CHAPTER

4

Using SAS Files

Introduction to SAS Files, Data Libraries, and Engines **83**
Types of SAS Files **84**
SAS Data Files (Member Type DATA) **85**
SAS Data Views (Member Type VIEW) **85**
Filename Extensions and Member Types **85**
Accessing SAS Data Files **86**
Specifying Pathnames **87**
Assigning Librefs with the LIBNAME Statement **88**
Engine/Host Options **91**
Omitting Engine Names From the LIBNAME Statement **91**
Assigning a Libref to Several Directories (Concatenating Directories) **91**
Understanding How Concatenated SAS Data Libraries Are Accessed **92**
Accessing Files for Input and Update **92**
Accessing Files for Output **92**
Accessing Data Sets with the Same Name **93**
Using Multiple Engines for a Library **93**
Using Environment Variables as Librefs **93**
Librefs Assigned by SAS **94**
SASUSER Data Library **94**
WORK Data Library **95**
USER Data Library **95**
Accessing Disk-Format Data Libraries **96**
Accessing Sequential-Format Data Libraries **96**
Reading and Writing SAS Files on Tape **96**
Reading and Writing Transport Formats on Tape **97**
Writing Sequential Data Sets to Named Pipes **97**
Accessing BMDP, OSIRIS, or SPSS Data Files **98**
The BMDP Engine **98**
The OSIRIS Engine **99**
The SPSS Engine **100**
Accessing SAS Files from Previous Releases or from Other Hosts **100**
Sharing Files **101**
Sharing Files in a Network **101**

Introduction to SAS Files, Data Libraries, and Engines

Your data can reside in different types of files, including SAS files and files that are formatted by other software products, such as database management systems. Under UNIX, a SAS file is a specially structured UNIX file. Although the UNIX operating environment manages the file for the SAS System by storing it, the operating system

cannot process it because of the structure built into the file by the SAS System. For example, you can list the filename with the **ls** command, but you cannot use the **vi** editor to edit the file. A SAS file can be permanent or temporary.

SAS files are stored in *SAS data libraries*. A SAS data library is a collection of SAS files within a UNIX directory. Any UNIX directory can be used as a SAS data library. (The directory can also contain files called *external files* that not managed by the SAS System. See Chapter 5, "Using External Files and Devices," on page 103 for how to access external files.) The SAS System stores temporary SAS files in a WORK data library (see "WORK Data Library" on page 95), which is automatically defined for you. You must specify a data library for each permanent SAS file.

SAS files and SAS data libraries are accessed through *engines*. An engine is set of routines that the SAS System must use to access the files in the data library. The SAS System can read from and, in some cases, write to the file by using the engine that is appropriate for that file type. For some file types, you need to tell the SAS System which engine to use. For others, the SAS System automatically chooses the appropriate engine. The engine that is used to create a SAS data set determines the format of the file.

SAS data libraries can be identified with *librefs*. A libref is a name by which you reference the file in your application. You can assign librefs by using the LIBNAME statement, by defining an environment variable, or through the New Library window.

Types of SAS Files

Different types of SAS files serve different functions. There are four different types of SAS files.

Data sets
: consist of descriptor information and data values organized as a table of rows and columns that can be processed by one of the engines. The descriptor information includes data set type, data set label, the names and labels of the columns in the data set, and so on. A SAS data set can also include *indexes* for one or more columns.

  SAS data sets are implemented in two forms:

  - If the data values and the data set's descriptor information are stored in one file, the SAS data set is called a *SAS data file*.
  - If the file simply contains information about where to obtain a data set's data values and descriptor information, the SAS data set is called a *SAS data view*.

  The default engine processes the data set as if the data file or data view and the indexes were a single entity.

  For more information, see "SAS Data Files (Member Type DATA)" on page 85 and "SAS Data Views (Member Type VIEW)" on page 85.

Catalogs
: are a special type of SAS file that can contain multiple entries. Many different types of entries can be kept in the same SAS catalog. For example, catalogs can contain entries created by SAS/AF and SAS/FSP software, windowing applications, key definitions, SAS/GRAPH graphs, and so on.

Stored program files
: are compiled DATA steps generated by the Stored Program Facility. For details on the Stored Program Facility, see *SAS Language Reference: Dictionary*.

Access descriptor files
describe the data formatted by other software products such as the ORACLE or the SYBASE database management systems. Descriptor files created by the ACCESS procedure in SAS/ACCESS software have the SAS member type of ACCESS.

SAS Data Files (Member Type DATA)

The SAS data file is probably the most frequently used type of SAS file. SAS data files are created in the DATA step and by some SAS procedures. There are two types of data files:

- *Native data files* store data values and their descriptor information in files formatted by the SAS System. These are the traditional SAS data sets familiar from previous versions of the SAS System.

  Native SAS data files created by the default engine can be indexed. An *index* is an auxiliary file created in addition to the data file it indexes. The index provides fast access to observations within a SAS data file by a variable or key. Under UNIX, indexes are stored as separate files but are treated as integral parts of the SAS data file by the SAS System.

  CAUTION:
  Do not remove index files using UNIX commands. Removing the index file can damage your SAS data set. Also, do not change its name or move it to a different directory. Use the DATASETS procedure to manage indexes. △

- *Interface data files* store data in files that have been formatted by other software and that the SAS System can only read. See "Accessing BMDP, OSIRIS, or SPSS Data Files" on page 98 for more information.

