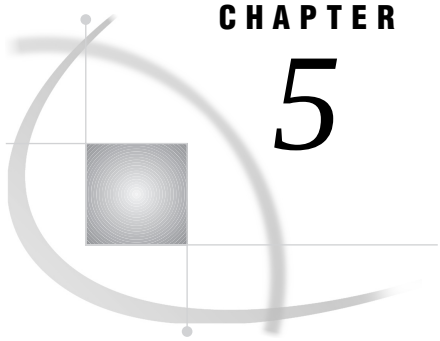

CHAPTER

5

Using External Files and Devices

Introduction to External Files and Devices

At times during a SAS session, you might want to use *external files*, that is, files that contain data or text or files in which you want to store data or text. These files are created and maintained by the operating system, not by the SAS System.

You can use external files in a SAS session to

- hold raw data to be read with the INPUT statements

- store printed reports created by a SAS procedure
- submit a file containing SAS statements for processing
- store data written with PUT statements.

For the SAS System, external files and devices can serve both as sources of input and as receivers of output. The input can be either raw data to be read in a DATA step or SAS statements to be processed by the SAS System. The output can be

- the SAS log, which contains notes and messages produced by the program
- the formatted output of SAS procedures
- data written with PUT statements in a DATA step.

You might also want to use peripheral devices such as a printer, plotter, or your own terminal. UNIX treats these I/O devices as if they were files. Each device is associated with a file, called a *special file*, which is treated as an ordinary disk file. When you write to a special file, the associated device is automatically activated. All special files reside in the **dev** directory or its subdirectories. Although there are some differences in how you use the various devices, the basic concept is the same for them all.

UNIX also enables you to use pipes to send data to and from operating system commands as if they were I/O devices.

The rest of this chapter describes how to specify an external file or device within a SAS session. Chapter 6, "Routing Output," on page 125 describes now to send the SAS log and SAS procedure output to an external file.

If you need to access an external file containing a transport data library, refer to *Moving and Accessing SAS Files across Operating Environments*.

Accessing an External File or Device

To read or write to an external file or device, refer to the file or device in the appropriate SAS statements in one of two ways:

- specifying the pathnames for the files.
- assigning a fileref to a device, one or more files, or a directory and using the fileref when you want to refer to the file or device.

In most cases, you will want to use a fileref. A fileref is nickname that you assign to a file or device. You simply assign the fileref once, and then use it as needed. Filerefs are especially useful when

- the pathname is long and has to be specified several times within a program
- the pathname might change. If the pathname changes, you need to change only the statement that assigns the fileref, not every reference to the file.

You can assign filerefs in the File Shortcuts window of the Explorer, with the FILENAME statement, with the FILENAME function,* or by defining the fileref as an environment variable.

To access the external file or device, you will need to specify its pathname or fileref in the appropriate SAS statements:

FILE
specifies the current output file for PUT statements.

* For a complete description of the FILENAME statement and the FILENAME function, see *SAS Language Reference: Dictionary*.

%INCLUDE
includes a file containing SAS source statements into the Program Editor.

INFILE
identifies an external file that you want to read with an INPUT statement.

Specifying Pathnames

You can reference an external file directly by specifying its pathname in the FILE, INFILE, or %INCLUDE statements, or you can reference the file indirectly by specifying a fileref and a pathname in the FILENAME statement and then using the fileref in the FILE, INFILE, or %INCLUDE statements.

Whether you reference a file directly or indirectly, you will need to specify its pathname in the appropriate statement. In most cases, you must enclose the name in quotes. For example, the following INFILE statement refers to the file **/users/pat/cars**:

```
infile '/users/pat/cars';
```

The following FILE statement directs output to the specified terminal:

```
file '/dev/ttyp1';
```

The level of specification depends on your current directory. You can use the character substitutions shown in Table 4.2 on page 88 to specify the pathname. You can also use wildcards as described in "Using Wildcards in Pathnames (Input Only)" on page 105.

You can omit the quotes on a filename if

- there is not already a fileref defined with that filename.
- the file has the filename extension expected by the statement that you are using to refer to the file. If you do not enclose a filename in quotes, the FILE and INFILE statements assume a file extension of **.dat**, and the %INCLUDE statement assumes a file extension of **.sas**.
- the file is in the current directory.
- the filename is all lower case characters.

For example, if the current directory is **/users/mkt/report** and it includes file **qtr.sas**, you can reference **qtr.sas** in any of the following statements:

```
%include '/users/mkt/report/qtr.sas';
%include 'qtr.sas';
file 'qtr.sas';
```

If there is no QTR fileref already defined, you can omit the quotes and the filename extension on the %INCLUDE statement:

```
%include qtr;
```

Using Wildcards in Pathnames (Input Only)

You can use the *, ?, and [ ]wildcards to specify pathnames in the FILENAME (only if the fileref is to be used for input), INFILE, and %INCLUDE statements and the INCLUDE command.

* matches one or more characters, except for the period at the beginning of filenames.

?	matches any single character.
[ ]	matches any single character from the set of characters defined within the brackets. You can specify a range of characters by specifying the starting character and ending character separated by a hyphen.

