Batch and Line Mode Processing in SAS® Viya™
PART 1  Running SAS Viya in Batch or Interactive Line Mode

Chapter 1 • Getting Started in Batch or Interactive Line Mode
  Running SAS in Batch Mode
  Running SAS in a Foreground or Background Process
  Interactive Line Mode in Linux Environments
  Executing Operating System Commands from Your SAS Session
  Customizing Your SAS Session By Using System Options
  Customizing Your SAS Session By Using Configuration and Autoexec Files
  Determining the Completion Status of a SAS Job in Linux Environments
  Exiting or Interrupting Your SAS Session in Linux Environments
  Checkpoint Mode and Restart Mode
  Ending a Process That Is Running as a SAS Server
  Interrupting a SAS Process and the Underlying DBMS Process

Chapter 2 • Connecting to SAS Cloud Analytic Services from the Command Line
  How to Connect to SAS Cloud Analytic Services
  Comparison of Batch Modes and Interactive Modes

Chapter 3 • Using SAS Files and Libraries
  Introduction to SAS Files, Libraries, and Engines in Linux Environments
  Common Types of SAS Files in Linux Environments
  Sharing SAS Files in Linux
  Referring to SAS Files By Using Librefs in Linux
  Specifying Pathnames in Linux
  Assigning a Libref to Several Directories (Concatenating Directories)
  Using Environment Variables as Librefs in Linux Environments
  Sasuser Library
  Work Library
  Using One-Level Names to Access Permanent Files (User Library)
  Support for Links in Linux

Chapter 4 • Using External Files and Devices
  Introduction to External Files and Devices in Linux
  Accessing an External File or Device in Linux Environments
  Specifying Pathnames in Linux Environments
  Assigning Filerefs to External Files or Devices with the FILENAME Statement
  Concatenating Filenames in Linux Environments
  Assigning a Fileref to a Directory (Using Aggregate Syntax)
  Using Environment Variables to Assign Filerefs in Linux Environments
  Filerefs Assigned by SAS in Linux Environments
  Reserved Filerefs in Linux Environments
  Sharing External Files in a Linux Environment
Contents

Reading from and Writing to Linux Commands (PIPE) .......................... 55

Chapter 5  •  Data Representation .......................................................... 59
  Numeric Variable Length and Precision in Linux Environments .......... 59
  Missing Values in Linux Environments ............................................. 60
  Reading and Writing Binary Data in Linux Environments ................. 60
  Converting a Linux Datetime Value to a SAS Datetime Value ............... 61

PART 2  Features Available in Batch and Line Mode 63

Chapter 6  •  Commands Available in Batch and Line Mode ...................... 65
  Dictionary ..................................................................................... 65

Chapter 7  •  Environment Variables Available in Batch and Line Mode .... 67
  Defining Environment Variables in Linux Environments ................. 67
  Dictionary ..................................................................................... 68

Chapter 8  •  Cleanwork ........................................................................ 71
  Cleanwork Utility ........................................................................... 71

Index ................................................................................................... 73
Early Adopter Software

THIS DOCUMENTATION FOR AN EARLY ADOPTER PRODUCT IS A PRELIMINARY DRAFT AND IS PROVIDED BY SAS INSTITUTE INC. ("SAS") ON AN "AS IS" BASIS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. SAS does not warrant that this documentation is complete, accurate, or similar to that which may be released to the general public, or that any such documentation will be released. The company shall not be liable whatsoever for any damages arising out of the use of this documentation, including any direct, indirect, or consequential damages. SAS reserves the right to alter or abandon use of this documentation at any time.
Part 1

Running SAS Viya in Batch or Interactive Line Mode

Chapter 1

Getting Started in Batch or Interactive Line Mode ........................ 3

Chapter 2

Connecting to SAS Cloud Analytic Services from the Command Line ............................................. 27

Chapter 3

Using SAS Files and Libraries .................................................. 31

Chapter 4

Using External Files and Devices .............................................. 45

Chapter 5

Data Representation ............................................................. 59
Chapter 1
Getting Started in Batch or Interactive Line Mode

Running SAS in Batch Mode ............................................. 4
  Introduction to Batch Mode ........................................... 4
  SAS Command ............................................................ 4
  Invoking SAS in Batch Mode ....................................... 4
  Syntax of the SAS Command ....................................... 5
  Submitting a Program to the Batch Queue .................... 5
  Writing Data from an External File Using Pipes .......... 5

Running SAS in a Foreground or Background Process .......... 6
  Interactive Line Mode in Linux Environments ........... 6
  Introduction to Interactive Line Mode ....................... 6
  Invoking SAS in Interactive Line Mode ..................... 7
  Exiting SAS in Interactive Line Mode ......................... 7
  Example: Invoke an Interactive SAS Session .............. 8

Executing Operating System Commands from Your SAS Session . 8
  Deciding Whether to Run an Asynchronous or Synchronous Task 8
  Executing a Single Linux Command ............................ 8
  Executing Several Linux Commands ........................... 9
  Changing the File Permissions for Your SAS Session .... 10
  Executing X Statements in Batch Mode ..................... 10

Customizing Your SAS Session By Using System Options .... 11
  Specify SAS System Options ...................................... 11
  Overriding the Default Value for a System Option ........ 11
  When the Value of a System Option Includes a Space ..... 13
  How SAS Processes System Options That Are Set More Than Once 13
  How SAS Processes System Options That Are Set in Multiple Places .. 13

Customizing Your SAS Session By Using Configuration and Autoexec Files . 14
  Introduction to Configuration and Autoexec Files ......... 14
  Creating a Configuration File .................................... 16
  Order of Precedence for Processing SAS Configuration Files 16
  Specifying a Configuration File for SAS to Use .......... 16

Determining the Completion Status of a SAS Job in Linux Environments .... 17

Exiting or Interrupting Your SAS Session in Linux Environments .... 18
  Methods for Exiting SAS in Interactive Line Mode ........ 18
  Methods for Interrupting or Terminating SAS ........... 18
  Messages in the Console Log (STDOUT) ...................... 20

Checkpoint Mode and Restart Mode .................................. 20
Running SAS in Batch Mode

Introduction to Batch Mode

To run SAS in batch mode, you specify your SAS program name in the SAS invocation command. You can run batch mode in the foreground, in the background by specifying an ampersand at the end of the SAS command, or submit your application to the batch queue by using the `batch`, `at`, `nohup`, or `cron` Linux commands. (For more information, see the Linux man pages for the `batch`, `at`, `nohup`, or `cron` commands.) If you start your application with one of these Linux commands and you log off from your system, then your application completes execution.

SAS Command

Ask your system administrator for the command that invokes SAS in batch mode at your site. At many sites, the command to invoke SAS is `sas`, but a different command might have been defined during the SAS installation process at your site. This documentation assumes that SAS is invoked by the `sas` command.

Note: Before you start your SAS session, review the different techniques for interrupting and terminating your SAS session. For more information, see “Exiting or Interrupting Your SAS Session in Linux Environments” on page 18. Also, if you cannot stop your SAS session, contact your system administrator.

Invoking SAS in Batch Mode

To invoke SAS in batch mode, you must specify a filename in the SAS command. For example, if `weekly.sas` is the file that contains the SAS statements to be executed, and you want to specify the NODATE and LINESIZE system options, you would enter the following command:

```
sas weekly.sas -nodate -linesize 90
```

The command would run the program in the foreground. If you want to run the program in the background, add the ampersand to the end of the command:

```
sas weekly.sas -nodate -linesize 90 &
```

For more information, see “Running SAS in a Foreground or Background Process” on page 6.

SAS creates a .log file and a .lst file in the current directory that contains the log and procedure output.
Syntax of the SAS Command

The general form of the SAS command is as follows:

```
sas <-option1...-option-n> <filename>
sas –sysin filename
```

You can use these arguments with the SAS command:

- `option1` ... `-option-n`
  specifies SAS system options to configure your session. For more information, see [SAS Viya System Options: Reference](#). If you omit any options, the SAS (or site-specific) default options are in effect.

`filename`

specifies the name of the file containing the SAS program to be executed. Specifying a filename on the SAS command invokes a batch SAS session.

If the file is not in the current directory, specify its full pathname. A `.sas` extension is inferred if the full pathname is not given.

**Note:** This command can fail in cases where an option does not recognize `filename`. In this case, `-sysin filename` is required.

Submitting a Program to the Batch Queue

To submit your program to the batch queue, you can use the `batch`, `at`, `nohup`, or `cron` commands. For example, you could submit `weekly.sas` from your shell prompt as follows:

```
$ at 2am
sas weekly.sas
<control-D>
warning: commands will be executed using /usr/bin/sh
job 8400.a at Wed Mar 16 02:00:00 2011
$
```

If you create a file that contains the SAS command (for example, `cmdfile.sh`) that is necessary to run your program, then you can enter the following command at your shell prompt:

```
at 2am < cmdfile.sh
```

SAS sends the output to a file that has the same name as the program. The output file has an extension of `.lst`. The log file writes to a file with an extension of `.log`. Both of these files are written to your current directory. See the Linux man pages for these commands for more information about submitting jobs to the batch queue. For more information about routing output, see [SAS Viya Universal Printing](#).

If you submit a file in batch mode, then a line that is greater than 256 bytes is truncated. An explicit message about this truncation is written to the SAS log.

Writing Data from an External File Using Pipes

You can use a Linux pipe to write data from an external file to a SAS program. For example, suppose that your data resides in the external file `mydata` and your SAS program `myprog.sas` includes this statement:

```
infile stdin;
```
Issue this command to have `myprog.sas` read data from the external file `mydata`:
```
cat mydata | sas myprog.sas
```

## Running SAS in a Foreground or Background Process

Linux is a multiprocessing operating system, so you can run multiple processes at the same time. For example, you can have one process running in the foreground and three in the background.

A foreground process executes while you wait for the prompt. That is, you cannot execute additional commands while the current command is being executed. After you enter a command, the shell starts a process to execute the command. After the system executes the command, the shell displays the prompt and you can enter additional commands. The following is an example of SAS executing as a foreground process:
```
sas
```

Running in the foreground enables you to access standard input and output.

A background process executes independently of the shell. After you enter a command, the shell starts a process to execute the command, and then issues the system prompt. You can enter other commands or start other background processes without waiting for your initial command to execute. The following is an example of the command that is used to execute a background process:
```
sas&
```

*Note:* Both the C shell and the Korn shell include commands that enable you to move jobs among three possible states: running in the foreground, running in the background, and suspended.

You can run SAS in batch mode as either a foreground process or a background process. To run in interactive line mode, you must start SAS as a foreground process.

## Interactive Line Mode in Linux Environments

### Introduction to Interactive Line Mode

In interactive line mode, you enter SAS statements line by line in response to prompts issued by SAS. SAS reads the source statements from the terminal as you enter them. DATA and PROC steps execute when one of the following occurs:

- a RUN, QUIT, or DATALINES statement is entered
- another DATA or PROC statement is entered
- the ENDSAS statement is entered

Invoke interactive line mode from the command line by specifying the SAS command without supplying a SAS program name.

To use interactive line mode, you must run SAS in the foreground. When the session begins, SAS initializes and provides a session prompt, similar to the following screen.
Figure 1.1  Interactive Line Mode Session in SAS

Invoking SAS in Interactive Line Mode

To start an interactive line mode session, invoke SAS without specifying a filename:

```
sas
```

By default, SAS log and procedure output (if any) appear on your display as each step executes.

You can also invoke SAS in interactive line mode and pass parameters to it:

```
sas -sysparm 'A B C'
```

The value A B C is assigned to the SYSPARM macro variable.

After you invoke SAS, the 1? prompt appears, and you can begin entering SAS statements. After you enter each statement, a line number prompt appears.

Exiting SAS in Interactive Line Mode

You can end the session by pressing the EOF key, usually Ctrl-D (see “Using Control Keys” on page 19) or by issuing the ENDSAS statement:

```
endsas;
```

The session ends after all SAS statements have executed.
Example: Invoke an Interactive SAS Session

To invoke an interactive SAS session, without specifying any SAS system options, enter
sas

To specify the WORK and MEMSIZE system options when you invoke SAS, you might enter this command:

sas -work /saswork -memsize 4G

Executing Operating System Commands from Your SAS Session

Deciding Whether to Run an Asynchronous or Synchronous Task

You can execute Linux commands from your SAS session either asynchronously or synchronously. When you run a command as an asynchronous task, the command executes independently of all other tasks that are currently running. To run a command asynchronously, you must use the SYSTASK statement. See “SYSTASK Statement” in SAS Viya Statements: Reference for information about executing commands asynchronously.

When you execute one or more Linux commands synchronously, you must wait for those commands to finish executing before you can continue working in your SAS session. You can use the CALL SYSTEM routine, %SYSEXEC macro program statement, and X statement to execute Linux commands synchronously. The CALL SYSTEM routine can be executed with a DATA step. The %SYSEXEC macro statement can be used inside macro definitions, and the X statement can be used outside of DATA steps and macro definitions. For more information, see “CALL SYSTEM Routine” in SAS Viya Functions and CALL Routines: Reference and “%SYSEXEC Statement” in SAS Viya Macro Language: Reference.

Executing a Single Linux Command

Single Commands

To execute only one Linux command, you can enter the X statement, CALL SYSTEM routine, or %SYSEXEC macro statement as follows:

X command;
CALL SYSTEM ('command');
%SYSEXEC command;

Note: When you use the %SYSEXEC macro statement, if the Linux command that you specify includes a semicolon, you must enclose the Linux command in a macro quoting function. For more information about quoting functions, see SAS Viya Macro Language: Reference.

