

**AFS User Guide**

**Version 3.6**

## **AFS User Guide: Version 3.6**

Copyright © 2000 IBM Corporation. All Rights Reserved

This edition applies to:

IBM AFS for AIX, Version 3.6  
IBM AFS for Digital Unix, Version 3.6  
IBM AFS for HP-UX, Version 3.6  
IBM AFS for Linux, Version 3.6  
IBM AFS for SGI IRIX, Version 3.6  
IBM AFS for Solaris, Version 3.6

and to all subsequent releases and modifications until otherwise indicated in new editions. This softcopy version is based on the printed edition of this book. Some formatting amendments have been made to make this information more suitable for softcopy.

### Revision History

Revision 3.6 April 2000  
First Edition

# Table of Contents

<b>About This Guide.....</b>	<b>i</b>
Audience and Purpose.....	i
Document Organization .....	i
How To Use This Document.....	i
Related Documents .....	ii
Typographical Conventions.....	ii
<b>1. An Introduction to AFS.....</b>	<b>1</b>
AFS Concepts .....	1
Client/Server Computing.....	1
Distributed File Systems.....	1
AFS Filespace and Local Filespace.....	1
Cells and Sites .....	2
Volumes and Mount Points.....	2
Volume Quotas .....	3
Using Files in AFS.....	3
The Cache Manager.....	3
Updating Copies of Cached Files .....	4
Multiple Users Modifying Files .....	4
AFS Security .....	4
Passwords and Mutual Authentication .....	4
Access Control Lists.....	5
Differences Between UNIX and AFS.....	5
File Sharing .....	6
Login and Authentication .....	6
File and Directory Protection .....	6
Machine Outages.....	7
Remote Commands .....	7
Differences in the Semantics of Standard UNIX Commands .....	8
Using AFS with NFS .....	8
<b>2. Using AFS.....</b>	<b>9</b>
Logging in and Authenticating with AFS .....	9
Logging In .....	9
To Log In Using an AFS-modified Login Utility .....	9
To Log In Using a Two-Step Login Procedure.....	9
Authenticating with AFS.....	10
Protecting Your Tokens with a PAG.....	10
Obtaining Tokens For Foreign Cells.....	10
The One-Token-Per-Cell Rule .....	10
Obtaining Tokens as Another User .....	11
Token Lifetime .....	11
Authenticating for DFS Access .....	11
To Authenticate with AFS .....	11
To Display Your Tokens .....	12
Example: Authenticating in the Local Cell .....	12
Example: Authenticating as a Another User .....	13

Example: Authenticating in a Foreign Cell .....	13
Limits on Failed Authentication Attempts .....	13
To Display Your Failed Authentication Limit and Lockout Time.....	13
Exiting an AFS Session.....	14
To Discard Tokens .....	14
Example: Unauthenticating from a Specific Cell .....	15
To Log Out.....	15
Accessing the AFS Filespace.....	15
AFS Pathnames .....	15
Example: Displaying the Contents of Another User’s Directory .....	16
Accessing Foreign Cells .....	16
Changing Your Password .....	17
To Display Password Expiration Date and Reuse Policy .....	17
To Change Your AFS Password .....	18
To Change Your UNIX Password.....	18
<b>3. Displaying Information about AFS.....</b>	<b>19</b>
Displaying Volume Quota .....	19
To Display Percentage of Quota Used.....	19
Example: Displaying Percentage of Quota Used .....	19
To Display Quota and Other Information about a Volume.....	20
Example: Display Quota and Other Information about a Volume.....	20
To Display Quota and Other Information about a Volume and Partition .....	20
Example: Displaying Quota and Other Information about a Volume and Partition.....	21
Locating Files and Directories .....	21
To Display a File or Directory’s Location .....	21
Example: Displaying Directory Location.....	22
Checking the Status of Server Machines .....	22
To Check File Server Machine Status .....	22
Example: Checking Server Machine Status .....	23
Determining Access to Foreign Cells .....	23
To Display Foreign Cells .....	23
Displaying Server Preference Ranks.....	24
To Display Server Preference Ranks .....	24
<b>4. Protecting Your Directories and Files.....</b>	<b>27</b>
Access Control Lists .....	27
Directory Level Access Control .....	27
The AFS ACL Permissions .....	27
The Four Directory Permissions.....	28
The Three File Permissions .....	29
The Eight Auxiliary Permissions.....	29
Shorthand Notation for Sets of Permissions.....	29
About Normal and Negative Permissions.....	30
Setting DFS ACLs .....	30
Using the System Groups on ACLs .....	31
Enabling Access to Subdirectories .....	31
Extending Access to Service Processes.....	31
Extending Access to Users from Foreign Cells.....	32

Displaying an ACL.....	32
To display an ACL.....	32
Example: Displaying the ACL on One Directory .....	33
Example: Displaying the ACLs on Multiple Directories .....	33
Changing an ACL.....	34
To Add, Remove, or Edit Normal ACL Permissions .....	34
Example: Adding a Single ACL Entry .....	36
Example: Setting Several ACL Entries on One Directory .....	36
To Add, Remove, or Edit Negative ACL Permissions.....	36
Example: Setting an Entry in the Negative Permissions Section .....	37
Example: Restoring Access by Removing an Entry from the Negative Permissions Section.....	37
Completely Replacing an ACL .....	38
To Replace an ACL Completely .....	38
Example: Replacing an ACL .....	39
Copying ACLs Between Directories.....	39
To Copy an ACL Between Directories.....	39
Example: Copying an ACL from One Directory to Another .....	40
How AFS Uses the UNIX Mode Bits .....	41
Example: Disabling Write Access for a File .....	41
<b>5. Using Groups.....</b>	<b>43</b>
About Groups.....	43
Suggestions for Using Groups Effectively .....	43
Group Names.....	44
Group-creation Quota .....	44
Displaying Group Information.....	44
To Display Group Membership.....	45
Example: Displaying the Members of a Group.....	45
Example: Displaying the Groups to Which a User Belongs .....	45
To Display the Groups a User or Group Owns.....	45
Example: Displaying the Groups a Group Owns .....	46
Example: Displaying the Groups a User Owns.....	46
To Display A Group Entry.....	46
Example: Listing Information about a Group .....	47
Example: Listing Group Information about a User .....	47
Creating Groups and Adding Members .....	48
To Create a Group.....	48
Example: Creating a Group.....	49
To Add Members to a Group.....	49
Example: Adding Members to a Group.....	49
Removing Users from a Group and Deleting a Group.....	50
To Remove Members from a Group.....	50
Example: Removing Group Members .....	50
To Delete a Group.....	50
Example: Deleting a Group.....	51
To Remove Obsolete ACL Entries .....	51
Example: Removing an Obsolete ACL Entry .....	51
Changing a Group's Owner or Name.....	51

To Change a Group's Owner .....	52
Example: Changing a Group's Owner to Another User .....	52
Example: Changing a Group's Owner to Itself .....	52
Example: Changing a Group's Owner to a Group.....	53
To Change a Group's Name.....	53
Example: Changing a Group's <i>group_name</i> Suffix .....	53
Example: Changing a Group's <i>owner_name</i> Prefix .....	53
Protecting Group-Related Information .....	54
Interpreting the Privacy Flags.....	54
To Set a Group's Privacy Flags.....	55
Example: Setting a Group's Privacy Flags .....	55
<b>6. Troubleshooting .....</b>	<b>57</b>
Problem: Cannot Access, Copy, or Save File.....	57
Problem: Accidentally Removed Your Entry from an ACL .....	59
Error Message: "afs: Lost contact with fileserver".....	59
Error Message: " <i>command</i> : Connection timed out" .....	60
Error Message: "fs: You don't have the required access rights on ' <i>file</i> '" .....	60
Error Message: "afs: failed to store file" .....	60
<b>A. Using the NFS/AFS Translator .....</b>	<b>61</b>
Requirements for Using the NFS/AFS Translator .....	61
Accessing AFS via the Translator.....	61
To Authenticate on a Supported Operating System.....	61
To Authenticate on an Unsupported Operating System .....	62
Troubleshooting the NFS/AFS Translator .....	63
Your NFS Client Machine is Frozen .....	63
NFS/AFS Translator Reboots.....	63
System Error Messages .....	64
<b>B. AFS Command Syntax and Online Help .....</b>	<b>65</b>
AFS Command Syntax.....	65
Command Syntax Example .....	65
Rules for Using AFS Commands.....	66
Spaces and Lines .....	66
Abbreviations and Aliases for Operation Codes .....	66
Omitting Argument Switches .....	66
Shortening Switches and Flags.....	67
Shortening Directory References .....	67
Commonly Used fs and pts Commands .....	68
About the fs Commands .....	68
About the pts Commands .....	68
Getting Help in AFS .....	69
Displaying Command Syntax and Aliases .....	69
Displaying Operation Code Descriptions .....	69
<b>Glossary .....</b>	<b>71</b>
<b>Index.....</b>	<b>81</b>

## About This Guide

This section describes the purpose, organization, and conventions of this document.

## Audience and Purpose

This guide describes concepts and procedures for accessing information stored in the AFS filesystem. It is intended for AFS users who are familiar with UNIX but not necessarily AFS.

The first chapter describes basic AFS concepts and guidelines for using it, and summarizes some of the differences between the UNIX file system and AFS. The remaining chapters explain how to perform basic AFS functions, including logging in, changing a password, listing information, protecting files, creating groups, and troubleshooting. Concepts important to a specific task or group of related tasks are presented in context, just prior to the procedures. Many examples are provided.

Instructions generally include only the commands and command options necessary for a specific task. For a complete list of AFS commands and description of all options available on every command, see the *IBM AFS Administration Reference*.

## Document Organization

This document is divided into the following chapters.

"An Introduction to AFS" on page 1 introduces the basic concepts and functions of AFS. To use AFS successfully, it is important to be familiar with the terms and concepts described in this chapter.

"Using AFS" on page 9 describes how to use AFS's basic features: how to log in and authenticate, unlog, log out, access AFS files and directories in AFS, and change your password.

"Displaying Information about AFS" on page 19 describes how to display information about AFS volume quota and location, file server machine status, and the foreign cells you can access.

"Protecting Your Directories and Files" on page 27 describes how to protect your data using AFS access control lists (ACLs).

"Using Groups" on page 43 describes how to create and manage groups.

"Troubleshooting" on page 57 outlines step-by-step diagnostic and corrective steps for specific problems.

"Appendix A, Using the NFS/AFS Translator" on page 61 describes how to use the NFS/AFS Translator to access the AFS filesystem from an NFS client machine.

"Appendix B, AFS Command Syntax and Online Help" on page 65 describes AFS command syntax and how to obtain online information about commands.

"Appendix C, Glossary" on page 71 defines terms used in the *IBM AFS User Guide*.

## How To Use This Document

Before you begin using AFS, read "An Introduction to AFS" on page 1. Next, follow the procedures outlined in "Using AFS" on page 9 to get started using AFS as an authenticated user. It describes how to access files in the AFS filesystem and how to end an AFS session. Consult the other chapters as you need to perform the tasks they describe.

## Related Documents

The AFS Documentation Kit also includes the following documents:

- The *IBM AFS Administration Reference* details the syntax of each AFS command and is intended for the experienced AFS administrator, programmer, or user. For each AFS command, the *IBM AFS Administration Reference* lists the command syntax, aliases and abbreviations, description, arguments, warnings, output, examples, and related topics. Commands are organized alphabetically.
- The *IBM AFS Administration Guide* describes concepts and procedures necessary for administering an AFS cell, as well as more extensive coverage of the topics in the *IBM AFS User Guide*.
- The *IBM AFS Quick Beginnings* provides instructions for installing AFS server and client machines.

## Typographical Conventions

This document uses the following typographical conventions:

- Command and option names appear in **bold type** in syntax definitions, examples, and running text. Names of directories, files, machines, partitions, volumes, and users also appear in **bold type**.
- Variable information appears in *italic type*. This includes user-supplied information on command lines and the parts of prompts that differ depending on who issues the command. New terms also appear in *italic type*.
- Examples of screen output and file contents appear in `monospace type`.

In addition, the following symbols appear in command syntax definitions, both in the documentation and in AFS online help statements. When issuing a command, do not type these symbols.

- Square brackets [ ] surround optional items.
- Angle brackets < > surround user-supplied values in AFS commands.
- A superscripted plus sign + follows an argument that accepts more than one value.
- The percent sign % represents the regular command shell prompt. Some operating systems possibly use a different character for this prompt.
- The number sign # represents the command shell prompt for the local superuser **root**. Some operating systems possibly use a different character for this prompt.
- The pipe symbol | in a command syntax statement separates mutually exclusive values for an argument.

For additional information on AFS commands, including a description of command string components, acceptable abbreviations and aliases, and how to get online help for commands, see "Appendix B, AFS Command Syntax and Online Help" on page 65.



# Chapter 1. An Introduction to AFS

This chapter introduces basic AFS concepts and terms. It assumes that you are already familiar with standard UNIX commands, file protection, and pathname conventions.

## AFS Concepts

AFS makes it easy for people to work together on the same files, no matter where the files are located. AFS users do not have to know which machine is storing a file, and administrators can move files from machine to machine without interrupting user access. Users always identify a file by the same pathname and AFS finds the correct file automatically, just as happens in the local file system on a single machine. While AFS makes file sharing easy, it does not compromise the security of the shared files. It provides a sophisticated protection scheme.

### Client/Server Computing

AFS uses a *client/server computing* model. In client/server computing, there are two types of machines. *Server machines* store data and perform services for client machines. *Client machines* perform computations for users and access data and services provided by server machines. Some machines act as both clients and servers. In most cases, you work on a client machine, accessing files stored on a file server machine.

### Distributed File Systems

AFS is a *distributed file system* which joins together the file systems of multiple file server machines, making it as easy to access files stored on a remote file server machine as files stored on the local disk. A distributed file system has two main advantages over a conventional centralized file system:

- Increased availability: A copy of a popular file, such as the binary for an application program, can be stored on many file server machines. An outage on a single machine or even multiple machines does not necessarily make the file unavailable. Instead, user requests for the program are routed to accessible machines. With a centralized file system, the loss of the central file storage machine effectively shuts down the entire system.
- Increased efficiency: In a distributed file system, the work load is distributed over many smaller file server machines that tend to be more fully utilized than the larger (and usually more expensive) file storage machine of a centralized file system.

AFS hides its distributed nature, so working with AFS files looks and feels like working with files stored on your local machine, except that you can access many more files. And because AFS relies on the power of users' client machines for computation, increasing the number of AFS users does not slow AFS performance appreciably, making it a very efficient computing environment.

## AFS Filespace and Local Filespace

AFS acts as an extension of your machine's local UNIX file system. Your system administrator creates a directory on the local disk of each AFS client machine to act as a gateway to AFS. By convention, this directory is called */afs*, and it functions as the root of the *AFS filesystem*.

Just like the UNIX file system, AFS uses a hierarchical file structure (a tree). Under the */afs* root directory are subdirectories created by your system administrator, including your home directory. Other directories that are at the same level of the local file system as */afs*, such as */usr*, */etc*, or */bin*, can either be located on your local disk or be links to AFS directories. Files relevant only to the local machine are usually stored on the local machine. All other files can be stored in AFS, enabling many users to share them and freeing the local machine's disk space for other uses.

**Note:** You can use AFS commands only on files in the AFS filesystem or the local directories that are links to the AFS filesystem.

## Cells and Sites

The *cell* is the administrative domain in AFS. Each cell's administrators determine how client machines are configured and how much storage space is available to each user. The organization corresponding to a cell can be a company, a university department, or any defined group of users. From a hardware perspective, a cell is a grouping of client machines and server machines defined to belong to the same cell. An AFS *site* is a grouping of one or more related cells. For example, the cells at the ABC Corporation form a single site.

By convention, the subdirectories of the */afs* directory are cellular filesystems, each of which contains subdirectories and files that belong to a single cell. For example, directories and files relevant to the ABC Corporation cell are stored in the subdirectory */afs/abc.com*.

While each cell organizes and maintains its own filesystem, it can also connect with the filesystem of other AFS cells. The result is a huge filesystem that enables file sharing within and across cells.

The cell to which your client machine belongs is called your *local cell*. All other cells in the AFS filesystem are termed *foreign cells*.

## Volumes and Mount Points

The storage disks in a computer are divided into sections called *partitions*. AFS further divides partitions into units called *volumes*, each of which houses a subtree of related files and directories. The volume provides a convenient container for storing related files and directories. Your system administrators can move volumes from one file server machine to another without your noticing, because AFS automatically tracks a volume's location.

You access the contents of a volume by accessing its *mount point* in the AFS filesystem. A mount point is a special file system element that looks and acts like a regular UNIX directory, but tells AFS the volume's name. When you change to a different directory (by using the **cd** command, for example) you sometimes *cross* a mount point and start accessing the contents of a different volume than before. You normally do not notice the crossing, however, because AFS automatically interprets mount points and

retrieves the contents of the new directory from the appropriate volume. You do not need to track which volume, partition, or file server machine is housing a directory's contents. If you are interested, though, you can learn a volume's location; for instructions, see "Locating Files and Directories" on page 21.

If your system administrator has followed the conventional practice, your home directory corresponds to one volume, which keeps its contents together on one partition of a file server machine. User volumes are typically named **user.username**. For example, the volume for a user named **smith** in the cell **abc.com** is called **user.smith** and is mounted at the directory **/afs/abc.com/usr/smith**.

Because AFS volumes are stored on different file server machines, when a machine becomes unavailable only the volumes on that machine are inaccessible. Volumes stored on other machines are still accessible. However, if a volume's mount point resides in a volume that is stored on an unavailable machine, the former volume is also inaccessible. For that reason, volumes containing frequently used directories (for example, **/afs** and **/afs/cellname**) are often copied and distributed to many file server machines.

## Volume Quotas

Each volume has a size limit, or *quota*, assigned by the system administrator. A volume's quota determines the maximum amount of disk space the volume can consume. If you attempt to exceed a volume's quota, you receive an error message. For instructions on checking volume quota, see "Displaying Volume Quota" on page 19.

Volumes have completely independent quotas. For example, say that the current working directory is **/afs/abc.com/usr/smith**, which is the mount point for the **user.smith** volume with 1000 free blocks. You try to copy a 500 block file from the current working directory to the **/afs/abc.com/usr/pat** directory, the mount point for the volume **user.pat**. However, you get an error message saying there is not enough space. You check the volume quota for **user.pat**, and find that the volume only has 50 free blocks.

## Using Files in AFS

### The Cache Manager

You can access the AFS filesystem only when working on an AFS client machine. The *Cache Manager* on that machine is your agent in accessing information stored in the AFS filesystem. When you access a file, the Cache Manager on your client machine requests the file from the appropriate file server machine and stores (*caches*) a copy of it on your client machine's local disk. Application programs on your client machine use the local, cached copy of the file. This improves performance because it is much faster to use a local file than to send requests for file data across the network to the file server machine.

