';
data out;
   infile mymember;
   input ... ;
run;
```

Multiple members of a PDS can be open for read access at the same time.

Reading from the Terminal

If you run SAS in interactive line mode or in noninteractive mode, you can read input from the terminal. These examples illustrate ways to define a terminal file.

In the first example, TERMINAL is specified as the device type in the FILENAME statement:

```
filename term1 terminal;
data one;
   infile term1;
   input ... ;
run;
```

In the next example, an asterisk is used in place of an physical file name to indicate that the file will be entered from the terminal:

```
filename term2  '*';
data out;
   infile term2;
   input ... ;
run;
```

Note: Enter "/*" to signify end-of-file after entering your input from the terminal. △

Reading Concatenated Data Sets

Multiple sequential data sets can be concatenated (via a JCL DD statement, a TSO ALLOCATE command, or a FILENAME statement) and read consecutively using one pair of INFILE/INPUT statements.

Sequential data sets and individual PDS or PDSE members can also be concatenated, as in the following example:

```
x alloc fi(in1)
   da('my.data1' 'my.pds(mem)' 'my.data2');
data mydata;
   infile in1;
   input ... ;
   /* SAS statements  */
run;
```

Here is an example of using the FILENAME statement to concatenate data sets:

```
filename in1 ('my.data1' 'my.pds(mem)' 'my.data2');
```

You can also concatenate external files that are stored on different types of devices and that have different characteristics.

If PDSs or PDSEs are concatenated and a member is specified in the INFILE statement, then SAS searches each PDS or PDSE for that member. SAS searches in the

order in which the PDSs appear in the DD statement, the ALLOCATE command, or the FILENAME statement or function. If the member is present in more than one of the PDSs, SAS retrieves the first one that it finds.

Reading from Multiple External Files

You can read from multiple external files either sequentially or alternately from multiple filerefs.

- To read from multiple external files sequentially, use the END= option or the EOF= option in each INFILE statement to direct program control to a new file after each file has been read. For example:

```
filename outrdr sysout=a pgm=intrdr
      recfm=fb lrecl=80;
data _null_;
   length dsn $ 44;
   input dsn $;
   infile dummy filevar=dsn end=end;
   file outrdr noprint notitles;
   do until(end);
      input;
      put _infile_;
      end;
cards;
PROD.PAYROLL.JCL(BACKUP)
PROD.PAYROLL.JCL(TRANS)
PROD.PAYROLL.JCL(PRINT)
;
run;
```

  See *SAS Language Reference: Dictionary* for more information about the END= and EOF= options of the INFILE statement.

- In order to alternately access multiple external files, the files must have different filerefs. You can partially process one file, go to a different file, and return to the original file. An INFILE statement must be executed each time you want to access a file, even if you are returning to a file that was previously accessed. The DATA step terminates when SAS encounters the EOF of any of the files. Consider the following example:

```
filename exfile1 'my.file.ex1';
filename exfile2 'my.file.ex2';
data mydata;
   infile exfile1;
   input ... ;
   /* SAS statements  */

   infile exfile2;
   input ... ;

   /* SAS statements  */

   infile exfile1;
   input ... ;

   /* SAS statements  */
```

```
run;
```

When there is more than one INFILE statement for the same fileref, with options specified in each INFILE statement, the options apply cumulatively to successive files.

Note: Multiple files inside concatenations cannot be accessed in this manner. △

Reading from Print Data Sets

When reading from a print data set, you can tell SAS to ignore the carriage-control character that is in column 1 of print data sets by specifying the SAS system option FILECC. For more information, see "FILECC" on page 342.

Getting Information about an Input Data Set

In the following example, data set information is printed in the SAS log. Control blocks are printed in hexadecimal format. Note that only the first 100 bytes of the JFCB are printed. The example can be used with either a sequential data set or a PDS.