Example 1: Executing a Linux Command By Using the X Statement

You can use the X statement to execute the ls Linux command (in a child shell) as follows:
Example 2: Executing a Linux Command By Using the CALL SYSTEM Routine

Inside a DATA step, you can use the CALL SYSTEM routine to execute a `cd` command, which changes the current directory of your SAS session:

```sas
data _null_;  
call system ('cd /users/smith/report');  
run;
```

The search for any relative (partial) filenames during the SAS session now begins in the `/users/smith/report` directory. When you end the session, your current directory is the directory in which you started your SAS session.

For more information about the CALL SYSTEM routine, see “CALL SYSTEM Routine” in SAS Viya Functions and CALL Routines: Reference.

How SAS Processes a Single Linux Command

When you specify only one command, SAS checks to see whether the command is `cd`, `pwd`, `setenv`, or `umask` and, if so, executes the SAS equivalent of these commands. The SAS `cd` and `pwd` commands are equivalent to their Bourne shell counterparts. The SAS `setenv` command is equivalent to its C shell namesake. The SAS `umask` command is equivalent to the numeric mode of the `umask` command supported by the Bourne, Korn, and C shells. These four commands are built into SAS because they affect the environment of the current SAS session. When executed by SAS software, they affect only the SAS environment and the environment of any shell programs started by the SAS session. They do not affect the environment of the shell program that began your SAS session.

If the command is not `cd`, `pwd`, or `setenv`, SAS starts a shell in which it executes the command that you specified. The shell that is used depends on the SHELL environment variable. If the command is `umask`, but you do not specify a mask, then SAS passes the command to the shell in which the current SAS session was started. For more information about the `umask` command, see “Changing the File Permissions for Your SAS Session” on page 10.

Executing Several Linux Commands

Executing Linux Commands

You can also use the X statement, CALL SYSTEM routine, and `%SYSEXEC` macro statement to execute several Linux commands:

```sas
X 'command-1;...command-n';
CALL SYSTEM ('command-1;...command-n');
%SYSEXEC quoting-function(command-1;...command-n);
```

Separate each Linux command with a semicolon (;).

Note: When you use the `%SYSEXEC` macro statement to execute several Linux commands, because the list of commands uses semicolons as separators, you must enclose the string of Linux commands in a macro quoting function. For more information about quoting functions, see SAS Viya Macro Language: Reference.
Example: Executing Several Commands Using the %SYSEXEC Macro

The following code defines and executes a macro called `pwdls` that executes the `pwd` and `ls -l` Linux commands:

```sas
%macro pwdls;
   %syexec %str(pwd;ls -l);
%mend pwdls;
%pwdls;
```

This example uses `%str` as the macro quoting function.

How SAS Processes Several Linux Commands

When you specify more than one Linux command (that is, a list of commands separated by semicolons), SAS passes the entire list to the shell and does not check for the `cd`, `pwd`, `setenv`, or `umask` commands, as it does when a command is specified by itself (without semicolons).

For more information about how SAS processes the `cd`, `pwd`, `setenv`, or `umask` commands, see “How SAS Processes a Single Linux Command” on page 9.

Changing the File Permissions for Your SAS Session

At invocation, a SAS session inherits the file permissions from the parent shell. Any file that you create inherits these permissions. If you want to change or remove file permissions from within SAS, issue the following command in the X statement: `umask`. The `umask` command applies a new "mask" to a file, that is, it sets new file permissions for any new file that you create. In this way, the `umask` command can provide file security by restricting access to new files and directories for the current process.

The default value for `umask` varies. Some systems, like Secure Linux, use mandatory access control, and the `umask` default is the same with or without Secure Linux enabled. Other systems use 022 as the default. System administrators can set their own default value, and you can check your default and change it in your own `.kshrc`, `.cshrc`, or `.profile` files. These values affect all child processes that are executed in the shell. Any subsequent file that you create during the current SAS session inherits the permissions that you specified. The permissions of a file created under a given mask are calculated in octal representation.

Note: The value of a mask can be either numeric or symbolic. For more information about this command, see the Linux man page for `umask`.

In addition, you can use the `PERMISSION=` option in the `FILE` or `FILENAME` statement to control the permissions for individual output files. For more information, see “FILE Statement” in SAS Viya Statements: Reference.

Executing X Statements in Batch Mode

If you run your SAS program in batch mode and if your operating system supports job control, the program is suspended when an X statement within the program needs input from the terminal.

If you run your SAS program from the batch queue by submitting it with the `at` or `batch` commands, SAS processes any X statements as follows:

- If the X statement does not specify a command, SAS ignores the statement.
• If any Linux command in the X statement attempts to get input, it receives an end-of-file (standard input is set to /dev/null).

• If any Linux command in the X statement writes to standard output or standard error, the output is mailed to you unless it was previously redirected.

Customizing Your SAS Session By Using System Options

Specify SAS System Options

You can use SAS system options to customize your SAS environment. SAS options can be specified in one or more ways:

• in a configuration file
• in the SASV9_OPTIONS environment variable
• in the SAS command
• in an OPTIONS statement (either in a SAS program or an autoexec file). An autoexec file contains SAS statements that are executed automatically when SAS is invoked. The autoexec file can be used to specify some SAS system options, as well as to assign librefs and filerefs to data sources that are used frequently.

Any options that do not affect the initialization of SAS, such as CENTER and NOCENTER, can be specified and changed at any time.

Some options can be specified only in a configuration file, in the SASV9_OPTIONS variable, or in the SAS command. These options determine how SAS initializes its interfaces with the operating system and the hardware; they are often called configuration options. After you start a SAS session, these options cannot be changed. Usually, configuration files specify options that you would not change very often. In those cases when you need to change an option just for one job, specify the change in the SAS command.

Overriding the Default Value for a System Option

The default values for SAS system options will be appropriate for many of your SAS programs. However, you can override a default setting using one or more of the following methods:

configuration file

Modify your current configuration file (see “Order of Precedence for Processing SAS Configuration Files” on page 16 ) or create a new configuration file. Specify SAS system options in the file by preceding each with a hyphen. For ON or OFF options, just list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword identifying the option followed by the option value. All SAS system options can appear in a configuration file.

For example, a configuration file might contain these option specifications:

- nocenter
- verbose
- linesize 64
SASV9_OPTIONS environment variable
Specify SAS system options in the SASV9_OPTIONS environment variable before you invoke SAS. See “Defining Environment Variables in Linux Environments” on page 67.

Settings that you specify in the SASV9_OPTIONS environment variable affect SAS sessions that are started when the variable is defined.

For example, in the Korn shell, you would use the following:

```bash
export SASV9_OPTIONS='-fullstimer -nodate'
```

SAS command
Specify SAS system options in the SAS command. Precede each option with a hyphen:

```
sas -option1 -option2...
```

For ON or OFF options, list the keyword corresponding to the appropriate setting. For options that accept values, list the keyword that identifies the option, followed by the option value. Here is an example:

```
sas -nodate -work mywork
```

Settings that you specify in the SAS command last for the duration of the SAS session or, for those options that can be changed within the session, until you change them. All options can be specified in the SAS command.

OPTIONS statement within a SAS session
Specify SAS system options in an OPTIONS statement at any point within a SAS session. The options are set for the duration of the SAS session or until you change them. When you specify an option in the OPTIONS statement, do not precede its name with a hyphen (-). If the option has an argument, use = after the option name. Here is an example:

```
options nodate linesize=72;
options editcmd='/usr/bin/xterm -e vi';
```

For more information about the OPTIONS statement, see “OPTIONS Statement” in SAS Viya Statements: Reference. Not all options can be specified in the OPTIONS statement.

OPTIONS statement in an autoexec file
Specify SAS system options in an OPTIONS statement in an autoexec file. An autoexec file contains SAS statements that are executed automatically when SAS is invoked. The autoexec file can be used to specify some SAS system options, as well as to assign librefs and filerefs to data sources that are used frequently. For example, your autoexec file could contain the following statements:

```
options nodate pagesize=80;
filename rpt '/users/myid/data/report';
```

In general, use quotation marks to enclose filenames and pathnames specified in the OPTIONS statement or the System Options window. Do not use quotation marks otherwise. Any exceptions are discussed under the individual option. To shorten filenames and pathnames that you specify, you can use the abbreviations listed in Table 3.1 on page 37.
When the Value of a System Option Includes a Space

If the value of a system option includes a space, you must enclose the value in quotation marks on the command line or in a config file. The following examples show the correct syntax:

- bufsize '3 k';
- bottommargin '2 in';

If the value of a system option does not include a space, you do not need to enclose the value in quotation marks:

- bufsize 3k;
- bottommargin 2in;

How SAS Processes System Options That Are Set More Than Once

If the same system option is set more than once in the SAS command, in a configuration file, or in the SASV9_OPTIONS environment variable, only the most recent specification is the value that SAS uses. The other specifications are ignored. For example, the NOFULLSTIMER option is ignored in the following SAS command:

```sas
sas -nofullstimer -fullstimer
```

The NOFULLSTIMER option is ignored in the following configuration file:

```sas
-nofullstimer
-linesize 80
-fullstimer
```

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLPROC, SASAUTOS, or SASHELP system option more than once, the most recent specification is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system option. For more information, see the “APPEND= System Option” in SAS Viya System Options: Reference and “INSERT= System Option” in SAS Viya System Options: Reference.

How SAS Processes System Options That Are Set in Multiple Places

System Options Set in Multiple Places

If the same system option is set in more than one place, only the most recent specification is the value that SAS uses. The following places are listed in order of precedence. For example, a setting made in the OPTIONS statement overrides any other setting. However, if you set a system option using the SASV9_OPTIONS environment variable, this setting overrides only the setting for the same system option in your configuration file.

Order of Precedence When System Options Are Processed

The order of precedence when system options are processed is as follows:

1. OPTIONS statement (from a SAS session or job).
2. An autoexec file that contains an OPTIONS statement (after SAS initializes). (An autoexec file contains SAS statements that are executed automatically when SAS is
invoked. The autoexec file can be used to specify some SAS system options, as well as to assign librefs and filerefs to data sources that are used frequently.)

3. SAS command.

4. SASV9_OPTIONS environment variable.

5. Configuration files (before SAS initializes). For more information, see “Order of Precedence for Processing SAS Configuration Files” on page 16.

For example, if a configuration file specifies NOFULLSTIMER, you can override the setting in the SAS command by specifying the FULLSTIMER system option.

By default, if you specify the HELPLOC, MAPS, MSG, SAMPLC, SASAUTOS, or SASHELP system option more than one time, the most recent value that is specified is the value that SAS uses. If you want to add additional pathnames to the pathnames already specified by one of these options, you must use the APPEND or INSERT system options to add the new pathname. For more information, see the “APPEND= System Option” in SAS Viya System Options: Reference and “INSERT= System Option” in SAS Viya System Options: Reference.

---

Customizing Your SAS Session By Using Configuration and Autoexec Files

Introduction to Configuration and Autoexec Files

Defining Configuration and Autoexec Files
You can customize your SAS session by defining configuration and autoexec files. You can use these files to specify system options and to execute SAS statements automatically whenever you start a SAS session. SAS system options control many aspects of your SAS session, including output destinations, the efficiency of program execution, and the attributes of SAS files and libraries. For a complete description of SAS system options, see SAS Viya System Options: Reference.

For SAS 9.4, the configuration file is typically named sasv9.cfg, and the autoexec file is named autoexec.sas. These files typically reside in the directory where SAS was installed. By default, this directory is the %SASROOT directory.

You can have customized configuration and autoexec files in your user home directory. If you do, then SAS uses the customizations specified in these files when you start a SAS session. For more information about the order of precedence SAS uses when processing configuration files, see “Order of Precedence for Processing SAS Configuration Files” on page 16.

SAS system options can be restricted by a Linux system administrator, so that once they are set by the administrator, they cannot be changed by a user. A system option can be restricted globally, by group, and by user. For more information, see the configuration guide for the Linux environment on the Technical Support Web site, and see “Restricted Options” in SAS Viya System Options: Reference.

Using the AUTOEXEC System Option
The AUTOEXEC system option specifies the autoexec file. The autoexec file contains SAS statements that are executed automatically when you invoke SAS, or when you start another SAS process. The autoexec file can contain any SAS statements. For
example, your autoexec file can contain LIBNAME statements for SAS libraries that you access routinely in SAS sessions.

SAS looks for the AUTOEXEC system option in the following places. It uses the first AUTOEXEC system option that it finds.

- in the command line
- in the SASV9_OPTIONS environment variable
- in the configuration file

If neither the AUTOEXEC nor NOAUTOEXEC system option is found, SAS looks for the autoexec file in three directories in the following order:

1. your current directory
2. your home directory
3. the `SASROOT` directory

SAS uses the first autoexec file that it finds to initialize the SAS session. If you want to see the contents of the autoexec file for your session, use the ECHOAUTO system option when you invoke SAS.

**Inserting and Appending Autoexec Files**

You can concatenate files in your autoexec file by using the following system options with the AUTOEXEC system option: "INSERT= System Option" in *SAS Viya System Options: Reference* and "APPEND= System Option" in *SAS Viya System Options: Reference*. The autoexec file is always a text file. If your filename contains embedded blanks or special characters, you must enclose the filename in quotation marks. Otherwise, quotation marks are optional when one or more filenames are specified.

You can use the following syntax to concatenate autoexec files:

```
-autexec "(/path1/autoexec.sas /path2/autoexec.sas /path3/autoexec.sas)"
```

You can use the following syntax with the INSERT system option:

```
-insert autoexec "a.sas" -insert autoexec "b.sas"
```

You can use the following syntax with the APPEND system option:

```
-append autoexec "a.sas" -append autoexec "b.sas"
```

If any file in a concatenated autoexec list does not exist or cannot be opened (for example, if you are not authorized for Read access), SAS issues error messages to the log. SAS terminates without executing any of the files in the list. The final SAS exit code is 103, which indicates system start-up failure.