SAS Data Views (Member Type VIEW)

A SAS data view contains only the information needed to derive the data values and the descriptor information. Depending on how the SAS data view is created, the actual data can be in other SAS data sets or in other vendors' files.

Views can be of two kinds:

- *Native SAS data views* contain information about data in one or more SAS data files or SAS data views. This type of view is created with the SQL procedure or DATA step.
- *Interface SAS data views* contain information about data formatted by other software products, for example, a database management system. The ACCESS procedure in SAS/ACCESS software creates such a view.

Filename Extensions and Member Types

Because the SAS System needs to distinguish between the different file types, it automatically assigns a certain extension to each file when it creates the file. Also, since a SAS file is a member of a SAS data library, the system assigns each file a SAS member type.

Table 4.1 on page 86 lists the file extensions and their corresponding SAS member types.

CAUTION:
Do not change the file extensions of SAS files. File extensions determine how the SAS System accesses files; changing them can cause unpredictable results. △

Table 4.1 File Extensions for SAS File Types

Version 6		Version 7 and Version 8			
Random Access Files	**Sequential Access Files**	**Random Access Files**	**Sequential Access Files**	**SAS Member Type**	**Description**
.sas	.sas	.sas	.sas	.sas	SAS program
.lst	.lst	.lst	.lst	.lst	Procedure output
.log	.log	.log	.log	.log	SAS log file
.ssd*nn*[1]	.sdq*nn*	.sas7bdat	.sas7sdat	DATA	SAS data file
.snx*nn*	.siq*nn*	.sas7bndx	.sas7sndx	INDEX	Data file index; not treated by the SAS System as a separate file
.sct*nn*	.scq*nn*	.sas7bcat	.sas7scat	CATALOG	SAS catalog
.ssp*nn*	.ssq*nn*	.sas7bpgm	.sas7spgm	PROGRAM	Stored program (DATA step)
.ssv*nn*	.svq*nn*	.sas7bvew	.sas7svew	VIEW	SAS data view
.ssa*nn*	.saq*nn*	.sas7bacs	.sas7sacs	ACCESS	Access descriptor file
.sst*nn*	.stq*nn*	.sas7baud	.sas7saud	AUDIT	Audit file
.sfd*nn*	.sfq*nn*	.sas7bfdb	.sas7sfdb	FDB	Consolidation database
.ssm*nn*	.smq*nn*	.sas7bmdb	.sas7smdb	MDDB	Multi-dimensional database
.sds*nn*	.soq*nn*	.sas7bods	.sas7sods	SASODS	Output delivery system file
.snm*nn*	.sqn*nn*	.sas7bdmd	.sas7sdmd	DMDB	Data mining database
.sit*nn*	.srq*nn*	.sas7bitm	.sas7ssitm	ITEMSTOR	Item store file
.sut*nn*	.suq*nn*	.sas7butl	.sas7sutl	UTILITY	Utility file
.spu*nn*	.spq*nn*	.sas7bput	.sas7sput	PUTILITY	Permanent utility file
.ssb*nn*	.sbq*nn*	.sas7bbak	.sas7sbak	BACKUP	Backup file

1 All Version 6 files end with a two-character code (*nn*) that identifies sets of compatible SAS files. See "Sharing Files" on page 101 for more information.

A UNIX directory can store a variety of files, but you might find it more practical to store files in separate directories according to their use. Also, you can keep libraries that are accessed by different engines in the same directory, but this is not recommended. See "Using Multiple Engines for a Library" on page 93 for more information.

Accessing SAS Data Files

If you want to read or write to a permanent SAS file, you can refer to the SAS file in one of two ways:

- refer to the data file directly by using its pathname in the appropriate statements (such as DATA, SET, MERGE, UPDATE, OUTPUT, and PROC).
- assign a libref to the SAS data library (directory) that contains the data file and use the libref as the first level of a two-level file name.

A libref is a nickname that you can use to refer to the data library during the SAS session or job. You will probably want to use a libref when:

- the data file pathname is long and must be specified several times within a program
- the pathname might change. If the pathname changes, you need to change only the statement assigning the libref, not every reference to the file.
- your application will be used on other platforms. Using librefs makes it easier to port an application to other operating environments.
- you need to concatenate libraries. See "Assigning a Libref to Several Directories (Concatenating Directories)" on page 91 for more information.

You can assign a libref with the LIBNAME statement or through the New Library window in the Explorer. To open the New Library window, select

File ► New ► Library

You can also use an environment variable as the libref.

After you have defined a libref, you can use the libref in one of two ways to access a permanent SAS data library:

- as the first level of a two-level SAS file name:

  libref.member-name

  where *libref* is the first-level name referring to the directory where the file is stored, and *member-name* specifies the name of the file being read or created.
- as the value of the USER= option. (See "USER Data Library" on page 95 for details.)

For example, these SAS statements access the data file FINAL.DATA in the library in the the directory **/users/myid/mydir**:

```
libname sales '/users/myid/mydir';
data sales.final;
```

Specifying Pathnames

Whether you specify a data file name directly in the various SAS statements or specify the data library name in a LIBNAME statement and then refer to the libref, the same rules apply for specifying UNIX directory and file pathnames.