```
filename in 'user.data';
data out;
   infile in jfcb=jf dscb=ds volumes=vol
             ucbname=ucb devtype=dev;
   if (_n_ = 1) then
      put @1 'Data Set Name:' @17 jf  $52.     /
          @4 'Volume ='       @20 vol $30.     /
          @4 'JFCB ='         @20 jf  $hex200. /
          @4 'DSCB ='         @20 ds  $hex188. /
          @4 'Devtype ='      @20 dev $hex48.  /
          @4 'Device Addr ='  @20 ucb $3.      ;
run;
```

Accessing Nonstandard Files

Accessing IMS-DL/I and CA-IDMS Databases

Both the SAS/ACCESS interface to IMS-DL/I and the SAS/ACCESS interface to CA-IDMS include a DATA step interface. Extensions for certain SAS statements (such as INFILE, FILE, PUT, and INPUT) enable you to format database-specific calls in a SAS DATA step. Therefore, you can access the IMS or CA-IDMS data directly, without using SAS/ACCESS view descriptors. If your site licenses these interfaces, see *SAS/ACCESS Interface to IMS-DL/I Software* and *SAS/ACCESS Interface to CA-IDMS Software: Reference* for more information.

Note: The DATA step interface for IMS-DL/I is a "read/write" interface. The DATA step interface for CA-IDMS is "read" only. △

Accessing ISAM Files

To read an ISAM file sequentially, include the ISAM keyword on the INFILE statement as in the following example:

```
data newdata;
   infile isamfile isam;
   input;
   /*  SAS statements  */
run;
```

Accessing VSAM Data Sets

Use the VSAM option to indicate that a fileref points to a VSAM external file.

- To read a VSAM file with an INPUT statement, specify the VSAM option in an INFILE statement:

```
filename in1 'prod.payroll';
data mydata;
   infile in1 vsam;
   input ...;
   /* SAS statements */
run;
```

Note: A VSAM file can be read sequentially without your having to specify the VSAM option. △

- To write to an empty VSAM file with a PUT statement, specify the VSAM option in a FILE statement:

```
filename out 'myid.newdata' disp=old;
data current;
   file out vsam;
   put ...;
   /* SAS statements */
run;
```

- To update a VSAM data set, include an INFILE statement and a FILE statement that point to the same fileref, and specify the VSAM type option in the DATA step:

```
filename mydata 'myid.newdata' disp=old;
data newdata;
   file mydata vsam;
   infile mydata vsam;
   /* SAS statements */
run;
```

Many VSAM-specific options are available with the INFILE and FILE statements. See "VSAM Options for the FILE and INFILE Statements under OS/390" on page 292 for details. For complete information about accessing VSAM data sets, see the *SAS Guide to VSAM Processing*.

Accessing the Volume Table of Contents (VTOC)

To access a disk's Volume Table of Contents (VTOC), specify the VTOC option in an INFILE statement. See "VTOC Options for the INFILE Statement under OS/390" on page 312 for more information.

Accessing UNIX System Services Files

IBM's UNIX System Services implements a directory-based file system that is very similar to the file systems used in UNIX. The SAS System under OS/390 enables you to read and write UNIX System Services files and to pipe data between SAS and UNIX System Services commands.

Allocating UNIX System Services Files

You can allocate a UNIX System Services file either externally (using a JCL DD statement or the TSO ALLOCATE command) or internally (using the SAS FILENAME statement or FILENAME function). For information about allocating UNIX System Services files externally, see your IBM documentation.

There are four ways to specify that a file is in UNIX System Services when you use the FILENAME statement or FILENAME function:

- Include a slash or tilde in the path name:

```
filename input1 '/u/sasusr/data/testset.dat';
filename input2 '~/data/testset2.dat';
```

- Specify HFS (for hierarchical file system) as the file type:

```
filename input hfs 'testset.dat';
```

- Specify HFS as the file prefix:

```
filename input 'HFS:testset.dat';
```

- Rely on the setting of the FILESYSTEM= system option:

```
options filesystem=HFS;
filename 'testset.dat';
```

You can also use these specifications in combination. For example, you can specify the UNIX System Services file type and use a slash in the pathname.

If you do not specify the entire pathname of a UNIX System Services file, then the directory component of the pathname is the working directory that was current when the file was allocated, not when the fileref is used. For example, if your working directory was **/usr/local/sasusr** when you allocated the file, then the following FILENAME statement associates the INPUT fileref with the following path:

```
/usr/local/sasusr/testset.dat
```

```
filename input hfs 'testset.dat';
```

If you change your current working directory to **/usr/local/sasusr/testdata** then the following statement still refers to **/usr/local/sasusr/testset.dat**, not to **/usr/local/sasusr/testdata/testset.dat**:

```
infile input;
```

Allocating a UNIX System Services Directory

To allocate a UNIX System Services directory, create the directory if necessary, then allocate the directory using any standard method, such as a JCL DD statement, a TSO ALLOCATE command, or a FILENAME statement such as those listed in “Allocating UNIX System Services Files” on page 93.

To open a particular file in a directory for input or output, you must specify the file name in the SAS INFILE or FILE statement, as described in “Accessing a Particular File in a UNIX System Services Directory” on page 95.

Specifying File-Access Permissions and Attributes

How you specify file-access permissions and attributes depends on which method you use to allocate a UNIX System Services file:

- When you use a JCL DD statement or a TSO ALLOCATE command to allocate a UNIX System Services file, you can use the PATHMODE and PATHOPTS options to specify file-access permissions and attributes for the file. If you later use the file's DDname in a SAS session, SAS uses the values of those options when it opens the file.

  For example, if you use the following TSO ALLOCATE command to allocate the DDname INDATA and SAS attempts to open it for output, then SAS issues an “insufficient authorization” error message and does not permit the file to be opened for output. (The ORDONLY value of PATHOPTS specifies "open for reading only.")

  ```
  alloc file(indata)
     path('/u/sasusr/data/testset.dat')
     pathopts(ordonly)
  ```

  In other words, you could use the DDname INDATA in a SAS INFILE statement, but not in a FILE statement. Similarly, if you specify OWRONLY, then you can use the DDname in a FILE statement but not in an INFILE statement.

  CAUTION:
  PATHOPTS values OAPPEND and OTRUNC take precedence over FILE statement options OLD and MOD. If you specify OAPPEND ("add new data to the end of the file"), the FILE statement option OLD does not override this behavior. Similarly, if you specify OTRUNC ("if the file exists, erase it and re-create it"), the FILE statement options OLD and MOD do not override this behavior. (See “Standard Host Options for the FILE Statement under OS/390” on page 290 for details about these FILE statement options.) △

- If you use the FILENAME statement or FILENAME function to allocate a UNIX System Services file, or if you use a JCL DD statement or a TSO ALLOCATE command but do not specify values for PATHMODE and PATHOPTS, then SAS uses the following values for those options:
  - For PATHMODE, SAS uses the file-access mode **-rw-rw-rw-**; however, this mode may be modified by the current file-mode creation mask. (For detailed information about the file-mode creation mask, see your IBM documentation.)
  - For PATHOPTS, the file-access mode that SAS supplies depends on how the fileref or DDname is being used:
    - If the fileref or DDname appears only in a FILE statement, SAS opens the file for writing only, and if the file does not exist, SAS creates it.
    - If the fileref appears only in an INFILE statement, SAS opens the file for reading only.

- If the fileref appears in both FILE and INFILE statements within the same DATA step, SAS opens the file for reading and writing. For the FILE statement, SAS also creates the file if it does not already exist.

Using UNIX System Services File Names in SAS Statements and Commands

To use an actual UNIX System Services filename (rather than a fileref or DDname) in a SAS statement or command, include a slash or tilde in the pathname, or use the HFS prefix with the filename. You can use a UNIX System Services file name anywhere that an external file name can be used, such as in a FILE or INFILE statement, in an INCLUDE or FILE command in the windowing environment, or in the SAS Explorer window. If the file is in the current directory, specify the directory component as **./**. For example:

```
include './testprg.sas'
```

Concatenating UNIX System Services Files

To associate a fileref with a concatenation of UNIX System Services files or directories, enclose the pathnames in parentheses. The fileref can be opened only for input. For example:

```
filename test ('data/test1.dat' 'data/test2.dat');
```

All of the pathnames in the concatenation must be for UNIX System Services files or directories. If your program reads data from different types of files in the same DATA step, you can use the EOF= option in each INFILE statement to direct program control to a new INFILE statement after each file has been read. (See *SAS Language Reference: Dictionary* for more information about the EOF= option of the INFILE statement.)

Accessing a Particular File in a UNIX System Services Directory

If you have associated a fileref with a UNIX System Services directory or with a concatenation of UNIX System Services directories, you can open a particular file in the directory for reading or writing by using an INFILE or FILE statement in the form shown below:

```
infile fileref(file);
file   fileref(file);
```

If you do not enclose *file* in quotes, then SAS appends a file extension to the file name. In the windowing environment commands INCLUDE and FILE, the file extension is ".sas". In the INFILE and FILE statements, the file extension is ".dat".

If the file is opened for input, SAS searches all of the directories that are associated with the fileref in the order in which they appear in the FILENAME statement or FILENAME function. If the file is opened for output, SAS creates the file in the first directory that was specified. If the file is opened for updating but does not exist, SAS creates the file in the first directory.

Piping Data between SAS and UNIX System Services Commands

To pipe data between SAS and UNIX System Services commands, you first specify the PIPE file type and the command in a FILENAME statement or FILENAME

function. Enclose the command in single quotes. For example, this FILENAME statement assigns the command **ls -lr** to the fileref OECMD:

```
filename oecmd pipe 'ls -lr';
```

To send the output from the command as input to the SAS System, you then specify the fileref in an INFILE statement. To use output from SAS as input to the command, you specify the fileref in a FILE statement.

You can associate more than one command with a single fileref. Commands are executed in the order in which they appear in the FILENAME statement or FILENAME function. For example:

```
filename oecmd pipe ('ls *.sas' 'ls *.data');
```

Piping Data from a UNIX System Services Command to SAS

When a pipe is opened for input by the INFILE statement, any output that the command writes to standard output or to standard error is available for input. For example, here is a DATA step that reads the output of the **ls -l** command and saves it in a SAS data set:

```
filename oecmd pipe 'ls -l';
data dirlist;
   infile oecmd truncover;
   input mode $ 1-10 nlinks 12-14 user $ 16-23
         group $25-32 size 34-40 lastmod $ 42-53
         name $ 54-253;
run;
```

Piping Data from SAS to an UNIX System Services Command

When a pipe is opened for output by the FILE statement, any lines that are written to the pipe by the PUT statement are sent to the command's standard input. For example, here is a DATA step that uses the UNIX System Services **od** command to write the contents of the file in hexadecimal format to the UNIX System Services file **dat/dump.dat**, as follows:

```
filename oecmd pipe 'od -x -tc - >dat/dump.dat';
data _null_;
   file oecmd;
   input line $ 1-60;
   put line;
cards;
The SAS System is an integrated system of software
products, enabling you to perform data management,
data analysis, and data presentation tasks.
;
run;
```

Host-Specific Options for UNIX System Services Files

Table 6.5 on page 97 shows which host-specific options are recognized by the FILENAME, FILE, and INFILE statements for UNIX System Services files and pipes. No other options are recognized, including such options specific to OS/390 as DISP, CLOSE, and DCB. Descriptions of the options follow the table.

Table 6.5 Host-Specific Options for UNIX System Services Files and Pipes

Option	FILENAME	FILE	INFILE
OLD	X	X	
MOD	X	X	
LRECL=	X	X	X
RECFM=	X	X	X

OLD
replaces the previous contents of the file. This is the default. This option has no effect on a pipe.

MOD
appends the output lines to the file. This option has no effect on a pipe.

LRECL=*value*
specifies the maximum number of characters in a line (unless the file has been opened with RECFM=N). The default is 255. Lines longer than *value* are truncated. *value* must be between 1 and 32,767, inclusive.

RECFM=*record-format*
specifies the record format of the file. Valid values are

F	specifies that all lines in the file have the length specified in the LRECL option. In output files, lines that are shorter than the LRECL value are padded on the right with blanks.
V | D	specifies that the lines in the file are of variable length, ranging from 1 character to the number of characters specified by LRECL=. This is the default.
P	specifies that the file has variable-length records and is in print format.
N	specifies that the file is in binary format. The file is treated as a byte stream; that is, line boundaries are not recognized.

Using the X Statement to Issue UNIX System Services Commands

To start the UNIX System Services shell, issue the following X statement:

```
x omvs;
```

Note: UNIX System Services commands are case sensitive. △

You can also use the X statement to issue any of three UNIX System Services commands:

x cd *directory*;
changes the current working directory to *directory*. If *directory* is omitted, the current working directory is changed to the working directory that was initially assigned to your login name.

x umask *mask*;
changes the current file-mode creation mask value to *mask*. According to UNIX conventions, *mask* is a one- to three-digit octal number. The file-mode creation mask modifies the file mode of new files. Each 1 bit in the file-mode creation mask causes the corresponding permission bit in the file mode to be disabled. If a bit is 0 in the mask, the corresponding file-mode bit can be enabled. For UNIX System Services files that are created by SAS, the file mode for new files is "-rw-rw-rw-"; however, this mode is modified by the current file-mode creation mask. For example, **x umask 022** ensures that each newly created file can be written to only by its owner. (For detailed information about the file-mode creation mask, see your IBM documentation.)

The new value is displayed in the SAS log. If *mask* is not specified, the current value is simply displayed in the SAS log; the current file-mode creation mask value remains unchanged.

x pwd;
displays your current working directory in the SAS log.

To issue other UNIX System Services commands, use the PIPE access method.

To issue a TSO command or CLIST that has the same name as one of the case-sensitive commands (a CLIST named CD, for example), either enter the command using uppercase characters, or use the **TSO:** prefix and enclose the command in quotes, as in the following examples:

```
x CD option1 option2 ...;
x 'tso:cd option1 option2 ...';
```

For more information about the X statement, see “X” on page 323.

Restrictions in SAS System Support for UNIX System Services

It is not possible to run SAS under the UNIX System Services shell. However, you can run the shell after you initialize SAS by using the **x omvs;** statement.

Writing Your Own I/O Access Methods

You can write your own I/O access method to replace the default SAS access method. This feature enables you to redirect external file I/O to a user-written program.

Note: The user-written I/O access method applies only to external files, not to SAS data sets. △

See your local SAS Support Consultant for additional information about writing I/O access methods.

Accessing SAS Statements from a Program

You can redirect your SAS statements to come from an external program rather than from a file by using the SYSINP= and PGMPARM= system options. SYSINP= specifies the name of the program, and PGMPARM= specifies a parameter that is passed to the program. For more information, see “SYSINP=” on page 412 and “PGMPARM=” on page 387.

Using the INFILE/FILE User Exit Facility

User exit modules enable you to inspect, modify, delete, or insert records in a DATA step. Here are some examples of how they may be used:

- encrypting and decrypting data
- compressing and decompressing data
- translating data from one character-encoding system to another.

This is an advanced topic. See “Sample Program” on page 471 for details.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OS/390 Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS® Companion for the OS/390® Environment, Verison 8

1–58025–523–X

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, November 1999