Because application programs use the cached copy of a file, any changes you make are not necessarily stored permanently to the central version stored on the file server machine until the file closes. At that point, the Cache Manager writes your changes back to the file server machine, where they replace the corresponding parts of the existing file. Some application programs close a file in this way each time you issue their **save** command (and then immediately reopen the file so that you can continue working). With other programs, issuing the **save** command writes the changes only to the local cached copy. If you use the latter type of text editor, you need to close the file periodically to make sure your changes are stored permanently.

If a file server machine becomes inaccessible, you can continue working with the local, cached copy of a file fetched from that machine, but you cannot save your changes permanently until the server machine is again accessible.

## Updating Copies of Cached Files

When the central version of a file changes on the file server machine, the AFS *File Server* process running on that machine advises all other Cache Managers with copies of that file that their version is no longer valid. AFS has a special mechanism for performing these notifications efficiently. When the File Server sends the Cache Manager a copy of a modifiable file, it also sends a *callback*. A callback functions as a promise from the File Server to contact the Cache Manager if the centrally stored copy of the file is changed while it is being used. If that happens, the File Server *breaks* the callback. If you run a program that requests data from the changed file, the Cache Manager notices the broken callback and gets an updated copy of the file from the File Server. Callbacks ensure that you are working with the most recent copy of a file.

**Note:** The callback mechanism does not guarantee that you immediately see the changes someone else makes to a file you are using. Your Cache Manager does not notice the broken callback until your application program asks it for more data from the file.

## Multiple Users Modifying Files

Like a standard UNIX file system, AFS preserves only the changes to a file that are saved last, regardless of who made the changes. When collaborating with someone on the same files, you must coordinate your work to avoid overwriting each other's changes. You can use AFS access control lists (ACLs) to limit the ability of other users to access or change your files, and so prevent them from accidentally overwriting your files. See "Protecting Your Directories and Files" on page 27.

## AFS Security

AFS makes it easy for many users to access the same files, but also uses several mechanisms to ensure that only authorized users access the AFS filesystem. The mechanisms include the following:

- Passwords and mutual authentication ensure that only authorized users access AFS filesystem
- Access control lists enable users to restrict or permit access to their own directories

### Passwords and Mutual Authentication

AFS uses two related mechanisms to ensure that only authorized users access the filesystem: passwords and mutual authentication. Both mechanisms require that a user prove his or her identity.

When you first identify yourself to AFS, you must provide the password associated with your username, to prove that you are who you say you are. When you provide the correct password, you become *authenticated* and your Cache Manager receives a *token*. A token is a package of information that is scrambled by an AFS authentication program using your AFS password as a key. Your Cache Manager can unscramble the token because it knows your password and AFS's method of scrambling.

The token acts as proof to AFS server programs that you are authenticated as a valid AFS user. It serves as the basis for the second means through which AFS creates security, called *mutual authentication*. Under mutual authentication, both parties communicating across the network prove their identities to one another. AFS requires mutual authentication whenever a server and client (most often, a Cache Manager) communicate with each other.

The mutual authentication protocol that AFS uses is designed to make it very difficult for people to authenticate fraudulently. When your Cache Manager contacts a File Server on your behalf, it sends the token you obtained when you authenticated. The token is encrypted with a key that only an AFS File Server can know. If the File Server can decrypt your token, it can communicate with your Cache Manager. In turn, the Cache Manager accepts the File Server as genuine because the File Server can decrypt and use the information in the token.

## Access Control Lists

AFS uses *access control lists (ACLs)* to determine who can access the information in the AFS filesystem. Each AFS directory has an ACL to specify what actions different users can perform on that directory and its files. An ACL can contain up to about 20 entries for users, groups, or both; each entry lists a user or group and the permissions it possesses.

The owner of a directory and system administrators can always administer an ACL. Users automatically own their home directories and subdirectories. Other non-owner users can define a directory's ACL only if specifically granted that permission on the ACL. For more information on ACLs, see "Protecting Your Directories and Files" on page 27 .

A group is composed of one or more users and client machines. If a user belongs to a group that appears on an ACL, the user gets all of the permissions granted to that group, just as if the user were listed directly on the ACL. Similarly, if a user is logged into a client machine that belongs to a group, the user has all of the permissions granted to that group. For instructions on defining and using groups, see "Using Groups" on page 43.

All users who can access your cell's filesystem, authenticated or not, are automatically assigned to a group called **system:anyuser**. For a discussion of placing the **system:anyuser** group on ACLs, see "Extending Access to Users from Foreign Cells" on page 32.

**Note:** You can use the UNIX mode bits to control access on specific files within an AFS directory; however, the effect of these mode bits is different under AFS than in the standard UNIX file system. See "File and Directory Protection" on page 6.

## Differences Between UNIX and AFS

AFS is designed to be similar to the UNIX file system. For instance, many of the basic UNIX file manipulation commands (**cp** for copy, **rm** for remove, and so on) are the same in AFS as they are as in UNIX. All of your application programs work as they did before. The following sections describe some of the differences between a standard UNIX file system and AFS.

### File Sharing

AFS enables users to share remote files as easily as local files. To access a file on a remote machine in AFS, you simply specify the file's pathname. In contrast, to access a file in a remote machine's UNIX file system, you must log into the remote machine or create a mount point on the local machine that points to a directory in the remote machine's UNIX file system.

AFS users can see and share all the files under the **/afs** root directory, given the appropriate privileges. An AFS user who has the necessary privileges can access a file in any AFS cell, simply by specifying the file's pathname. File sharing in AFS is not restricted by geographical distances or operating system differences.

### Login and Authentication

To become an authenticated AFS user, you need to provide a password to AFS.

- On machines that use an AFS-modified login utility, logging in is a one-step process; your initial login automatically authenticates you with AFS.
- On machines that do not use an AFS-modified login utility, you must perform two steps.
  1. Log in to your local machine.
  2. Issue the **klog** command with the **-setpag** argument to authenticate with AFS and get your token.

Your system administrator can tell you whether your machine uses an AFS-modified login utility or not. Then see the login instructions in "Logging in and Authenticating with AFS" on page 9.

AFS authentication passwords are stored in special AFS database, rather than in the local password file (**/etc/passwd** or equivalent). If your machine uses an AFS-modified login utility, you can change your password with a single command. If your machine does not use an AFS-modified login utility, you must issue separate commands to change your AFS and local passwords. See "Changing Your Password" on page 17.

### File and Directory Protection

AFS does not rely on the mode bit protections of a standard UNIX system (though its protection system does interact with these mode bits). Instead, AFS uses an access control list (ACL) to control access to

each directory and its contents. The following list summarizes the differences between the two methods:

- UNIX mode bits specify three types of access permissions: **r (read)**, **w (write)**, and **x (execute)**. An AFS ACL uses seven types of permissions: **r (read)**, **l (lookup)**, **i (insert)**, **d (delete)**, **w (write)**, **k (lock)**, and **a (administer)**. For more information, see "The AFS ACL Permissions" on page 27 and "How AFS Uses the UNIX Mode Bits" on page 41.
- The three sets of mode bits on each UNIX file or directory enable you to grant permissions to three users or groups of users: the file or directory's owner, the group that owns the file or directory, and all other users. An ACL can accommodate up to about 20 entries, each of which extends certain permissions to a user or group. Unlike standard UNIX, a user can belong to an unlimited number of groups, and groups can be defined by both users and system administrators. See "Using Groups" on page 43.
- UNIX mode bits are set individually on each file and directory. An ACL applies to all of the files in a directory. While at first glance the AFS method possibly seems less precise, in actuality (given a proper directory structure) there are no major disadvantages to directory-level protections and they are easier to establish and maintain.

## Machine Outages

The kinds of failures you experience when a standard UNIX file system goes down are different than when one or more individual AFS file server machines become unavailable. When a standard UNIX file system is inaccessible, the system simply locks up and you can lose changes to any files with which you were working.

When an AFS file server machine becomes inaccessible, you cannot access the files on that machine. If a copy of the file is available from another file server machine, however, you do not necessarily even notice the server outage. This is because AFS gives your cell's system administrators the ability to store copies of popular programs on multiple file servers. The Cache Manager chooses between the copies automatically; when one copy becomes unavailable, the Cache Manager simply chooses another.

If there are no other copies of a file that is stored on an inaccessible server machine, you can usually continue to use the copy stored in your client machine's local AFS cache. However, you cannot save changes to files stored on an inaccessible file server machine until it is accessible again.

## Remote Commands

The UNIX *remote commands* enable you to run programs on a remote machine without establishing a connection to it by using a program such as **telnet**. Many of the remote commands (such as **ftp**, **rnp**, and **rsh**) remain available in AFS, depending on how your administrators have configured them. If the remote machine has a Cache Manager, your token is used there also and you are authenticated while the remote command runs. If the remote machine does not run a Cache Manager, you receive the following message:

```
Warning: unable to authenticate.
```

In this case, you are logged into the remote machine's UNIX file system, but you are not authenticated to AFS. You can access the local files on the remote machine and the AFS directories that grant access to the **system:anyuser** group, but you cannot access protected AFS directories.

## Differences in the Semantics of Standard UNIX Commands

This section summarizes differences in the functionality of some commonly issued UNIX commands.

### **chmod**

Only members of the **system:administrators** group can use this command to turn on the `setuid`, `setgid` or sticky mode bits on AFS files. (For more information about this group, see "Using the System Groups on ACLs" on page 31.)

### **chown**

Only members of the **system:administrators** group can issue this command on AFS files.

### **chgrp**

Only members of the **system:administrators** group can issue this command on AFS files and directories.

### **groups**

If the user's AFS tokens are identified by a process authentication group (PAG), the output of this command includes two large numbers. For a description of PAGs, see "Authenticating with AFS" on page 10.

### **inetd**

The AFS version of this daemon authenticates remote issuers of the AFS-modified **rcp** and **rsh** commands with AFS.

### **login utilities**

AFS-modified login utilities both log you into the local UNIX file system and authenticate you with AFS.

### **ln**

You cannot use this command to create a hard link between files that reside in different AFS directories. You must add the **-s** option to create a symbolic link instead.

## Using AFS with NFS

Some cells use the Networking File System (NFS) in addition to AFS. If you work on an NFS client machine, your system administrator can configure it to access the AFS filespace through a program called the *NFS/AFS Translator*<sup>TM</sup>. See "Appendix A, Using the NFS/AFS Translator" on page 61.



## Chapter 2. Using AFS

This chapter explains how to perform four basic AFS tasks: logging in and authenticating with AFS, ending an AFS session, accessing the AFS filesystem, and changing your password.

### Logging in and Authenticating with AFS

To access the AFS filesystem as an authenticated user, you must both log into an AFS client machine's local (UNIX) file system and authenticate with AFS. When you log in, you establish your local system identity. When you authenticate, you prove your identity to AFS and obtain a token, which your Cache Manager uses to prove your authenticated status to the AFS server processes it contacts on your behalf. Users who are not authenticated (who do not have a token) have limited access to AFS directories and files.

#### Logging In

On machines that use an AFS-modified login utility, you log in and authenticate in one step. On machines that do not use an AFS-modified login utility, you log in and authenticate in separate steps. To determine which type of login utility your machine uses, you can check for AFS tokens after logging in, or ask your system administrator, who can also tell you about any differences between your login procedure and the two methods described here.

#### To Log In Using an AFS-modified Login Utility

Provide your username at the `login:` prompt that appears when you establish a new connection to a machine. Then provide your password at the `Password:` prompt as shown in the following example. (Your password does not echo visibly on the screen.)

```
login: username
Password: password
```

If you are not sure which type of login utility is running on your machine, it is best to issue the **tokens** command to check if you are authenticated; for instructions, see "To Display Your Tokens" on page 12. If you do not have tokens, issue the **klog** command as described in "To Authenticate with AFS" on page 11.

#### To Log In Using a Two-Step Login Procedure

If your machine does not use an AFS-modified login utility, you must perform a two-step procedure:

1. Log in to your client machine's local file system by providing a user name and password at the **login** program's prompts.
2. Issue the **klog** command to authenticate with AFS. Include the command's **-setpag** argument to associate your token with a special identification number called a *PAG* (for *process authentication group*). For a description of PAGs, see "Protecting Your Tokens with a PAG" on page 10.

```
% klog -setpag
```

```
Password: your_AFS_password
```

**Note:** If your machine uses a two-step login procedure, you can choose to use different passwords for logging in and authenticating. It is simplest to use the same one for both, though. Talk with your system administrator.

## Authenticating with AFS

To work most effectively in the AFS filesystem, you must authenticate with AFS. When you do, your Cache Manager is given a token as proof of your authenticated status. It uses your token when requesting services from AFS servers, which accept the token as proof of your authenticated status. If you do not have a token, AFS servers consider you to be the **anonymous** user and your access to AFS filesystem is limited: you have only the ACL permissions granted to the **system:anyuser** group.

You can obtain new tokens (reauthenticate) at any time, even after using an AFS-modified login utility, which logs you in and authenticates you in one step. Issue the **klog** command as described in "To Authenticate with AFS" on page 11.

## Protecting Your Tokens with a PAG

To make your access to AFS as secure as possible, it is best to associate your tokens with a unique identification number called a *PAG* (for *process authentication group*). AFS-modified login utilities automatically create a PAG and associate the new token with it. To create a PAG when you use the two-step login procedure, include the **klog** command's **-setpag** flag. If you do not use this flag, your tokens are associated with your UNIX UID number instead. This type of association has two potential drawbacks:

- Anyone who can assume your local UNIX identity can use your tokens. The local superuser **root** can always use the UNIX **su** command to assume your UNIX UID, even without knowing your password.
- In some environments, certain programs cannot use your tokens even when it is appropriate for them to do so. For example, printing commands such as **lp** or **lpr** possibly cannot access the files you want to print, because they cannot use your tokens.

## Obtaining Tokens For Foreign Cells

A token is valid only in one cell (the cell whose AFS authentication service issued it). The AFS server processes in any other cell consider you to be the **anonymous** user unless you have an account in the cell and authenticate with its AFS authentication service.

To obtain tokens in a foreign cell, use the **-cell** argument to the **klog** command. You can have tokens for your home cell and one or more foreign cells at the same time.

## The One-Token-Per-Cell Rule

You can have only one token per cell for each PAG you have obtained on a client machine. If you already have a token for a particular cell and issue the **klog** command, the new token overwrites the existing one. Getting a new token is useful if your current token is almost expired but you want to continue accessing AFS files. For a discussion of token expiration, see "Token Lifetime" on page 11.

To obtain a second token for the same cell, you must either login on a different machine or establish another separate connection to the machine where you already have a token (by using the **telnet** utility, for example). You get a new PAG for each separate machine or connection, and can use the associated tokens only while working on that machine or connection.

## Obtaining Tokens as Another User

You can authenticate as another username if you know the associated password. (It is, of course, unethical to use someone else's tokens without permission.) If you use the **klog** command to authenticate as another AFS username, you retain your own local (UNIX) identity, but the AFS server processes recognize you as the other user. The new token replaces any token you already have for the relevant cell (for the reason described in "The One-Token-Per-Cell Rule" on page 10).

## Token Lifetime

Tokens have a limited lifetime. To determine when your tokens expire, issue the **tokens** command as described in "To Display Your Tokens" on page 12. If you are ever unable to access AFS in a way that you normally can, issuing the **tokens** command tells you whether an expired token is a possible reason.

Your cell's administrators set the default lifetime of your token. The AFS authentication service never grants a token lifetime longer than the default, but you can request a token with a shorter lifetime. See the **klog** reference page in the *IBM AFS Administration Reference* to learn how to use its **-lifetime** argument for this purpose.

## Authenticating for DFS Access

If your machine is configured to access a DCE cell's DFS filesystem by means of the AFS/DFS Migration Toolkit, you can use the **dlog** command to authenticate with DCE. The **dlog** command has no effect on your ability to access AFS filesystem.

If your system administrator has converted your AFS account to a DCE account and you are not sure of your DCE password, use the **dpass** command to display it. You must be authenticated as the AFS user whose AFS account was converted to a DCE account, and be able to provide the correct AFS password. Like the **dlog** command, the **dpass** command has no functionality with respect to AFS.

For more information on using the **dlog** and **dpass** commands, see your system administrator.

## To Authenticate with AFS

If your machine is not using an AFS-modified login utility, you must authenticate after login by issuing the **klog** command. You can also issue this command at any time to obtain a token with a later expiration

date than your current token.

```
% klog [-setpag] [-cell <cell name>]
Password: your_AFS_password
```

where

### **-setpag**

Associates the resulting tokens with a PAG (see "Protecting Your Tokens with a PAG" on page 10). Include this flag the first time you obtain a token for a particular cell during a login session or connection. Do not include it when refreshing the token for a cell during the same session.

### **-cell**

Names the cell for which to obtain the token. You must have an account in the cell.

Your password does not echo visibly appear on the screen. When the command shell prompt returns, you are an authenticated AFS user. You can use the **tokens** command to verify that you are authenticated, as described in the following section.

## **To Display Your Tokens**

Use the **tokens** command to display your tokens.

```
% tokens
```

The following output indicates that you have no tokens:

```
Tokens held by the Cache Manager:
--End of list--
```

If you have one or more tokens, the output looks something like the following example, in which the tokens for AFS UID 1022 in the **abc.com** cell expire on August 3 at 2:35 p.m. The tokens for AFS UID 9554 in the **stateu.edu** cell expire on August 4 at 1:02 a.m.

```
Tokens held by the Cache Manager:
User's (AFS ID 1022) tokens for afs@abc.com [Expires Aug  3 14:35]
User's (AFS ID 9554) tokens for afs@stateu.edu [Expires Aug  4  1:02]
--End of list--
```

## **Example: Authenticating in the Local Cell**

Suppose that user **terry** cannot save a file. He uses the **tokens** command and finds that his tokens have expired. He reauthenticates in his local cell under his current identity by issuing the following command:

```
% klog
Password: terry's_password
```

The he issues the **tokens** command to make sure he is authenticated.

```
% tokens
Tokens held by the Cache Manager:
User's (AFS ID 4562) tokens for afs@abc.com [Expires Jun 22 14:35]
--End of list--
```

### Example: Authenticating as a Another User

Now **terry** authenticates in his local cell as another user, **pat**. The new token replaces **terry**'s existing token, because the Cache Manager can store only one token per cell per login session on a machine.

```
% klog pat
Password: pat's_password
% tokens
Tokens held by the Cache Manager:
User's (AFS ID 4278) tokens for afs@abc.com [Expires Jun 23 9:46]
--End of list--
```

### Example: Authenticating in a Foreign Cell

Now **terry** authenticates in the **stateu.edu** cell where his account is called **ts09**.

```
% klog ts09 -cell stateu.edu
Password: ts09's_password
% tokens
Tokens held by the Cache Manager:
User's (AFS ID 4562) tokens for afs@abc.com [Expires Jun 22 14:35]
User's (AFS ID 8346) tokens for afs@stateu.edu [Expires Jun 23 1:02]
--End of list--
```

### Limits on Failed Authentication Attempts

Your system administrator can choose to limit the number of times that you fail to provide the correct password when authenticating with AFS (using either an AFS-modified login utility or the **klog** command). If you exceed the limit, the AFS authentication service refuses further authentication attempts for a period of time set by your system administrator. The purpose of this limit is to prevent unauthorized users from breaking into your account by trying a series of passwords.