Specify directory and file pathnames in quotes. The level of specification depends on your current directory. For example, if **/u/1999/budgets** is not your current directory, then to access the data file named **may**, you must specify the entire pathname:

```
data '/u/1999/budgets/may';
```

If you wanted to use a libref, you would specify:

```
libname budgets '/u/1999/budgets';
data budgets.may;
```

If **/u/1999/budgets** is your current directory, you could specify only the data file names:

```
data 'quarter1';
merge 'jan' 'feb' 'mar';
run;
```

If you wanted to use a libref, you would specify:

```
libname budgets '.';
data budgets.quarter1;
merge budgets.jan budgets.feb budgets.mar;
run;
```

You can use the character substitutions shown in Table 4.2 on page 88 to specify pathnames.

Table 4.2 Character Substitutions in Pathnames

Characters	Meaning
~/	$HOME/ Can be used only at the beginning of a pathname.
~*name*/	*name*'s home directory (taken from file **/etc/passwd**). Can be used only at the beginning of a pathname.
!sasroot	name of **sasroot** directory (see Appendix 1, "The sasroot Directory," on page 317). Specified only at the beginning of a pathname.
.	current working directory
..	parent of current working directory
$*VARIABLE*	environment variable *VARIABLE*

Assigning Librefs with the LIBNAME Statement

Use the LIBNAME statement to associate a libref with a SAS data library. The general form of the LIBNAME statement is

LIBNAME *libref* <*engine*> '*SAS-data-library*' <*options*> <*engine/host-options*>;

LIBNAME *libref* <*engine*> ('*library-1*<,...'*library-n*'>) <*options*>;

LIBNAME *libref* ('*library-1*' | *libref- 1* ... '*library-n*' | *libref-n*);

LIBNAME *libref* CLEAR | _ALL_ CLEAR;

LIBNAME *libref* LIST| _ALL_ LIST;

libref
is any valid libref as doc umented in *SAS Language Reference: Dictionary*.
The SAS System reserves some librefs for special system libraries. See "Librefs Assigned by SAS" on page 94 for more information.

engine
is one of the library engines supported under UNIX. "Details" on page 245 describes the two types of engines: *library engines* and *view engines*. See Table 4.3 on page 90 for engine names and descriptions. If no engine name is specified, the SAS System determines which engine to use as described in "Omitting Engine Names From the LIBNAME Statement" on page 91.

'*SAS-data-library*'
differs according to the engine name that you specify and according to your current working directory. Table 4.3 on page 90 describes what each engine

expects for this argument. Specify directory pathnames as described in "Specifying Pathnames" on page 87. You cannot create directories with the LIBNAME statement. The directory that you specify must already exist. Enclose the data library name in quotes. Remember that UNIX filenames are case-sensitive.

'library-n' | libref-n
are pathnames or librefs (that have already been assigned) for the data libraries that you want to access with the same libref. Use these forms of the LIBNAME statement when you want to concatenate data libraries. Separate the pathnames with either commas or blank spaces. Enclose library pathnames in quotes. Do not enclose librefs in quotes. See "Assigning a Libref to Several Directories (Concatenating Directories)" on page 91 for more information.

options
are LIBNAME statement options that are available in all operating environments. See *SAS Language Reference: Dictionary* for information about these options.

engine/host-options
can be any of the options described in "Engine/Host Options" on page 91.

ALL
refers to all librefs currently defined. You can use this keyword when you are listing or clearing librefs.

CLEAR
clears the specified libref or, if you specify _ALL_, clears all librefs that are currently defined. SASUSER, SASHELP, and WORK remain assigned.

Note: When you clear a libref defined by an environment variable, the variable remains defined, but it is no longer considered a libref. You can still reuse it, either as a libref or a fileref. See "Using Environment Variables as Librefs" on page 93 for more information. △

The SAS System automatically clears the association between librefs and their respective data libraries at the end of your job or session. If you want to associate an existing libref with a different SAS data library during the current session, you do not have to end the session or clear the libref. The SAS System automatically reassigns the libref when you issue a LIBNAME statement for the new SAS data library.

LIST
prints to the SAS log the engine, pathname, file format, access permissions, and so on, that are associated with the specified libref or, if you specify _ALL_, prints this information for all librefs that are currently defined. Librefs defined as environment variables appear only if you have already used those librefs in a SAS statement.

The association between the libref and the SAS data library lasts as long as the SAS job or session, unless you use the LIBNAME statement either to clear the association or to associate the libref with another SAS data library.

Table 4.3 on page 90 describes each of the engines that you can specify in the LIBNAME statement and tells what each engine expects for the *SAS-data-library* argument.