Wildcards are supported for input only. You cannot use wildcards in the FILE statement.

The following example reads input from every file in the current directory that begins with the string **wild** and ends with **.dat**:

```
filename wild 'wild*.dat';
data;
   infile wild;
   input;
run;
```

The following example reads input from every file in every subdirectory of the current working directory:

```
filename subfiles '*/*';
data;
   infile subfiles;
   input;
run;
```

If new files are added to any of the subdirectories, they can be accessed with the **subfiles** fileref without changing the FILENAME statement.

You can also use wildcards in filenames, but not in directory names, when you use aggregate syntax:

```
filename curdir ".";
data;
  infile curdir('wild*');
  input;
run;
```

The period in the FILENAME statement refers to the current directory. See Table 4.2 on page 88 for information on other characters substitutions available on UNIX.

The following statement associates the fileref **myref** with all files that begin with alphabetic characters. Files beginning with numbers or other characters such as the period or tilde are excluded.

```
filename myref '[a-zA-Z]*.dat';
```

The following statement associates **myref** with any file beginning with **sales** (in either uppercase, lowercase, or mixed case) and a year between 1990 and 1999:

```
filename myref '[Ss][Aa][Ll][Ee][Ss]199[0-9].dat';
```

Assigning Filerefs with the FILENAME Statement

The most common way to assign a fileref to an external file or device is with the FILENAME statement. There are several forms of the FILENAME statement, depending on the type of device you want to access.

FILENAME *fileref* <*device-type*> *'external-file'* <*host-options*>;

FILENAME *fileref device-type* <*'external-file'*><*host-options*>;

FILENAME *fileref* CLEAR | _ALL_ CLEAR;

FILENAME *fileref* LIST | _ALL_ LIST;

FILENAME *fileref* ("*pathname-1*" ... "*pathname-n*");

FILENAME *fileref directory-name*;

FILENAME *fileref* <*access-method*> '*external-file*' *access-information*;

fileref
is the name by which you reference the file.

device-type
specifies a device, such as a disk, terminal, printer, pipe, and so on. The device-type keyword must follow *fileref* and precede *pathname*. Table 5.1 on page 108 describes the valid device types. DISK is the default device type.

'*external-file*'
differs according to device type. Table 5.1 on page 108 shows the information appropriate to each device. Remember that UNIX filenames are case-sensitive. See "Specifying Pathnames" on page 105 for more information.

"*pathname-n*"
are pathnames for the files that you want to access with the same fileref. Use this form of the FILENAME statement when you want to concatenate filenames. Concatenating filenames is available only for DISK files, so you do not have to specify the *device-type*. Separate the pathnames with either commas or blank spaces. Enclose each pathname in quotes. Table 4.2 on page 88 shows character substitutions you can use when specifying a pathname. If the fileref that you are defining is to be used for input, then you can also use wildcards as described in "Using Wildcards in Pathnames (Input Only)" on page 105. Remember that UNIX filenames are case-sensitive.

directory-name
specifies the directory that contains the files that you want to access. For more information, see "Assigning a Fileref to a Directory (Using Aggregate Syntax)" on page 111.

host-options
control how the external file is processed. See "FILENAME" on page 237 for an explanation of these options.

access-method
can be CATALOG, SOCKET, FTP, or URL. Table 5.1 on page 108 describes the information expected by these access methods.

access-information
differs according to the access method. Table 5.1 on page 108 shows the information appropriate to each access method.

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

CLEAR
clears the specified fileref or, if you specify _ALL_, clears all filerefs that are currently defined.

Note: When you clear a fileref that is defined by an environment variable, the variable remains defined but is no longer considered a fileref. You can still reuse it, either as a fileref or a libref. See "Using Environment Variables to Assign Filerefs" on page 112 for more information. △

LIST
prints to the SAS log the pathname of the specified fileref or, if you specify _ALL_, lists the definition for all filerefs that are currently defined. Filerefs defined as environment variables appear only if you have already used those filerefs in a SAS statement. If you are using the Bourne shell or the Korn shell, the SAS System cannot determine the name of a preopened file, so it displays the following string instead of a filename:

```
<File Descriptor number>
```

See “Using Environment Variables to Assign Filerefs” on page 112 for more information.