To determine if your user account is subject to this limit, ask your system administrator or issue the **kas examine** command as described in "To Display Your Failed Authentication Limit and Lockout Time" on page 13.

The following message indicates that you have exceeded the limit on failed authentication attempts.

```
Unable to authenticate to AFS because ID is locked - see your system admin
```

## To Display Your Failed Authentication Limit and Lockout Time

Issue the **kas examine** command to determine if there is a limit on the number of unsuccessful authentication attempts for your user account and any associated lockout time. You can examine only your own account. The fourth line of the output reports the maximum number of times you can provide an incorrect password before being locked out of your account. The `lock time` field on the next line reports how long the AFS authentication service refuses authentication attempts after the limit is exceeded.

```
% kas examine your_username  
Password for your_username: your_AFS_password
```

The following example displays the output for the user **pat**, who is allowed nine failed authentication attempts. The lockout time is 25.5 minutes.

```
User data for pat  
key (15) cksum is 3414844392, last cpw: Thu Oct 21 16:05:44 1999  
password will expire: Fri Nov 26 20:44:36 1999  
9 consecutive unsuccessful authentications are permitted.  
The lock time for this user is 25.5 minutes.  
User is not locked.  
entry never expires. Max ticket lifetime 100.00 hours.  
last mod on Wed Aug 18 08:22:29 1999 by admin  
permit password reuse
```

## Exiting an AFS Session

Because logging in and authenticating with AFS are distinct operations, you must both logout and unauthenticate (issue the **unlog** command to discard your tokens) when exiting an AFS session. Simply logging out does not necessarily destroy your tokens.

You can use the **unlog** command any time you want to unauthenticate, not just when logging out. For instance, it is a good practice to unauthenticate before leaving your machine unattended, to prevent other users from using your tokens during your absence. When you return to your machine, issue the **klog** command to reauthenticate, as described in "To Authenticate with AFS" on page 11.

Do not issue the **unlog** command when you are running jobs that take a long time to complete, even if you are logging out. Such processes must have a token during the entire time they need authenticated access to AFS.

If you have tokens from multiple cells and want to discard only some of them, include the **unlog** command's **-cell** argument.

## To Discard Tokens

Issue the **unlog** command to discard your tokens:

```
% unlog -cell <cell name>+
```

Omit the **-cell** argument to discard all of your tokens, or use it to name each cell for which to discard tokens. It is best to provide the full name of each cell (such as **stateu.edu** or **abc.com**).

You can issue the **tokens** command to verify that your tokens were destroyed, as in the following example.

```
% tokens
Tokens held by the Cache Manager:
--End of list--
```

### Example: Unauthenticating from a Specific Cell

In the following example, a user has tokens in both the **accounting** and **marketing** cells at her company. She discards the token for the **acctg.abc.com** cell but keeps the token for the **mktg.abc.com** cell.

```
% tokens
Tokens held by the Cache Manager:
User's (AFS ID 35) tokens for afs@acctg.abc.com [Expires Nov 10 22:30]
User's (AFS ID 674) tokens for afs@mktg.abc.com [Expires Nov 10 18:44]
--End of list--
% unlog -cell acctg.abc.com
% tokens
Tokens held by the Cache Manager:
User's (AFS ID 674) tokens for afs@mktg.abc.com [Expires Nov 10 18:44]
--End of list--
```

### To Log Out

After you have unauthenticated, log out by issuing the command appropriate for your machine type, which is possibly one of the following.

```
% logout

or

% exit

or

% <Ctrl-d>
```

## Accessing the AFS Filespace

While you are logged in and authenticated, you can access files in AFS just as you do in the UNIX file system. The only difference is that you can access potentially many more files. Just as in the UNIX file system, you can only access those files for which you have permission. AFS uses access control lists (ACLs) to control access, as described in "Protecting Your Directories and Files" on page 27.

## AFS Pathnames

AFS pathnames look very similar to UNIX file system names. The main difference is that every AFS pathname begins with the AFS root directory, which is called **/afs** by convention. Having **/afs** at the top of every AFS cell's filesystem links together their filesystems into a global filesystem.

**Note for Windows users:** Windows uses a backslash (\) rather than a forward slash (/) to separate the elements in a pathname. Otherwise, your access to AFS filesystem is much the same as for users working on UNIX machines.

The second element in AFS pathnames is generally a cell's name. For example, the ABC Corporation cell is called **abc.com** and the pathname of every file in its filesystem begins with the string **/afs/abc.com**. Some cells also create a directory at the second level with a shortened name (such as **abc** for **abc.com** or **stateu** for **stateu.edu**), to reduce the amount of typing necessary. Your system administrator can tell you if your cell's filesystem includes shortened names like this. The rest of the pathname depends on how the cell's administrators organized its filesystem.

To access directories and files in AFS you must both specify the correct pathname and have the required permissions on the ACL that protects the directory and the files in it.

## Example: Displaying the Contents of Another User's Directory

The user **terry** wants to look for a file belonging to another user, **pat**. He issues the **ls** command on the appropriate pathname.

```
% ls /afs/abc.com/usr/pat/public
doc/                directions/
guide/              jokes/
library/
```

## Accessing Foreign Cells

You can access files not only in your own cell, but in any AFS cell that you can reach via the network, regardless of geographical location. There are two additional requirements:

- Your Cache Manager's list of foreign cells must include the cell you want to access. Only the local superuser **root** can edit the list of cells, but anyone can display it. See "Determining Access to Foreign Cells" on page 23.
- The ACL on the directory that houses the file, and on every parent directory in the pathname, must grant you the necessary permissions. The simplest way for the directory's owner to extend permission to foreign users is to put an entry for the **system:anyuser** group on the ACL.

The alternative is for the foreign cell's administrator to create an account for you, essentially making you a local user in the cell. The directory's owner creates an ACL entry for you as for any other local user. To authenticate in the foreign cell, issue the **klog** command with the **-cell** argument.



For further discussion of directory and file protection, see "Protecting Your Directories and Files" on page 27.

## Changing Your Password

In cells that use an AFS-modified login utility, the password is the same for both logging in and authenticating with AFS. In this case, you use a single command, **kpasswd**, to change the password.

If your machine does not use an AFS-modified login utility, there are separate passwords for logging into the local file system and authenticating with AFS. (The two passwords can be the same or different, at your discretion.) In this case, use the **kpasswd** command to change your AFS password and the UNIX **passwd** command to change your UNIX password.

Your system administrator can improve cell security by configuring several features that guide your choice of password. Keep them in mind when you issue the **kpasswd** command:

- Limiting the amount of time your password is valid. This improves your cell's security by limiting the amount of time an unauthorized user has to try to guess your password. Your system administrator needs to tell you when your password is due to expire so that you can change it in time. The administrator can configure the AFS-modified login utility to report this information automatically each time you log in. You can also use the **kas examine** command to display the password expiration date, as instructed in "To Display Password Expiration Date and Reuse Policy" on page 17.

You can change your password prior to the expiration date, but your system administrator can choose to set a minimum time between password changes. The following message indicates that the minimum time has not yet passed.

```
kpasswd: password was not changed because you changed it too
recently; see your system administrator
```

- Enforcing password quality standards, such as a minimum length or inclusion of nonalphabetic characters. The administrator needs to tell you about such requirements so that you do not waste time picking unacceptable passwords.
- Rejecting a password that is too similar to the last 20 passwords you used. You can use the **kas examine** command to check whether this policy applies to you, as instructed in "To Display Password Expiration Date and Reuse Policy" on page 17. The following message indicates that the password you have chosen is too similar to a previous password.

```
kpasswd: Password was not changed because it seems like a reused password
```

## To Display Password Expiration Date and Reuse Policy

Issue the **kas examine** command to display your password expiration date and reuse policy. You can examine only your own account. The third line of the output reports your password's expiration date. The last line reports the password reuse policy that applies to you.

```
% kas examine your_username
```

```
Password for your_username: your_AFS_password
```

The following example displays the output for the user **pat**.

```
User data for pat
key (15) cksum is 3414844392, last cpw: Thu Oct 21 16:05:44 1999
password will expire: Fri Nov 26 20:44:36 1999
9 consecutive unsuccessful authentications are permitted.
The lock time for this user is 25.5 minutes.
User is not locked.
entry never expires. Max ticket lifetime 100.00 hours.
last mod on Wed Aug 18 08:22:29 1999 by admin
don't permit password reuse
```

## To Change Your AFS Password

Issue the **kpasswd** command, which prompts you to provide your old and new passwords and to confirm the new password. The passwords do not echo visibly on the screen.

```
% kpasswd
Old password: current_password
New password (RETURN to abort): new_password
Retype new password: new_password
```

## To Change Your UNIX Password

Issue the UNIX **passwd** command, which prompts you to provide your old and new passwords and to confirm the new password. The passwords do not echo visibly on the screen. On many machines, the **passwd** resides in the **/bin** directory, and you possibly need to type the complete pathname.

```
% passwd
Changing password for username.
Old password: current_password
New password: new_password
Retype new passwd: new_password
```

## Chapter 3. Displaying Information about AFS

This chapter explains how to display information that can help you use AFS more effectively. It includes the following sections.

- "Displaying Volume Quota" on page 19
- "Locating Files and Directories" on page 21.
- "Checking the Status of Server Machines" on page 22
- "Determining Access to Foreign Cells" on page 23
- "Displaying Server Preference Ranks" on page 24

### Displaying Volume Quota

By convention, the files in your home directory are stored together in a single volume. (For information about volumes, see "Volumes and Mount Points" on page 2.) To allocate your cell's available disk space as fairly as possible, your system administrators impose a size limit, or *quota*, on each volume. You cannot store more data in a volume than its quota allows. If a volume is close to its quota, you sometimes cannot save changes you have made to files stored in the volume.

The amount of space available on the partition that houses the volume also limits how large the volume can grow. If the disk partition is full, you can become unable to save changes to a file even though the volume is not close to its quota.

Check the quota on your home volume periodically to make sure you have adequate space. Also, if you encounter problems saving a file, check the quota of the volume in which the file is stored. Use the following commands to display volume quota.

- The **fs quota** command lists the percentage of the volume quota used.
- Both the **fs listquota** and **fs examine** commands list the volume name, its maximum size (quota), and its current size. They also report the following additional information.
  - The **fs listquota** command lists the percentage used of both the volume and the partition.
  - The **fs examine** command lists the partition's size, the amount of space currently used, and any messages associated with the volume.

### To Display Percentage of Quota Used

Issue the **fs quota** command to display the percentage of the quota currently used for the volume that contains a specified directory or file.

```
% fs quota [<dir/file path>*]
```

where *dir/file path* specifies the pathname of a file or directory in each volume for which to display quota information. If you do not provide a pathname, the output reports quota information for the volume that contains the current working directory.

### Example: Displaying Percentage of Quota Used

The following example displays the percentage of quota used for the volumes that contain two user home directories in the ABC Corporation cell.

```
% cd /afs/abc.com/usr
% fs quota terry pat
34% of quota used.
85% of quota used.
```

### To Display Quota and Other Information about a Volume

Issue the **fs listquota** command to display the following information:

- The name of the volume that houses each specified file or directory
- The quota, expressed as a number of kilobytes (1024 indicates one megabyte)
- The current size of the volume (the number of kilobytes of currently used)
- The percentage of the quota used
- The percentage of space used on the disk partition housing the volume

The command's syntax is as follows.

```
% fs listquota [<dir/file path>+]
```

where *dir/file path* specifies the pathname of a file or directory in each volume for which to display quota information. If you do not provide a pathname, the output reports quota information for the volume that contains the current working directory.

### Example: Display Quota and Other Information about a Volume

The following example displays quota information about the volume that houses the home directory of user **terry**.

```
% fs listquota ~terry
Volume Name      Quota   Used    % Used  Partition
user.terry       10000  3400    34%     86%
```

### To Display Quota and Other Information about a Volume and Partition

Issue the **fs examine** command to display the following information about a volume and the partition it resides on:

- The volume's ID number (abbreviated in the output as *vid*)
- The volume name

- The volume's quota and current size, in kilobytes
- The number of kilobyte blocks available on the disk partition housing the volume and the total size of that partition
- An *off-line message* associated with the volume, if any, as set by a system administrator

The command's syntax is as follows.

```
% fs examine [<dir/file path>']
```

where *dir/file path* specifies the pathname of a file or directory in each volume for which to display quota information. If you do not provide a pathname, the output reports quota information for the volume that contains the current working directory.

### Example: Displaying Quota and Other Information about a Volume and Partition

The following example displays quota and other information about the volume that houses the current working directory.

```
% fs examine
Volume status for vid = 536871122 named user.terry
Current disk quota is 10000
Current blocks used are 5745
The partition has 1593 blocks available out of 99162
```

## Locating Files and Directories

Normally, you do not need to know which file server machine stores the volume containing a file or directory. Given the pathname to a file, the Cache Manager on your client machine automatically accesses the appropriate server machine.

If you become unable to access a file, however, it can be useful to know which file server machine houses it. You can then check whether the File Server process or machine is functioning correctly, as described in "Checking the Status of Server Machines" on page 22. Or, if your system administrators schedule downtime for a machine, you can learn whether the outage is likely to prevent you from accessing certain files.

### To Display a File or Directory's Location

Issue the **fs whereis** command to display the file server machine on which a file or directory is stored.

```
% fs whereis [<dir/file path>']
```

where *dir/file path* specifies the pathname of each file or directory for which you want location information. If you do not provide a pathname, the output reports the machine housing the volume that contains the current working directory.

If the output mentions more than one machine, there is a copy of the volume at each site (the volume is *replicated*). Your system administrators can choose to replicate volumes that contain information many people need to use, both for load balancing reasons and to make the information available even if there is an outage on one machine that houses the volume.

### Example: Displaying Directory Location

The following example displays the names of the server machines that house the home volumes for users **terry** and **pat**.

```
% cd /afs/abc.com/usr
% fs whereis terry pat
File /afs/abc.com/usr/terry is on host fs2.abc.com
File /afs/abc.com/usr/pat is on host fs3.abc.com
```

## Checking the Status of Server Machines

Sometimes one or more server machines in your cell become inaccessible due to hardware problems, software problems, or routine maintenance. During the outage, you cannot access files stored on those machines or save any changes you have made to files that are stored on those machines. (Your Cache Manager possibly has copies of the files stored locally, which you can still work with.)

To check the status of server machines, use the **fs checkservers** command. If a server machine has more than one network interface address (is *multihomed*), the Cache Manager sends the status-checking message to all of the machine's interfaces. If at least one of the server's interfaces replies, the command's output reports the machine as accessible. If there is no reply from any of the interfaces, the output reports the machine as inaccessible but displays only one of the interfaces (usually the one with the best preference rank; see "Displaying Server Preference Ranks" on page 24).

To check the status of different groups of server machines, combine the **fs checkservers** command's options as indicated:

- To check file server machines in the local cell only, do not include any options
- To check file server machines in a particular foreign cell only, include the **-cell** argument
- To check every file server machine that your Cache Manager has contacted in any cell, include the **-all** flag

It can take several minutes for the command shell prompt to return, because the **fs** command interpreter waits a timeout period before concluding that an unresponsive machine is really inaccessible. To have the command shell prompt return immediately, add the ampersand (**&**), which runs the **fs checkservers** command in the background.

### To Check File Server Machine Status

Issue the **fs checkservers** command to check the status of file server machines.

```
% fs checkservers [-cell <cell to check>] [-all] [&]
```

where

#### **-cell**

Names each cell for which to check server machine status. Do not combine this argument and the **-all** flag.

#### **-all**

Checks the status of all server machines. Do not combine this flag and the **-cell** argument.

The following message indicates that all server machines replied to the Cache Manager's status-checking message:

```
All servers are running.
```

Otherwise, a message like the following lists the inaccessible machines:

```
These servers unavailable due to network or server problems: list of machines.
```

### **Example: Checking Server Machine Status**

The following example checks the status of every file server machine the Cache Manager has contacted in any cell. Two machines are not responding.

```
% fs checkservers -all &
These servers unavailable due to network or server problems:
    fs1.abc.com server7.stateu.edu.
```

## **Determining Access to Foreign Cells**

The Cache Manager maintains a list of foreign cells that it knows how to reach. A cell must appear in the list for you to access its AFS filesystem. (In addition, the ACL on each directory in the pathname to the file must grant you the necessary permissions, and your system administrator must mount the cell in the local AFS filesystem--by convention, just under the **/afs** directory.)

### **To Display Foreign Cells**

Issue the **fs listcells** command to display the cells you can access from this client machine. It can take several minutes for the command shell prompt to return. The Cache Manager stores the machines as IP addresses, but has the addresses translated to names before displaying them. To have the command shell prompt return immediately, use the ampersand (**&**) to run the **fs listcells** command in the background as in the following example.

```
% fs listcells &
```

```
Cell abc.com on hosts
    db1.abc.com
    db2.abc.com
    db3.abc.com
Cell test.abc.com on hosts
    test4.abc.com.
Cell stateu.edu on hosts
    sv5.stateu.edu.
    sv2.stateu.edu.
    sv11.stateu.edu.
Cell def.com on hosts
    serverA.def.com
```

## Displaying Server Preference Ranks

The Cache Manager stores a list of preference ranks for file server machines. When it needs to access a file or directory, the Cache Manager compares the ranks of the file server machines that house the relevant volume. It first tries to access the volume on the machine with the best rank. (If a file server machine is multihomed--has more than one network interface--the Cache Manager actually assigns a separate rank to each interface.)

The Cache Manager assigns a default rank to a file server machine interface by comparing its own IP address to the interface's IP address. It assigns a better rank to interfaces that are on its own subnetwork or network than to interfaces on other networks. Therefore, the ranks bias the Cache Manager to fetch files from file server machines that are close in terms of network distance, which tends to reduce network traffic and help the Cache Manager deliver data to applications more quickly.

The Cache Manager stores each rank as a pairing of a file server machine interface's IP address and an integer rank from the range **0** to **65,534**. A lower number is a better rank. To display the server preference ranks on the local client machine, use the **fs getserverprefs** command.

The Cache Manager stores a separate but similar set of ranks for Volume Location (VL) Servers, which tell the Cache Manager the location of volumes that house files and directories. To display those ranks, add the **-vlservers** flag to the **fs getserverprefs** command.