Table 4.3 Engine Names and Descriptions

Engine Type	Name (alias)	Description	SAS-data-library
default	V8 (BASE) V7	enables you to create new SAS data files and access existing SAS data files that were created with Version 7 or Version 8. The V7 and V8 engines are identical. This engine enables read access to data files created with some earlier versions of SAS, but this engine is the only one that supports Version 7 and Version 8 catalogs. This engine allows for data set indexing and compression and is also documented in *SAS Language Reference: Dictionary*.	is the pathname of the directory containing the library.
sequential	V8TAPE (TAPE) V7TAPE V6TAPE	accesses SAS data files that were created in a sequential format, whether on tape or on disk. This engine requires less overhead than the default engine because sequential access is simpler than random access. This engine is also documented in *SAS Language Reference: Dictionary*.	is the name of the *special file* (see "Introduction to External Files and Devices" on page 103) that is associated with the sequential device, such as **`/dev/rmt/0mn`**.
compatibility	V6	accesses any data file created by Release 6.09 through 6.12.	is the pathname of the directory containing the library.
servers	SPDS	enables communication between a client session and a data server. You must have the Scalable Performance Data Server licensed on your client machine to use this engine. Refer to *Scalable Performance Data Server User's Guide, Version 2* for more information.	is the logical LIBNAME domain name for an SPDS data library on the server machine. The name server resolves the domain name into the physical path for the library.
	MDDB	enables communication between a client session and an MDDB server. You must have SAS/MDDB Server licensed either or your client machine or on your server machine to use this engine. Refer to *SAS MDDB Server Software: Administration Guide* for complete information.	
transport	XPORT	accesses transport data sets. This engine creates machine-independent SAS transport files that can be used under all hosts running Release 6.09 or later of the SAS System. This engine is documented in *Moving and Accessing SAS Files across Operating Environments*.	is the pathname of either a sequential device or a disk file.
interface	BMDP	provides read-only access to BMDP files. This engine is available only on AIX, HP-UX, and Solaris.	is the pathname of the data file.

Engine Type	Name (alias)	Description	SAS-data-library
	OSIRIS	provides read-only access to OSIRIS files.	is the pathname of the data file.
	SPSS	provides read-only access to SPSS files	is the pathname of the data file.

Engine/Host Options

The LIBNAME statement accepts the FILELOCKS option:

FILELOCKS=NONE | FAIL | CONTINUE

This option specifies whether file locking is on or off for the library that you are defining. This LIBNAME statement option works like the FILELOCKS system option, except that it applies only to the library that you are defining. See "FILELOCKS" on page 266 for more information.

You can also specify any of the options supported by the SPDS engines. SPDS is the Scalable Performance Data Server. Refer to *Scalable Performance Data Server User's Guide, Version 2* for a description of these options.

Omitting Engine Names From the LIBNAME Statement

It is always more efficient to specify the engine name than to have the SAS System determine the correct engine. However, if you omit an engine name in the LIBNAME statement or if you define an environment variable to serve as a libref, the SAS System determines the appropriate engine.

If you have specified the ENGINE= system option, SAS uses the engine name that you specified. See "ENGINE" on page 265 for a discussion of the ENGINE= system option.

Note: The ENGINE= system option specifies the default engine for data libraries on disk only. △

If you did not specify the ENGINE= system option, SAS looks at the extensions of the files in the given directory and uses these rules to determine an engine:

- If all the SAS data sets in the library were created by the same engine, the libref is assigned to that engine.
- If there are no SAS data sets in the given directory, the libref is assigned to the default engine.
- If there are SAS data sets from more than one engine, the system issues a message about finding mixed engine types and assigns the libref to the default engine.

Assigning a Libref to Several Directories (Concatenating Directories)

You can use the LIBNAME statement to assign librefs and engines to one or more directories, including the working directory.

If you have SAS data sets located in multiple directories, you can treat these directories as a single SAS data library by specifying a single libref and concatenating the directory locations, as in the following example:

```
libname income ('revenue','costs');
```

This statement indicates that the two directories, **revenue** and **costs**, are to be treated as a single SAS data library.

If you have already assigned librefs to your SAS data libraries, you can use these librefs to indicate that you want to concatenate the data libraries, as in this example:

```
libname income ('corpsale','retail');
libname costs ('salaries','expenses');
libname profits (income,'capgain',costs);
```

This statement indicates that the five directories, **corpsale**, **retail**, **salaries**, **expenses**, and **capgain**, are to be treated as a single SAS data library.

When you concatenate SAS data libraries, the SAS System uses a protocol for accessing the libraries, which depends on whether you are accessing the libraries for read, write, or update. (A *protocol* is a set of rules.) See "Understanding How Concatenated SAS Data Libraries Are Accessed" on page 92 for more information.

See *SAS Language Reference: Dictionary* for complete documentation on the LIBNAME statement.

Understanding How Concatenated SAS Data Libraries Are Accessed

When you use the concatenation feature to specify more than one physical directory for a libref, the SAS System uses the protocol shown in the following sections to determine which directory is accessed. (The protocol illustrated by these examples applies to all SAS statements and procedures that access SAS files, such as the DATA, UPDATE, and MODIFY statements in the DATA step, and the SQL and APPEND procedures.)

Accessing Files for Input and Update

When a SAS data set is accessed for input or update, the first SAS data set that is found by that name is the one that is accessed. For example, if you submit the following statements and the data set OLD.SPECIES exists in both directories, the one in the **mysasdir** directory is the one that is printed:

```
libname old ('mysasdir','saslib');
proc print data=old.species;
run;
```

The same would be true if you opened OLD.SPECIES for update with the FSEDIT procedure.

Accessing Files for Output

If the data set is accessed for output, it is always written to the first directory, provided that the directory exists. If the directory does not exist, an error message is displayed. For example, if you submit the following statements, the SAS System writes the OLD.SPECIES data set to the first directory (**mysasdir**) and replaces any existing data set with the same name:

```
libname old ('mysasdir','saslib');
data old.species;
x=1;
y=2;
run;
```

If a copy of the OLD.SPECIES data set exists in the second directory, it is not replaced.