**Differences between Configuration and Autoexec Files**

The differences between configuration files and autoexec files are as follows:

- Configuration files can contain only SAS system option settings. Autoexec files can contain any valid SAS statement. For example, you might want to create an autoexec file that includes an OPTIONS statement to change the default values of various system options and LIBNAME and FILENAME statements for the SAS libraries and external files that you use most often.

- Configuration files are processed before SAS initializes while autoexec files are processed immediately after SAS initializes but before it processes any source statements. An OPTIONS statement in an autoexec file is equivalent to submitting an OPTIONS statement as the first statement of your SAS session.
Creating a Configuration File

To create a configuration file, follow these steps:

1. Use a text editor to write the SAS system options into a Linux file. Save the file as either sasv9.cfg or .sasv9.cfg.

2. Specify one or more system options on each line. Use the same syntax that you would use for specifying system options with the SAS command, but do not include the SAS command itself. For example, a configuration file might contain the following lines:
   -nocenter
   -verbose
   -linesize 64
   -work /users/myid/tmp

3. Save and close the configuration file.

Order of Precedence for Processing SAS Configuration Files

SAS is shipped with a default configuration file in the $SASROOT directory. Your on-site SAS personnel can edit this configuration file so that it contains whichever options are appropriate to your site.

You can also create one or more of your own configuration files. SAS reads option settings from each of these files in the following order:

1. sasv9.cfg in the $SASROOT directory.
2. sasv9_local.cfg in the $SASROOT directory.
3. .sasv9.cfg in your home directory. (Notice the leading period.)
4. sasv9.cfg in your home directory.
5. sasv9.cfg in your current directory.
6. any restricted configuration files. Restricted configuration files contain system options that are set by the site administrator and cannot be changed by the user. Options can be restricted globally, by group, or by user. For more information about restricted configuration files, see the configuration guide for the Linux environment.

For future releases of SAS, the names of these files will change accordingly.

For each system option, SAS uses the last setting that it encounters. Any other settings are ignored. For example, if the WORKPERMS system option is specified in sasv9.cfg in the $SASROOT directory and in sasv9.cfg in your current directory, SAS uses the value specified in sasv9.cfg in your current directory.

Specifying a Configuration File for SAS to Use

When you specify a configuration file for SAS to use, you bypass the search of the configuration files listed in “Order of Precedence for Processing SAS Configuration Files” on page 16.

Note: SAS still processes any restricted configuration files that exist. The settings in these files take precedence over the settings in the configuration file that you specify.
If you set both SASV9_OPTIONS and SASV9_CONFIG, SAS always uses SASV9_OPTIONS. SASV9_CONFIG is used only if you do not use --config in the command line.

To specify a configuration file, complete one of the following steps:

- specify a configuration file with the CONFIG system option in the SAS command:
  
  ```
  sas -config filename
  ```

- specify a configuration file in the SASV9_OPTIONS environment variable. See “Defining Environment Variables in Linux Environments” on page 67. For example, in the Korn shell, you would use the following:
  
  ```
  export SASV9_OPTIONS='-config filename'
  ```

- define the environment variable SASV9_CONFIG. See “Defining Environment Variables in Linux Environments” on page 67. For example, in the Korn shell, you would use the following:
  
  ```
  export SASV9_CONFIG=filename
  ```

`filename` is the name of a file that contains SAS system options.

If you have specified a configuration file in the SASV9_OPTIONS or SASV9_CONFIG environment variables, you can prevent SAS from using that file by specifying NOCONFIG in the SAS command.

If SAS cannot find SASV9_OPTIONS, the following message is written to the SAS log:

```
ERROR: Cannot open [/fullpath/filename]: No such file or directory.
```
### Exiting or Interrupting Your SAS Session in Linux Environments

**Methods for Exiting SAS in Interactive Line Mode**

Use one of the following methods to exit a SAS session in interactive line mode:

- Use `endsas;`.
- Use Ctrl+D if this control key sequence is your EOF command.

**Methods for Interrupting or Terminating SAS**

**Interrupting or Terminating SAS**

In addition to the methods for exiting SAS, SAS provides methods for interrupting or terminating a SAS session. SAS does not recommend that you use these methods until you have tried to exit SAS by one of the methods listed in “Methods for Exiting SAS in Interactive Line Mode” on page 18. Typically, you interrupt or terminate a SAS session in batch mode or interactive line mode if you have a process that is not ending as expected (such as for an infinite loop).

You can interrupt or terminate SAS in the following ways:

- Press the interrupt or quit control key. Interrupt displays a dialog box while quit forces a shutdown. Using the quit control key is not recommended.
- Enter the Linux `kill` command. Use this command when all other methods of exiting SAS have failed. By default, the kill command is `kill -15` (SIGTERM).

Using the Linux `kill -9` command on a SAS process that is running might corrupt data sets that are open for Write or Update access.
Interrupting a SAS Process
If you are running SAS in interactive line mode or in batch mode in the foreground, then you can use either of the following methods to interrupt SAS:

- Press the control key sequence that is set to interrupt in the shell that invoked SAS. In most cases, this control key sequence is Ctrl+C. See the man page for the `stty` command to determine the appropriate control key sequence for your environment.
- Use the `-SIGINT` option in the `kill` command. For more information, see “Using the Linux kill Command” on page 19.

The interrupt signal sends a request to the supervisor to handle an interrupt. The interrupt signal is not handled until a safe point in the code is reached. A safe point is one that allows the interrupt handler to be run safely. The supervisor responds as soon as possible with a prompt that requests what type of interrupt action you want to take. During this time, normal processing of a DATA step or PROC step is suspended.

Using Control Keys
Control keys enable you to interrupt or terminate your session by pressing the interrupt or quit key sequence. However, control keys can be used only when your SAS program is running in interactive line mode or in batch mode in the foreground. You cannot use control keys to stop a background job.

Note: You cannot use control keys to stop a batch job that has been submitted with the `batch`, `at`, `nohup`, or `cron` command.

Because control keys vary from system to system, issue the `stty` command to determine which key sends which signal. The `stty` command varies considerably among Linux operating environments, so check the Linux man page for `stty` on your system before using the command. Usually, one of these forms of the command prints all of the current terminal settings:

```
stty
stty -a
stty everything
```

The output should contain lines similar to these:

```
intr = ^C; quit = ^\; erase = ^H;
kill = ^U; eof = ^D; eol = ^@
```

The caret (^) represents the Ctrl key. In this example, Ctrl+C is the interrupt key and Ctrl +\ is the quit key. Quit is a more forceful termination and might result in data corruption.

Using the Linux kill Command
Note: Use the `kill` command only after you have tried all other methods to exit your SAS session.

The `kill` command sends an interrupt or terminate signal to SAS, depending on which signal you specify. You can use the `kill` command to interrupt or terminate a SAS session running in any mode. The `kill` command cannot be issued from within a SAS session. You must issue it from another terminal or from another window (if your terminal permits it).

The format of the `kill` command is:

```
kill <-signal-name> pid
```

To send the interrupt signal, specify `-SIGINT`. To send the terminate signal, specify `-SIGTERM`. Use the `ps` command and its options to determine the process identification number (pid) of the SAS session that you want to interrupt or terminate.
The results of using the `ps` command differ in different operating environments. See the Linux man page for your operating environment for specific information about the `ps` command and its options. Adding options helps determine which process you want to kill if you have more than one SAS process running. Also, servers leave a process identification number in their start-up directories. You can use this number with the `kill` command to identify the process that you want to stop.

The following table lists some of the important `kill` signals.

**Table 1.2 Description of Important Kill Signals**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SIGNULL</td>
<td>Checks access to process identifier.</td>
</tr>
<tr>
<td>1</td>
<td>SIGHUP</td>
<td>Causes SAS to terminate.</td>
</tr>
<tr>
<td>2</td>
<td>SIGINT</td>
<td>Causes SAS to interrupt the session. SIGINT is very similar to SIGQUIT.</td>
</tr>
<tr>
<td>3</td>
<td>SIGQUIT</td>
<td>Causes a more forceful shutdown than SIGTERM. It does not cause a core dump.</td>
</tr>
<tr>
<td>9</td>
<td>SIGKILL</td>
<td>Brings down SAS. Use this option only after all attempts to exit SAS have failed. Using SIGKILL can cause data corruption.</td>
</tr>
<tr>
<td>15</td>
<td>SIGTERM</td>
<td>Causes SAS to terminate.</td>
</tr>
</tbody>
</table>

For more information, see the Linux man pages for the `ps` and `kill` commands.

**Messages in the Console Log (STDOUT)**

If SAS encounters an error or warning condition when the SAS log is not available, then any messages that SAS issues are written to the console log. Normally, the SAS log is unavailable only early in SAS initialization and late in SAS termination.

If you are using the `-STDIO` option, the log is displayed in stderr, and the listing is displayed in stdout.

**Checkpoint Mode and Restart Mode**

**Overview of Checkpoint Mode and Restart Mode**

When used together, checkpoint mode and restart mode create an environment where batch programs that terminate before completing can be resubmitted without rerunning steps or labeled code sections that have already completed. Execution resumes with
either the DATA or PROC step or the labeled code section that was executing when the failure occurred.

A labeled code section is the SAS code that begins with `label:` outside of a DATA or PROC step and ends with the `RUN` statement that precedes the next `label:` that is outside of a DATA or PROC step. Labels must be unique. Consider using labeled code sections when you want to group DATA or PROC steps that might need to be grouped together because the data for one is dependent on the other.

The following example program has two labeled code sections. The first labeled code section begins with the label `readSortData:` and ends with the `RUN` statement for `PROC SORT DATA=mylib.mydata;`. The second labeled code section starts with the `label` `report:` and ends with the `RUN` statements for `PROC REPORT DATA=mylib.mydata;`.

```sas
READSORTDATA:
data mylib.mydata;
...more sas code...
run;

PROC SORT DATA=mylib.mydata;
...more sas code...
run;

REPORT:
PROC REPORT DATA=mylib.mydata;
...more sas code...;
run;
ENDREADSORTREPORT;
```

**Note:** The use of `label:` in checkpoint mode and restart mode is valid only outside of a DATA or PROC statement. Checkpoint mode and restart mode for labeled code sections are not valid for labels within a DATA step or macros.

Checkpoint mode and restart mode can be enabled for either DATA and PROC steps or for labeled code sections, but not both simultaneously. To use checkpoint mode and restart mode on a step-by-step basis, use the step checkpoint mode and the step restart mode. To use checkpoint mode and restart mode based on groups of code sections, use the label checkpoint mode and the label restart mode. Each group of code is identified by a unique label. If you use labels, all steps in a SAS program must belong to a labeled code section.

When checkpoint mode is enabled, SAS records information about DATA and PROC steps or labeled code sections in a checkpoint library. When a batch program terminates prematurely, you can resubmit the program in restart mode to complete execution. In restart mode, global statements are re-executed, macro definitions are recompiled, and macros are re-executed. SAS reads the data in the checkpoint library to determine which steps or labeled code sections completed. Program execution resumes with the step or the label that was executing when the failure occurred.

The checkpoint-restart data contains information only about the DATA and PROC steps or the labeled code sections that completed and the step or labeled code sections that did not complete. The checkpoint-restart data does not contain the following information:

- information about macro variables and macro definitions
- information about SAS data sets
- information that might have been processed in the step or labeled code section that did not complete
Note: Checkpoint mode is not valid for batch programs that contain the DM statement to submit commands to SAS. If checkpoint mode is enabled and SAS encounters a DM statement, checkpoint mode is disabled and the checkpoint catalog entry is deleted.

As a best practice, if you use labeled code sections, add a label at the end of your program. When the program completes successfully, the label is recorded in the checkpoint-restart data. If the program is submitted again in restart mode, SAS knows that the program has already completed successfully.

If a DATA or PROC step must be re-executed, you can add the global statement CHECKPOINT EXECUTE_ALWAYS immediately before the step. This statement tells SAS to always execute the following step without considering the checkpoint-restart data. It is applicable only to the step that follows the statement. For more information, see “CHECKPOINT EXECUTE_ALWAYS Statement” in SAS Viya Statements: Reference.

You enable checkpoint mode and restart mode for DATA and PROC steps by using system options when you start the batch program in SAS:

- STEPCHKPT system option enables checkpoint mode, which indicates to SAS to record checkpoint-restart data
- STEPCHKPTLIB system option identifies a user-specified checkpoint-restart library
- STEPRESTART system option enables restart mode, ensuring that execution resumes with the DATA or PROC step indicated by the checkpoint-restart library.

You enable checkpoint mode and the restart mode for labeled code sections by using these system options when you start the batch program in SAS:

- LABELCHKPT system option enables checkpoint mode for labeled code sections, which indicates to SAS to record checkpoint-restart data.
- LABELCHKPTLIB system option identifies a user-specified checkpoint-restart library
- LABELRESTART system option enables restart mode, ensuring that execution resumes with the labeled code section indicated by the checkpoint-restart library.

If you use the Work library as your checkpoint-restart library, you can use the CHKPTCLEAN system option to have the files in the Work library erased after a successful execution of your batch program.

For information, see the following system options in SAS Viya System Options: Reference:

- “STEPCHKPT System Option” in SAS Viya System Options: Reference
- “STEPCHKPTLIB= System Option” in SAS Viya System Options: Reference
- “STEPRESTART System Option” in SAS Viya System Options: Reference
- “LABELCHKPT System Option” in SAS Viya System Options: Reference
- “LABELCHKPTLIB= System Option” in SAS Viya System Options: Reference
- “LABELRESTART System Option” in SAS Viya System Options: Reference
- “CHKPTCLEAN System Option” in SAS Viya System Options: Reference
Requirements for Using Checkpoint Mode and Restart Mode

In order for checkpoint mode and restart mode to work successfully, the number and order of the DATA and PROC steps or labeled code sections in the batch program must not change between SAS invocations. By specifying the ERRORABEND and ERRORCHECK system options when SAS starts, SAS terminates for most error conditions in order to maintain valid checkpoint-restart data.