If the default ranks do not seem to result in the best performance, your system administrator can change them. Ask your system administrator about the ranks if appropriate.

### To Display Server Preference Ranks

Issue the **fs getserverprefs** command to display the file server machine preference ranks used by the Cache Manager on the local machine. To display VL Server ranks, add the **-vlservers** flag. By default, the Cache Manager has the IP address of each interface translated into a hostname before displaying it. To bypass the translation and display IP addresses, include the **-numeric** flag. This can significantly speed up the command's output.

```
% fs getserverprefs [-numeric] [-vlservers]
```

The following example displays the file server machine preference ranks for a client machine in the **abc.com** cell. The ranks of the file server machines in that cell are lower than the ranks of the file server



machines from the foreign cell, **def.com**. Because the **-numeric** flag is not used, the output displays hostnames. The appearance of an IP address for two machines indicates that translating them was not possible.

```
% fs getserverprefs
fs2.abc.com          20007
fs3.abc.com          30002
fs1.abc.com          20011
fs4.abc.com          30010
server1.def.com      40002
192.12.105.34        40000
server6.def.com      40012
192.12.105.37        40005
```



## Chapter 4. Protecting Your Directories and Files

This chapter explains how to protect AFS files and directories by defining permissions on an access control list.

### Access Control Lists

AFS augments and refines the standard UNIX scheme for controlling access to files and directories. Instead of using mode bits to define access permissions for individual files, as UNIX does, AFS stores an *access control list (ACL)* with each directory. It defines which users and groups can access the directory and the files it contains, and in what manner. An ACL can store up to about 20 entries, each of which pairs a user or group and a set of permissions. AFS defines seven permissions rather than the three that UNIX uses.

Another refinement to the standard UNIX protection scheme is that users can define their own protection *groups* and then place the groups on ACLs as though they were individual users. A group can include both users and machines. Each user who belongs to a group inherits all of the permissions granted to the group on the ACL. Similarly, all users who are logged into a machine that belongs to a group inherits all of the permissions granted to the group. You can create groups to place on ACLs and also use groups that other users have created. To learn more about group creation, see "Using Groups" on page 43.

In addition, AFS defines two system groups called **system:anyuser** and **system:authuser**. By placing them on ACLs, you can grant access to large numbers of users at once. See "Using the System Groups on ACLs" on page 31.

Although AFS uses ACLs to protect files and directories, it also uses the UNIX mode bits to a limited extent. See "How AFS Uses the UNIX Mode Bits" on page 41.

### Directory Level Access Control

As noted, AFS associates an ACL with each directory, and it applies to all of the files stored in the directory. Files do not have separate ACLs. Defining access at the directory level has several consequences:

- The permissions on a directory's ACL apply to all of the files in the directory. When you move a file to a different directory, you effectively change its permissions to those on its new directory's ACL. Changing a directory's ACL changes the protection on all the files in it.
- When you create a subdirectory, it inherits the current ACL of its parent directory. You can then set the subdirectory's ACL to be different from its parent's. However, do not make the ACL on the parent directory more restrictive than on a subdirectory, because that can prevent users from accessing the subdirectory even when they have the necessary permissions on its ACL. Specifically, a user must have the **l (lookup)** permission (defined in "The AFS ACL Permissions" on page 27) on the parent directory to reach its subdirectories.

As a general rule, it makes sense to grant fairly liberal access to your home directory. If you need to protect certain files more closely, place them in subdirectories that have more restrictive ACLs.

## The AFS ACL Permissions

There are seven standard AFS ACL permissions. Functionally, they fall into two groups: one that applies to the directory itself and one that applies to the files.

### The Four Directory Permissions

The four permissions in this group are meaningful with respect to the directory itself. For example, the **i** (**insert**) permission does not control addition of data to a file, but rather creation of a new file or subdirectory.

#### The **l** (**lookup**) permission

This permission functions as something of a gate keeper for access to the directory and its files, because a user must have it in order to exercise any other permissions. In particular, a user must have this permission to access anything in the directory's subdirectories.

This permission enables a user to issue the following commands:

- The **ls** command to list the names of the files and subdirectories in the directory
- The **ls -ld** command to obtain complete status information for the directory element itself
- The **fs listacl** command to examine the directory's ACL

This permission does not enable a user to read the contents of a file in the directory or to issue the **ls -l** or **fs listacl** commands with a filename as the argument. Those operations require the **r** (**read**) permission, which is described in "The Three File Permissions" on page 29.

Similarly, this permission does not enable a user to issue the **ls**, **ls -l**, **ls -ld**, or **fs listacl** commands against a subdirectory of the directory. Those operations require the **l** permission on the ACL of the subdirectory itself.

#### The **i** (**insert**) permission

This permission enables a user to add new files to the directory, either by creating or copying, and to create new subdirectories. It does not extend into any subdirectories, which are protected by their own ACLs.

#### The **d** (**delete**) permission

This permission enables a user to remove files and subdirectories from the directory or move them into other directories (assuming that the user has the **i** permission on the ACL of the other directories).

#### The **a** (**administer**) permission

This permission enables a user to change the directory's ACL. Members of the **system:administrators** group implicitly have this permission on every directory (that is, even if that group does not appear on the ACL). Similarly, the owner of a directory implicitly has this permission on its ACL and those of all directories below it.

## The Three File Permissions

The three permissions in this group are meaningful with respect to files in a directory, rather than the directory itself or its subdirectories.

### The r (read) permission

This permission enables a user to read the contents of files in the directory and to issue the **ls -l** command to stat the file elements.

### The w (write) permission

This permission enables a user to modify the contents of files in the directory and to issue the **chmod** command to change their UNIX mode bits.

### The k (lock) permission

This permission enables a user to run programs that issue system calls to lock files in the directory.

## The Eight Auxiliary Permissions

AFS provides eight additional permissions that do not have a defined meaning. They are denoted by the uppercase letters **A, B, C, D, E, F, G, and H**.

Your system administrator can choose to write application programs that assign a meaning to one or more of the permissions, and then place them on ACLs to control file access by those programs. Use the **fs listacl** and **fs setacl** commands to display and set the auxiliary permissions on ACLs just like the standard seven.

## Shorthand Notation for Sets of Permissions

You can combine the seven permissions in any way in an ACL entry, but certain combinations are more useful than others. Four of the more common combinations have corresponding shorthand forms. When using the **fs setacl** command to define ACL entries, you can provide either one or more of the individual letters that represent the permissions, or one of the following shorthand forms:

### **all**

Represents all seven standard permissions (**rlidwka**)

### **none**

Removes the entry from the ACL, leaving the user or group with no permission

### **read**

Represents the **r (read)** and **l (lookup)** permissions

### **write**

Represents all permissions except **a (administer)**: **rlidwk**

## About Normal and Negative Permissions

ACLs enable you both to grant and to deny access to a directory and the files in it. To grant access, use the **fs setacl** command to create an ACL entry that associates a set of permissions with a user or group, as described in "Changing an ACL" on page 34. When you use the **fs listacl** command to display an ACL (as described in "Displaying an ACL" on page 32), such entries appear underneath the following header, which uses the term *rights* to refer to permissions:

```
Normal rights
```

There are two ways to deny access:

1. The recommended method is simply to omit an entry for the user or group from the ACL, or to omit the appropriate permissions from an entry. Use the **fs setacl** command to remove or edit an existing entry. In most cases, this method is enough to prevent access of certain kinds or by certain users. You must take care, however, not to grant the undesired permissions to any groups to which such users belong.
2. The more explicit method for denying access is to place an entry on the *negative permissions* section of an ACL, by including the **-negative** flag to the **fs setacl** command. For instructions, see "To Add, Remove, or Edit Negative ACL Permissions" on page 36. The **fs listacl** command displays the negative permissions section of an ACL underneath the following header:

```
Negative rights
```

When determining what type of access to grant to a user, AFS first examines all of the entries in the normal permissions section of the ACL. It then subtracts any permissions associated with the user (or with groups to which the user belongs) on the negative permissions section of the ACL. Therefore, negative permissions always cancel out normal permissions.

Negative permissions can be confusing, because they reverse the usual meaning of the **fs setacl** command. In particular, combining the **none** shorthand and the **-negative** flag is a double negative: by removing an entry from the negative permissions section of the ACL, you enable a user once again to obtain permissions via entries in the normal permissions section. Combining the **all** shorthand with the **-negative** flag explicitly denies all permissions.

It is useless to create an entry in the negative permissions section if an entry in the normal permissions section grants the denied permissions to the **system:anyuser** group. In this case, users can obtain the permissions simply by using the **unlog** command to discard their tokens. When they do so, AFS recognizes them as the **anonymous** user, who belongs to the **system:anyuser** group but does not match the entries on the negative permissions section of the ACL.

## Setting DFS ACLs

If your machine is configured to access a DCE cell's DFS filespace via the AFS/DFS Migration Toolkit, then you can use the AFS **fs listacl** and **fs setacl** commands to display and set the ACLs on DFS directories and files that you own. However, DFS uses a slightly different set of permissions and a different syntax for ACL entries. See the DFS documentation or ask your system administrator.

## Using the System Groups on ACLs

AFS defines two *system groups* that grant access to a large number of users at once when placed on an ACL. However, you cannot control the membership of these groups, so consider carefully what kind of permissions you wish to give them. (You do control the membership of the groups you own; see "Using Groups" on page 43.)

### **system:anyuser**

Includes anyone who can access the cell's file tree, including users who have tokens in the local cell, users who have logged in on a local AFS client machine but have not obtained tokens (such as the local superuser **root**), and users who have connected to a local machine from outside the cell. Creating an ACL entry for this group is the only way to extend access to AFS users from foreign cells, unless your system administrator creates local authentication accounts for them.

### **system:authuser**

Includes all users who have a valid AFS token obtained from the local cell's AFS authentication service.

The third system group, **system:administrators**, includes a small group of administrators who have extensive permissions in the cell. You do not generally need to put this group on your ACLs, because its members always have the **a (administer)** permission on every ACL, even if the group does not appear on it.

## Enabling Access to Subdirectories

A user must have the **I** permission on a directory to access its subdirectories in any way. Even if users have extensive permissions on a subdirectory, they cannot access it if the parent directory's ACL does not grant the **I** permission.

You can grant the **I** permission in one of three ways: grant it to a system group (**system:anyuser** or **system:authuser**), grant it to individual users, or grant it to one or more groups of users defined by you or other users (see "Using Groups" on page 43). Granting the **I** permission to the **system:anyuser** group is the easiest option and is generally secure because the permission only enables users to list the contents of the directory, not to read the files in it. If you want to enable only locally authenticated users to list a directory's contents, substitute the **system:authuser** group for the **system:anyuser** group. Your system administrator has possibly already created an entry on your home directory's ACL that grants the **r** and **I** permissions to the **system:anyuser** group.

## Extending Access to Service Processes

It is sometimes necessary to grant more extensive permissions to the **system:anyuser** group so that processes that provide printing and mail delivery service can work correctly. For example, printing processes sometimes need the **r** permission in addition to the **l** permission. A mail delivery process possibly needs the **i** permission to place new messages in your mail directory. Your system administrator has probably already created the necessary ACL entries. If you notice an ACL entry for which the purpose is unclear, check with your system administrator before removing it.

## Extending Access to Users from Foreign Cells

The only way to grant access to users from foreign cells who do not have an account in your cell is to put the **system:anyuser** group on an ACL. Remember, however, that such an entry extends access to everyone who can reach your cell, not just the AFS users from foreign cells that you have in mind.

## Displaying an ACL

To display the ACL associated with a file or directory, issue the **fs listacl** command.

**Note for AFS/DFS Migration Toolkit users:** If the machine on which you issue the **fs listacl** command is configured to access a DCE cell's DFS filespace via the AFS/DFS Migration Toolkit, you can use the command to display the ACL on DFS files and directories. To display a DFS directory's Initial Container or Initial Object ACL instead of the regular one, include the **fs listacl** command's **-id** or **-if** flag. For more information, ask your system administrator. The **fs** command interpreter ignores the **-id** and **-if** flags if you include them when displaying an AFS ACL.

### To display an ACL

1. Issue the **fs listacl** command.

```
% fs listacl [<dir/file path>*]
```

where

**la**

Is an acceptable alias for **listacl** (and **lista** is the shortest acceptable abbreviation).

***dir/file path***

Names one or more files or directories for which to display the ACL. For a file, the output displays the ACL on its directory. If you omit this argument, the output is for the current working directory. Partial pathnames are interpreted relative to the current working directory. You can also use the following notation on its own or as part of a pathname:

.

(A single period). Specifies the current working directory.



..

(Two periods). Specifies the current working directory's parent directory.

\*

(The asterisk). Specifies each file and subdirectory in the current working directory. The ACL displayed for a file is always the same as for its directory, but the ACL for each subdirectory can differ.

The output for each file or directory specified as *dir/file path* begins with the following header to identify it:

```
Access list for dir/file path is
```

The `Normal rights` header appears on the next line, followed by lines that each pair a user or group name and a set of permissions. The permissions appear as the single letters defined in "The AFS ACL Permissions" on page 27, and always in the order **rlidwka**. If there are any negative permissions, the `Negative rights` header appears next, followed by pairs of negative permissions.

If the following error message appears instead of an ACL, you do not have the permissions needed to display an ACL. To specify a directory name as the *dir/file path* argument, you must have the **l** (**lookup**) permission on the ACL. To specify a filename, you must also have the **r** (**read**) permission on its directory's ACL.

```
fs: You don't have the required access permissions on 'dir/file path'
```

## Example: Displaying the ACL on One Directory

The following example displays the ACL on user **terry**'s home directory in the ABC Corporation cell:

```
% fs la /afs/abc.com/usr/terry
Access list for /afs/abc.com/usr/terry is
Normal rights:
  system:authuser rl
  pat rlw
  terry rlidwka
Negative rights:
  terry:other-dept rl
  jones rl
```

where **pat**, **terry**, and **jones** are individual users, **system:authuser** is a system group, and **terry:other-dept** is a group that **terry** owns. The list of normal permissions grants all permissions to **terry**, the **rlw** permissions to **pat**, and the **rl** permissions to the members of the **system:authuser** group.

The list of negative permissions denies the **rl** permissions to **jones** and the members of the **terry:other-dept** group. These entries effectively prevent them from accessing **terry**'s home directory in any way; they cancel out the **rl** permissions extended to the **system:authuser** group, which is the only entry on the normal permissions section of the ACL that possibly applies to them.

## Example: Displaying the ACLs on Multiple Directories

The following example illustrates how you can specify pathnames in different ways, and the appearance of the output for multiple directories. It displays the ACL for three directories: the current working directory (which is a subdirectory of user **terry**'s home directory), the home directory for user **pat**, and another subdirectory of **terry**'s home directory called **plans**.

```
% fs listacl . /afs/abc.com/usr/pat ../plans
Access list for . is
Normal rights:
    system:anyuser rl
    pat:dept rliw
Access list for /afs/abc.com/usr/pat is
Normal rights:
    system:anyuser rl
    pat rlidwka
    terry rliw
Access list for ../plans is
Normal rights:
    terry rlidwka
    pat rlidw
```

## Changing an ACL

To add, remove, or edit ACL entries, use the **fs setacl** command. By default, the command manipulates entries on the normal permissions section of the ACL. To manipulate entries on the negative permissions section, include the **-negative** flag as instructed in "To Add, Remove, or Edit Negative ACL Permissions" on page 36.

You can change any ACL on which you already have the **a** permission. You always have the **a** permission on the ACL of every directory that you own, even if you accidentally remove that permission from the ACL. (The **ls -ld** command reports a directory's owner.) Your system administrator normally designates you as the owner of your home directory and its subdirectories, and you possibly own other directories also.

If an ACL entry already exists for the user or group you specify, then the new permissions completely replace the existing permissions rather than being added to them. In other words, when issuing the **fs setacl** command, you must include all permissions that you want to grant to a user or group.

**Note for AFS/DFS Migration Toolkit users:** If the machine on which you issue the **fs setacl** command is configured to access a DCE cell's DFS file space via the AFS/DFS Migration Toolkit, you can use the command to set the ACL on DFS files and directories. To set a DFS directory's Initial Container or Initial Object ACL instead of the regular one, include the **fs setacl** command's **-id** or **-if** flag. For more information, ask your system administrator. The **fs** command interpreter ignores the **-id** and **-if** flags if you include them when setting an AFS ACL.

## To Add, Remove, or Edit Normal ACL Permissions

Issue the **fs setacl** command to edit entries in the normal permissions section of the ACL. To remove an entry, specify the **none** shorthand as the permissions. If an ACL entry already exists for a user or group, the permissions you specify completely replace those in the existing entry.

```
% fs setacl -dir <directory>+ -acl <access list entries>+
```

where

### sa

Is an acceptable alias for **setacl** (and **seta** is the shortest acceptable abbreviation).

### -dir

Names one or more directories to which to apply the ACL entries defined by the **-acl** argument. Partial pathnames are interpreted relative to the current working directory. You can also use the following notation on its own or as part of a pathname:

.

(A single period). If used by itself, sets the ACL on the current working directory.

..

(Two periods). If used by itself, sets the ACL on the current working directory's parent directory.

\*

(The asterisk). Sets the ACL on each of the subdirectories in the current working directory. You must precede it with the **-dir** switch, since it potentially designates multiple directories. The **fs** command interpreter generates the following error message for each file in the directory:

```
fs: 'filename': Not a directory
```

If you specify only one directory (or file) name, you can omit the **-dir** and **-acl** switches. For more on omitting switches, see "Appendix B, AFS Command Syntax and Online Help" on page 65.

### -acl

Specifies one or more ACL entries, each of which pairs a user or group name and a set of permissions. Separate the pairs, and the two parts of each pair, with one or more spaces.

To define the permissions, provide either:

- One or more of the letters that represent the standard or auxiliary permissions (**rlidwka** and **ABCDEFGH**), in any order
- One of the four shorthand notations:
  - **all** (equals **rlidwka**)

- **none** (removes the entry)
- **read** (equals **rl**)
- **write** (equals **rlidwk**)

On a single command line, you can combine user and group entries. Also, you can both combine individual letters and use the shorthand notations, but not within a single pair.

### Example: Adding a Single ACL Entry

Either of the following example commands grants user **pat** the **r** and **l** permissions on the ACL of the **notes** subdirectory of the current working directory. They illustrate how it is possible to omit the **-dir** and **-acl** switches when you name only one directory.

```
% fs sa notes pat rl
% fs sa pat read
```

### Example: Setting Several ACL Entries on One Directory

The following example edits the ACL for the current working directory. It removes the entry for the **system:anyuser** group, and adds two entries: one grants all permissions except **a** to the members of the **terry:colleagues** group and the other grants the **r** and **l** permissions to the **system:authuser** group.

```
% fs sa -dir . -acl system:anyuser none terry:colleagues write \
system:authuser rl
```

### To Add, Remove, or Edit Negative ACL Permissions

Issue the **fs setacl** command with the **-negative** flag to edit entries in the negative permissions section of the ACL. To remove an entry, specify the **none** shorthand as the permissions. If an ACL entry already exists for a user or group, the permissions you specify completely replace those in the existing entry.

```
% fs setacl -dir <directory>+ -acl <access list entries>+ -negative
```

where

#### **sa**

Is an acceptable alias for **setacl** (and **seta** is the shortest acceptable abbreviation).