Accessing Data Sets with the Same Name

If you use the DATA and SET statements to access data sets with the same name, the DATA statement uses the output rules and the SET statement uses the input rules. For example, suppose you submit the following statements and TEST.SPECIES originally exists only in the second directory, **mysasdir**:

```
libname test ('sas','mysasdir');
data test.species;
set test.species;
if value1='y' then
   value2=3;
run;
```

The DATA statement opens TEST.SPECIES for output according to the output rules; that is, the SAS System opens a data set in the first of the concatenated libraries (**sas**). The SET statement opens the existing TEST.SPECIES data set in the second (**mysasdir**) directory, according to the input rules. Therefore, the original TEST.SPECIES data set is not updated. After the data step executes, two TEST.SPECIES data sets exist, one in each directory.

Using Multiple Engines for a Library

You can assign multiple librefs to a single directory, and specify a different engine with each libref. For example, after the following statements are executed, data sets that are referenced by ONE are created and accessed using the default engine, while data sets that are referenced by TWO are created and accessed using the sequential engine:

```
libname one v7 '/users/myid/educ';
libname two tape '/users/myid/educ';
```

Note: Keeping different types of data libraries in one directory is not recommended because you must remember the appropriate engine for accessing each library. SAS cannot determine the right engine for accessing libraries in a directory that contains libraries of different types. See "Omitting Engine Names From the LIBNAME Statement" on page 91 for more information. △

Using Environment Variables as Librefs

An environment variable can also be used as a libref. The variable name must be in all upper case characters, and the variable value must be the full pathname of the directory, that is, the name of the directory must begin with a slash.

Suppose you want to use the data library in **/users/mydir/educ**, and you want to refer to it with the EDUC environment variable. You can define the variable at two times:

- before you invoke the SAS System. See "Defining Environment Variables" on page 17. For example, in the Korn shell, you would use

  ```
  export EDUC=/users/mydir/educ
  ```

- after you invoke the SAS System by using the X statement (see "Executing Operating System Commands from Your SAS Session" on page 12) and the SAS **setenv** command:

```
x setenv EDUC /users/mydir/educ;
```

You cannot specify an engine when you define a libref as an environment variable, so the SAS System determines which engine to use as described in "Omitting Engine Names From the LIBNAME Statement" on page 91.

After the libref is defined, you can use it to access data sets stored in the library:

```
proc print data=educ.class;
run;
```

Note: If a variable and a libref have the same name but refer to different files, the SAS System uses the libref. △

Librefs Assigned by SAS

The SAS System automatically defines four librefs:

SASHELP
: contains a group of catalogs that contain information that is used to control various aspects of your SAS session. The SASHELP library is in the sasroot directory. See Appendix 1, "The sasroot Directory," on page 317.

SAUSER
: contains SAS catalogs that enable you to tailor features of the SAS System for your needs. If the defaults in the SASHELP library are not suitable for your applications, you can modify them and store your personalized defaults in your SASUSER library.

USER
: allows you to read, create, and write files in a SAS data library other than WORK without specifying a libref as part of the SAS file name.

WORK
: is the temporary, or scratch, library automatically defined by the SAS System at the beginning of each SAS session or job. The WORK library stores two types of temporary files: those you create and those created internally by the SAS System as part of normal processing.

These librefs and the LIBRARY libref are reserved librefs. If your site also has SAS/GRAPH software or SAS/GIS software, the MAPS or GISMAPS librefs might also be automatically defined. All these libraries are described in *SAS Language Reference: Dictionary*. SASUSER, USER, and WORK have operating system dependencies.

SASUSER Data Library

When you invoke the SAS System, it looks for one special directory in which to store a data library with the SASUSER libref. If this directory does not exist, the SAS System uses the SASUSER system option to create it. The configuration file usually specifies the directory as follows:

```
-sasuser ~/sasuser
```

This specification tells the SAS System to create the SASUSER library in the **sasuser** subdirectory of your home directory. You can permit read-only access to the SASUSER

library by using the RSASUSER system option. See Chapter 17, "SAS System Options," on page 253 for details on the SASUSER and RSASUSER system options.

Once the SASUSER library has been created, the SAS System automatically assigns the same libref to it each time you start a SAS session. It cannot be cleared or reassigned during a SAS session. If you delete the library, the SAS System re-creates it the next time you start a session.

The SAS System stores your user profile in the SASUSER data library. Your SASUSER.PROFILE catalog contains the tailoring features you specify for the SAS System. By default, this information is taken from the SASHELP library. When you save changes to function key definitions, window attributes, and other information from SAS sessions, the SAS System stores the changes in the SASUSER.PROFILE catalog.

You can, of course, store other data sets and catalogs there as well. Because the SAS System assigns the libref for you, you do not need to use a LIBNAME statement before referencing this library.

WORK Data Library

The WORK data library is the temporary library that is automatically defined by the SAS System at the beginning of each SAS session or job. The WORK data library stores temporary SAS files that you create as well as files created internally by the SAS System.

To access files in the WORK data library, simply specify a one-level name for the file. The libref WORK is automatically assigned to these files unless you have assigned the USER libref.

When you invoke the SAS System, it assigns the WORK libref to a subdirectory of the directory specified in the WORK system option described in Chapter 17, "SAS System Options," on page 253. This subdirectory is usually named **SAS_work*code*** where ***code*** is a 12–character code based on the process ID of the SAS session. This libref cannot be cleared or reassigned during a SAS session.