The checkpoint-restart library can be a user-specified library or, if no library is specified, the checkpoint-restart data is saved to the Work library. Always start SAS with the NOWORKTERM and NOWORKINIT system options regardless of whether the checkpoint-restart data is saved to a user-specified library or to the Work library. SAS writes the name of the Work library to the SAS log.

Operating Environment Information

Under UNIX and z/OS operating environments, consider always assigning a checkpoint-restart library when you use the STEPCHKPT option or the LABELCHKPT option. If your site sets the CLEANWORK utility to run at regular intervals, data in the Work library might be lost. Under z/OS, it might not be practical for your site to reuse the Work library in a batch session.

The labels for labeled code sections must be unique. If SAS enters restart mode for a label that is a duplicate label, SAS starts at the first label. The code between the duplicate labels might rerun needlessly.

Setting Up and Executing Checkpoint Mode and Restart Mode

To set up checkpoint mode and restart mode, make the following modifications to your batch program:

• Add the CHECKPOINT EXECUTE_ALWAYS statement before any DATA and PROC steps that you want to execute each time the batch program is submitted.

• If your checkpoint-restart library is a user-defined library, you must add the LIBNAME statement that defines the checkpoint-restart libref as the first statement in the batch program. If you use the Work library as your checkpoint library, no LIBNAME statement is necessary.

Once the batch program has been modified, you start the program using the appropriate system options:

• For checkpoint-restart data that is saved in the Work library, start a batch SAS session that specifies these system options:

  • SYSIN, if required in your operating environment, names the batch program.
  • STEPCHECKPT or LABELCHECKPT enables checkpoint mode.
  • NOWORKTERM saves the Work library when SAS ends.
  • NOWORKINIT does not initialize the Work library when SAS starts.
  • ERRORCHECK STRICT puts SAS in syntax-check mode when an error occurs in the LIBNAME, FILENAME, %INCLUDE, and LOCK statements.
  • ERRORABEND specifies whether SAS terminates for most errors.
  • CHKPTCLEAN specifies whether to erase files in the Work library and delete the Work library if the batch program runs successfully.
In the Windows operating environment, the following SAS command starts a batch program in checkpoint mode using the Work library as the checkpoint-restart library:

```
sas -sysin 'c:\mysas\myprogram.sas' -stepchkpt -noworkterm -noworkinit
     -errorcheck strict -errorabend -chkptclean
```

- For checkpoint-restart data that is saved in a user-specified library, start a batch SAS session that includes these system options:
  - SYSIN, if required in your operating environment, names the batch program.
  - STEPCHKPT or LABELCHKPT enables checkpoint mode.
  - STEPCHKPTLIB or LABELCHKPTLIB specifies the libref of the library where SAS saves the checkpoint-restart data.
  - NOWORKTERM saves the Work library when SAS ends.
  - NOWORKINIT does not initialize the Work library when SAS starts.
  - ERRORCHECK STRICT puts SAS in syntax-check mode when an error occurs in the LIBNAME, FILENAME, %INCLUDE, and LOCK statements.
  - ERRORABEND specifies whether SAS terminates for most errors.

In the Windows operating environment, the following SAS command starts a batch program in checkpoint mode using a user-specified checkpoint-restart library:

```
sas -sysin 'c:\mysas\myprogram.sas' -labelchkpt -labelchkptlib mylibref
     -noworkterm -noworkinit -errorcheck strict -errorabend
```

In this case, the first statement in MyProgram.sas is the LIBNAME statement that defines the `MyLibref` libref.

### Restarting Batch Programs

To resubmit a batch SAS session using the checkpoint-restart data that is saved in the Work library, include these system options when SAS starts:

- SYSIN, if required in your operating environment, names the batch program.
- STEPCHKPT or LABELCHKPT continues checkpoint mode.
- STEPRESTART or LABELRESTART enables restart mode, indicating to SAS to use the checkpoint-restart data.
- NOWORKINIT starts SAS using the Work library from the previous SAS session.
- NOWORKTERM saves the Work library when SAS ends.
- ERRORCHECK STRICT puts SAS in syntax-check mode when an error occurs in the LIBNAME, FILENAME, %INCLUDE, and LOCK statements.
- ERRORABEND specifies whether SAS terminates for most errors.
- CHKPTCLEAN specifies whether to erase files in the Work library if the batch program runs successfully.

In the Windows operating environment, the following SAS command resubmits a batch program whose checkpoint-restart data was saved to the Work library:

```
sas -sysin 'c:\mysas\mysasprogram.sas' -stepchkpt -steprechpt -noworkterm
     -noworkinit -errorcheck strict -errorabend -chkptclean
```
By specifying the NOWORKTERM system options and either the STEPCHKPT or LABELCHKPT system option, checkpoint mode continues to be enabled once the batch program restarts.

To resubmit a batch SAS session using the checkpoint-restart data that is saved in a user-specified library, include these system options when SAS starts:

- SYSIN, if required in your operating environment, names the batch program.
- STEPCHKPT or LABELCHKPT continues checkpoint mode.
- STEPRESTART or LABELRESTART enables restart mode, indicating to SAS to use the checkpoint-restart data.
- STEPCHKPTLIB or LABELCHKPTLIB specifies the libref of the checkpoint-restart library.
- NOWORKTERM saves the Work library when SAS ends.
- NOWORKINIT does not initialize the Work library when SAS starts.
- ERRORCHECK STRICT puts SAS in syntax-check mode when an error occurs in the LIBNAME, FILENAME, %INCLUDE, and LOCK statements.
- ERRORABEND specifies whether SAS terminates for most errors.

In the Windows operating environment, the following SAS command resubmits a batch program whose checkpoint-restart data was saved to a user-specified library:

```
sas -sysin 'c:\mysas\mysasprogram.sas' -labelchkpt -labelrestart -labelchklib
  -noworkterm -noworkinit mylibref -errorcheck strict -errorabend
```

### Ending a Process That Is Running as a SAS Server

If you are using another SAS server, use the Linux scripts that shipped with the servers to stop the process. You can also use these scripts to start (or restart) a server, as well as determine whether the server is already running. For more information about these scripts, contact your site administrator.

*Note:* If the server does not respond to the Linux script, then you can use the `kill` command to end the server process. For more information, see “Using the Linux kill Command” on page 19.

### Interrupting a SAS Process and the Underlying DBMS Process

*CAUTION:* Interrupting a SAS process and the underlying DBMS process might kill all jobs that are running on your DBMS. Interrupting a SAS or DBMS process should be an exception. Use care when you construct your queries. For example, if SAS sends SQL to an RDBMS, there is no way to interrupt the SQL statements because SAS no longer has control of them. The statements are running in the RDBMS.
When you interrupt a SAS process, you might terminate the current query. If you are using the current query to create a new data set, then the data set is still created even if the query is terminated. If you are using the current query to overwrite a data set, then the data set is not overwritten if the query is terminated. In most cases, you do not receive a warning that the query did not complete.

*Note:* In this section, SAS process refers to a series of events. It is not the process on the operating system. When you interrupt or terminate a SAS process, the process on the operating system might still be running.

In many cases (such as using Oracle in Linux environments), when you interrupt or terminate a query on a server, the following processes stop:

- Processing of current extractions. For example, if you forgot to include a WHERE clause in your SQL query and are now extracting one billion rows into SAS, issuing an interrupt stops the SAS process and the extract step in the DBMS.

- Processing of queries that are in progress on the server. For example, you might have a very complex extract query that runs for a long time before producing a result. Issuing an interrupt stops the SAS and DBMS processes. As a result, the complex extract query running on your DBMS server is interrupted and terminated.

- Processing an update, delete, or insert. For example, you are updating, deleting, or inserting many rows in your DBMS. An interrupt stops the SAS and DBMS processes.
Chapter 2
Connecting to SAS Cloud Analytic Services from the Command Line

How to Connect to SAS Cloud Analytic Services
Requirements to Connect to SAS Cloud Analytic Services
Example Program that Connects to SAS Cloud Analytic Services
Comparison of Batch Modes and Interactive Modes

How to Connect to SAS Cloud Analytic Services

Requirements to Connect to SAS Cloud Analytic Services
Before you can run a program that connects to the SAS Cloud Analytic Services (CAS) server, your user credentials must have permission to access the CAS server. Your system administrator typically manages users and permissions. For more information, see SAS Cloud Analytic Services: Authorization.

In addition, the CAS server must be running before your program can connect to it. Your system administrator typically manages the servers that are used by SAS. For more information, see SAS Viya Administration: Servers.

To run a program that connects to the CAS server from the command line:

• Specify the CAS server. You can provide this value with the CASHOST= option in the OPTIONS statement.
  
  Note: In SAS Studio, the CAS server is set automatically and does not need to be specified.

• Specify the server port value. The default value for the server port is 5570. You can provide this value with the CASPORT= option in the OPTIONS statement.
  
  Note: In SAS Studio, the CAS server port value is set automatically and does not need to be specified.

• Specify the CAS statement in your SAS program to start your CAS session.

• As a best practice, at the end of the program, specify the TERMINATE option in the CAS statement. For example, this statement ends the CAS session called mysess:

  cas mysess terminate;

When you run a program from the command line, such as when you run a program at a scheduled time overnight, be certain to specify the CAS server value and CAS server port value. Otherwise, your SAS program does not connect to the CAS server.
Example Program that Connects to SAS Cloud Analytic Services

The following code connects to the CAS server:

```sas
/* These options must be included to connect to the CAS server in batch mode */
options cashost='host-name' casport=xxxxx;

cas casauto;
caslib _all_ assign;

proc cas;
  session casauto;

  /* Load source data (Cars) into a table in CAS */
  table.loadTable result=r /
    caslib="hps"
    path="carssashelp.sashdat"
    casOut={name="cars",replace=true};

  /* View variable information */
  table.columnInfo / table="cars";

  /* View table. The TO= parameter is similar to OBS=. */
  table.fetch /
    format=true
    sortby={
      {name="make",order="descending"},
      {name="model",order="descending"}
    }
    table="cars.sashdat"
    to=10;
  run;
  quit;

cas casauto terminate;
```

See Also
CAS statement

Comparison of Batch Modes and Interactive Modes

There are two methods of running a program in batch mode. One method is to run a program from the command line without viewing it in an interface. The other is to run the program as a background submission from SAS Studio. To do this, you select the program name and run it without opening it in the Program Editor.
### Table 2.1 Comparison of Batch Mode from the Command Line or in SAS Studio

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| From the command line                       | Issue the SAS command, and specify the name of the program to run. To run the program mySASprog.sas from the command line, issue this command:  \sas\ mySASprog  
You can specify options in the invocation command:  \sas\ -lang u8 mySASprog  
The log and results are in separate files that begin with the name of the program that you are running. For example, if you run the program mySASprog.sas, then the log and results are in the files mySASprog.log and mySASprog.lst, respectively. |
| In SAS Studio (as a background submission)  | Right-click on a program in the navigation pane, and submit the program to run in the background. This runs the program in SAS Studio without first opening it in the Program Editor.  
The log and results appear in SAS Studio.  
For more information, see “Using SAS Studio” in *SAS Studio: User’s Guide*. |

### Table 2.2 Comparison of Interactive Mode from the Command Line or in SAS Studio

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| From the command line                       | Invoke the command that starts SAS without specifying a SAS program to run. Specify any options that you want enabled during your interactive SAS session:  \sas\ -lang u8  
This begins the interactive session. For more information, see “Introduction to Interactive Line Mode” on page 6. |
| In SAS Studio                                | Click the interactive icon ![Interactive Icon](image)  
When this icon is active, you can run selected sections of code within a SAS program. |
Chapter 3
Using SAS Files and Libraries

Introduction to SAS Files, Libraries, and Engines in Linux Environments ........................................... 32
  SAS Files ......................................................................................................................................................... 32
  SAS Libraries and Librefs ................................................................................................................................. 32
  Engines ............................................................................................................................................................ 32

Common Types of SAS Files in Linux Environments ..................................................................................... 33
  SAS Data Sets .................................................................................................................................................. 33

Sharing SAS Files in Linux ................................................................................................................................... 33
  Sharing SAS Files ............................................................................................................................................ 33
  Options to Use for File Locking: SAS Files ........................................................................................................ 34
  File Locking for SAS Files: The FILELOCKS Statement Option .................................................................... 34
  File Locking for SAS Files: The FILELOCKS System Option ........................................................................ 34
  Waiting to Use a Locked File ............................................................................................................................... 34
  When FILELOCKS=CONTINUE ....................................................................................................................... 34

Referring to SAS Files By Using Librefs in Linux .............................................................................................. 35
  What Is a Libref? ............................................................................................................................................... 35
  Assigning Librefs ............................................................................................................................................ 35
  Permanently Assigning a Libref ........................................................................................................................ 35
  Accessing a Permanent SAS Library By Using a Libref .................................................................................. 36

Specifying Pathnames in Linux .......................................................................................................................... 36
  Rules for Specifying Directory and Pathnames ................................................................................................ 36
  Example 1: Access a File That Is Not in the Current Directory ...................................................................... 36
  Example 2: Access a File in the Current Directory ......................................................................................... 36
  Valid Character Substitutions in Pathnames ..................................................................................................... 37

Assigning a Libref to Several Directories (Concatenating Directories) .............................................................. 37
  Introduction to Concatenating Directories ........................................................................................................ 37
  How SAS Accesses Concatenated Libraries ..................................................................................................... 38
  Accessing Files for Input and Update ............................................................................................................... 38
  Accessing Files for Output .................................................................................................................................. 38
  Accessing Data Sets with the Same Name ......................................................................................................... 38

Using Environment Variables as Librefs in Linux Environments ...................................................................... 39

Librefs Assigned by SAS in Linux Environments .............................................................................................. 39

Sasuser Library .................................................................................................................................................... 40
  What Is the Sasuser Library? ............................................................................................................................ 40
  Contents of the Sasuser Library ........................................................................................................................ 40
  Sasuser.Profile Catalog .................................................................................................................................... 41
  Sasuser.Registry Catalog ................................................................................................................................... 42
Introduction to SAS Files, Libraries, and Engines in Linux Environments

SAS Files

What Is a SAS File?
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 Linux, a SAS file is a specially structured Linux file. Although the Linux operating environment manages the file for SAS by storing it, the operating system cannot process it because of the structure built into the file by SAS. 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.