Table 5.1 Device Information in the FILENAME Statement

Device or Access Method	Function	External-file
CATALOG	references a SAS catalog as a flat file.	is a valid two-, three-, or four-part SAS catalog name followed by catalog options needed. Refer to *SAS Language Reference: Dictionary* for a description of catalog options.
DISK	associates the fileref with a DISK file.	is either the pathname for a single file, or if you are concatenating filenames, a list of pathnames separated by blanks or commas and enclosed in parentheses. The level of specification depends on your location in the file system. Table 4.2 on page 88 shows character substitutions that you can use when specifying a UNIX pathname. See “Using DISK Files” on page 109 for more information.
DUMMY	associates a fileref with a null device.	none. See “Debugging Code With DUMMY Devices” on page 110 for more information.
EMAIL	sends electronic mail to an address.	is an address and email options. See “Sending Electronic Mail from Within the SAS System (EMAIL)” on page 116 for more information.
FTP	reads or writes to a file from any machine on a network that is running an FTP server.	is the pathname of the external file on the remote machine followed by FTP options. See “Assigning Filerefs to Files on Other Systems (FTP and SOCKET access types)” on page 111 and *SAS Language Reference: Dictionary* for more information. If you are transferring a file from the OS/390 operating environment and you want to access the file by using one of the S370 formats, the file must be of type RECFM=U before you transfer it to UNIX.
PIPE	reads input from or writes output to a UNIX command.	is a UNIX command. See “Reading from and Writing to UNIX Commands (PIPE)” on page 114 and Chapter 6, “Routing Output,” on page 125 for details.
PLOTTER	sends output to a plotter.	is a device name and plotter options. See “Using PRTFILE and PRINT with a Fileref” on page 131and “Using the PRINTTO Procedure” on page 133 for details.
PRINTER	sends output to a printer.	is a device name and printer options. See “Sending Output to PRINTER Devices” on page 110, “Using PRTFILE and PRINT with a Fileref” on page 131, and “Using the PRINTTO Procedure” on page 133 for details.

Device or Access Method	Function	External-file
SOCKET	reads and writes information over a TCP/IP socket.	depends on whether the SAS application is a server application or a client application. In a client application, external-file is the name or IP address of the host and the TCP/IP port number to connect to followed by any TCP/IP options. In a server application, it is the port number to create for listening, followed by the SERVER keyword, and then any TCP/IP options. See "Assigning Filerefs to Files on Other Systems (FTP and SOCKET access types)" on page 111 and *SAS Language Reference: Dictionary* for details.
TAPE	associates a fileref with a tape.	is the pathname for a tape device. The name specified should be the name of the special file associated with the tape device. See "Processing Files on TAPE" on page 121 for more information.
TEMP	associates a fileref with an external file stored in the WORK data library.	none. See "Using Temporary Files (TEMP Device Type)" on page 110 for more information.
TERMINAL	associates a fileref with a terminal.	is the pathname of a terminal. See "Accessing TERMINAL Devices Directly" on page 110 for more information.
URL	allows you to access remote files using the URL of the file.	is the name of the file that you want to read from or write to on a URL server. The URL must be in one of these forms: **http://*hostname/file*** **http://*hostname*:*portno*/*file*** Refer to *SAS Language Reference: Dictionary* for more information.
XPRINTER	sends output to the default printer that was set up through the Printer Setup dialog.	none. See Chapter 6, "Routing Output," on page 125 for more information.

Using DISK Files

The most common use of the FILENAME statement is to access DISK files. The FILENAME syntax for a DISK file is

FILENAME *fileref* <DISK> *'pathname'* <*options*>;

The following FILENAME statement associates the fileref MYFILE with the external file **/users/mydir/myfile**, which is stored on a disk device:

```
filename myfile disk '/users/mydir/myfile';
```

The following FILENAME statement assigns a fileref of PRICES to the file **/users/pat/cars**. The FILE statement then refers to the file using the fileref:

```
filename prices '/users/pat/cars';
data current.list;
   file prices;
   ...PUT statements...
run;
```

See "Concatenating Filenames" on page 111 for more information on using DISK files.

Debugging Code With DUMMY Devices

You can substitute the DUMMY device type for any of the other device types. This device type serves as a tool for debugging your SAS code without actually reading or writing to the device. After debugging is complete, replace the DUMMY device name with the proper device type, and your program will access the specified device type.

The FILENAME syntax for a DUMMY file is

FILENAME *fileref* DUMMY *'pathname'*<*options*>;

Output to DUMMY devices is discarded.

Sending Output to PRINTER Devices

The PRINTER device type allows you to send output directly to a printer. The FILENAME syntax to direct a file to a PRINTER is

FILENAME *fileref* PRINTER *'*<*printer*> <*printer-options*>*'* <*options*>;

For example, this SAS program sends the output file to the BLDG3 printer:

```
filename myfile printer 'bldg3';

data test;
   file myfile;
   put 'This will appear in bldg3 .';
run;
```

See “Using PRTFILE and PRINT with a Fileref” on page 131 and “Using the PRINTTO Procedure” on page 133 for more information.

Using Temporary Files (TEMP Device Type)

The TEMP device type associates a fileref with a temporary file stored in the same directory as the WORK data library. (See “WORK Data Library” on page 95.) Using the TEMP device type enables you to create a file that lasts only as long as the SAS session.

The FILENAME syntax for a TEMP file is

FILENAME *fileref* TEMP <*options*>;

For example, this FILENAME statement associates TMP1 with a temporary file:

```
filename tmp1 temp;
```

Accessing TERMINAL Devices Directly

To access a terminal directly, use the TERMINAL device type. The FILENAME syntax to associate a file with a terminal is

FILENAME *fileref* TERMINAL <*'terminal-pathname'*> <*options*>;

The *terminal-pathname* must be a pathname of the special file associated with the terminal. Check with your system administrator for details. Enclose the name in quotes. If you omit the terminal pathname, the fileref is assigned to your terminal.