#### **-dir**

Names one or more directories to which to apply the negative ACL entries defined by the **-acl** argument. For a detailed description of acceptable values, see "To Add, Remove, or Edit Normal ACL Permissions" on page 34.

**-acl**

Specifies one or more ACL entries, each of which pairs a user or group name and a set of permissions. Separate the pairs, and the two parts of each pair, with one or more spaces. For a detailed description of acceptable values, see "To Add, Remove, or Edit Normal ACL Permissions" on page 34. Keep in mind that the usual meaning of each permission is reversed.

**-negative**

Places the entries defined by the **-acl** argument on the negative permissions section of the ACL for each directory named by the **-dir** argument.

**Example: Setting an Entry in the Negative Permissions Section**

User **terry** has granted all access permissions except **a** to the group **terry:team** on her **plans** subdirectory.

```
% cd /afs/abc.com/usr/terry
% fs listacl plans
Access control list for plans is
Normal rights:
    system:anyuser rl
    terry:team rlidwk
    terry rlidwka
```

However, **terry** notices that one of the members of the group, user **pat**, has been making inappropriate changes to files. To prevent this without removing **pat** from the group or changing the permissions for the **terry:team** group, **terry** creates an entry on the negative permissions section of the ACL that denies the **w** and **d** permissions to **pat**:

```
% fs setacl plans pat wd -negative
% fs listacl plans
Access control list for plans is
Normal rights:
    system:anyuser rl
    terry:team rlidwk
    terry: rlidwka
Negative rights:
    pat wd
```

**Example: Restoring Access by Removing an Entry from the Negative Permissions Section**

In the previous example, user **terry** put **pat** on the negative permissions section of ACL for the **plans** subdirectory. But the result has been inconvenient and **pat** has promised not to change files any more. To enable **pat** to exercise all permissions granted to the members of the **terry:team** group, **terry** removes the entry for **pat** from the negative permissions section of the ACL.

```
% fs setacl plans pat none -negative
```

```
% fs listacl plans
Access control list for plans is
Normal rights:
    system:anyuser rl
    terry:team rlidwk
    terry rlidwka
```

## Completely Replacing an ACL

It is sometimes simplest to clear an ACL completely before defining new permissions on it, for instance if the mix of normal and negative permissions makes it difficult to understand how their interaction affects access to the directory. To clear an ACL completely while you define new entries, include the **-clear** flag on the **fs setacl** command. When you include this flag, you can create entries on either the normal permissions or the negative permissions section of the ACL, but not on both at once.

Remember to create an entry for yourself. As the owner of the directory, you always have the **a** (**administer**) permission required to replace a deleted entry, but the effects of a missing ACL entry can be confusing enough to make it difficult to realize that the problem is a missing entry. In particular, the lack of the **l** (**lookup**) permission prevents you from using any shorthand notation in pathnames (such as a period for the current working directory or two periods for the parent directory).

## To Replace an ACL Completely

Issue the **fs setacl** command with the **-clear** flag to clear the ACL completely before setting either normal or negative permissions. Because you need to grant the owner of the directory all permissions, it is better in most cases to set normal permissions at this point.

```
% fs setacl -dir <directory>+ -acl <access list entries>+ -clear [-negative]
```

where

**sa**

Is an acceptable alias for **setacl** (and **seta** is the shortest acceptable abbreviation).

**-dir**

Names one or more directories to which to apply the ACL entries defined by the **-acl** argument. For a detailed description of acceptable values, see "To Add, Remove, or Edit Normal ACL Permissions" on page 34.

**-acl**

Specifies one or more ACL entries, each of which pairs a user or group name and a set of permissions. Separate the pairs, and the two parts of each pair, with one or more spaces. Remember to grant all permissions to the owner of the directory. For a detailed description of acceptable values, see "To Add, Remove, or Edit Normal ACL Permissions" on page 34.

**-clear**

Removes all entries from each ACL before creating the entries indicated by the **-acl** argument.

**-negative**

Places the entries defined by the **-acl** argument on the negative permissions section of each ACL.

**Example: Replacing an ACL**

The following example clears the ACL on the current working directory and creates entries that grant all permissions to user **terry** and all permissions except **a** to user **pat**.

```
% fs setacl . terry all pat write -clear
% fs listacl .
Access control list for . is
Normal rights:
  terry rlidwka
  pat rlidwk
```

**Copying ACLs Between Directories**

The **fs copyacl** command copies a source directory's ACL to one or more destination directories. It does not affect the source ACL at all, but changes each destination ACL as follows:

- If an entry on the source ACL does not exist on the destination ACL, the command copies it to the destination ACL.
- If an entry on the destination ACL does not also exist on the source ACL, the command does not remove it unless you include the **-clear** flag, which overwrites the destination ACL completely.
- If an entry is on both ACLs, the command changes the destination ACL entry to match the source ACL entry.

To copy an ACL, you must have the **I** permission on the source ACL and the **a** permission on each destination ACL. If you identify the source directory by naming a file in it, you must also have the **r** permission on the source ACL. To display the permissions you have on the two directories, use the **fs listacl** command as described in "Displaying an ACL" on page 32.

**Note for AFS/DFS Migration Toolkit users:** If the machine on which you issue the **fs copyacl** command is configured for access to a DCE cell's DFS filespace via the AFS/DFS Migration Toolkit, you can use the command to copy ACLs between DFS files and directories also. The command includes **-id** and **-if** flags for altering a DFS directory's Initial Container and Initial Object ACLs as well as its regular ACL; for details, ask your system administrator. You cannot copy ACLs between AFS and DFS directories, because they use different ACL formats. The **fs** command interpreter ignores the **-id** and **-if** flags if you include them when copying AFS ACLs.

## To Copy an ACL Between Directories

Issue the **fs copyacl** command to copy a source ACL to the ACL on one or more destination directories.

```
% fs copyacl -fromdir <source directory> -todir <destination directory>+ \
[-clear]
```

where

### **co**

Is the shortest acceptable abbreviation for **copyacl**.

### **-fromdir**

Names the source directory from which to copy the ACL. Partial pathnames are interpreted relative to the current working directory. If this argument names a file, the ACL is copied from its directory.

### **-todir**

Names each destination directory to which to copy the source ACL. Partial pathnames are interpreted relative to the current working directory. Filenames are not acceptable.

### **-clear**

Completely overwrites each destination directory's ACL with the source ACL.

## Example: Copying an ACL from One Directory to Another

In this example, user **terry** copies the ACL from her home directory (the current working directory) to its **plans** subdirectory. She begins by displaying both ACLs.

```
% fs listacl . plans
Access list for . is
Normal rights:
    terry rlidwka
    pat rlidwk
    jones rl
Access list for plans is
Normal rights:
    terry rlidwka
    pat rl
    smith rl

% fs copyacl -from . -to plans

% fs listacl . plans
Access list for . is
Normal rights:
    terry rlidwka
    pat rlidwk
    jones rl
```



```

Access list for plans is
Normal rights:
    terry rlidwka
    pat rlidwk
    jones rl
    smith rl

```

## How AFS Uses the UNIX Mode Bits

Although AFS protects data primarily with ACLs rather than mode bits, it does not ignore the mode bits entirely. An explanation of how mode bits work in the UNIX file system is outside the scope of this document, and the following discussion assumes you understand them; if necessary, see your UNIX documentation. Also, the following discussion does not cover the `setuid`, `setgid` or sticky bits. If you need to understand how those bits work on AFS files, see the *IBM AFS Administration Guide* or ask your system administrator.

AFS uses the UNIX mode bits in the following way:

- It uses the initial bit to distinguish files and directories. This is the bit that appears first in the output from the `ls -l` command and shows the hyphen (-) for a file or the letter `d` for a directory.
- It does not use any of the mode bits on a directory. The AFS ACL alone controls directory access.
- For a file, the owner (first) set of bits interacts with the ACL entries that apply to the file in the following way. AFS does not use the group or world (second and third sets) of mode bits at all.
  - If the first `r` mode bit is not set, no one (including the owner) can read the file, no matter what permissions they have on the ACL. If the bit is set, users also need the `r` and `l` permissions on the ACL of the file's directory to read the file.
  - If the first `w` mode bit is not set, no one (including the owner) can modify the file. If the `w` bit is set, users also need the `w` and `l` permissions on the ACL of the file's directory to modify the file.
  - There is no ACL permission directly corresponding to the `x` mode bit, but to execute a file stored in AFS, the user must also have the `r` and `l` permissions on the ACL of the file's directory.

When you issue the UNIX `chmod` command on an AFS file or directory, AFS changes the bits appropriately. To change a file's mode bits, you must have the AFS `w` permission on the ACL of the file's directory. To change a directory's mode bits, you must have the `d`, `i`, and `l` permissions on its ACL.

### Example: Disabling Write Access for a File

Suppose `terry` is chairing a committee that is writing a proposal. As each section is approved, she turns off write access to that file to prevent further changes. For example, the following `chmod` command turns off the `w` mode bits on the file `proposal.chap2`. This makes it impossible for anyone to change the file, no matter what permissions are granted on the directory ACL.

```

% chmod -w proposal.chap2
% ls -l
-rw-r--r-- 1 terry      573 Nov 10 09:57 conclusion

```

*Chapter 4. Protecting Your Directories and Files*

```
-r--r--r-- 1 terry      573 Nov 15 10:34 intro
-r--r--r-- 1 terry      573 Dec  1 15:07 proposal.chap2
-rw-r--r-- 1 terry      573 Nov 10 09:57 proposal.chap3
-rw-r--r-- 1 terry      573 Nov 10 09:57 proposal.chap4
```

## Chapter 5. Using Groups

This chapter explains how to create groups and discusses different ways to use them.

### About Groups

An AFS *group* is a list of specific users that you can place on access control lists (ACLs). Groups make it much easier to maintain ACLs. Instead of creating an ACL entry for every user individually, you create one entry for a group to which the users belong. Similarly, you can grant a user access to many directories at once by adding the user to a group that appears on the relevant ACLs.

AFS client machines can also belong to a group. Anyone logged into the machine inherits the permissions granted to the group on an ACL, even if they are not authenticated with AFS. In general, groups of machines are useful only to system administrators, for specialized purposes like complying with licensing agreements your cell has with software vendors. Talk with your system administrator before putting a client machine in a group or using a machine group on an ACL.

To learn about AFS file protection and how to add groups to ACLs, see "Protecting Your Directories and Files" on page 27.

### Suggestions for Using Groups Effectively

There are three typical ways to use groups, each suited to a particular purpose: private use, shared use, and group use. The following are only suggestions. You are free to use groups in any way you choose.

- *Private use*: you create a group and place it on the ACL of directories you own, without necessarily informing the group's members that they belong to it. Members notice only that they can or cannot access the directory in a certain way. You retain sole administrative control over the group, since you are the owner.

The existence of the group and the identity of its members is not necessarily secret. Other users can see the group's name on an ACL when they use the **fs listacl** command, and can use the **pts membership** command to display + the groups to which they themselves belong. You can, however, limit who can display the members of the group, as described in "Protecting Group-Related Information" on page 54.

- *Shared use*: you inform the group's members that they belong to the group, but you are the group's sole owner and administrator. For example, the manager of a work group can create a group of all the members in the work group, and encourage them to use it on the ACLs of directories that house information they want to share with other members of the group.

**Note:** If you place a group owned by someone else on your ACLs, the group's owner can change the group's membership without informing you. Someone new can gain or lose access in a way you did not intend and without your knowledge.

- *Group use*: you create a group and then use the **pts chown** command to assign ownership to a group--either another group or the group itself (the latter type is a *self-owned* group). You inform the

members of the owning group that they all can administer the owned group. For instructions for the **pts chown** command, see "To Change a Group's Owner" on page 52.

The main advantage of designating a group as an owner is that several people share responsibility for administering the group. A single person does not have to perform all administrative tasks, and if the group's original owner leaves the cell, there are still other people who can administer it.

However, everyone in the owner group can make changes that affect others negatively: adding or removing people from the group inappropriately or changing the group's ownership to themselves exclusively. These problems can be particularly sensitive in a self-owned group. Using an owner group works best if all the members know and trust each other; it is probably wise to keep the number of people in an owner group small.

## Group Names

The groups you create must have names with two parts, in the following format:

*owner\_name:group\_name*

The *owner\_name* prefix indicates which user or group owns the group (naming rules appear in "To Create a Group" on page 48). The *group\_name* part indicates the group's purpose or its members' common interest. Group names must always be typed in full, so a short *group\_name* is most practical. However, names like **terry:1** and **terry:2** that do not indicate the group's purpose are less useful than names like **terry:project**.

Groups that do not have the *owner\_name* prefix possibly appear on some ACLs; they are created by system administrators only. All of the groups you create must have an *owner\_name* prefix.

## Group-creation Quota

By default, you can create 20 groups, but your system administrators can change your *group-creation quota* if appropriate. When you create a group, your group quota decrements by one. When a group that you created is deleted, your quota increments by one, even if you are no longer the owner. You cannot increase your quota by transferring ownership of a group to someone else, because you are always recorded as the creator.

If you exhaust your group-creation quota and need to create more groups, ask your system administrator. For instructions for displaying your group-creation quota, see "To Display A Group Entry" on page 46.

## Displaying Group Information

You can use the following commands to display information about groups and the users who belong to them:

- To display the members of a group, or the groups to which a user belongs, use the **pts membership** command.
- To display the groups that a user or group owns, use the **pts listowned** command.

- To display general information about a user or group, including its name, AFS ID, creator, and owner, use the **pts examine** command.

**Note:** The **system:anyuser** and **system:authuser** system groups do not appear in a user's list of group memberships, and the **pts membership** command does not display their members. For more information on the system groups, see "Using the System Groups on ACLs" on page 31.

## To Display Group Membership

Issue the **pts membership** command to display the members of a group, or the groups to which a user belongs.

```
% pts membership <user or group name or id>*
```

where *user or group name or id* specifies the name or AFS UID of each user for which to display group membership, or the name or AFS GID of each group for which to display the members. If identifying a group by its AFS GID, precede the GID with a hyphen (-) to indicate that it is a negative number.

## Example: Displaying the Members of a Group

The following example displays the members of the group **terry:team**.

```
% pts membership terry:team
Members of terry:team (id: -286) are:
  terry
  smith
  pat
  johnson
```

## Example: Displaying the Groups to Which a User Belongs

The following example displays the groups to which users **terry** and **pat** belong.

```
% pts membership terry pat
Groups terry (id: 1022) is a member of:
  smith:friends
  pat:accounting
  terry:team
Groups pat (id: 1845) is a member of:
  pat:accounting
  sam:managers
  terry:team
```

## To Display the Groups a User or Group Owns

Issue the **pts listowned** command to display the groups that a user or group owns.

```
% pts listowned <user or group name or id>*
```

where *user or group name or id* specifies the name or AFS UID of each user, or the name or AFS GID of each group, for which to display group ownership. If identifying a group by its AFS GID, precede the GID with a hyphen (-) to indicate that it is a negative number.

## Example: Displaying the Groups a Group Owns

The following example displays the groups that the group **terry:team** owns.

```
% pts listowned -286
Groups owned by terry:team (id: -286) are:
terry:project
terry:planners
```

## Example: Displaying the Groups a User Owns

The following example displays the groups that user **pat** owns.

```
% pts listowned pat
Groups owned by pat (id: 1845) are:
pat:accounting
pat:plans
```

## To Display A Group Entry

Issue the **pts examine** command to display general information about a user or group, including its name, AFS ID, creator, and owner.

```
% pts examine <user or group name or id>*
```

where *user or group name or id* specifies the name or AFS UID of each user, or the name or AFS GID of each group, for which to display group-related information. If identifying a group by its AFS GID, precede the GID with a hyphen (-) to indicate that it is a negative number.

The output includes information in the following fields:

### Name

For users, this is the character string typed when logging in. For machines, the name is the IP address; a zero in address field acts as a wildcard, matching any value. For most groups, this is a name of the form *owner\_name:group\_name*. Some groups created by your system administrator do not have the *owner\_name* prefix. See "Group Names" on page 44.

**id**

This is a unique identification number that the AFS server processes use internally. It is similar in function to a UNIX UID, but operates in AFS rather than the UNIX file system. Users and machines have positive integer AFS user IDs (UIDs), and groups have negative integer AFS group IDs (GIDs).

**owner**

This is the user or group that owns the entry and so can administer it.

**creator**

The name of the user who issued the **pts createuser** and **pts creatgroup** command to create the entry. This field is useful mainly as an audit trail and cannot be changed.

**membership**

For users and machines, this indicates how many groups the user or machine belongs to. For groups, it indicates how many members belong to the group. This number cannot be set explicitly.

**flags**

This field indicates who is allowed to list certain information about the entry or change it in certain ways. See "Protecting Group-Related Information" on page 54.

**group quota**

This field indicates how many more groups a user is allowed to create. It is set to 20 when a user entry is created. The creation quota for machines or groups is meaningless because it not possible to authenticate as a machine or group.

**Example: Listing Information about a Group**

The following example displays information about the group **pat:accounting**, which includes members of the department that **pat** manages. Notice that the group is self-owned, which means that all of its members can administer it.

```
% pts examine pat:accounting
Name: pat:accounting, id: -673, owner: pat:accounting, creator: pat,
membership: 15, flags: S-M--, group quota: 0
```

**Example: Listing Group Information about a User**

The following example displays group-related information about user **pat**. The two most interesting fields are **membership**, which shows that **pat** belongs to 12 groups, and **group quota**, which shows that **pat** can create another 17 groups.

```
% pts examine pat
Name: pat, id: 1045, owner: system:administrators, creator: admin,
membership: 12, flags: S-M--, group quota: 17
```

## Creating Groups and Adding Members

Use the **pts creategroup** command to create a group and the **pts adduser** command to add members to it. Users and machines can belong to groups, but other groups cannot.

When you create a group, you normally become its owner automatically. This means you alone can administer it: add and remove members, change the group's name, transfer ownership of the group, or delete the group entirely. If you wish, you can designate another owner when you create the group, by including the **-owner** argument to the **pts creategroup** command. If you assign ownership to another group, the owning group must already exist and have at least one member. You can also change a group's ownership after creating it by using the **pts chown** command as described in "Changing a Group's Owner or Name" on page 51.

### To Create a Group

Issue the **pts creategroup** command to create a group. Your group-creation quota decrements by one for each group.

```
% pts creategroup -name <group name>+ [-owner <owner of the group>]
```

where

#### **cg**

Is an alias for **creategroup** (and **creatcg** is the shortest acceptable abbreviation).