Case Sensitivity in Data Set Names
In Linux operating environments, SAS data set names are written in all lowercase characters. Because of this requirement, SAS reads only data set names that are written in all lowercase characters.

If you use the Linux utilities `mv` or `cp` to rename SAS data set names with uppercase or mixed-case characters, SAS can no longer read the data set names.

Linux is case sensitive. Therefore, a data set named `xxx.sas7bdat` is not the same as a data set named `XXX.sas7bdat`. In fact, both of these data sets can reside in the same directory as completely different data sets.

SAS Libraries and Librefs

SAS files are stored in SAS libraries. A SAS library is a collection of SAS files within a Linux directory. Any Linux directory can be used as a SAS library. SAS stores temporary SAS files in a Work library, which is automatically defined for you. You must specify a library for each permanent SAS file. For more information, see “Work Library” on page 42.

SAS libraries can be identified with librefs. A libref is a name by which you reference the directory in your application. For more information about how to assign a libref, see “Referring to SAS Files By Using Librefs in Linux” on page 35.

Engines

SAS files and SAS libraries are accessed through engines. An engine is a set of routines that SAS must use to access the files in the library. SAS 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 SAS which engine to use. For others, SAS automatically
chooses the appropriate engine. The engine that is used to create a SAS data set determines the format of the file.

Common Types of SAS Files in Linux Environments

SAS Data Sets

SAS Data Files (Member Type DATA)
The SAS data file is probably the most frequently used type of SAS file. These files have the extension `sas7bdat`. SAS data files are created in the DATA step and by some SAS procedures.

Native SAS data files store data values and file metadata in files formatted by SAS. Native SAS data files that are 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 Linux, indexes are stored as separate files, but are treated as integral parts of the SAS data file by SAS.

CAUTION:
Do not remove index files using Linux 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.

SAS Views (Member Type VIEW)
A SAS view contains only the information needed to derive the data values and the descriptor information. Depending on how the SAS view is created, the actual data can be located in other SAS data sets or in other vendors’ files.

Views can be of two kinds:

- Native SAS views contain information about data in one or more SAS data files or SAS views. This type of view is created with the SQL procedure or DATA step.

- Interface SAS views contain information about data formatted by other software products such as a database management system.

Sharing SAS Files in Linux

Sharing SAS Files

If more than one SAS process has Write access to a SAS file (a data set, catalog, library, and so on) at the same time, you would get unpredictable results if the file was updated. SAS locks the file to prevent more than one user from having Write access to a file. When one SAS process opens a file with Write access, other processes are blocked from Write access until the first process closes the file. SAS provides statement and system options to override this file protection. However, in almost all cases, you should leave file protection turned on.
Options to Use for File Locking: SAS Files

You can turn off file locking for SAS files in the following ways:

- Use the FILELOCKS option in the LIBNAME statement.
- Use the FILELOCKS system option.

File Locking for SAS Files: The FILELOCKS Statement Option

By default, SAS restricts Write access to one user. The FILELOCKS option in the LIBNAME statement overrides the default and allows multiple users to have Write access to a file. SAS files that are opened under the libref in the LIBNAME statement are the files that are locked. Multiple users have Read access to files.

The FILELOCKS statement option applies to most (but not all) of the SAS I/O files (for example, data sets and catalogs) that are opened under the libref in the LIBNAME statement.

For more information, see “LIBNAME Statement (Linux)” in SAS Viya Statements: Reference.

File Locking for SAS Files: The FILELOCKS System Option

By default, SAS restricts Write access to one user. The FILELOCKS system option overrides this default for both SAS files and external files and allows multiple users to have Write access to a file. The FILELOCKS system option enables you to apply a behavior globally to individual files that are opened.

You can use the FILELOCKS system option in the OPTIONS statement or on the command line. You can specify multiple instances of the FILELOCKS system option. Each instance is added to an internal table of paths and settings. The FILELOCKS system option applies to most (but not all) of the SAS I/O files (for example, data sets and catalogs) that are opened under the libref in the LIBNAME statement. For more information, see “FILELOCKS System Option” in SAS Viya System Options: Reference and “LIBNAME Statement (Linux)” in SAS Viya Statements: Reference.

Waiting to Use a Locked File

If you want to use a SAS file that is locked by another process, you can use the FILELOCKWAIT option in the LIBNAME statement to specify how long SAS waits for the locked file to become available to another process. The FILELOCKWAIT statement option affects only those files that are opened under the libref in a LIBNAME statement. For more information, see “LIBNAME Statement (Linux)” in SAS Viya Statements: Reference.

When FILELOCKS=CONTINUE

By default, SAS restricts Write access to one user. When you use the FILELOCKS=CONTINUE option, SAS fails to open a file if that file is locked by another user, and writes an error message to the log. However, if SAS returns a message that identifies some other error, then SAS disregards the lock on the file, opens the file, and continues to execute the job.
Referring to SAS Files By Using Librefs in Linux

What Is a Libref?

A libref is an alias that you can use to refer to a library during a SAS session or job. You will probably want to use a libref when one of the following is true:

• 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. For more information, see “Assigning a Libref to Several Directories (Concatenating Directories)” on page 37.

Assigning Librefs

Methods for Assigning Librefs

You can use the following methods to assign a SAS libref:

• LIBNAME statement

• LIBNAME function

A libref assignment remains in effect for the duration of the SAS job, session, or process unless you clear the libref or use the same libref in another LIBNAME statement or LIBNAME function.

If you assign a libref from a SAS process, that libref is valid only within that SAS process. If you clear a libref from within a SAS process, that libref is not cleared from other SAS processes.

Using the LIBNAME Statement

The LIBNAME statement identifies a SAS library to SAS, associates an engine with the library, enables you to specify options for the library, and assigns a libref to it. For information, see “LIBNAME Statement (Linux)” in SAS Viya Statements: Reference.

Using the LIBNAME Function

The LIBNAME function takes the same arguments and options as the LIBNAME statement. For more information, see “LIBNAME Function” in SAS Viya Functions and CALL Routines: Reference.

Permanently Assigning a Libref

You might want to save a libref so that it is valid between SAS sessions. You can assign a libref permanently by using one of the following methods:

• Specify the LIBNAME statement or LIBNAME function in an autoexec file. For more information, see “LIBNAME Statement (Linux)” in SAS Viya Statements:
Use environment variables as librefs. Include these environment variables in your start-up files so that these variables are set when SAS is invoked.

Accessing a Permanent SAS Library By Using a Libref

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

- as the first level of a two-level SAS filename:
  
  `libref.member-name`
  
  where `libref` is the first-level name referring to the directory where the file is stored, and `member-name` is the name of the file being read or created.

- as the value of the USER= option. For more information, see “Using One-Level Names to Access Permanent Files (User Library)” on page 43.

For example, these SAS statements access the Final data set in the Sales library that is stored in the `/users/myid/mydir` directory:

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

Specifying Pathnames in Linux

Rules for Specifying Directory and Pathnames

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

Specify directory and file pathnames in quotation marks. The level of specification depends on your current directory.

Example 1: Access a File That Is Not in the Current Directory

If `/u/2011/budgets` is not your current directory, then to access the data file named May, you must specify the entire pathname:

```sas
data '/u/2011/budgets/may';
```

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

```sas
libname budgets '/u/2011/budgets';
data budgets.may;
```

Example 2: Access a File in the Current Directory

If `/u/2011/budgets` is your current directory, you could specify only the filenames:

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

Note: If you omit the quotation marks, then SAS assumes that these data sets are stored in the Work directory.

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

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

**Valid Character Substitutions in Pathnames**

You can use the character substitutions in the following table to specify pathnames.

**Table 3.1 Character Substitutions in Pathnames**

<table>
<thead>
<tr>
<th>Characters</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>~/</td>
<td>$HOME/</td>
</tr>
<tr>
<td></td>
<td>Can be used only at the beginning of a pathname.</td>
</tr>
<tr>
<td>~name/</td>
<td>name's home directory (taken from file <code>/etc/passwd</code>). Can be used only at the beginning of a pathname.</td>
</tr>
<tr>
<td>!sasroot</td>
<td>name of the sasroot directory</td>
</tr>
<tr>
<td></td>
<td>Specified only at the beginning of a pathname.</td>
</tr>
<tr>
<td>.</td>
<td>current working directory.</td>
</tr>
<tr>
<td>..</td>
<td>parent of current working directory.</td>
</tr>
<tr>
<td>$VARIABLE</td>
<td>environment variable VARIABLE.</td>
</tr>
</tbody>
</table>

**Assigning a Libref to Several Directories (Concatenating Directories)**

**Introduction to Concatenating Directories**

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

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

```sas
libname income ('/u/2011/revenue', '/u/2011/costs');
```

This statement indicates that the two directories, `/u/2011/revenue` and `/u/2011/costs`, are to be treated as a single SAS library.
If you have already assigned librefs to your SAS libraries, you can use these librefs to indicate that you want to concatenate the libraries, as in this example:

```sas
libname income ('/u/2011/corpsale', '/u/2011/retail');
libname costs ('/u/2011/salaries', '/u/2011/expenses');
libname profits (income, costs, '/u/2011/capgain');
```

This statement indicates that the five directories, `/u/2011/corpsale`, `/u/2011/retail`, `/u/2011/salaries`, `/u/2011/expenses`, and `/u/2011/capgain`, are to be treated as a single SAS library called `profits`.

### How SAS Accesses Concatenated Libraries

When you concatenate SAS libraries, SAS uses a protocol for accessing the libraries, which depends on whether you are accessing the libraries for read, write, or update.

SAS uses the protocol 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:

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

#### 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, SAS writes the `old.species` data set to the first directory (`mysasdir`), and replaces any existing data set with the same name:

```sas
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. When you execute the following statements, assume that `test.species` originally exists only in the second directory, `mysasdir`. Execute the following statements:

```sas
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, SAS 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 directory (mysasdir), 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 Environment Variables as Librefs in Linux Environments

An environment variable can be used as a libref. The variable name must be in all uppercase 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.

Note: SAS on Linux does not support the assignment of the User libref using the USER environment variable.

Supposed that you want to use the library in `/users/mydir/educ`, and you want to refer to it with the EDUC environment variable. You can define the variable at the following times:

- Before you invoke SAS. See “Defining Environment Variables in Linux Environments” on page 67. For example, in the Korn shell, you would use:
  ```
  export EDUC=/users/mydir/educ
  ```

- After you invoke SAS. You can use the X statement (see “Executing Operating System Commands from Your SAS Session” on page 8) and the SAS `setenv` command:
  ```
  x setenv EDUC /users/mydir/educ;
  ```

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 libraries, SAS uses the libref.

Librefs Assigned by SAS in Linux Environments

SAS automatically defines three 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 `1SASROOT` directory.
Sasuser contains SAS catalogs that enable you to customize features of SAS (such as window size, font settings, and printer entries) 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.

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

These librefs and the library libref are reserved librefs. Sasuser and Work have operating system dependencies.

---

**Sasuser Library**

**What Is the Sasuser Library?**

The Sasuser library contains the customizations (such as window size and positioning, colors, fonts, and printer entries) that you specified for your SAS session. When you invoke SAS, it looks for the Sasuser directory to find these customizations. If this directory does not exist, SAS uses the SASUSER system option to create it. The default directory is set in the system configuration file (sasv9.cfg) and is usually similar to the following:

```
-sasuser ~/sasuser.v94
```

This specification tells SAS to create a directory for the Sasuser libref in your home directory. To determine the value of this directory for your system, use PROC OPTIONS or libname sasuser LIST.

You can permit Read-Only access to the Sasuser library by using the RSASUSER system option. For more information, see “SASUSER= System Option” in SAS Viya System Options: Reference and “RSASUSER System Option” in SAS Viya System Options: Reference.

After the Sasuser library has been created, SAS automatically assigns the same Sasuser 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, SAS re-creates it the next time you start a session. Because SAS assigns the libref for you, you do not need to use a LIBNAME statement before referencing this library.

**Contents of the Sasuser Library**

Your customizations are stored in one of the following locations in the Sasuser library:

- “Sasuser.Profile Catalog” on page 41
- “Sasuser.Registry Catalog” on page 42
Overview of the Sasuser.Profile Catalog

The Sasuser.Profile catalog is the profile.sas7bcat file in your Sasuser library. This catalog enables you to customize how you work with SAS. SAS uses this catalog to store function key definitions, fonts for graphics applications, window attributes, and other information from interactive windowing procedures. SAS saves changes that you make to function key definitions, window attributes (such as size, color, and position), PMENU settings, and so on, in the Sasuser.Profile catalog. The information in the Sasuser.Profile catalog is accessed automatically by SAS when you need it for processing.

How SAS Accesses the Sasuser.Profile Catalog

SAS creates the Sasuser.Profile catalog the first time it tries to find it and it does not exist. If you are using an interactive windowing environment, then creating the Sasuser.Profile catalog occurs during system initialization in your first SAS session. If you are using one of the other modes of execution, the Sasuser.Profile catalog is created the first time you execute a SAS procedure that requires it.