The WORKINIT and WORKTERM system options control the creation and deletion of the WORK data library. See *SAS Language Reference: Dictionary* for details.

Note: If a SAS session is terminated improperly (for example, using the **kill -9** command), the SAS System will not delete the **SAS_work*code*** directory. You may want to use the **cleanwork** command to delete these straggling directories (see Appendix 2, "Tools for the System Administrator," on page 319). △

USER Data Library

SAS data sets are referenced with a one- or two-level name. The two-level name is of the form *libref.member-name* where *libref* refers to the SAS data library in which the data set resides and *member-name* refers to the particular *member* within that library. The one-level name is of the form *member-name* (without a *libref*). In this case, the SAS System stores the files in the temporary WORK data library. To override this action and have files with one-level names stored in a permanent library, first assign the USER libref to an existing directory. To refer to temporary SAS files while USER is assigned, use a two-level name with WORK as the libref. You have three ways to assign the USER libref:

- Assign the USER libref directly using the LIBNAME statement:

```
libname user '/users/myid/mydir';
```

- Specify the USER system option before you start the session. For example, you can assign the USER libref when you invoke the SAS System:

```
sas -user /users/myid/mydir
```

- Specify the USER= system option after you start the session. First, assign a libref to the permanent library. Then use the USER= system option in an OPTIONS statement to equate that libref to USER. For example, these statements assign the libref USER to the directory with libref MINE:

```
libname mine '/users/myid/mydir';
options user=mine;
```

See Chapter 17, "SAS System Options," on page 253 for details on the USER system option.

Accessing Disk-Format Data Libraries

You will probably create and access data libraries on disk more than any other type of library. The default engine and the compatibility engines allow read, write, and update access to SAS files on disk. They also support indexing and compression.

In the following example, the IN libref is assigned to a directory that contains the STATS1 data set:

```
libname in '/users/myid/myappl';
proc print data=in.stats1;
run;
```

Remember, when the LIBNAME statement is issued, the *SAS-data-library* that it specifies must already exist. For example, if you want to create the SAS data set ORDERS in a directory, use the X statement to issue the **mkdir** UNIX command, and then use the LIBNAME statement to associate the libref with the directory.

```
x mkdir /users/publish/books;
libname books '/users/publish/books';
data books.orders;
  more SAS statements
run;
```

By default, the LIBNAME statement associates the V8 engine with the directory.

Accessing Sequential-Format Data Libraries

The sequential engines enable you to access data libraries in sequential format on tape or disk. The sequential engines do not support indexing and compression of observations.

Note: Before using sequential engines, read the information about sequential data libraries in *SAS Language Reference: Dictionary*. △

Reading and Writing SAS Files on Tape

You can write SAS files directly to tape using the TAPE engine; however, it is more efficient to use a staging directory so that the files can be processed directly from disk. You can use the UNIX **tar** command to move SAS data sets between the staging directory and tape. (Do not use the UNIX **cp** command.)

A SAS library on tape can contain one or more SAS data sets; however, only one SAS data set from a given library on tape can be accessed at any given point in a SAS job.

To access Version 8 SAS files on tape, you can specify the V8TAPE or TAPE engine in the LIBNAME statement:

LIBNAME *libref* V8TAPE '*tape-device-pathname*';

The *tape-device-pathname* must be a pathname for a tape device; it should be the name of the special file associated with the tape device. (Check with your system administrator for details.) The name must be enclosed in quotes. You cannot specify remote tape devices in the LIBNAME statement.

Multi-volume tape libraries are supported if you specify the TAPECLOSE=LEAVE system option when you start your SAS session.

For example, the following LIBNAME statement assigns the libref SEQ2 to the **/dev/tape2** tape device. Because the tape device is specified, the engine does not have to be specified.

```
libname seq2 '/dev/tape2';
```

Reading and Writing Transport Formats on Tape

Transport formats on tape are handled in a manner similar to external files. Read "Processing Files on TAPE" on page 121 before continuing with this topic.

For example, the following SAS statements issue the UNIX **mt** command to rewind the tape and create a transport file using the xport engine and PROC CPORT:

```
x 'mt -t /dev/rmt/0mn rewind';
libname tranfile xport '/dev/rmt/0mn';
proc cport library=sasuser file=tranfile;
run;
```

The following statements import the transport file into the WORK data library:

```
x 'mt -t /dev/rmt/0mn rewind';
libname tranfile xport '/dev/rmt/0mn';
proc cimport infile=tranfile library=work;
run;
```

Writing Sequential Data Sets to Named Pipes

You can send output to and read input from the operating system by using named pipes. For example, you may want to compress a data set or send it to a tape management system without creating intermediate files.

You can read from and write to named pipes from within your SAS session by specifying the pipe name in the LIBNAME statement:

LIBNAME *libref* <TAPE> '*pipename*';

Since you cannot position a pipe file, SAS uses the TAPE engine to ensure sequential access. You do not have to specify the engine name; TAPE is assumed.

For example, suppose you want to create a SAS data set and compress the data set without creating an intermediate, uncompressed data set. Create a named pipe (such as **mypipe**) and enter the **compress** command:

```
mknod mypipe p compress <mypipe >sasds.Z
```

In your SAS session, assign a libref to the pipe and begin writing to the data set:

```
libname x 'mypipe';
data x.a;
  ...more SAS statements...
output;
run;
```

The data is sent to **mypipe**, compressed, and written to the data set. When SAS closes the data set, the compress finishes, and you have a compressed, sequential data set in **sasds.Z**.