#### **-name**

Names each group to create. The name must have the following format:

```
owner_name:group_name
```

The *owner\_name* prefix must accurately indicate the group's owner. By default, you are recorded as the owner, and the *owner\_name* must be your AFS username. You can include the **-owner** argument to designate another AFS user or group as the owner, as long as you provide the required value in the *owner\_name* field:

- If the owner is a user, it must be the AFS username.
- If the owner is another regular group, it must match the owning group's *owner\_name* field. For example, if the owner is the group **terry:associates**, the owner field must be **terry**.
- If the owner is a group without an *owner\_name* prefix, it must be the owning group's name.

The name can include up to 63 characters including the colon. Use numbers and lowercase letters, but no spaces or punctuation characters other than the colon.

#### **-owner**

Is optional and assigns ownership to a user other than yourself, or to a group. If you specify a group, it must already exist and have at least one member. (This means that to make a group self-owned, you must issue the **pts chown** command after using this command to create the group, and the **pts adduser** command to add a member. See "Changing a Group's Owner or Name" on page 51.)



Do not name a machine as the owner. Because no one can authenticate as a machine, there is no way to administer a group owned by a machine.

### Example: Creating a Group

In the following example user **terry** creates a group to include all the other users in his work team, and then examines the new group entry.

```
% pts creategroup terry:team
group terry:team has id -286
% pts examine terry:team
Name: terry:team, id: -286, owner: terry, creator: terry,
membership: 0, flags: S----, group quota: 0.
```

### To Add Members to a Group

Issue the **pts adduser** command to add one or more users to one or more groups. You can always add members to a group you own (either directly or because you belong to the owning group). If you belong to a group, you can add members if its fourth privacy flag is the lowercase letter **a**; see "Protecting Group-Related Information" on page 54.

```
% pts adduser -user <user name>+ -group <group name>+
```

You must add yourself to groups that you own, if that is appropriate. You do not belong automatically just because you own the group.

**Note:** If you already have a token when you are added to a group, you must issue the **klog** command to reauthenticate before you can exercise the permissions granted to the group on ACLs.

where

#### **-user**

Specifies the username of each user to add to the groups named by the **-group** argument. Groups cannot belong to other groups.

#### **-group**

Names each group to which to add users.

### Example: Adding Members to a Group

In this example, user **terry** adds himself, **pat**, **indira**, and **smith** to the group he just created, **terry:team**, and then verifies the new list of members.

```
% pts adduser -user terry pat indira smith -group terry:team
```

```
% pts members terry:team
Members of terry:team (id: -286) are:
  terry
  pat
  indira
  smith
```

## Removing Users from a Group and Deleting a Group

You can use the following commands to remove groups and their members:

- To remove a user from a group, use the **pts removeuser** command
- To delete a group entirely, use the **pts delete** command
- To remove deleted groups from ACLs, use the **fs cleanacl** command

When a group that you created is deleted, your group-creation quota increments by one, even if you no longer own the group.

When a group or user is deleted, its AFS ID appears on ACLs in place of its AFS name. You can use the **fs cleanacl** command to remove these obsolete entries from ACLs on which you have the **a (administer)** permission.

### To Remove Members from a Group

Issue the **pts removeuser** command to remove one or more members from one or more groups. You can always remove members from a group that you own (either directly or because you belong to the owning group). If you belong to a group, you can remove members if its fifth privacy flag is the lowercase letter **r**; see "Protecting Group-Related Information" on page 54. (To display a group's owner, use the **pts examine** command as described in "To Display A Group Entry" on page 46.)

```
% pts removeuser -user <user name>+ -group <group name>+
```

where

#### **-user**

Specifies the username of each user to remove from the groups named by the **-group** argument.

#### **-group**

Names each group from which to remove users.

### Example: Removing Group Members

The following example removes user **pat** from both the **terry:team** and **terry:friends** groups.

```
% pts removeuser pat -group terry:team terry:friends
```

## To Delete a Group

Issue the **pts delete** command to delete a group. You can always delete a group that you own (either directly or because you belong to the owning group). To display a group's owner, use the **pts examine** command as described in "To Display A Group Entry" on page 46.

```
% pts delete <user or group name or id>*
```

where *user or group name or id* specifies the name or AFS UID of each user, or the name or AFS GID of each group, to delete. If identifying a group by its AFS GID, precede the GID with a hyphen (-) to indicate that it is a negative number.

## Example: Deleting a Group

In the following example, the group **terry:team** is deleted.

```
% pts delete terry:team
```

## To Remove Obsolete ACL Entries

Issue the **fs cleanacl** command to remove obsolete entries from ACLs after the corresponding user or group has been deleted.

```
% fs cleanacl [<dir/file path>*]
```

where *dir/file path* name each directory for which to clean the ACL. If you omit this argument, the current working directory's ACL is cleaned.

## Example: Removing an Obsolete ACL Entry

After the group **terry:team** is deleted, its AFS GID (-286) appears on ACLs instead of its name. In this example, user **terry** cleans it from the ACL on the **plans** directory in his home directory.

```
% fs listacl plans
Access list for plans is
Normal rights:
    terry rlidwka
    -268 rlidwk
    sam rliw
% fs cleanacl plans
% fs listacl plans
Access list for plans is
Normal rights:
    terry rlidwka
    sam rliw
```

## Changing a Group's Owner or Name

To change a group's owner, use the **pts chown** command. To change its name, use the **pts rename** command.

You can change the owner or name of a group that you own (either directly or because you belong to the owning group). You can assign group ownership to another user, another group, or the group itself. If you are not already a member of the group and need to be, use the **pts adduser** command before transferring ownership, following the instructions in "To Add Members to a Group" on page 49.

The **pts chown** command automatically changes a group's *owner\_name* prefix to indicate the new owner. If the new owner is a group, only its *owner\_name* prefix is used, not its entire name. However, the change in *owner\_name* prefix command does not propagate to any groups owned by the group whose owner is changing. If you want their *owner\_name* prefixes to indicate the correct owner, you must use the **pts rename** command.

Otherwise, you normally use the **pts rename** command to change only the *group\_name* part of a group name (the part that follows the colon). You can change the *owner\_name* prefix only to reflect the actual owner.

### To Change a Group's Owner

Issue the **pts chown** command to change a group's name.

```
% pts chown <group name> <new owner>
```

where

#### **group name**

Specifies the current name of the group to which to assign a new owner.

#### **new owner**

Names the user or group that is to own the group.

### Example: Changing a Group's Owner to Another User

In the following example, user **pat** transfers ownership of the group **pat:staff** to user **terry**. Its name changes automatically to **terry:staff**, as confirmed by the **pts examine** command.

```
% pts chown pat:staff terry
% pts examine terry:staff
Name: terry:staff, id: -534, owner: terry, creator: pat,
membership: 15, flags: S0m--, group quota: 0.
```

### Example: Changing a Group's Owner to Itself

In the following example, user **terry** makes the **terry:team** group a self-owned group. Its name does not change because its *owner\_name* prefix is already **terry**.

```
% pts chown terry:team terry:team
% pts examine terry:team
Name: terry:team, id: -286, owner: terry:team, creator: terry,
membership: 6, flags: SOm--, group quota: 0.
```

## Example: Changing a Group's Owner to a Group

In this example, user **sam** transfers ownership of the group **sam:project** to the group **smith:cpa**. Its name changes automatically to **smith:project**, because **smith** is the *owner\_name* prefix of the group that now owns it. The **pts examine** command displays the group's status before and after the change.

```
% pts examine sam:project
Name: sam:project, id: -522, owner: sam, creator: sam,
membership: 33, flags: SOm--, group quota: 0.
% pts chown sam:project smith:cpa
% pts examine smith:project
Name: smith:project, id: -522, owner: smith:cpa, creator: sam,
membership: 33, flags: SOm--, group quota: 0.
```

## To Change a Group's Name

Issue the **pts rename** command to change a group's name.

```
% pts rename <old name> <new name>
```

where

### **old name**

Specifies the group's current name.

### **new name**

Specifies the complete new name to assign to the group. The *owner\_name* prefix must correctly indicate the group's owner.

## Example: Changing a Group's *group\_name* Suffix

The following example changes the name of the **smith:project** group to **smith:fiscal-closing**. The group's *owner\_name* prefix remains **smith** because its owner is not changing.

```
% pts examine smith:project
Name: smith:project, id: -522, owner: smith:cpa, creator: sam,
membership: 33, flags: SOm--, group quota: 0.
% pts rename smith:project smith:fiscal-closing
% pts examine smith:fiscal-closing
Name: smith:fiscal-closing, id: -522, owner: smith:cpa, creator: sam,
membership: 33, flags: SOm--, group quota: 0.
```

### Example: Changing a Group's *owner\_name* Prefix

In a previous example, user **pat** transferred ownership of the group **pat:staff** to user **terry**. Its name changed automatically to **terry:staff**. However, a group that **terry:staff** owns is still called **pat:plans**, because the change to a group's *owner\_name* that results from the **pts chown** command does not propagate to any groups it owns. In this example, a member of **terry:staff** uses the **pts rename** command to change the name to **terry:plans** to reflect its actual ownership.

```
% pts examine pat:plans
Name: pat:plans, id: -535, owner: terry:staff, creator: pat,
  membership: 8, flags: SOM--, group quota: 0.
% pts rename pat:plans terry:plans
% pts examine terry:plans
Name: terry:plans, id: -535, owner: terry:staff, creator: pat,
  membership: 8, flags: SOM--, group quota: 0.
```

## Protecting Group-Related Information

A group's *privacy flags* control who can administer it in various ways. The privacy flags appear in the *flags* field of the output from the **pts examine** command; see "To Display A Group Entry" on page 46. To set the privacy flags for a group you own, use the **pts setfields** command as instructed in "To Set a Group's Privacy Flags" on page 55.

### Interpreting the Privacy Flags

The five privacy flags always appear, and always must be set, in the following order:

**s**

Controls who can issue the **pts examine** command to display the entry.

**o**

Controls who can issue the **pts listowned** command to list the groups that a user or group owns.

**m**

Controls who can issue the **pts membership** command to list the groups a user or machine belongs to, or which users or machines belong to a group.

**a**

Controls who can issue the **pts adduser** command to add a user or machine to a group.

**r**

Controls who can issue the **pts removeuser** command to remove a user or machine from a group.

Each flag can take three possible types of values to enable a different set of users to issue the corresponding command:

- A hyphen (-) means that the group's owner can issue the command, along with the administrators who belong to the **system:administrators** group.
- The lowercase version of the letter means that members of the group can issue the command, along with the users indicated by the hyphen.
- The uppercase version of the letter means that anyone can issue the command.

For example, the flags `S0mar` on a group entry indicate that anyone can examine the group's entry and list the groups that it owns, and that only the group's members can list, add, or remove its members.

The default privacy flags for groups are `S-M--`, meaning that anyone can display the entry and list the members of the group, but only the group's owner and members of the **system:administrators** group can perform other functions.

## To Set a Group's Privacy Flags

Issue the **pts setfields** command to set the privacy flags on one or more groups.

```
% pts setfields -nameorid <user or group name or id>*
               -access <set privacy flags>
```

where

### **-nameorid**

Specifies the name or AFS GID of each group for which to set the privacy flags. If identifying a group by its AFS GID, precede the GID with a hyphen (-) to indicate that it is a negative number.

### **-access**

Specifies the privacy flags to set for each group. Observe the following rules:

- Provide a value for all five flags in the order **somar**.
- Set the first flag to lowercase **s** or uppercase **S** only.
- Set the second flag to the hyphen (-) or uppercase **O** only. For groups, AFS interprets the hyphen as equivalent to lowercase **o** (that is, members of a group can always list the groups that it owns).
- Set the third flag to the hyphen (-), lowercase **m**, or uppercase **M**.
- Set the fourth flag to the hyphen (-), lowercase **a**, or uppercase **A**. The uppercase **A** is not a secure choice, because it permits anyone to add members to the group.
- Set the fifth flag to the hyphen (-) or lowercase **r** only.

## Example: Setting a Group's Privacy Flags

The following example sets the privacy flags on the **terry:team** group to set the indicated pattern of administrative privilege.

```
% pts setfields terry:team -access S0m--
```

## *Chapter 5. Using Groups*

- Everyone can issue the **pts examine** command to display general information about it (uppercase **S**).
- Everyone can issue the **pts listowned** command to display the groups it owns (uppercase **O**).
- The members of the group can issue the **pts membership** command to display the group's members (lowercase **m**).
- Only the group's owner, user **terry**, can issue the **pts adduser** command to add members (the hyphen).
- Only the group's owner, user **terry**, can issue the **pts removeuser** command to remove members (the hyphen).



## Chapter 6. Troubleshooting

This chapter explains how to investigate and solve some problems you can sometimes encounter when working with AFS files. To use the instructions, find the heading that describes your problem or matches the error message you received.

### Problem: Cannot Access, Copy, or Save File

1. Issue the **tokens** command to verify that you have valid tokens. For complete instructions, see "To Display Your Tokens" on page 12.

```
% tokens
```

- If your tokens are valid, proceed to Step "2" on page 57.
- If you do not have tokens for the relevant cell, or they are expired, issue the **klog** command to authenticate. For complete instructions, see "To Authenticate with AFS" on page 11. Then try accessing or saving the file again. If you are not successful, proceed to Step "2" on page 57.

```
% klog
```

2. Issue the **fs checkservers** command to check the status of file server machines. For complete instructions, see "Checking the Status of Server Machines" on page 22.

```
% fs checkservers &
```

- If the following message appears, proceed to Step "3" on page 57.

```
All servers are running.
```

- Output like the following indicates that your Cache Manager cannot reach the indicated file server machines.

```
These servers unavailable due to network or server problem:  
list of machines.
```

Issue the **fs whereis** command to check if the file you are attempting to access or save is stored on one of the inaccessible file server machines. For complete instructions, see "Locating Files and Directories" on page 21.

```
% fs whereis <dir/file path>
```

If your file is stored on an inaccessible machine, then you cannot access the file or save it back to the File Server until the machine is again accessible. If your file is on a machine that is not listed as inaccessible, proceed to Step "3" on page 57.

3. Issue the **fs listacl** command to verify that you have the permissions you need for accessing, copying, or saving the file. For complete instructions, see "To display an ACL" on page 32.

```
% fs listacl <dir/file path>
```

You need the indicated permissions:

- To access, copy, or save a file, you must have the **l (lookup)** permission on the directory and on all directories above it in the pathname.
- To save changes to an existing file, you must in addition have the **w (write)** permission. To create a new file, you must in addition have the **i (insert)** and **w** permissions.
- To copy a file between two directories, you must in addition have the **r (read)** permission on the source directory and the **i** permission on the destination directory.

If you do not have the necessary permissions but own the directory, you always have the **a (administer)** permission even if you do not appear on the ACL. Issue the **fs setacl** command to grant yourself the necessary permissions. For complete instructions, see "Changing an ACL" on page 34.

```
% fs setacl -dir <directory>* -acl <access list entries>*
```

If you do not have the necessary permissions and do not own the directory, ask the owner or a system administrator to grant them to you. If they add you to a group that has the required permissions, you must issue the **klog** command to reauthenticate before you can exercise them.

If you still cannot access the file even though you have the necessary permissions, contact your system administrator for help in investigating further possible causes of your problem. If you still cannot copy or save the file even though you have the necessary permissions, proceed to Step "4" on page 58.

4. If copying a file, issue the **fs listquota** command to check whether the volume into which you are copying it, or the partition that houses that volume, is almost full. For saving, check the volume and partition that contain the directory into which you are saving the file. For complete instructions, see "Displaying Volume Quota" on page 19.

```
% fs listquota <dir/file path>
```

The command produces output as in the following example:

```
% fs listquota /afs/abc.com/usr/terry
Volume Name      Quota    Used    % Used  Partition
user.terry       10000    3400    34%     86%
```

- If the value in the `Partition` field is not close to 100%, the partition is not almost full. Check the value in the `% Used` field. If it is close to 100%, then the volume is almost full. If possible, delete files from the volume that are no longer needed, or ask your system administrator to increase the volume's quota.

If the value in the `% Used` field is not close to 100% (is, say, 90% or less), then it is unlikely that you are exceeding the volume's quota, unless the file is very large or the volume's quota is small. Contact your system administrator for help in investigating further possible causes of your problem.

- If the value in the `Partition` field is very close to 100%, the partition is possibly nearly full. However, server machine partitions are usually very large and can still have enough space for an average file when nearly full. You can either ask your system administrator about the partition's

status, or issue the **fs examine** command. The final line in its output reports how many kilobyte blocks are still available on the partition. For complete instructions, see "Displaying Volume Quota" on page 19.

```
% fs examine <dir/file path>
```

If there is enough free space on the partition but you still cannot save the file, ask your system administrator for help in investigating further possible causes of your problem.

## Problem: Accidentally Removed Your Entry from an ACL

1. If you own the directory from which you have accidentally removed your ACL entry, then you actually still have the **a (administer)** permission even if it does not appear on the ACL. You normally own your home directory and all of its subdirectories, for instance. Issue the **fs setacl** command to grant yourself all other permissions. For complete instructions, see "To Add, Remove, or Edit Normal ACL Permissions" on page 34.

```
% fs setacl -dir <directory> -acl <your_username> all
```

For *directory*, provide the complete pathname to the directory (for example, */afs/abc.com/usr/your\_username*). This is necessary because AFS cannot interpret pathname abbreviations if you do not have the **I (lookup)** permission.

2. If you do not own the directory, issue the **fs listacl** to check if any remaining entries grant you the permissions you need (perhaps you belong to one or more groups that appear on the ACL). For complete instructions, see "To display an ACL" on page 32.

```
% fs listacl <dir/file path>
```

- The following message displays the directory's ACL. If you need permissions that no entry currently grants you, ask the directory's owner or your system administrator for help.

```
Access list for <dir/file path> is
Normal rights
list of entries
```

- If the command returns the following error message instead of an ACL, then you do not have the **I** permission.

```
fs: You don't have the required access rights on 'dir/file path'
```

Ask the directory's owner or your system administrator to grant you the permissions you need. If they add you to a group that has the required permissions, you must issue the **klog** command to reauthenticate before you can exercise them.

## Error Message: "afs: Lost contact with fileserver"

Issue the **fs checkservers** command to check the status of file server machines. For complete instructions, see "Checking the Status of Server Machines" on page 22.

```
% fs checkservers &
```

- If the following message appears, ask your system administrator for assistance in diagnosing the cause of the `Lost contact` error message.

```
All servers are running.
```

- Output like the following indicates that your Cache Manager cannot reach the indicated file server machines. You must wait until they are again accessible before continuing to work with the files that are stored on them.

```
These servers unavailable due to network or server problem:  
list_of_machines.
```

## Error Message: "*command*: Connection timed out"

Issue the **fs checkservers** command as described in "Error Message: afs: Lost contact with fileserver" on page 59.