When the Sasuser.Profile Catalog Does Not Exist

If the Sasuser.Profile catalog does not exist, then, at invocation, SAS checks for the Sashelp.Profile catalog. (This catalog exists only if you have copied your Sasuser.Profile catalog to the Sashelp library.) If the Sashelp.Profile catalog exists, then SAS copies it to the Sasuser library, and this catalog becomes your new Sasuser.Profile catalog. If the Sashelp.Profile catalog does not exist, then SAS creates Sasuser.Profile using the default settings for a SAS session. The default settings for your SAS session are stored in several catalogs in the Sashelp library. If you make changes to key settings or other options, then the new information is stored in your Sasuser.Profile catalog. To restore the original default settings to the Sasuser.Profile catalog, use the CATALOG procedure or the CATALOG window to delete entries from your Sasuser.Profile catalog. By default, SAS then uses the corresponding entry from the Sashelp library.

Checking for an Uncorrupted Sasuser.Profile Catalog

When you invoke SAS, SAS checks for an existing, uncorrupted Sasuser.Profile catalog. If the catalog is found, SAS copies the Sasuser.Profile catalog to Sasuser.Profbak. This backup catalog is used if Sasuser.Profile becomes corrupted.

If you invoke SAS and determine that your customizations have been lost, then your Sasuser.Profile catalog is either corrupted or locked by another SAS session that was started with the same user ID. If either of these conditions are true, then SAS uses Sashelp.Profile or Sasuser.Profbak to replace the locked or corrupted Sasuser.Profile catalog.

If Your Sasuser.Profile Catalog Is Locked or Corrupted

If your Sasuser.Profile catalog is locked, then SAS checks for Sashelp.Profile. If Sashelp.Profile exists, SAS copies it to Work.Profile, and then saves the customizations to the Work.Profile catalog instead of the Sasuser.Profile catalog. This Work.Profile catalog is used for the duration of the SAS session. Because the contents of the Work directory are temporary, any customizations that you save to the Work.Profile catalog are lost at the end of the SAS session.
If your Sasuser.Profile catalog is corrupted, SAS copies the corrupted catalog to Sasuser.Badpro.SAS, and then checks for Sasuser.Profbak. If Sasuser.Profbak exists, then SAS copies it to Sasuser.Profile. Any changes that you made to the Sasuser.Profile catalog during the previous session are lost. If your Sasuser.Profile catalog is being used by multiple SAS sessions, then you can specify the RSASUSER system option to permit Read-Only access to the Sasuser library. Because this permission is Read-Only, you will not be able to save any customizations to your Sasuser.Profile catalog during that SAS session.

**Sasuser.Registry Catalog**

**Overview of the Sasuser.Registry Catalog**

The Sasuser.Registry catalog is the registry.sas7bitm file in your Sasuser library. If you change any Universal Printing entries or libref assignments during a SAS session, then SAS saves the changes in the Sasuser.Registry catalog.

**How SAS Accesses the Sasuser.Registry Catalog**

At invocation, SAS looks in the **Sasuser** directory to see whether it can write to the Sasuser.Registry catalog. If SAS cannot write to this catalog, then the following warning appears in the SAS log:

```
WARNING: Unable to open SASUSER.REGISTRY. WORK.REGISTRY will be used instead.
NOTE: All registry changes will be lost at the end of the session.
```

If SAS can read the Sasuser.Registry catalog, then SAS copies the Sasuser.Registry catalog to create a Work.Registry catalog (in the Work library). This Work.Registry catalog is used for the duration of the SAS session. Because the contents of the Work library are temporary, then any customizations that you save to the Work.Registry catalog are lost at the end of the SAS session. However, the customizations saved in the Sasuser.Registry catalog still exist.

If SAS cannot read the Sasuser.Registry catalog, then SAS creates the Work.Registry catalog using the default settings for a SAS session. In this case, SAS issues an additional warning to the SAS log:

```
WARNING: Unable to copy SASUSER.REGISTRY to WORK.REGISTRY.
```

**Work Library**

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

To access files in the Work library, 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 SAS, it assigns the Work libref to a subdirectory of the directory specified in the WORK system option described in “**WORK System Option**” in *SAS Viya System Options: Reference*. This subdirectory is usually named `SAS_workcode_nodename`, and has the following characteristics:
workcode

is a 12-character code. The first four characters are randomly generated numbers. The next eight characters are based on the hexadecimal process identification number of the SAS session.

nodename

is the name of the Linux computer where the SAS process is running.

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 library. For more information, see “WORKINIT System Option” in SAS Viya System Options: Reference and “WORKTERM System Option” in SAS Viya System Options: Reference.

Note: If a SAS session is terminated improperly (for example, with the kill —9 command), SAS does not delete the SAS_workcode_nodename directory. You might want to use the “Cleanwork Utility” on page 71 to delete the directories.

Using One-Level Names to Access Permanent Files (User Library)

Introduction to One-Level Names

SAS data sets are referenced with a one- or two-level name. The two-level name has the form libref\_member-name, where libref refers to the SAS library in which the data set resides, and member-name refers to the particular member within that library. The one-level name has the form member-name (without a libref). In this case, SAS stores the files in the temporary Work library. To override this action and store files with one-level names in a permanent library, you must 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.

Techniques for Assigning the User 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 SAS session. For example, you can assign the User libref when you invoke SAS:

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

- Specify the USER= system option after you start the SAS 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;
  ```

For more information, see “USER= System Option” in SAS Viya System Options: Reference.
Note: SAS on Linux does not support the assignment of the User libref using the USER environment variable.

---

Support for Links in Linux

SAS provides limited support for hard links and symbolic links in Linux environments. You can create links that point to a SAS data set or SAS catalog. If you reference the link in a SAS program, SAS follows the link to find the data set or catalog.

For example, you can create a symbolic link in the /tmp directory to the /home/user/mydata.sas7bdat data set by entering the following command at the Linux prompt:

```bash
ln -s /home/user/mydata.sas7bdat /tmp/mydata.sas7bdat
```

The following SAS code uses the symbolic link in the /tmp directory to find the mydata.sas7bdat data set. This code does not change the symbolic link, but it does sort the data in the data set.

```sas
libname tmp '/tmp';

proc sort data=tmp.mydata;
    by myvariable;
run;
```

Note: SAS does not support links for a version data set or for a data set that has an index.
Chapter 4
Using External Files and Devices

Introduction to External Files and Devices in Linux ......................... 46
Accessing an External File or Device in Linux Environments ............... 46
  Specifying a Pathname or a Fileref .................................. 46
  What Is a Fileref? .................................................. 47
Specifying Pathnames in Linux Environments .................................. 47
  Rules for Specifying Pathnames ................................... 47
  Omitting Quotation Marks in a Filename .............................. 47
  Working with Mixed Case or Uppercase Filenames ....................... 48
  Interpreting the Messages in the SAS Log ............................ 48
  Using Wildcards in Pathnames (Input Only) ........................... 49
Assigning Filerefs to External Files or Devices with the FILENAME Statement . 50
  Introduction to the FILENAME Statement ................................ 50
  Accessing DISK Files ................................................. 50
  Debugging Code with DUMMY Devices ................................ 50
  Using Temporary Files (TEMP Device Type) ............................ 51
  Assigning Filerefs to Files on Other Systems (SFTP Access Type) .... 51
Concatenating Filenames in Linux Environments ............................... 51
Assigning a Fileref to a Directory (Using Aggregate Syntax) ............... 52
  Introduction to Aggregate Syntax .................................. 52
  Assigning a Fileref to Several Directories ............................ 52
Using Environment Variables to Assign Filerefs in Linux Environments ..... 53
  Reading a Data File ................................................ 53
  Writing to an External File ......................................... 53
Filerefs Assigned by SAS in Linux Environments ............................... 53
  Filerefs for Standard Input, Standard Output, and Standard Error .... 53
  File Descriptors in the Bourne and Korn Shells ....................... 54
Reserved Filerefs in Linux Environments ...................................... 54
Sharing External Files in a Linux Environment ................................ 54
  Sharing External Files .............................................. 54
  Options to Use for File Locking: External Files ...................... 55
  File Locking for External Files: The LOCKINTERNAL Statement Option ... 55
  File Locking for External Files: The FILELOCKS System Option ...... 55
Reading from and Writing to Linux Commands (PIPE) ......................... 55
  What Are Pipes? .................................................. 55
  Syntax of the FILENAME Statement to Assign a Fileref to a Pipe .... 56
Introduction to External Files and Devices in Linux

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. You can create, read, write, and delete external files from within SAS.

You can use external files in a SAS session to perform the following functions:
- 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 SAS, 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 into a DATA step or into SAS statements to be processed by SAS. The output can be one of the following:
- 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

When you run SAS in batch mode or in interactive line mode, Linux enables you to use pipes to send data to and from operating system commands as if they were I/O devices.

Accessing an External File or Device in Linux Environments

Specifying a Pathname or a Fileref

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

FILE
   specifies the current output file for PUT statements.

%INCLUDE
   includes a file that contains SAS source statements that are executed when you submit a program.

   TIP If you use %INCLUDE, the line limit is 6000 bytes.

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

In the SAS statement, refer to the file or device in one of two ways:
- Specify the pathnames for the external files. For more information, see “Specifying Pathnames in Linux Environments” on page 47.
Assign a fileref to a device, one or more files, or a directory, and use the fileref when you want to refer to the file, directory, or device.

In most cases, you should use a fileref.

What Is a Fileref?

A fileref is a nickname that you assign to a file or device. You assign the fileref once, and then use it as needed. Filerefs are especially useful under the following conditions:

- 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 with the FILENAME statement, with the FILENAME function, or by defining the fileref as an environment variable.

Note: For a complete description of the FILENAME statement and the FILENAME function, see “FILENAME Statement” in SAS Viya Statements: Reference and “FILENAME Function” in SAS Viya Functions and CALL Routines: Reference.

Specifying Pathnames in Linux Environments

Rules for Specifying Pathnames

You can reference an external file directly by specifying its pathname in the FILE, INFILE, or %INCLUDE statements. 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 need to specify its pathname in the appropriate statement. In most cases, you must enclose the name in quotation marks. For example, the following INFILE statement refers to the file /users/pat/cars:

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

The following FILE statement directs output to a specified special device file:

```
file '/dev/ttypl';
```

Note: If a filename has leading blanks, then the blanks are trimmed.

The level of specification depends on your current directory. You can use the character substitutions shown in Table 3.1 on page 37 to specify the pathname. You can also use wildcards as described in “Using Wildcards in Pathnames (Input Only)” on page 49.

Omitting Quotation Marks in a Filename

You can omit the quotation marks in a filename if one of the following is true:

- There is not already a fileref defined with that filename.
- The file has the file extension that is expected by the statement that you are using to refer to the file. If you do not enclose a filename in quotation marks, the FILE and INFILE statements assume a file extension of `.dat`, and the %INCLUDE statement assumes a file extension of `.sas`. 
• The file is located in the current directory.
• The filename is written with all lowercase 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:

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

If there is no `qtr` fileref already defined, you can omit the quotation marks and the file extension in the `%INCLUDE` statement:

```sas
%include qtr;
```

**Working with Mixed Case or Uppercase Filenames**

File names in the Linux operating system are case sensitive. This means that a file named `PROGRAM` is not the same as a file named `program`. When you reference the name of a file that is written in mixed case or uppercase, and that filename is not enclosed in quotation marks, SAS converts the filename to lowercase. If the filename does not have a file extension, SAS adds the missing file extension.

For example, if you specify `%include code(PROGRAM);` in your program, SAS converts the filename `PROGRAM` to lowercase, and adds an extension of `.sas` to the filename. `PROGRAM` becomes `program.sas`.

**Interpreting the Messages in the SAS Log**

When you execute the following program, SAS converts `TEMP` to `temp`, and adds an extension of `.sas` to the filename:

```sas
filename inc_code 'your-directory';
%include inc_code(TEMP);
```

SAS writes the following messages to the SAS log:

```
WARNING: Physical file does not exist, A.../your-directory/TEMP.sas.
ERROR: Cannot %INCLUDE member TEMP in the aggregate INC_CODE.
```

The warning message shows only the original filename (`TEMP.sas`), and not the lowercase conversion (`temp.sas`). This situation might cause confusion if a file named `TEMP.sas` does exist.

To avoid this confusion, include the file extension with the filename if the filename contains an extension, or enclose the mixed case or uppercase filename in quotation marks if the filename does not have an extension. For example:

```sas
%include code(TEMP.sas);
%include code("TEMP");
```

In both of these cases, SAS does not convert `TEMP` to lowercase.
Using Wildcards in Pathnames (Input Only)

Descriptions of the Valid Wildcards
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.

Example 1: Selecting Files By Including a Wildcard in a String
The following example reads input from every file in the current directory that begins with the string wild and ends with .dat:

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

Example 2: Reading Each File in the Current Directory
The following example reads input from every file in every subdirectory of the current working directory:

```plaintext
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.

Example 3: Wildcards in Filenames When Using Aggregate Syntax
You can also use wildcards in filenames, but not in directory names, when you use aggregate syntax:

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

In the example above, the period in the FILENAME statement refers to the current directory.
See “Valid Character Substitutions in Pathnames” on page 37 for information about character substitutions available in Linux.

**Example 4: Associating a Fileref with Multiple Files**
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 2010 and 2019:

```
filename myref '[Ss][Aa][Ll][Ee][Ss][0-9]201[0-9].dat';
```

---

**Assigning Filerefs to External Files or Devices with the FILENAME Statement**

**Introduction to 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 that you want to access. For more information, see “FILENAME Statement” in *SAS Viya Statements: Reference*.

**Accessing DISK Files**
The most common use of the FILENAME statement is to access DISK files. The FILENAME syntax for a DISK file is the following:

```
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;
```

For more information about using DISK files, see “Concatenating Filenames in Linux Environments” on page 51.