For example, this FILENAME statement associates the fileref HERE with your terminal:

```
filename here terminal;
```

The following FILENAME statement associates the fileref THATFILE with another terminal:

```
filename thatfile terminal '/dev/tty3';
```

Assigning Filerefs to Files on Other Systems (FTP and SOCKET access types)

You can access files on other systems in your network by using the SOCKET and FTP access methods. The forms of the FILENAME statement are

FILENAME *fileref* FTP *'external-file'* <*ftp-options*>;

FILENAME *fileref* SOCKET *'external-file'* <*tcpip-options*>;

FILENAME *fileref* SOCKET *':portno'* SERVER <*tcpip-options*>;

These access methods are documented in *SAS Language Reference: Dictionary*. Under UNIX, the FTP access method supports an additional option:

MACH=*'machine'*
identifies which entry in the **.netrc** file should be used to get the username and password. Consult the UNIX man page for more information on the **.netrc** file. You cannot specify the MACH option together with the HOST option in the FILENAME statement.

The file that you want to transfer must be of type RECFM=U before you transfer it to UNIX.

CAUTION:
When you use the FTP access method to create a remote file, the UNIX permissions for that file are set to *-rw-rw-rw-*, which makes the file world-readable and world-writeable. See the man page for **chmod** for information on changing file permissions. △

Concatenating Filenames

You can concatenate filenames in the FILENAME, %INCLUDE, and INFILE statements. Concatenating filenames allows you to read those files sequentially.

FILENAME *fileref* ("*pathname-1*" ... "*pathname-n*");

%INCLUDE '("*filename-1*" ... "*filename-n*")';

%INCLUDE "('*filename-1*' ... '*filename-n*')";

INFILE '("*filename-1*" ... "*filename-n*")';

INFILE "('*filename-1*' ... '*filename-n*')";

You can enclose the pathnames in single or double quotes and separate them with commas or blank spaces. You can use the characters shown in Table 4.2 on page 88 and the wildcards described in "Using Wildcards in Pathnames (Input Only)" on page 105 to specify the pathnames.

Assigning a Fileref to a Directory (Using Aggregate Syntax)

Aggregate syntax allows you to assign a fileref to a directory and then work with any file in that directory by specifying its filename in parentheses after the fileref.

FILENAME *fileref directory-name*;

Aggregate syntax is especially useful when you have to refer to several files in one directory.

To refer to a file in the directory, specify the fileref followed by the individual filename in parentheses. For example, you can refer to the file **cars.dat** in the directory **/users/pat** as shown in this example:

```
filename prices '/users/pat';
data current.list;
   file prices(cars);
   ...other SAS statements...
run;
```

You can also use aggregate syntax with filerefs that have been defined using environment variables (see "Using Environment Variables to Assign Filerefs" on page 112). For example:

```
x setenv PRICES /users/pat;
data current.list;
   file prices(cars);
   ...other SAS statements...
run;
```

Assigning a Fileref to Several Directories

In the FILENAME statement, you can concatenate directory names and use the fileref to refer to any file within those directories:

FILENAME *fileref* ("*directory-1*" ... "*directory-n*");

When you concatenate directory names, you can use aggregate syntax to refer to a file in one of the directories. For example, assume that the **report.sas** file resides in the directory associated with the MYPROGS environment variable. When the SAS System executes the following code, it searches for **report.sas** in the pathnames that are specified in the FILENAME statement and it executes the program.

```
filename progs ("$MYPROGS" "/users/mkt/progs");
%inc progs(report);
```

The SAS System searches the pathnames in the order specified in the FILENAME statement until

- it finds the first file with the specified name. Even if you use wildcards (see "Using Wildcards in Pathnames (Input Only)" on page 105) in the filename, SAS matches only one file.
- it encounters a filename in the list of pathnames that you specified in the FILENAME statement.

Using Environment Variables to Assign Filerefs

An environment variable can also be used as a fileref to refer to DISK files. The variable name must be in all uppercase characters, and the variable value must be the full pathname of the external file; that is, the filename must begin with a slash.

Suppose that you want to read the data file **/users/myid/educ.dat**, but you want to refer to it with the INED 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 use

```
export INED=/users/myid/educ.dat
```

- 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 INED /users/myid/educ.dat;
```

After INED is associated with the file **/users/myid/educ.dat**, you can use INED as a fileref to refer to the file in the INFILE statement:

```
infile ined;
```

The same method applies if you want to write to an external file. For example, you can define OUTFILE before you invoke the SAS System:

```
OUTFILE=/users/myid/scores.dat
export OUTFILE
```

Then, use the environment variable name as a fileref to refer to the file:

```
file outfile;
```

Note: If a variable and a fileref have the same name but refer to different files, the SAS System uses the fileref. For example, the %INCLUDE statement below refers to file **/users/myid/this_one**. △