## Error Message: "fs: You don't have the required access rights on '*file*'"

You do not have the ACL permissions you need to perform the operation you are attempting. If you own the directory and have accidentally removed yourself from the ACL, see "Problem: Accidentally Removed Your Entry from an ACL" on page 59. Otherwise, ask the directory's owner or your system administrator to grant you the appropriate permissions.

## Error Message: "afs: failed to store file"

Follow the instructions in "Problem: Cannot Access, Copy, or Save File" on page 57.

## Appendix A. Using the NFS/AFS Translator

Some cells use the Network File System (NFS) in addition to AFS. If you work on an NFS client machine, your system administrator can configure it to access the AFS filespace through a program called the *NFS/AFS Translator*<sup>™</sup>. If you have an AFS account, you can access AFS as an authenticated user while working on your NFS client machine. Otherwise, you access AFS as the **anonymous** user.

**Note:** Acceptable NFS/AFS Translator performance requires that NFS is functioning correctly.

### Requirements for Using the NFS/AFS Translator

For you to use the NFS/AFS Translator, your system administrator must configure the following types of machines as indicated:

- An *NFS/AFS translator machine* is an AFS client machine that also acts as an NFS server machine. Its Cache Manager acts as the surrogate Cache Manager for your NFS client machine. Ask your system administrator which translator machines you can use.
- Your NFS client machine must have an NFS mount to a translator machine. Most often, your system administrator mounts the translator machine's **/afs** directory and names the mount **/afs** as well. This enables you to access the entire AFS filespace using standard AFS pathnames. It is also possible to create mounts directly to subdirectories of **/afs**, and to give NFS mounts different names on the NFS client machine.

Your access to AFS is much more extensive if you have an AFS user account. If you do not, the AFS servers recognize you as the **anonymous** user and only grant you the access available to members of the **system:anyuser** group.

If your NFS client machine uses an operating system that AFS supports, your system administrator can configure it to enable you to issue many AFS commands on the machine. Ask him or her about the configuration and which commands you can issue.

### Accessing AFS via the Translator

If you do not have an AFS account or choose not to access AFS as an authenticated user, then all you do to access AFS is provide the pathname of the relevant file. Its ACL must grant the necessary permissions to the **system:anyuser** group.

If you have an AFS account and want to access AFS as an authenticated user, the best method depends on whether your NFS machine is a supported type. If it is, use the instructions in "To Authenticate on a Supported Operating System" on page 61. If it is not a supported type, use the instructions in "To Authenticate on an Unsupported Operating System" on page 62.

#### To Authenticate on a Supported Operating System

1. Log into the NFS client machine using your NFS username.

2. Issue the **klog** command. For complete instructions, see "To Authenticate with AFS" on page 11.

```
% klog -setpag
```

## To Authenticate on an Unsupported Operating System

1. Log onto the NFS client machine using your NFS username.
2. Establish a connection to the NFS/AFS translator machine you are using (for example, using the **telnet** utility) and log onto it using your AFS username (which is normally the same as your NFS username).
3. If the NFS/AFS translator machine uses an AFS-modified login utility, then you obtained AFS tokens in Step "2" on page 62. To check, issue the **tokens** command, which is described fully in "To Display Your Tokens" on page 12.

```
% tokens
```

If you do not have tokens, issue the **klog** command, which is described fully in "To Authenticate with AFS" on page 11.

```
% klog -setpag
```

4. Issue the **knfs** command to associate your AFS tokens with your UNIX UID on the NFS client machine where you are working. This enables the Cache Manager on the translator machine to use the tokens properly when you access AFS from the NFS client machine.

If your NFS client machine is a system type for which AFS defines a system name, it can make sense to add the **-sysname** argument. This argument helps the Cache Manager access binaries specific to your NFS client machine, if your system administrator has used the `@sys` variable in pathnames. Ask your system administrator if this argument is useful for you.

```
% knfs <host name> [<user ID (decimal)>] \
    [-sysname <host's '@sys' value>]
```

where

### **host name**

Specifies the fully-qualified hostname of your NFS client machine (such as **nfs52.abc.com**).

### **user ID**

Specifies your UNIX UID or equivalent (not your username) on the NFS client machine. If your system administrator has followed the conventional practice, then your UNIX and AFS UIDs are the same. If you do not know your local UID on the NFS machine, ask your system administrator for assistance. Your system administrator can also explain the issues you need to be aware of if your two UIDs do not match, or if you omit this argument.

### **-sysname**

Specifies your NFS client machine's system type name.

5. **(Optional)** Log out from the translator machine, but do not unauthenticate.
6. Work on the NFS client machine, accessing AFS as necessary.
7. When you are finished accessing AFS, issue the **knfs** command on the translator machine again. Provide the same *host name* and *user ID* arguments as in Step "4" on page 62, and add the **-unlog** flag to destroy your tokens. If you logged out from the translator machine in Step "5" on page 63, then you must first reestablish a connection to the translator machine as in Step "2" on page 62.

```
% knfs <host name> [<user ID (decimal)>] -unlog
```

## Troubleshooting the NFS/AFS Translator

Acceptable performance by the NFS/AFS translator depends for the most part on NFS. Sometimes, problems that appear to be AFS file server outages, broken connections, or inaccessible files are actually caused by NFS outages.

This section describes some common problems and their possible causes. If other problems arise, contact your system administrator, who can ask the AFS Product Support group for assistance if necessary.

**Note:** To avoid degrading AFS performance, the Cache Manager on the translator machine does not immediately send changes made on NFS client machines to the File Server. Instead, it checks every 60 seconds for such changes and sends them then. It can take longer for changes made on an NFS client machine to be saved than for changes made on an AFS client machine. The save operation must complete before the changes are visible on NFS client machines that are using a different translator machine or on AFS client machines.

### Your NFS Client Machine is Frozen

If your system administrator has used the recommended options when creating an NFS mount to an NFS/AFS translator machine, then the mount is both *hard* and *interruptible*:

- A hard mount means that the NFS client retries its requests if it does not receive a response within the expected time frame. This is useful because requests have to pass through both the NFS and AFS client software, which can sometimes take longer than the NFS client expects. However, it means that if the NFS/AFS translator machine actually becomes inaccessible, your NFS client machine can become inoperative (*freeze* or *hang*).
- If the NFS mount is interruptible, then in the case of an NFS/AFS translator machine outage you can press **<Ctrl-c>** or another interrupt signal to halt the NFS client's repeated attempts to access AFS. You can then continue to work locally, or can NFS-mount another translator machine. If the NFS mount is not interruptible, you must actually remove the mount to the inaccessible translator machine.

## NFS/AFS Translator Reboots

If you have authenticated to AFS and your translator machine reboots, you must issue the **klog** command (and **knfs** command, if appropriate) to reauthenticate. If you used the **knfs** command's **-sysname** argument to define your NFS client machine's system name, use it again.

## System Error Messages

This section explains possible meanings for NFS error messages you receive while accessing AFS filesystem.

```
stale NFS client
```

```
Getpwd: can't read
```

Both messages possibly means that your translator machine was rebooted and cannot determine the pathname to the current working directory. To reestablish the path, change directory and specify the complete pathname starting with **/afs**.

```
NFS server translator_machine is not responding still trying.
```

The NFS client is not getting a response from the NFS/AFS translator machine. If the NFS mount to the translator machine is a hard mount, your NFS client continues retrying the request until it gets a response (see "Your NFS Client Machine is Frozen" on page 63). If the NFS mount to the translator machine is a soft mount, the NFS client stops retrying after a certain number of attempts (three by default).



## Appendix B. AFS Command Syntax and Online Help

The AFS commands available to you are used to authenticate, list AFS information, protect directories, create and manage groups, and create and manage ACLs. There are three general types of commands available to all AFS users: file server commands, protection server commands, and miscellaneous commands. This chapter discusses the syntax of these AFS commands, the rules that must be followed when issuing them, and ways of accessing help relevant to them.

### AFS Command Syntax

Most AFS commands use the following syntax:

```
command_suite operation_code -switch <value>[+] -flag
```

The *command suite* indicates the general type of command and the server process that performs the command. Regular AFS users have access to two main command suites and a miscellaneous set of commands:

- The **fs** command suite is used to issue file server commands that interact with the File Server process.
- The **pts** command suite is used to issue protection-related commands.
- The miscellaneous commands are not associated with any command suite.

The *operation code* indicates the action that the command performs. Miscellaneous commands have operation codes only.

A command can have multiple *options*, which can be *arguments* or *flags*:

- Arguments are used to supply additional information for use by the command. They consist of a paired *switch* and *instance*. A switch defines the type of argument and is always preceded by a hyphen; arguments can take multiple instances if a plus sign (+) appears after the instance. An instance represents some variable piece of information that is used by the command. Arguments can be optional or required.
- Flags are used to direct a command to perform in a specific way (for example, to generate a specific type of output). Flags are always preceded by a hyphen and are always optional.

### Command Syntax Example

In the following AFS command

```
% fs setacl -dir $HOME -acl pat all terry none -negative
```

- **fs** is the command suite.
- **setacl** is the *operation code*, which directs the File Server process to set an access control list.
- **-dir \$HOME** and **-acl pat all terry none** are *arguments*.

- **-dir** and **-acl** are switches; **-dir** indicates the name of the directory on which to set the ACL, and **-acl** defines the entries to set on it.
- **\$HOME** and **pat all terry none** are *instances* of the arguments. **\$HOME** defines a specific directory for the directory argument. The **-acl** argument has two instances specifying two ACL entries: **pat all** and **terry none**.
- **-negative** is a flag; it directs the command to put the access list entries on the negative rather than the normal permissions list.

## Rules for Using AFS Commands

This section describes the rules to follow when using AFS commands.

### Spaces and Lines

Separate each command element (command suite, operation code, switches, instances, and flags) with a space. Multiple instances of an argument are also separated by a space.

Type all AFS commands on one line, followed by a carriage return. Some commands in this document appear on more than one line, but that is for legibility only.

### Abbreviations and Aliases for Operation Codes

You can type operation codes in one of three ways:

- You can type the operation code in full.
- You can abbreviate the operation code to the shortest form that distinguishes it from the other operation codes in its command suite.
- You can use the alias for the operation code, if one exists.

For example, the **fs listacl** command can be issued as follows:

- **fs listacl** (full command)
- **fs lista** (abbreviation)
- **fs la** (alias)

The *IBM AFS Administration Reference* provides information on the full and abbreviated command syntax as well as any aliases for all of the commands discussed in this guide.

### Omitting Argument Switches

You can omit an argument's switch if the command takes only one argument, or if the following conditions are met.

- All of the command's required arguments appear in the order prescribed by the syntax statement.
- No switches are used on any arguments, even if they are in the correct order.
- There is only one value for each argument. The important exception to this condition is if the final required argument accepts multiple values; in this case, it is acceptable to provide multiple values without providing the switch.

For example, the following two commands are equivalent:

```
% fs setacl -dir /afs/abc.com/usr/terry/private -acl pat rl
% fs setacl /afs/abc.com/usr/terry/private pat rl
```

However, the following is not an acceptable short form because the arguments are not in the prescribed order:

```
% fs setacl -acl pat rl /afs/abc.com/usr/terry/private
```

## Shortening Switches and Flags

If you are required to use a switch, or if you decide to use a flag, you can often shorten the name of that switch or flag provided that the shortened form still distinguishes it from the command's other flags and switches.

For example, when you issue the **fs setacl** command, you can abbreviate all of the switches and flags of the command to their initial letter because they all begin with a different letter. However, when you issue the **knfs** command, the **-host** argument and **-help** flag both begin with the letter **h**, so the shortest unambiguous abbreviations are **-ho** and **-he** respectively.

## Shortening Directory References

Most AFS command arguments that require directory or pathnames instances accept one or more of the following short forms:

- A single period (.) indicates the current working directory.
- Two periods (..) indicate the parent directory of the current working directory.
- The \$HOME environment variable indicates the issuer's home directory.

For example, if the user **terry** wants to grant **r (read)** and **l (lookup)** permissions on his home directory to his manager **pat**, **terry** can issue the following command.

```
% fs setacl -dir $HOME -acl pat rl
```

If the current working directory is **terry**'s home directory, he can issue the following command.

```
% fs setacl -dir . -acl pat rl
```

Both of the previous examples are acceptable short forms for the following command:

```
% fs setacl -dir /afs/abc.com/usr/terry -acl pat rl
```

## Commonly Used fs and pts Commands

This section provides additional information on the commonly used AFS **fs** and **pts** commands. For more detailed information, see the *IBM AFS Administration Reference*.

### About the fs Commands

Some **fs** commands extend UNIX file system semantics by invoking file-related functions that UNIX does not provide (setting access control lists, for example). Other **fs** commands help you control the performance of the Cache Manager running on your local client machine.

All **fs** commands accept the optional **-help** flag. It has the same function as the **fs help** command: it prints a command's online help message on the screen. Do not provide other options at the same time as this flag. It overrides them, and the only effect of issuing the command is to display the help message.

The privilege required for issuing **fs** commands varies. The necessary privileges for the **fs** commands described in this guide include the following:

- Having certain permissions on a directory's access control list. For example, creating and removing mount points requires **a (administer)**, **i (insert)**, and **d (delete)** permissions for the directory in which the mount point resides.
- Belonging to the **system:administrators** group (see "Using the System Groups on ACLs" on page 31).
- No privilege. Many **fs** commands simply list information and so do not require any special privilege.

### About the pts Commands

The **pts** command suite is the interface through which you can create protection groups and add members to them. System administrators who belong to a special system group called **system:administrators** group can manipulate any group, and also create the user and machine entries that can belong to groups. Users who do not belong to the **system:administrators** group can always list the information associated with the group entries they own, as well as their own user entries. Depending on the setting of an entry's privacy flags, regular users can sometimes access and manipulate group entries in certain ways.

All **pts** commands accept optional arguments and flags. They are listed in the command descriptions in the *IBM AFS Administration Reference* and are described here in detail:

**[-cell <cell name>]**

This argument indicates that the command runs in the indicated cell. The issuer can abbreviate the *cell name* value to the shortest form that distinguishes it from the other cells listed in the **/usr/vice/etc/CellServDB** file on the client machine on which the command is issued. By default, commands are executed in the local cell as defined

- First, by the value of the environment variable **AFSCELL**. (This variable is normally not defined by default. If you are working in another, nonlocal cell for an extended period of time, you can set the variable to the name of that cell.)
- Second, in the **/usr/vice/etc/ThisCell** file on the client machine on which the command is issued.

**[-force]**

This flag directs the **pts** command interpreter to continue executing the command, if possible, even if it encounters problems during the command's execution. The command interpreter performs as much of the requested operation as possible, rather than halting if it encounters a problem. The command interpreter reports any errors it encounters during the command's execution. This flag is especially useful if you provide many instances for an argument; if one of the instances is invalid, the command reports the error and proceeds with the remaining arguments.

**[-help]**

This flag has the same function as the **pts help** command: it prints the command's online help message on the screen. Do not provide other options at the same time as this flag. It overrides them, and the only effect of issuing the command is to display the help message.

## Getting Help in AFS

AFS online help consists of basic syntax messages. The AFS distribution also includes help in HTML format which your system administrator can make available to you.

### Displaying Command Syntax and Aliases

To display a brief description of a command, its syntax statement, and alias if any, use the **help** operation code. For example, to display the online help entry for the **fs listacl** command, enter the following command:

```
% fs help listacl
fs listacl: list access control list
aliases: la
Usage: fs listacl [-path <dir/file path>+] [-id] [-if] [-help]
```

To display the syntax statement only, use the **-help** flag, which is available on most AFS commands. For example, to display the syntax statement for the **fs setacl** command, enter the following command:

```
% fs setacl -help
Usage: fs setacl -dir <directory>+ -acl <access list entries>+ [-clear] [-negative]
[-id] [-if] [-help]
```

### Displaying Operation Code Descriptions

To display a short description of all of a command suite's operation codes, issue the **help** operation code without any other arguments. For example, the **fs help** command displays a short description of every operation code in the **fs** command suite.

To display a list of the commands in a command suite that concern a certain type of object, provide a relevant keyword argument to the **apropos** operation code. For example, if you want to set an ACL but cannot remember which **fs** command to use, issue the following command:

*Appendix B. AFS Command Syntax and Online Help*

```
% fs apropos set
setacl: set access control list
setcachesize: set cache size
setcell: set cell status
setclientaddr: set client network interface addresses
setquota: set volume quota
setserverprefs: set file server ranks
setvol: set volume status
sysname: get/set sysname (i.e. @sys) value
```

The following message indicates that there are no commands whose names or descriptions include the keyword string you have provided:

```
Sorry, no commands found
```

**Note:** If the keyword you provide has spaces in it, enclose it in double quotes (" ").

# Glossary

## A

### **a (administer) Permission**

The ACL permission that allows the possessor to change the entries on the ACL .

### **a Privacy Flag**

The fourth privacy flag on a group, which enables the possessor to add members to it.

### **Access Control List (ACL)**

A list associated with an AFS directory that specifies what actions a user or group can perform on the directory and the files in it. There are seven access permissions: **a (administer)**, **d (delete)**, **i (insert)**, **k (lock)**, **l (lookup)**, **r (read)**, and **w (write)**.

### **ACL Entry**

An entry on an ACL that pairs a user or group with specific access permissions.

### **Alias**

An alternative name for an AFS command.

### **all ACL Shorthand**

A shorthand notation used with the **fs setacl** command to represent all seven permissions.

### **Anonymous**

The identity assigned to a user who does not have a valid token for the local cell.

### **Argument**

The portion of a command that names an entity to be affected by the command. Arguments consist of two parts: a *switch* and one or more *instances*. Some AFS commands take one or more arguments.

## **Authenticate**

To become recognized as a valid AFS user by providing the correct password. Authenticate by logging onto a machine that uses an AFS-modified login utility or by issuing the **klog** command. Only authenticated users can perform most AFS actions.

## **B**

### **Byte, kilobyte**

A unit of measure used to measure usage of space in a volume or on a partition. A kilobyte block is equal to 1024 bytes.

## **C**

### **Cache Manager**

A set of modifications to the operating system on a client machine which enables users on the machine to access files stored in AFS. The Cache Manager requests files from the File Server and stores (*caches*) a copy of each file on the client machine's local disk. Application programs then use the cached copy, which eliminates repeated network requests to file server machines.

### **Cached File**

A copy of a file that the Cache Manager stores on a workstation's local disk.

### **Callback**

A promise from the File Server to contact the Cache Manager if the centrally stored copy of the file changes while the Cache Manager has a cached copy. If the file is altered, the File Server *breaks* the callback. The next time an application program asks for data from the file, the Cache Manager notices the broken callback and retrieves an updated copy of the file from the File Server. Callbacks ensure the user is working with the most recent copy of a file.