*Note:* If a filename has leading blanks, then blanks are trimmed.

**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.

Here is the FILENAME syntax for a DUMMY file:

FILENAME fileref DUMMY 'pathname' <options>;

Output to DUMMY devices is discarded.

**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 library. (See “Work Library” on page 42.) Using the TEMP device type enables you to create a file that lasts only as long as the SAS session.

Here is the FILENAME syntax for a TEMP file:

FILENAME fileref TEMP <options>;

For example, this FILENAME statement associates Tmp1 with a temporary file:

filename tmp1 temp;

**Assigning Filerefs to Files on Other Systems (SFTP Access Type)**

You can access files on other systems in your network by using the SFTP access method. Here are the forms of the FILENAME statement:

FILENAME fileref SFTP 'external-file' <sftp-options>;

These access methods are documented in “FILENAME Statement” in *SAS Viya Statements: Reference*.

---

**Concatenating Filenames in Linux Environments**

You can concatenate filenames in the FILENAME, %INCLUDE, and INFILE statements. Concatenating filenames enables 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 quotation marks and separate them with commas or blank spaces. You can use the characters shown in Table 3.1 on page 37 and the wildcards described in “Using Wildcards in Pathnames (Input Only)” on page 49 to specify the pathnames.
Assigning a Fileref to a Directory (Using Aggregate Syntax)

**Introduction to Aggregate Syntax**

**Aggregate Syntax**

Aggregate syntax enables 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.

```plaintext
FILENAME fileref directory-name;
```

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

**Example 1: Referring to a File Using Aggregate Syntax**

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:

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

**Example 2: Using Aggregate Syntax with Filerefs Defined by Environment Variables**

You can also use aggregate syntax with filerefs that have been defined using environment variables. For more information, see “Using Environment Variables to Assign Filerefs in Linux Environments” on page 53. This example accesses the file cars.dat in the directory that is referenced by the PRICES environment variable:

```plaintext
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:

```plaintext
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 SAS executes the following code, it searches for Report.sas in the pathnames that are specified in the FILENAME statement and it executes the program.

```plaintext
filename progs ("$MYPROGS" "/users/mkt/progs");
```
SAS 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 49) in the filename, SAS matches only one file.

Using Environment Variables to Assign Filerefs in Linux Environments

Reading a Data File

If 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 SAS, see “Defining Environment Variables in Linux Environments” on page 67. For example, in the Korn shell, you use the following:

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

- After you invoke SAS by using the X statement (see “Executing Operating System Commands from Your SAS Session” on page 8) 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;
  ```

Writing to an External File

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

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

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

  ```
  file OUTFILE;
  ```

Filerefs Assigned by SAS in Linux Environments

Filerefs for Standard Input, Standard Output, and Standard Error

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 SAS, 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.

**File Descriptors in the Bourne and Korn Shells**

Each file has an assigned internal file descriptor. 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.

If you are using the Bourne shell or the Korn shell, SAS assigns filerefs of the following form to files that have a file descriptor 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 assign the appropriate file descriptor to the file to be processed.

---

**Reserved Filerefs in Linux Environments**

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.

---

**Sharing External Files in a Linux Environment**

**Sharing External Files**

If more than one user has simultaneous Write access to an external file, or if a single user has Write access to the same file from different SAS sessions, the results of sharing the
file can be unpredictable. To remedy this situation, you can use a statement option or a system option to restrict Write access to one user, while allowing multiple users Read access. For more information, see “Sharing SAS Files” on page 33.

**Options to Use for File Locking: External Files**

File locking applies to all files that are opened. You can turn off file locking for external files in the following ways:

- Use the LOCKINTERNAL option in the FILENAME statement.
- Use the FILELOCKS system option.

**File Locking for External Files: The LOCKINTERNAL Statement Option**

You can control file locking for external files by using the LOCKINTERNAL option in the FILENAME statement. The AUTO option value locks a file exclusively for Write access, or non-exclusively for Read access. For example, if a file is opened for update or output, then all other access from internal processes are blocked. If a file is opened for input, then other users can also open the file for input. In this case, opening the file for update and output is blocked. The SHARED option value allows for all of the behavior of the AUTO option, except that the file can be shared by one writer and multiple readers. The external file that is associated with the fileref is the file that is locked. By default, multiple users can simultaneously read an external file. For more information, see “FILENAME Statement” in SAS Viya Statements: Reference.

**File Locking for External Files: The FILELOCKS System Option**

You can control file locking for external files (as well as for SAS files) by using the FILELOCKS system option. This option enables you to apply a behavior globally to individual files or directories. Using FILELOCKS restricts writer access to one user. With file locking turned on, multiple SAS sessions are able to simultaneously read the same file. You can use FILELOCKS at start-up, in the OPTIONS statement, or in the command line. You can specify multiple instances of the FILELOCKS option. Each instance is added to an internal table of paths and settings. For more information, see “FILELOCKS System Option” in SAS Viya System Options: Reference.

---

**Reading from and Writing to Linux Commands (PIPE)**

**What Are Pipes?**

Pipes enable your SAS application to receive input from any Linux command that writes to standard output and to route output to any Linux command that reads from standard input. In Linux 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:

```
ls | wc -w
```
Syntax of the FILENAME Statement to Assign a Fileref to a Pipe

Under Linux, you can use the FILENAME statement to assign filerefs not only to external files and I/O devices, but also to a pipe. Here is the syntax of the FILENAME statement:

FILENAME fileref PIPE 'Linux-command' <options>;

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

PIPE
- identifies the device-type as a Linux pipe.

'Linux-command'
- is the name of a Linux command, executable program, or shell script to which you want to route output or from which you want to read input. The commands must be enclosed in either double or single quotation marks.

options
- control how the external file is processed. For an explanation of these options, see “FILENAME Statement” in SAS Viya Statements: Reference.

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 SAS assumes that the input comes from a Linux command. If the fileref is used in a FILE statement, then SAS assumes that the output goes to a Linux command.

Using the Fileref for Reading

Specifying a Fileref for Reading

When the fileref is used for reading, the specified Linux 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.

Example 1: Sending the Output of the Process Command to a SAS DATA Step

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 SAS:

```sas
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 in the system, including the name of the command that started the task.

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.

**Example 2: Using the Stdin Fileref to Read Input**

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 the following file, now called `ps.sas`:

```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 is stored in `ps.lst`, and the log is stored in `ps.log`.

**Using the Fileref for Writing**

**Specifying a Fileref for Writing**

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

**Example: Starting a Remote Shell and Printing Output**

Consider this FILENAME statement:

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

Any data sent to the `letterq` fileref is passed to the Linux command, which starts a remote shell on the computer named Alpha. Note that the form of the command that starts a remote shell varies among the various Linux operating systems. The shell then prints the `letterq` output on the printer identified by the destination BLDGA3. Any messages that are produced by the `lp` command are sent to the SAS log.
Chapter 5
Data Representation

Numeric Variable Length and Precision in Linux Environments

The default length of numeric variables in SAS data sets is 8 bytes. (You can control the
length of SAS numeric variables with the LENGTH or ATTRIB statements in the DATA
step.)

The issue of numeric precision affects the return values of almost all SAS math functions
and many numeric values returned from SAS procedures. Numeric values in SAS for
Linux are represented as IEEE double-precision floating-point numbers. The decimal
precision of a full 8-byte number is effectively 15 decimal digits.

The following table specifies the significant digits and largest integer that can be stored
exactly in SAS numeric variables.

Table 5.1 Significant Digits and Largest Integer by Length for SAS Variables under Linux

<table>
<thead>
<tr>
<th>Length in Bytes</th>
<th>Significant Digits Retained</th>
<th>Largest Integer Represented Exactly</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>8,192</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2,097,152</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>536,870,912</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>137,438,953,472</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>35,184,372,088,832</td>
</tr>
</tbody>
</table>
When you are specifying variable lengths, keep in mind that the length of a variable affects both the amount of disk space used and the number of I/O operations required to read and write the data set.

If you know that the value of a numeric variable will be an integer between -8192 and 8192 inclusive, you can use a length of 3 to store the number and thus save space in your data set. For example:

```sas
data mydata;
  length num 3;
  ...more SAS statements...
run;
```

Numeric dummy variables (variables whose only purpose is to hold 0 or 1) can be stored in a variable whose length is 3 bytes.

**CAUTION:**

*Use the LENGTH statement to reduce length only for variables whose values are always integers.* Fractional numbers lose precision if they are truncated. In addition, you must ensure that the values of your variable are always represented exactly in the number of bytes that you specify. You can do this programmatically in a DATA step with the TRUNC function. No warnings or errors are issued when the length that you specify in the LENGTH statement results in the truncation of data.

### Missing Values in Linux Environments

In SAS on Linux, missing values are represented by IEEE Not-a-Number values. An IEEE Not-a-Number value is an IEEE floating-point bit pattern that represents something other than a valid numeric value. These numbers are not computationally derivable.

### Reading and Writing Binary Data in Linux Environments

Different computers store numeric binary data in different forms. If you try to move binary data in flat files across systems that are incompatible, problems will occur. A safer way to move data is by using SAS data sets.

SAS provides several sets of informats and formats for handling binary data. Some of these informats and formats are host dependent. For example, the IBw.d, PDw.d, PIBw.d, and RBw.d informats and formats read and write data in native mode. That is, they use the byte-ordering system that is standard for the computer.

For more information about all of the informats and formats, see *SAS Viya Formats and Informats: Reference*. 
Converting a Linux Datetime Value to a SAS Datetime Value

A Linux datetime value is stored as the number of seconds since January 1, 1970. A SAS datetime value is stored as the number of seconds since January 1, 1960. To convert a Linux datetime value to a SAS datetime value, you must add 10 years in seconds to the Linux datetime value.

The INTNX function converts a Linux datetime value to a SAS datetime value, as shown in the example below:

```sas
data Linux_to_SAS;
  input Linux_datetime;
  /* The INTNX function accounts for leap years. */
  SAS_datetime = intnx('DTyear',Linux_datetime,10,'s');
  format SAS_datetime datetime20.;
datalines;
1285560000
1313518500
1328414200;
proc print data=Linux_to_SAS;
run;
```

The following output displays the results.

*Figure 5.1  Conversion of a Linux Datetime Value to a SAS Datetime Value*

<table>
<thead>
<tr>
<th>Obs</th>
<th>UNIX_datetime</th>
<th>SAS_datetime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1285560000</td>
<td>26SEP2010:04:00:00</td>
</tr>
<tr>
<td>2</td>
<td>1313518500</td>
<td>15AUG2011:18:15:00</td>
</tr>
<tr>
<td>3</td>
<td>1328414200</td>
<td>04FEB2012:03:56:40</td>
</tr>
</tbody>
</table>

For more information, see “INTNX Function” in *SAS Viya Functions and CALL Routines: Reference*. 
Part 2

Features Available in Batch and Line Mode

Chapter 6
Commands Available in Batch and Line Mode ............... 65

Chapter 7
Environment Variables Available in Batch and Line Mode .... 67

Chapter 8
Cleanwork ......................................................... 71
Chapter 6

Commands Available in Batch and Line Mode

---

**Dictionary**

**SETENV Command: Linux**

Defines an environment variable and assigns a value to it.

**Syntax**

```bash
SETENV <variable-name> <variable-value>

UNSETENV variable-name
```

**Required Argument**

`variable-name`

- specifies a Linux environment variable that you can set. This value is required when you use the UNSETENV command.

**Optional Argument**

`variable-value`

- specifies the value of a Linux environment variable.

**Details**

The SETENV command can be used to define an environment variable and assign a value to it. The value of an environment variable can be retrieved from within the SAS session using the SYSGET function during autoexec processing. The command `x setenv a /tmp; sets a=/tmp`. The command `x echo $a;` results in the value `/tmp`.

The UNSETENV command removes an environment variable. The memory for the entry and the environment variable is released.
X Command: Linux

Enables you to enter Linux commands without ending the SAS session.

Syntax

\[
X \ command \\
X \ 'command-1; \ command-2....<; \ command-n'>
\]

Required Argument

\[\text{command}\]

 specified a Linux command.

Details

When you enter the X command, SAS starts a shell to execute the commands that you specified. The commands that you enter are processed differently, depending on whether you enter one command or more than one command.

See Also

“Executing Operating System Commands from Your SAS Session” on page 8
Chapter 7
Environment Variables Available in Batch and Line Mode

Defining Environment Variables in Linux Environments

What Is a Linux Environment Variable?
Linux environment variables are variables that apply to both the current shell and to any subshells that it creates (for example, when you send a job to the background or execute a script). If you change the value of an environment variable, the change is passed forward to subsequent shells, but not backward to the parent shell.

In a SAS session, you can use the SASV9_OPTIONS environment variable to specify system options and the SASV9_CONFIG environment variable to specify a configuration file. Any changes that you make to an environment variable after initialization of a SAS session are not recognized.

You can also use environment variables as filerefs and librefs in various statements and commands. Filerefs and librefs consist of uppercase letters, digits, and the underscore character in environment variable names. Other characters are not recognized by SAS. For more information, see “Using Environment Variables as Librefs in Linux Environments” on page 39 or “Using Environment Variables to Assign Filerefs in Linux Environments” on page 53.

Note: A SAS/ACCESS product initializes the environment variables that it needs when loading. For more information, see the documentation for your SAS/ACCESS product.
How to Define an Environment Variable for Your Shell

**Defining Environmental Variables**
The way in which you define an environment variable depends on the shell that you are running. (To determine which shell you are running, type `ps` at the command prompt or `echo $SHELL` to see the current value of the SHELL environment variable.)

**Bourne and Korn Shells**
In the Bourne shell and in the Korn shell, use the `export` command to export one or more variables to the environment. For example, these commands make the value of the variable `scname` available to all subsequent shell scripts:

```
$ scname=phonelist
$ export scname
```

In the Korn shell, you can combine these commands into one command:

```
$ export scname=phonelist
```

If you change the value of `scname`, then the new value affects both the shell variable and the environment variable. If you do not export a variable, only the shell script in which you define has access to its value.

**C Shell**
In the C shell (csh and tcsh), you set (define and export) environment variables with the `setenv` (set environment) command. For example, this command is equivalent to the commands shown previously:

```
% setenv scname phonelist
```

**Displaying the Value of an Environment Variable**
To display the values of individual environment variables, use the `echo` command and parameter substitution. An example is `echo $SHELL`, which returns the current value of the SHELL environment variable. Use the `env` (or `printenv`) command to display all environment variables and their current values.

---

**Dictionary**

**AUTHINFO Environment Variable**
Specifies a file where user ID and passwords are kept for authentication, typically called the Authinfo file.

**Details**
Use the AUTHINFO environment variable to set the location of the Authinfo file on your system. This variable can be set to the location of one or more files. Use a ‘;’ to separate multiple filenames. Set the value of the AUTHINFO environment variable
using the instructions in “How to Define an Environment Variable for Your Shell” on page 68.

The value that you specify typically includes the path and filename for the Authinfo file on your system. For example, when working with a C shell, you might specify the AUTHINFO environment variable as follows:

```
setenv authinfo '$HOME/authInfo-file'
```

If the AUTHINFO= system option is set in a program that you run in batch mode, the system option value overrides the value that is set for the AUTHINFO environment variable.

**See Also**

- “Authinfo Authentication Method” in *SAS Viya Administration: Authentication*
- “Create an Authinfo File” in *SAS Viya Administration: Authentication*

---

**PATHENCODING Environment Variable**

Specifies the encoding for external file references and directory references when the encoding is different from the SAS session encoding.

**Details**

Set the value of the PATHENCODING environment variable using the instructions in “How to Define an Environment Variable for Your Shell” on page 68.

The encoding value that you assign specifies the encoding for external file references and directory references that are accessed from within a SAS program. Specify a value for this environment variable when external file encoding and directory encoding are different from the SAS session encoding. SAS uses the default session encoding when referencing external files and directories. The PATHENCODING environment variable provides an alternative encoding for external file and directory references. PATHENCODING is valid only for files that are located on disk. When the PATHENCODING environment variable has a valid encoding value, SAS transcodes the pathname from the SAS session encoding into the specified encoding.

For a list of common encoding values, see “Common Encoding Methods” in *SAS Viya National Language Support (NLS): Reference Guide*.

The pathnames that you specify within a SAS program must be entered in the SAS session encoding. Do not specify pathnames in the encoding that you specify for the PATHENCODING environment variable.

To specify a PATHENCODING value of UTF-8 in a SAS session that uses English, the characters in a pathname must contain characters that are recognized by both the session encoding and the encoding that is specified by PATHENCODING. As a result, to specify a PATHENCODING value of UTF-8 in a SAS session that uses English (LANG=EN), you must specify a SAS session encoding of UTF-8 or SAS_U8.

---

**SASV9_CONFIG Environment Variable**

Specifies the configuration file that is referenced when you start a SAS session.
Details

Set the value of the SASV9_CONFIG environment variable using the instructions in “How to Define an Environment Variable for Your Shell” on page 68.

The file specification that you assign to SASV9_CONFIG specifies the path and name of the configuration file that the SAS session uses. This configuration file contains all of the SAS system options that you want to use in a SAS session. For example, in a Korn shell, you might assign the custom.cfg file in your home directory to SASV9_CONFIG as follows:

```bash
> export sasv9_config=/u/<user_id>/custom.cfg
```

---

SASV9_OPTIONS Environment Variable

Specifies the list of SAS system options that are automatically used when you start a SAS session.

Details

Use the SASV9_OPTIONS environment variable to specify a list of SAS system options that are automatically used when you start a SAS session. This is useful if you typically set the same SAS system options each time you work with SAS.

Here is an example of a command to set SASV9_OPTIONS:

```bash
export sasv9_options = '-nonotes -echo "SAS is running"'
```


Chapter 8
Cleanwork

Cleanwork Utility

Delete any leftover Work directories, utility directories, or both, whose associated SAS process has ended.

```
cleanwork directory <n> <v> <hostmatch> <log logfile>
```

- `directory` names the directory that contains the Work directory, the Utility directory, or both directories. That is, you can specify multiple directory paths in the cleanwork command. The directory name must match the value specified in the WORK system option or the value specified in the UTILLOC system option.

- `Tip` Unless the cleanwork command is run by root, user permissions might prevent you from removing a directory.

- `-n` specifies that SAS lists the directories that contain entries that can be removed.

- `-v` specifies verbose output.

- `hostmatch` specifies the name of a host from which you can remove Work directories that might still be active in a Network File System (NFS).

- `log logfile` specifies that the output from cleanwork should be sent to a log file. The `-log` option is useful when you run cleanwork from a cron (scheduled) job.

You provide the name of the log file (`logfile`) for the cleanwork output. The `logfile` name can include a path or only the filename. If no path is provided, the file is generated in the directory from which you called the cleanwork utility. Enclose the filename in single quotation marks if you use characters that are considered special by the shell.

Details

The cleanwork utility removes any subdirectories that were assigned to the Work library or directories assigned by the UTILLOC system option. The cleanwork command
removes only those files that are associated with defunct SAS processes. Each subdirectory name has a format of the form:

SAS_workcode_nodename
SAS_utilcode_nodename

**workcode** or **utilcode**

is a 12-character code. The first four characters are randomly generated numbers. The next eight characters are based on the hexadecimal representation of the process ID of the associated SAS process. Files that are associated with active processes are not removed.

**nodename**

specifies the name of the Linux system where the SAS process is running.

For example, if you are working on nodename *jupiter*, then the cleanwork command removes all directories with inactive processes on *jupiter*. The cleanwork command does not remove a directory that is associated with an orphaned process if that process is still active. In this case, you need to manually kill the process, and then rerun cleanwork.

**See Also**

“Work Library” on page 42
Index

**Special Characters**

- %INCLUDE statement  
  - concatenating filenames 51  
  - specifying pathnames 47  
- %SYSEXEC macro statement 9

**A**

- aggregate syntax 52  
- asynchronous tasks  
  - executing 8  
- autoexec files  
  - configuration files versus 15

**B**

- background process 6  
- batch mode 4  
  - executing X statements 10  
- batch programs  
  - restarting 24  
- binary data 60  
- binary values  
  - reading and writing 60  
- Bourne shell  
  - defining environment variables 68  
  - file descriptors 54

**C**

- C shell  
  - defining environment variables 68  
- CALL SYSTEM routine 9  
- case  
  - in data set names 32  
  - mixed case or uppercase filenames 48  
- catalogs  
  - Sasuser.Profile 41  
  - Sasuser.Registry 42  
- checkpoint mode 20  
  - requirements 23  
  - setting up and executing 23  
- commands  
  - executing several 9  
- issuing from SAS sessions 66  
  - piping to/from 55  
  - synchronous versus asynchronous 8  
  - completion code 17  
  - completion status of jobs 17  
  - concatenating directories 38  
  - concatenating filenames 51  
  - configuration files  
    - autoexec files versus 15  
    - creating 16  
    - order of precedence 16  
    - overriding system option default values 11  
    - specifying 16  
  - console log 20  
  - control keys  
    - terminating SAS sessions 19

**D**

- DATA member type 33
- data representation 59  
  - binary data 60  
  - missing values 60  
  - numeric variables 59  
- data sets 33  
  - case sensitivity in names of 32  
- SAS data files 33  
  - with same name 38
- DATA step  
  - sending Linux command output to 55  
- DATALINES fileref 54  
- date conversion 61  
- DBMS processes  
  - interrupting 25  
- devices  
  - assigning and deassigning filerefs 50  
  - debugging code with DUMMY devices 50  
  - using with SAS 46  
- directories  
  - assigning and deassigning filerefs 52  
  - assigning librefs to several 38  
  - concatenating 38
unused Work and utility directories 71
DISK files 50
DUMMY devices
debugging code with 50

E
gines 32
environment variables 67
as librefs 39
defining 68
returning value of 68
error processing
 checkpoint mode and restart mode 20
 errors
 console log 20
 executing SAS statements
 autoexec file 15
 exit status
 SAS jobs 17
 exiting SAS
 preferred methods 18
 external devices 46
 external files 32, 46
 assigning and deassigning filerefs 50
 concatenating filenames 51
 specifying pathnames 47
 wildcards in pathnames 50
 writing data from, with pipes 5

F
file descriptors 54
file locking 33
 FILELOCKS statement option 34
 FILELOCKS system option 34
 FILELOCKS=CONTINUE 34
 options for 34
 waiting to use locked files 34
 file permissions
 changing for SAS sessions 10
 FILELOCKS statement 34
 FILELOCKS= system option 34
 set to CONTINUE 34
 FILENAME statement
 assigning filerefs to directories 52
 assigning filerefs to external files or devices 50
 assigning filerefs to pipes 56
 concatenating filenames 51
 specifying pathnames 47
 filenames
 concatenating 51
 interpreting log messages 48
 mixed case or uppercase 48
 omitting quotation marks in 47
filerefs 47
 assigned by SAS 53
 assigning and deassigning 50
 assigning and deassigning, directories 52
 assigning and deassigning, pipes 55
 reserved 54
 foreground process 6
 formats
 for binary data 60
 FTP access method 51

H
halting execution
 DBMS processes 25
 SAS processes 19
 hard links 44

I
IEEE Not-a-Number values 60
indexes 33
INFILE statement
 concatenating filenames 51
 specifying pathnames 47
 informs
 for binary data 60
 interactive line mode 6
 interactive SAS session 8
 interface SAS views 33
 interrupting SAS 18
 control keys 19
 kill command 19
 messages in console log 20
 SAS processes 26
 invoking SAS sessions
 as foreground or background process 6
 interactive line mode 6

J
jobs
 completion status of 17

K
kill command 19
Korn shell
defining environment variables 68
 file descriptors 54

L
length of numeric variables 59
LIBNAME function
Index 75

assigning librefs 35
LIBNAME statement
assigning librefs 35
libraries 32
accessing with librefs 36
Work library 42, 71
librefs 35
accessing permanent SAS libraries 36
assigned by SAS 39
assigning to several directories 38
assigning with LIBNAME function 35
assigning with LIBNAME statement 35
environment variables as 39
permanently assigning 35
referring to SAS files 36
Sashelp libref 39
Sasuser libref 40
User libref 43
WORK libref 40
links 44
locking files
See file locking
log
console log 20
interpreting messages about filenames 48
LOG fileref 54

M
MEMSIZE system option
example 8
missing values 60
mixed case filenames 48

N
native data files 33
native SAS views 33
Not-a-Number values 60
numeric variables 59
length and precision 59

O
one-level names 43
OPTIONS statement
overriding system option default values 12
overriding system option default values 11

P
PATHENCODING environment variable 69
pathnames 47
character substitutions in 37
specifying 36
pausing execution
DBMS processes 25
SAS processes 19
permanent files
accessing with one-level names 43
permanent librefs 35
permanent SAS libraries 36
pipes
data to/from Linux commands 55
writing data from external files 5
precision of numeric variables 59
PRINT fileref 54

Q
quitting SAS 18
quotation marks
omitting in filenames 47

R
reading binary data 60
reserved filerefs 54
restart mode 20
requirements 23
restarting batch programs 24
return code 17
routing output
piping to/from commands 55
running SAS
as foreground or background process 6
interactive line mode 6

S
SAS
interrupting 18
running in background process 6
running in foreground process 6
terminating 18
SAS command
overriding system option default values 12
syntax 5
SAS data files 33
SAS files 32
accessing for input and update 38
accessing for output 38
concatenating filenames 51
one-level names for accessing permanent files 43
referring to with librefs 36
specifying pathnames 47
SAS jobs
  completion status 17
SAS processes
  terminating 19
SAS programs
  submitting to batch queue 5
SAS servers
  ending processes for running 25
SAS sessions
  file permissions for 10
  interactive line mode 6
  issuing commands from 66
SAS views 33
Sashelp libref 39
Sasuser library
  contents of 40
  Sasuser.Profile catalog 41
  Sasuser.Registry catalog 42
Sasuser libref 40
Sasuser.Profile catalog 41
  checking for uncorrupted 41
  how SAS accesses 41
  when catalog does not exist 41
  when locked or corrupted 41
Sasuser.Registry catalog 42
  how SAS accesses 42
SASV9_CONFIG environment variable
  69
SASV9_OPTIONS environment variable
  70
  overriding system option default values 12
SETENV command 65
sharing files
  file locking 34
shells
  defining environment variables 68
  starting remote shells 57
SOCKET access method 51
standard error
  filerefs for 53
standard input/output
  filerefs for 53
  reading input from standard input 57
starting SAS sessions
  as foreground or background process 6
  interactive line mode 6
symbolic links 44
synchronous tasks 8
system options
  overriding default values 11
  set in one place 13
  specifying 11
T
TEMP devices 51
temporary files 51
terminating execution
  DBMS processes 25
  SAS processes 19
terminating SAS 18
  control keys 19
  kill command 19
  messages in console log 20
U
uppercase filenames 48
User libref 43
  assigning 43
utilities directory
  deleting when unused 71
V
variables
  numeric 59
VIEW member type 33
W
wildcards
  in pathnames 50
Work library 42
  deleting unused Work directories 71
WORK libref 40
WORK system option
  example 8
  writing binary data 60
X
X command 66
X statement
  executing in batch mode 10
  executing several commands 9
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore for additional books and resources.