```
filename ABC '/users/myid/this_one';
x setenv ABC /users/myid/that_one;
%include ABC;
```

Filerefs Assigned by the SAS System

Often a command's arguments or options tell the command what to use for input and output, but in case they do not, the shell supplies you with three standard files: one for input (*standard input*), one for output (*standard output*), and one for error messages (*standard error*). By default, these files are all associated with your terminal: standard input with your keyboard, and both standard output and standard error with your terminal's display. When you invoke the SAS System, it assigns a fileref to each file that it opens, including the three standard files. SAS assigns the filerefs STDIN, STDOUT, and STDERR to standard input, standard output, and standard error, respectively.

Each file has an internal *file descriptor* assigned to it. By default, 0 is the file descriptor for standard input, 1 is the file descriptor for standard output, and 2 is the file descriptor for standard error. As other files are opened, they get other file descriptors. In the Bourne shell and in the Korn shell, you can specify that data be written to or be read from a file using the file descriptor as described in "File Descriptors in the Bourne and Korn Shells" on page 114.

File Descriptors in the Bourne and Korn Shells

If you are using the Bourne shell or the Korn shell, the SAS System assigns filerefs of the following form to files that have a file descriptor (see "Filerefs Assigned by the SAS System" on page 113) larger than 2.

```
FILDES number
```

number is a two-digit representation of the file descriptor. You can use these filerefs in your SAS applications.

For example, if you invoke SAS with the following command, then the operating environment opens the file **sales_data** and assigns file descriptor 4 to it:

```
sas salespgm 4< sales_data
```

SAS assigns the fileref FILDES04 to the file and executes the application **salespgm**. When the application reads input from FILDES04, it reads the file **sales_data**. Using file descriptors as filerefs enables you to use the same application to process data from different files without changing the application to refer to each file. In the command that you use to invoke the application, you simply assign the appropriate file descriptor to the file to be processed.

Reserved Filerefs

The following filerefs are reserved.

DATALINES fileref in the INFILE statement
: specifies that input data immediately follow a DATALINES statement. You need to use INFILE DATALINES only when you want to specify options in the INFILE statement to read instream data.

LOG fileref in the FILE statement
: specifies that output lines produced by PUT statements be written to the SAS log. LOG is the default destination for output lines.

PRINT fileref in the FILE statement
: specifies that output lines produced by PUT statements be written to the same print file as output produced by SAS procedures.

Reading from and Writing to UNIX Commands (PIPE)

Under UNIX, you can use the FILENAME statement to assign filerefs not only to external files and I/O devices, but also to a pipe. Pipes enable your SAS application to receive input from any UNIX command that writes to standard output and to route output to any UNIX command that reads from standard input. In UNIX commands, the pipe is represented by a vertical bar (|). For example, to find the number of files in your directory, you could redirect the output of the **ls** command through a pipe to the **wc** (word count) command by entering

```
ls | wc -w
```

The syntax of the FILENAME statement is

FILENAME *fileref* PIPE *'UNIX-command'* <*options*>;

fileref
is the name by which you reference the pipe from the SAS System.

PIPE
identifies the device-type as a UNIX pipe.

'*UNIX-command*'
is the name of a UNIX command, executable program, or shell script to which you want to route output or from which you want to read input. The command(s) must be enclosed in either double or single quotes.

options
control how the external file is processed. See "FILENAME" on page 237 for an explanation of these options.

Whether you are using the command as input or output depends on whether you use the *fileref* in a reading or writing operation. For example, if the fileref is used in an INFILE statement, then the SAS System assumes that the input comes from a UNIX command; if the fileref is used in a FILE statement, then the SAS System assumes that the output goes to a UNIX command.

Using the Fileref for Reading

When the fileref is used for reading, the specified UNIX command executes, and any output sent to its standard output or standard error is read through the fileref. In this case, the standard input of the command is connected to **/dev/null**.

For example, the following SAS program uses the PIPE device-type keyword to send the output of the **ps** (process) command to a SAS DATA step. The resulting SAS data set contains data about every process currently running the SAS System:

```
filename ps_list pipe "ps -e|grep 'sas'";
data sasjobs;
   infile ps_list;
   length process $ 80;
   input process $ char80.;
run;
proc print data=sasjobs;
run;
```

The **ps -e** command produces a listing of all active processes on the system, including the name of the command that started the task. In BSD-based UNIX systems, you use the **ps -ax** command.

The operating environment uses pipes to send the output from **ps** to the **grep** command, which searches for every occurrence of the string **'sas'**. The FILENAME statement connects the output of the **grep** command to the fileref PS_LIST. The DATA step then creates a data set named SASJOBS from the INFILE statement that points to the input source. The INPUT statement reads the first 80 characters on each input line.

In the next example, the STDIN fileref is used to read input through a pipe into the SAS command which in turn executes the SAS program. By placing the piping operation outside the SAS program, the program becomes more general. The program in the previous example has been changed and stored in file **ps.sas**:

```
data sasjobs;
   infile stdin;
   length process $ 80;
   input process $ char80.;
run;
proc print data=sasjobs;
```

```
run;
```

To run the program, use pipes to send the output of **ps** to **grep** and from **grep** into the SAS command:

```
ps -e|grep 'sas'|sas ps.sas &
```

The output will be stored in **ps.lst**; the log in **ps.log** as described in "The Default Routings for the SAS Log and Procedure Output" on page 126.

Using the Fileref for Writing

When the fileref is used for writing, the output from the SAS System is read in by the specified UNIX command, which then executes.

In this example, any data sent to the MAIL fileref are piped to the **mail** command and sent to user PAT:

```
filename mail pipe 'mail pat';
```

Consider this FILENAME statement:

```
filename letterq pipe 'remsh alpha lp -dbldga3';
```

Any data sent to the LETTERQ fileref are passed to the UNIX command, which starts a remote shell on the machine named ALPHA.* The shell then prints the LETTERQ output on the printer identified by the destination BLDGA3. Any messages produced by the **lp** command are sent to the SAS log.

Sending Electronic Mail from Within the SAS System (EMAIL)

The SAS System lets you send electronic mail using SAS functions in a DATA step or in SCL. Sending e-mail from within the SAS System allows you to

- use the logic of the DATA step or SCL to subset e-mail distribution based on a large data set of e-mail addresses.
- send e-mail automatically upon completion of a SAS program that you submitted for batch processing.
- direct output through e-mail based on the results of processing.
- send e-mail messages from within a SAS/AF frame application, customizing the user interface.

Initializing Electronic Mail

Because of the wide range of e-mail programs available, the SAS System sends all mail through an external shell script. The SAS System provides two scripts, located in **!SASROOT/utilities/bin**:

sasmailer
does not support aliases or attachments.

sasm.elm.mime
supports aliases using ELM and attachments using MIME.

* The form of the command that starts a remote shell varies among the various UNIX operating systems.

Note: You or your system administrator will probably have to customize these scripts before you can use them with your specific e-mail program. △

Specify the name of the script you will be using by setting the EMAILSYS system option. You can specify the EMAILSYS system option in the CONFIG.SAS file or when invoking your SAS session:

-**EMAILSYS** *name-of-script*

Using the DATA Step or SCL to Send Electronic Mail

In general, a DATA step or SCL code that sends electronic mail has the following components:

- a FILENAME statement with the EMAIL device-type keyword
- options specified on the FILENAME or FILE statements indicating the e-mail recipients, subject, and any attached files
- PUT statements that contain the body of the message
- PUT statements that contain special e-mail directives (of the form !EM_*directive*!) that can override the e-mail attributes (TO, CC, SUBJECT, ATTACH) or perform actions (such as SEND, ABORT, and start a NEWMSG).

Syntax of the FILENAME Statement for Electronic Mail

To send electronic mail from a DATA step or SCL, issue a FILENAME statement of the following form:

FILENAME *fileref* EMAIL '*address*' <*email-options*>;

The FILENAME statement accepts the following *email-options*:

fileref
is a valid fileref.

'*address*'
is the destination e-mail address of the user to which you want to send e-mail. You must specify an address here, but you can override its value with the TO e-mail option.

email-options
can be any of the following:

TO=*to-address*
specifies the primary recipients of the electronic mail. If an address contains more than one word, enclose it in single quotes. To specify more than one address, enclose the group of addresses in parentheses and each address in single quotes. For example, **to='joe@somplace.org'** and **to=('joe@smplc.org' 'jane@diffplc.org')** are valid TO values.

CC=*cc-address*
specifies the recipients you want to receive a copy of the electronic mail. If an address contains more than one word, enclose it in single quotes. To specify more than one address, enclose the group of addresses in parentheses and each address in single quotes. For example, **cc='joe@somplace.org'** and **cc=('joe@smplc.org' 'jane@diffplc.org')** are valid CC values.

SUBJECT='*subject*'
specifies the subject of the message. If the subject text is longer than one word, enclose it in single quotes. For example, **subject=Sales** and **subject='June Report'** are valid subjects. Any subject not enclosed in quotes is converted to upper case.

ATTACH='*pathname*'
specifies the full pathname of one or more files to attach to the message. Enclose *pathname* in single quotes. To attach more than one file, enclose the group of file names in parentheses. For example, **attach='opinion.txt'** and **attach=('june98.txt' 'july98.txt')** are valid file attachments.

Note: Not all external scripts support file attachments or all types of file attachments. Scripts that do not accept attachments should not send mail if an attachment is attempted. Otherwise, the message could say "here's the graph you wanted," but the graph would not be included. △

You can also specify the *email-options* in the FILE statement inside the DATA step. Options that you specify in the FILE statement override any corresponding options that you specified in the FILENAME statement.

In your DATA step, after using the FILE statement to define your e-mail fileref as the output destination, use PUT statements to define the body of the message.

You can also use PUT statements to specify e-mail directives that change the attributes of your electronic message or perform actions with it. Specify only one directive in each PUT statement; each PUT statement can contain only the text associated with the directive it specifies.

The directives that change the attributes of your message are

!EM_TO! *addresses*
Replace the current primary recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotes.

!EM_CC! *addresses*
Replace the current copied recipient addresses with *addresses*. In the PUT statement, specify *addresses* without single quotes.

!EM_SUBJECT! *subject*
Replace the current subject of the message with *subject*.

!EM_ATTACH! *pathname*
Replace the names of any attached files with *pathname*.

The directives that perform actions are

!EM_SEND!
Sends the message with the current attributes. By default, the message is automatically sent at the end of the DATA step. If you use this directive, the SAS System sends the message when it encounters the directive, *and* again at the end of the DATA step.

!EM_ABORT!
Aborts the current message. You can use this directive to stop the SAS System from automatically sending the message at the end of the DATA step.

!EM_NEWMSG!
Clears all attributes of the current message, including TO, CC, SUBJECT, ATTACH, and the message body.

Example: Sending E-Mail from the DATA Step

Suppose that you want to share a copy of your CONFIG.SAS file with your coworker Jim, whose user ID is JBrown. If your e-mail program handles alias names and attachments, you could send it by submitting the following DATA step:

```
filename mymail email 'JBrown'
         subject='My CONFIG.SAS file'
         attach='config.sas';

data _null_;
  file mymail;
  put 'Jim,';
  put 'This is my CONFIG.SAS file.';
  put 'I think you might like the
       new options I added.';
run;
```

The following example sends a message and two attached files to multiple recipients. It specifies the e-mail options in the FILE statement instead of the FILENAME statement:

```
filename outbox email 'ron@acme.com';

data _null_;
  file outbox
     to=('ron@acme.com' 'lisa@acme.com')
     /* Overrides value in */
     /* filename statement */

     cc=('margaret@yourcomp.com'
         'lenny@laverne.abc.com')
     subject='My SAS output'
     attach=('results.out' 'code.sas')
     ;
  put 'Folks,';
  put 'Attached is my output from the
      SAS program I ran last night.';
  put 'It worked great!';
run;
```

You can use conditional logic in the DATA step to send multiple messages and control which recipients get which message. For example, suppose you want to send customized reports to members of two different departments. If your e-mail program handles alias names and attachments, your DATA step might look like the following:

```
filename reports email 'Jim';

data _null_;
  file reports;
  infile cards eof=lastobs;
  length name dept $ 21;
  input name dept;
  put '!EM_TO!' name;
   /* Assign the TO attribute      */
```

```
  put '!EM_SUBJECT! Report for ' dept;
   /* Assign the SUBJECT attribute */

  put name ',';
  put 'Here is the latest report for ' dept '.';
  if dept='marketing' then
    put '!EM_ATTACH! mktrept.txt';
  else
    /* ATTACH the appropriate report */

    put '!EM_ATTACH! devrept.txt';
  put '!EM_SEND!';
    /* Send the message */

  put '!EM_NEWMSG!';
    /* Clear the message attributes */

  return;
lastobs: put '!EM_ABORT!';
    /* Abort the message before the */
    /*   RUN statement causes it to */
    /*   be sent again.

  datalines;
Susan          marketing
Jim            marketing
Rita           development
Herb           development
;
run;
```

The resulting e-mail message and its attachments are dependent on the department to which the recipient belongs.

Note: You must use the !EM_NEWMSG! directive to clear the message attributes between recipients. The !EM_ABORT! directive prevents the message from being automatically sent at the end of the DATA step. △

Example: Sending E-Mail Using SCL Code

The following example is the SCL code behind a frame entry design for e-mail. The frame entry includes several text entry fields that let the user enter information:

mailto	the user ID to send mail to
copyto	the user ID to copy (CC) the mail to
attach	the name of a file to attach
subject	the subject of the mail
line1	the text of the message

The frame entry also contains a pushbutton called SEND that causes this SCL code (marked by the **send:** label) to execute.

```
send:

   /* set up a fileref */
```

```
   rc = filename('mailit','userid','email');

   /* if the fileref was successfully set up
      open the file to write to */

   if rc = 0 then do;
      fid = fopen('mailit','o');
      if fid > 0 then do;

         /* fput statements are used to
            implement writing the
            mail and the components such as
            subject, who to mail to, etc. */


         fputrc1  = fput(fid,line1);
         rc = fwrite(fid);

         fputrc2  = fput(fid,'!EM_TO! '||mailto);
         rc = fwrite(fid);
         fputrc3  = fput(fid,'!EM_CC! '||copyto);
         rc = fwrite(fid);

         fputrc4  = fput(fid,'!EM_ATTACH! '||attach);
         rc = fwrite(fid);
         fputrc5  = fput(fid,'!EM_SUBJECT! '||subject);
         rc = fwrite(fid);

         closerc  = fclose(fid);
      end;
   end;
return;

cancel:
   call execcmd('end');
return;
```

Processing Files on TAPE

Tape devices are inherently slow and should be used on a regular basis only for archiving files or for transferring data from one system to another.

There are four UNIX commands that are frequently used to process tape files on UNIX:

mt	positions the tape (winds forward and rewinds). On AIX, this command is **tctl**.
dd	converts, reblocks, translates, and copies files.
cat	concatenates, copies, and prints files.
tar	saves and restores archive files.
remsh	connects to the specified host and executes the specified command.

For a complete description of these commands, refer to the man pages.

In addition, you will almost always need to use a no-rewind device and the SAS system option TAPECLOSE=LEAVE to get the results you want. The example in this section assume the use of a no-rewind device and TAPECLOSE=LEAVE.

You can use either the TAPE device type or the PIPE device type to process tape files.

Using the TAPE Device Type

To use the TAPE device type, enter the FILENAME statement as follows:

FILENAME *fileref* TAPE *'tape-device-pathname'* <*options*>;

The *tape-device-pathname* is the pathname of the special file associated with the tape device. Check with your system administrator for details. Enclose the name in quotes.

For example, this FILENAME statement associates YR1999 with a file stored on a tape that is mounted on device **/dev/tp0**:

```
filename  yr1999 tape '/dev/tp0';
```

Using the PIPE Device Type

You can also use the PIPE device type together with UNIX **dd** command to process the tape:

FILENAME *fileref* **PIPE** *'UNIX-commands'*;

UNIX-commands are the commands needed to process the tape.

Using the PIPE device type and the **dd** command can process the tape more efficiently than the TAPE device type, and it allows you to use remote tape drives. However, using UNIX commands in your application means that the application will have to be modifed if it is ported to a non-UNIX environment.

For example, the following DATA step writes an external file to tape:

```
options tapeclose=leave;
x 'mt -t /dev/rmt/0mn rewind';
filename outtape pipe 'dd of=/dev/rmt/0mn 2> /dev/null';
data _null_;
   file outtape;
   put '1 one';
   put '2 two';
   put '3 three';
   put '4 four';
   put '5 five';
run;
```

The following DATA step reads the file from tape:

```
options tapeclose=leave;
x 'mt -t /dev/rmt/0mn rewind';
filename intape pipe 'dd if=/dev/rmt/0mn 2> /dev/null';
data numbers;
   infile intape pad;
   input digit word $8.;
run;
```

If the tape drive that you want to access is a remote tape drive, you can access the remote tape drive by adding **remsh *machine-name*** to the X and FILENAME

statements. For example, if the remote machine name is **wizard**, then you could read and write tape files on **wizard** by modifiying the X and FILENAME statements as follows:

```
x 'remsh wizard mt -t /dev/rmt/0mn rewind';
filename intape pipe 'remsh wizard \
                dd if=/dev/rmt/0mn 2> /dev/null';
```

Working with External Files Created on the Mainframe

There are three main points to remember when dealing with tapes on UNIX that were created on a mainframe:

- UNIX does not support IBM standard label tapes. IBM standard label tapes contain user data files and labels, which themselves are files on the tape. To process the user data files on these tapes, use a no-rewind device (such as **/dev/rmt/0mn**) and the **mt** command with the **fsf *count*** subcommand to position the tape to the desired user data file. The formula for calculating ***count*** is

  count = (3 x *user_data_file_number*) - 2

- UNIX does not support multi-volume tapes. To process multi-volume tapes on UNIX, the contents of each tape must be copied to disk using the **dd** command. After all of the tapes have been unloaded, you can use the **cat** command to concatenate all of the pieces in the correct order. You can then process the concatenated file on disk.

- You must know the DCB characteristics of the file. The records in files that are created on a mainframe are not delimited with end-of-line characters, so you must specify the original DCB parameters on the INFILE or FILENAME statement. In the INFILE statement, specify the record length, record format, and block size with the LRECL, RECFM, and BLKSIZE host options. In the FILENAME statement, if you use the PIPE device-type and the **dd** command, you must also specify the block size with the **ibs** subcommand. For more information about host options on the INFILE statement, see "INFILE" on page 242. For more information about the **ibs** subcommand, refer to the man page for the **dd** command.

Example: Multi-volume, Standard Label Tapes

Suppose that you are given a two-reel, multi-volume, standard label tape set containing a mainframe external file and told that the record length is 7 and the record format is fixed. You will need to unload the data portion of each tape into disk files, concatenate the two disk files, and process the resultant file.

Make sure that the first tape is in the tape drive, then use the **mt** command to rewind the tape, skip over the label file, and position the tape at the beginning of the user data file. In this case, the user data file that you want to access is the first (and only) user data file on the tape. To skip over the label and position the tape at the beginning of the user data file, use the **fsf *count*** subcommand. Using the formula in "Working with External Files Created on the Mainframe" on page 123, the **fsf** count value is 1.

```
mt -t /dev/rmt/0mn rewind
mt -t /dev/rmt/0mn fsf 1
dd if=/dev/rmt/0mn of=/tmp/tape1 ibs=7
```

Repeat this process with the second tape, then concatenate the two disk files into one file.

```
mt -t /dev/rmt/0mn rewind
mt -t /dev/rmt/0mn fsf 1
dd if=/dev/rmt/0mn of=/tmp/tape2 ibs=7

cat /tmp/file1 /tmp/file2 > /tmp/ebcdic.numbers
```

You can then use the following DATA step to refer to the concatenated file (**/tmp/ebcdic.numbers**) and to convert the data using the appropriate EBCDIC informats:

```
filename ibmfile '/tmp/ebcdic.numbers';
data numbers;
   infile ibmfile lrecl=7 recfm=f;
   length digit 8 temp $ 1 word $ 6;
   input temp $ebcdic1. word $ebcdic6.;
   digit=input(temp,8.);
   drop temp;
run;
```

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