If you begin writing to a named pipe before the task on the other end (in this case, the **compress** command) begins reading, your SAS session will be suspended until the task begins to read.

Accessing BMDP, OSIRIS, or SPSS Data Files

f Version 7 of the SAS System includes three interface library engines, BMDP, OSIRIS and SPSS, that enable you to access external data directly from a SAS program. All these engines are read-only.

Because they are sequential, these engines cannot be used with the POINT= option on the SET statement or with the FSBROWSE, FSEDIT, or FSVIEW procedures. You can use PROC COPY, PROC DATASETS, or a DATA step to copy the system file to a SAS data set and then perform these functions on the SAS data set. Also, some procedures (such as PROC PRINT) give a warning message about the engine being sequential.

With these engines, the physical filename associated with a libref is an actual filename, not a directory. This is an exception to the rules concerning librefs.

You can also use the CONVERT procedure to convert BMDP, OSIRIS and SPSS files to SAS data files. See “CONVERT” on page 225 for more information.

The BMDP Engine

The BMDP interface library engine enables you to read BMDP files from the BMDP statistical software package directly from a SAS program. The BMDP engine is a read-only engine. The following discussion assumes you are familiar with the BMDP save file terminology.*

Note: This engine is available for AIX, HP-UX, and Solaris. △

To read a BMDP save file, issue a LIBNAME statement that explicitly specifies the BMDP engine. In this case, the LIBNAME statement takes this form:

LIBNAME *libref* BMDP *'filename'*;

In this form of the LIBNAME statement, *libref* is a SAS libref and *filename* is the BMDP physical filename. If the libref appears previously as a fileref, omit *filename* because the physical filename associated with the fileref is used. This engine can only read save files created under UNIX.

Because there can be multiple save files in a single physical file, you reference the CODE= value as the member name of the data set within the SAS language. For example, if the save file contains CODE=ABC and CODE=DEF and the libref is MYLIB, you reference them as MYLIB.ABC and MYLIB.DEF. All CONTENT types are treated the same, so even if member DEF is CONTENT=CORR, it is treated as CONTENT=DATA.

* See the documentation provided by BMDP Statistical Software Inc. for more information.

If you know that you want to access the first save file in the physical file or if there is only one save file, refer to the member name as _FIRST_. This is convenient if you do not happen to know the CODE= value.

For example, assume that the physical file MYBMDP.DAT contains the save file ABC. The following SAS code associates the libref MYLIB with the BMDP physical file and runs the CONTENTS and PRINT procedures on the save file:

```
libname mylib bmdp 'mybmdp.dat';
proc contents data=mylib.abc;
run;
proc print data=mylib.abc;
run;
```

The following example uses the LIBNAME statement to associate the libref MYLIB2 with the BMDP physical file. Then it prints the data for the first save file in the physical file:

```
libname mylib2 bmdp 'mybmdp.dat';
proc print data=mylib2._first_;
run;
```

The OSIRIS Engine

The Inter-University Consortium on Policy and Social Research (ICPSR) uses the OSIRIS file format for distribution of its data files. The SAS System provides the OSIRIS interface library engine to support the many users of the ICPSR data and to be compatible with PROC CONVERT.

The OSIRIS engine allows you to read OSIRIS data and dictionary files directly from a SAS program. The following discussion assumes you are familiar with the OSIRIS file terminology and structure. If you are not, refer to the documentation provided by the ICPSR.

To read an OSIRIS file, issue a LIBNAME statement that explicitly specifies the OSIRIS engine. The syntax of the LIBNAME statement in this case is

LIBNAME *libref* OSIRIS '*data-filename*' DICT='*dictionary-filename*';

libref
is a SAS libref.

'*data-filename*'
is the physical filename of the data file. If the libref appears also as a fileref, omit the data filename.

'*dictionary-filename*'
is the physical filename of the dictionary file. The dictionary filename can also be an environment variable or a fileref, but if it is either of those, do not enclose it in quotes. The DICT= option is required.

OSIRIS data files do not have member names. Therefore, use whatever member name you like.

To use the same dictionary file with different data files, code a separate LIBNAME statement for each one.

Since the OSIRIS software does not run outside the MVS environment, the layout of an OSIRIS data dictionary is consistent across operating environments. However, the OSIRIS engine is designed to accept a data dictionary from any other operating environment on which the SAS System runs. It is important that the dictionary and data files not be converted from EBCDIC to ASCII; the engine expects EBCDIC data.

The dictionary file should consist of fixed-length records of length 80. The data file should contain records large enough to hold the data described in the dictionary.

In the following example, the data file is **/users/myid/osr/dat**, and the dictionary file is **/users/myid/osr/dic**. The example associates the libref MYLIB with the OSIRIS files and runs a PROC CONTENTS and PROC PRINT on the data.

```
libname mylib osiris '/users/myid/osr/dat'
   dict='/users/myid/osr/dic';
proc contents data=mylib._first_;
run;
proc print data=mylib._first_;
run;
```

The SPSS Engine

The SPSS interface library engine allows you to read only SPSS export files. This engine does not read Release 9 and SPSS-X native files. The following discussion assumes that you are familiar with the SPSS save file terminology and structure. If you are not, refer to the documentation provided by SPSS.