### **Cell**

An independently administered site running AFS, consisting of a collection of file server machines and client machines defined to belong to the cell. A machine can belong to only one cell at a time.

### **Client Machines**

Computers that perform computations for users. Users normally work on a client machine, accessing files stored on a file server machine.



### **Client/Server Computing**

A computing system in which two types of computers (client machines and server machines) perform different specialized functions.

### **Command**

A string of characters indicating an action for an AFS server to perform. For a description of AFS command syntax, see "Appendix B, AFS Command Syntax and Online Help" on page 65.

### **Command Suite**

A group of AFS commands with related functions. The command suite name is the first word in many AFS commands.

### **Complete Pathname**

A full specification of a file's location in AFS, starting at the root of the filespace (by convention mounted at the `/afs` directory) and specifying all the directories the Cache Manager must pass through to access the file. The names of the directories are separated by slashes.

## **D**

### **d (delete) Permission**

The ACL permission that enables the possessor to remove elements from a directory.

### **Directory**

A logical structure containing a collection of files and other directories.

### **Distributed File System**

A file system that joins the file systems of individual machines. Files are stored on different machines in the network but are accessible from all machines.

## **F**

### **File**

A collection of information stored and retrieved as a unit.

### **File Server Machine**

A type of machine that stores files and transfers them to client machines on request.

### **Flag**

Part of a command that determines how the command executes, or the type of output it produces.

### **Foreign Cell**

A cell other than the cell to which the client machine belongs. If the client machine is appropriately configured, users can access the AFS filespace in foreign cells as well as the local cell, and can authenticate in foreign cells in which they have AFS accounts.

## **G**

### **Group**

A defined list of users, which can be placed on a directory's ACL to extend a set of permissions to all of its members at once.

### **Group-owned Group**

A group owned by another group. All members of the owning group can administer the owned group; the members of the owned group do not have administer permissions themselves.

## **H**

### **Hierarchical File Structure**

A method of storing data in directories that are organized in a tree structure.

### **Home Directory**

A directory owned by a user and dedicated to storage of the user's personal files.

## **I**

### **i (insert) Permission**

The ACL permission that enables the possessor to add files or subdirectories to a directory.

## **Instance**

The part of a command string that defines the entity to affect.

## **K**

### **k (lock) Permission**

See the **k (lock) Permission** entry. The ACL permission that enables programs to place advisory locks on a file.

## **Kilobyte**

A unit of measure used to measure usage of space in a volume or on a partition. A kilobyte is equal to 1024 bytes. The term *kilobyte block* is sometimes used when referring to disk space.

## **L**

### **l (lookup) Permission**

The ACL permission that enables the possessor to list the contents of a directory and display its ACL.

## **Local Cell**

The cell to which the user's account and client machine belong.

## **lock Permission**

See the **k (lock) Permission** entry.

## **Login**

The process of establishing a connection to a client machine's local file system as a specific user.

## **Logout**

The process of ending a connection to the local file system.

## M

### m Privacy Flag

The third privacy flag on a group, which enables the possessor to list the members of a group or the groups to which a user belongs.

### Mode Bits

A set of permissions that the UNIX file system associates with a file or directory to control access to it. They appear in the first field of the output from the **ls -l** command.

### Mount Point

A special type of directory that associates a location in the AFS file space with a volume. It acts like a standard UNIX directory in that users can change directory to it and list its contents with the UNIX **cd** and **ls** commands.

### Mutual Authentication

A procedure through which two parties prove their identities to one another. AFS server and client processes normally mutually authenticate as they establish a connection.

## N

### NFS/AFS Translator

A program that enables users on NFS client machines to access files in the AFS filesystem.

### none ACL Shorthand

A shorthand notation used with the **fs setacl** command to delete an entry from an ACL.

## O

### o Privacy Flag

The second privacy flag on a group, which enables the possessor to list groups owned by the user or group.

### Operation Code

The second word in an AFS command that belongs to a suite. It indicates the command's function.

**Owner of a Group**

The person or group who can administer a group.

**P**

**Parent Directory**

The directory in which a directory or file resides.

**Partition**

A logical section of a disk in a computer.

**Password**

A unique, user-defined string of characters validating the user's system identity. The user must correctly enter the password in order to be authenticated.

**Permission**

A certain type of access granted on an ACL. Anyone who possesses the permission can perform the action.

**Q**

**Quota**

The size limit of a volume, assigned by the system administrator and measured in kilobyte blocks.

**R**

**r (read) Permission**

The ACL permission that enables the possessor to examine the contents of a file.

**r Privacy Flag**

The fifth privacy flag on a group, which enables the possessor to remove members from it.

**read ACL Shorthand**

A shorthand notation used with the **fs setacl** command to represent the **r** and **l** permissions.

**Relative Pathname**

A pathname that does not begin at the root of the AFS or local filespace and so represents a file or directory's location with respect to the current working directory.

**Remote Commands**

Commands used to run programs on a remote machine without establishing a persistent connection to it.

**S**

**s Privacy Flag**

The first privacy flag on a group, which enables the possessor to list general information about it.

**Self-owned Group**

A group that owns itself, enabling all of its members to administer it.

**Server**

A program or machine that provides a specialized service to its clients, such as storing and transferring files or performing authentication.

**Subdirectory**

A directory that resides in another directory in the file system hierarchy.

**Switch**

The part of a command string defining the type of an argument. It is preceded by a hyphen.

**Syntax Statement**

A specification of the options available on a command and their ordering.

### **System Administrator**

A user who is authorized to administer an AFS cell.

### **System Groups**

Groups that AFS defines automatically to represent users who share certain characteristics. See the following three entries.

#### **System:administrators group**

A system group that includes users authorized to administer AFS.

#### **System:anyuser group**

A system group that includes everyone who can gain access the cell's AFS filespace. It includes unauthenticated users, who are assigned the identity **anonymous**.

#### **System:authuser group**

A system group that includes all users who currently have valid AFS tokens for the local cell.

## **T**

### **Token**

A collection of data that the AFS server processes accept as evidence that the possessor has successfully proved his or her identity to the cell's AFS authentication service. AFS assigns the identity **anonymous** to users who do not have a token.

## **U**

### **UNIX Mode Bits**

See the **Mode Bits** entry.

### **Username**

A character string entered at login that uniquely identifies a person in the local cell.

## **V**

### **Volume**

A structure that AFS uses to group a set of files and directories into a single unit for administrative purposes. The contents of a volume reside on a single disk partition and must be mounted in the AFS filesystem to be accessible.

## **W**

### **w (write) Permission**

The ACL permission that enables the possessor to modify the contents of a file.

### **write ACL Shorthand**

A shorthand notation used with the **fs setacl** command to represent all permissions except the **a** permission.



# Index

## A

- a ACL permission, 28
- a privacy flag on groups, 54
- access control list
  - (see ACL)
- access permissions on ACL, 28
  - (see also ACL)
- access to AFS filesystem
  - ACL entries control, 35
  - controlling at directory level, 27
  - controlling for subdirectories, 31
  - enabling for service processes, 32
  - enabling for users from foreign cells, 32
  - failures, troubleshooting, 57
  - format of pathnames, 16
  - from NFS client machines, 61
  - granting and denying to users, 27
- ACL
  - accidentally removed yourself, 59
  - auxiliary permissions, 29
  - clearing, 38
  - compared to UNIX mode bits, 6
  - copying between directories, 39
  - creating negative entry, 36
  - creating normal entry, 35
  - described, 5, 27
  - displaying, 32
  - foreign users on, 31
  - negative permissions, 30
  - normal permissions, 30
  - normal vs. negative permissions, 30
  - permissions defined, 28
  - removing obsolete entries, 50
  - replacing all entries, 38
  - setting, 34
  - shorthand notation for grouping sets of permissions, 29
- adding
  - ACL entry to negative permissions section, 36
  - ACL entry to normal permissions section, 35
  - users to groups, 48
- administer ACL permission, 28

- AFS
  - accessing filesystem, 16
  - accessing from NFS client machine, 61
  - filesystem as extension of local filesystem, 2
  - security, 4
  - sharing information, 1
  - transparent access, 1
  - UIDs and GIDs, 47
- afs (/afs) directory
  - as root of AFS filesystem, 2, 16
- afs: failed to store file (error message), 60
- all shorthand for ACL permissions, 29
- apropos operation code, 69
- arguments to AFS commands, 65
- authentication
  - as another user, 11
  - defined, 5
  - in a foreign cell, 10
  - limits on consecutive failed attempts, 13
  - mutual, 4
  - to AFS on NFS client machines, 61
  - tokens as proof, 10
  - with DCE for DFS access, 11
- auxiliary ACL permissions, 29

## C

- Cache Manager
  - described, 3
  - displaying file server preferences, 24
  - tokens, use of, 10
- caching files, 3
- callbacks, 4
- cells
  - defined, 2
  - local vs. foreign, 2
- changing
  - ACLs, 34
  - AFS password, 18
  - group name, 52
  - group owner, 52
  - UNIX password, 18
- checking
  - tokens, 12
- chgrp command, 8
- chmod command, 8, 41
- chown command, 8

- clearing all ACL entries, 38
- client machine, 1, 3
- client/server computing, 1
- commands
  - AFS, issuing on NFS client machine, 61
  - chgrp, 8
  - chmod, 8, 41
  - chown, 8
  - dlog, 11
  - dpass, 11
  - fs checkservers, 22
  - fs cleanacl, 51
  - fs copyacl, 40
  - fs examine, 20
  - fs getserverprefs, 24
  - fs listacl, 32
  - fs listcells, 23
  - fs listquota, 20
  - fs quota, 19
  - fs setacl, 35, 36
  - fs whereis, 21
  - ftp, 7
  - groups, 8
  - inetd, 8
  - kas examine, 14, 17
  - klog, 11
  - knfs, 62
  - kpasswd, 18
  - ln, 8
  - login, 9
  - passwd, 18
  - pts adduser, 49
  - pts chown, 52
  - pts creategroup, 48
  - pts delete, 51
  - pts examine, 46
  - pts listowned, 46
  - pts membership, 45
  - pts removeuser, 50
  - pts rename, 53
  - pts setfields, 55
  - rcp, 7
  - rsh, 7
  - suite organization for AFS, 65
  - syntax for AFS, 65
  - tokens, 12
  - unlog, 14
- communication

- among cells and sites, 2
- between clients and servers, 1
- connection timed out (error message), 60
- copying
  - ACL between directories, 39
  - files, inability to, 57
- creating
  - ACL as copy of another, 39
  - ACL entry in negative permissions section, 36
  - ACL entry in normal permissions section, 35
  - groups, 48

## D

- d ACL permission, 28
- delete ACL permission, 28
- deleting groups, 50
- denying access with negative ACL entry, 36
- directories
  - accessing AFS, 15
  - copying ACLs between, 39
  - denying access, 36
  - displaying location, 21
  - granting access, 35
  - inability to access, 57
  - replacing ACL, 38
  - setting access control list, 27
  - shorthand notation for referencing, 67
- disk partition
  - consequences when full, 19
  - displaying percentage of space used, 20
  - displaying space available and total size, 20
  - use in AFS, 2
- displaying
  - ACL entries, 32
  - directory/file location, 21
  - disk partition percentage space used, 20
  - disk partition space available and total size, 20
  - file or directory location, 21
  - group creator, 46
  - group information, 44
  - group owner, 46
  - group-creation quota, 46
  - groups owned by a group, 46

- password expiration date, 17
- password reuse policy, 17
- percentage of volume quota used, 19
- tokens, 12
- volume quota with other information, 20, 20

distributed file system, 1

dlog command, 11

dpass command, 11

## E

- erasing all ACL entries, 38
- error messages, troubleshooting, 60
- examples
  - adding a user to an ACL, 36
  - adding members to a group, 49
  - authenticating, 12
  - authenticating as another user, 13
  - authenticating in a foreign cell, 13
  - changing group name, 53
  - changing group owner, 52
  - checking status of file servers, 23
  - copying ACL between directories, 40
  - creating a group, 49
  - creating a self-owned group, 52
  - creating entry on negative permissions section of ACL, 37
  - deleting a group, 51
  - displaying ACL for single directory, 33
  - displaying ACLs for multiple directories, 34
  - displaying group information about a user, 47
  - displaying groups a group owns, 46
  - displaying groups a user owns, 46
  - displaying information about group, 47
  - displaying members of a group, 45
  - displaying volume information, 21
  - displaying volume quota and other information, 20
  - displaying volume quota percentage used, 20
  - locating multiple files, 22
  - removing deleted groups from ACLs, 51
  - removing group members, 50
  - replacing an ACL, 39

- setting group's privacy flags, 55
- unauthenticating from selected cells, 15
- using chmod, 41
- volume/mount point interaction, 3

exiting an AFS session, 14

## F

failed to store file (error message), 60

file server machines

- checking status, 22

files

- accessing AFS, 15
- caching, 3
- denying access, 4, 36
- displaying location, 21
- granting access, 35
- inability to access, copy or save, 57
- sharing, 4, 6
- updating, 4

flags on AFS commands, 65

foreign cells

- accessing, 16
- defined, 2
- enabling access, 23

format of AFS pathnames, 16

fs commands

- checkservers, 22
- cleanacl, 51
- copyacl, 40
- examine, 20
- getserverprefs, 24
- getting help, 68
- help flag, 68
- introduction, 68
- listacl, 32
- listcells, 23
- listquota, 20
- privileges required, 68
- quota, 19
- setacl, 35
  - completely replacing ACL, 38
  - with -negative flag, 36
- whereis, 21

ftp command, 7

## G

- GID, AFS, 47
- granting access to AFS filesystem, 35
- group use of group, 44
- group-creation quota
  - defined, 44
  - displaying, 46
- groups
  - adding members, 48, 49
  - changing name, 52
  - changing owner, 52
  - creating, 48
  - creation quota, 44
  - creator, displaying, 46
  - deleting, 50
  - displaying information, 44
  - group use, 44
  - group-owned groups, 44
  - listing groups owned, 46
  - machines as members, 43
  - naming conventions, 44
  - owner as administrator, 48
  - owner, displaying, 46
  - privacy flags, 54
  - private use, 43
  - removing members, 50
  - rules for assigning ownership, 48
  - self-owned groups, 44
  - shared use, 43
- groups command, 8

## H

- help
  - examples, 69
  - online for AFS commands, 69
  - operation code in AFS command suites, 69

## I

- i ACL permission, 28
- inetd command, 8
- insert ACL permission, 28
- instances to AFS commands, 65

## K

- k ACL permission, 29
- kas commands
  - examine, 14, 17
- keyword for apropos command, 69
- klog command, 11
- knfs command, 62
- kpasswd command, 18

## L

- l ACL permission, 28
- lifetime of tokens, 11
- limits on authentication attempts, 14
- ln command, 8
- local cell, defined, 2
- local machine, 2
- local password file (/etc/passwd), 6
- lock ACL permission, 29
- logging in, 9
- logging out, 14
- login utility, 8, 9
- lookup ACL permission, 28
- lost contact with fileserver (error message), 60

## M

- m privacy flag on groups, 54
- machines
  - as members of groups, 43
  - client, 1
  - server, 1
- mode bits (UNIX)
  - interpretation in AFS, 41
- mount points defined, 3
- mutual authentication, 4

## N

- negative ACL permissions
  - defined, 30
  - setting, 30
- NFS
  - accessing AFS from client, 61
  - issuing AFS commands on NFS client machine, 61

- NFS/AFS Translator, 61
- none shorthand for ACL permissions, 29
- normal ACL permissions
  - defined, 30
  - setting, 30

## O

- o privacy flag on groups, 54
- online help, 69
- operation codes in AFS commands
  - abbreviating, 66
  - defined, 65

## P

- PAG, 10
- passwd
  - command, 18
  - file, 6
- password, 4
  - changing AFS, 18
  - changing UNIX, 18
  - expiration date, displaying, 17
  - reuse policy, displaying, 17
- pathnames, 16
- permissions on ACL
  - defined, 28
  - displaying, 32
  - normal vs. negative, 30
  - setting, 34
  - shorthand for, 29
- privacy flags on groups, 54
- private use of group, 43
- process authentication group (PAG), 10
- protection
  - for files and directories, 27
  - group-related information, 54
- Protection Database, 68
- pts commands
  - adduser, 49
  - cell argument, 68
  - chown, 52
  - creategroup, 48
  - delete, 51
  - examine, 46
  - force flag, 69

- getting help, 69
- help flag, 69
- listowned, 46
- membership, 45
- privilege required, 68
- removeuser, 50
- rename, 53
- setfields, 55

## Q

- quitting an AFS session, 14

## R

- r ACL permission, 29
- r privacy flag on groups, 54
- rcp command, 7
- read ACL permission, 29
- read shorthand for ACL permissions, 29
- remote commands, 7
- removing
  - all ACL entries, 38
  - obsolete ACL entries, 50
  - users from groups, 50
- replacing
  - all entries on ACL, 38
- root of AFS file space, 2
- rsh command, 7
- rules for assigning group names, 48

## S

- s privacy flag on groups, 54
- saving files
  - inability to, 57
  - on inaccessible file server machines, 22
- security in AFS, 4
- self-owned group, 44
- server machines defined, 1
- setpag argument to klog command, 10
- setting permissions on ACL, 34
- shared use of group, 43
- shorthand notation for ACL permissions, 29
- site defined, 2
- status of file server machines, 22
- subdirectories, accessing, 27, 31

- suite, defined for AFS command, 65
- switches on AFS commands
  - abbreviating, 67
  - defined, 65
  - omitting, 66
- syntax of AFS commands described, 65
- system groups
  - using on ACLs, 31
- system:administrators group, 31
- system:anyuser group
  - controlling access by foreign users, 16
  - using on ACLs, 31
- system:authuser group, 31

## T

- tokens
  - as proof of authentication, 5, 10
  - command, 12
  - destroying, 14
  - displaying, 12
  - getting, 11
  - lifetime, 11
  - use in mutual authentication, 5
- troubleshooting
  - accidental removal from ACL, 59
  - error messages, 60
  - inability to access, copy or save file, 57

## U

- UID, AFS, 47
- unauthenticating, 14
- UNIX, differences with AFS
  - commands, 7
  - file access/protection, 6
  - file transfer, 6
  - login, 6
  - mode bits, interpretation, 41
  - passwords, 6
  - sharing files, 6
- unlog command, 14
- users
  - account lockout time, 14
  - adding as group members, 49
  - displaying group information, 44

- displaying number of group memberships, 46
- listing groups owned, 46
- removing from groups, 50

## V

- volume quota, 19
  - displaying percentage used, 19
  - displaying with other information, 20, 20
- volumes
  - accessing via mount points, 3
  - defined, 2
  - volume/mount point interaction, 3

## W

- w ACL permission, 29
- write ACL permission, 29
- write shorthand for ACL permissions, 29

## Y

- you don't have the required access rights (error message), 60