To read an SPSS export file, issue a LIBNAME statement that explicitly specifies the SPSS engine. The syntax of the LIBNAME statement in this case is

LIBNAME *libref* SPSS '*filename*';

Libref is a SAS libref and *filename* is the physical filename. If the libref appears also as a fileref, omit *filename*; the physical filename associated with the fileref is used.

Export files can originate from any operating environment. Export files must be transported to and from your operating environment in text format. If they are transported in binary format, other operating environments will not be able to read them.

Because SPSS-X files do not have internal names, refer to them by any member name you like.

The following example associates the libref MYLIB with the physical file **/users/myid/mydir/myspssx.spp** in order to run the CONTENTS and PRINT procedures on the save file:

```
libname mylib spss '/users/myid/mydir/myspssx.spp';
proc contents data=mylib._first_;
proc print data=mylib._first_;
run;
```

In the next example, the FILENAME statement associates the fileref MYLIB2 with the **/users/myid/mydir/aspssx.spp** SPSS physical file, and the LIBNAME statement associates the libref with the SPSS engine. The PRINT procedure prints the data from the save file.

```
filename mylib2 '/users/myid/mydir/aspssx.spp';
libname mylib2 spss;
proc print data=mylib2._first_;
run;
```

Accessing SAS Files from Previous Releases or from Other Hosts

For information about moving a file to or from a release that precedes Release 6.07, refer to *Moving and Accessing SAS Files across Operating Environments.* For information about moving a file to or from Release 6.07 through Version 8, refer to *Moving and Accessing SAS Files across Operating Environments*.

Sharing Files

If more than one user accesses a SAS file at the same time or if a single user has access to the same file from different SAS sessions, the results are unpredictable. By default, the FILELOCKS system option is set to FAIL, which prevents simultaneous access to the same SAS file. (See "FILELOCKS" on page 266.) If FILELOCKS has been set to NONE, then you should do one of the following:

- Make sure that your **sasuser** directory is unique for each SAS session. Typically, the system administrator assigns this directory in the system configuration file. The specification in that file or in your personal configuration file helps ensure that the directory is unique as long as you run only one SAS session at a time.

  If you run two or more SAS sessions simultaneously, you can guarantee unique user files by specifying different **sasuser** directories for each session. In the first session, you can use

  ```
  -sasuser ~/sasuser
  ```

  In the *n*th session, you can use

  ```
  -sasuser ~/sasusern
  ```

  For details, see "Processing Configuration Files" on page 17 and "SASUSER" on page 290. The RSASUSER option can be used to control modifications to the SASUSER library when it is shared by several users (see "RSASUSER" on page 286).
- If you run two or more SAS sessions simultaneously (either by using the X statement or by invoking it from two different windows) and you use the same PROFILE catalog, do not do anything within the SAS session to change that catalog (for example, using the WSAVE command or changing key assignments) because both sessions may use the same one.
- If you and other people use the same data sets, avoid writing to them at the same time.

Sharing Files in a Network

The SAS System can be licensed to run on one or more workstations in a network of similar machines. The license specifically lists the workstations that the SAS System can run on. Other workstations in the network may have access to the executable files for the SAS System but not be able to run the SAS System.

If the licensed workstations are connected via NFS mounts so that they share a file system, they can all share a single copy of the SAS System executables, although this is not necessary. They can also share SAS files. However, if a SAS session attempts to update a data set or catalog, it must obtain an exclusive file lock on that file to prevent other sessions from accessing that file.

When the SAS System is installed on workstations of different types that are connected via NFS, each type of workstation must have its own copy of the SAS System executables. Catalogs and data sets, however, may be shared between certain combinations of machine types.

If the data set or catalog you want to process exists on your network but cannot be accessed with the LIBNAME statement because it resides on a different type of workstation, you have several alternatives:

- You can log in to the remote machine and convert the file to SAS transport format using the CPORT procedure, copy the transport file to your machine or access it via NFS, and import it to your type of machine format.

- You can log in to the remote machine and perform the work there. This alternative works best when the file is used rarely, or if the original file changes often.
- You can do part of your work on your machine and the other part on the remote machine. One example of this alternative would be to run a set of statements on a small test case on the local machine, and then submit the real work to be done on the remote machine. Similarly, you might want to subset a large data set on another machine, and then do local analysis on that subset. This can be accomplished with SAS/CONNECT software.

To allow machines that cannot share SAS data sets or catalogs to coexist in the same networked file system, the SAS System adds a two-digit suffix to the file extensions. For example, if two SAS data sets named TEST are created on machines with different underlying architectures, one will be stored in file **test.ssd01** and the other will be stored in file **test.ssd02**.

The SAS System may hang when accessing data over NFS mounts if the FILELOCKS option is set to **FAIL** or **CONTINUE**. To alleviate the problem, make sure that all NFS filelocking daemons are running on both machines (usually **statd** and **lockd**).

Note: To test whether there is a problem with file locking, you can set the **FILELOCKS** system option to **NONE** temporarily. It is recommended that you do not set **FILELOCKS** to **NONE** permanently. △

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for UNIX Environments, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS® Companion for UNIX Environments, Version 8

ISBN 1–58025–502–7

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

1st printing, October 1999