

# Chapter 8: The AFS File System



Fermilab is using AFS (Andrew File System) as a distributed file service, and it is installed on several machines on site in a production environment, including the FNALU cluster. This chapter discusses the basic concepts of AFS and provides information on the commands used to manage your files and directories in the AFS environment.



IBM, the vendor for AFS, maintains a user's guide, a system administrator's guide and other documentation online at <http://www-3.ibm.com/software/stormgmt/afs/manuals/Library/index.html>. Additional AFS information for webmasters is maintained under the Computing Division web help page <http://www.fnal.gov/cd/webgroup/webhelp/webhelp.html>, called *AFS Cheat Sheet for Webmasters*.

## 8.1 Introduction to AFS

---

AFS is a distributed file service, in other words, a shared file system. “AFS space” is a UNIX directory/file area starting at `/afs` that can be shared between computers. This is handy when large numbers of people need to access files in an area. Fermilab has a *cell* in AFS space, which is simply the area under the AFS root directory belonging to Fermilab. Fermilab's cell is `/afs/fnal.gov/`. Any directory under this cell, e.g., `/afs/fnal.gov/x/y/z/`, has exactly the same contents when viewed/manipulated on one computer as on another, provided both computers implement AFS with the Fermilab cell. There is also a symbolic link (see section 7.3.5 *Reference a file: ln*) to a shorthand version of the cell name, which is simply `/afs/fnal`. Other cells are also accessible. They are listed as directories under `/afs`, e.g., the CERN cell at `/afs/cern.ch`. Users are required to authenticate to AFS; AFS authentication is now subsumed into Kerberos authentication.

## 8.2 How to Determine if AFS is Installed on your System

---

Typically, if AFS is installed on your system, your login directory is under a subdirectory of `/afs/fnal.gov/files/home/`. If you're still not sure if AFS is installed, issue the command:

```
% ps -ef | grep afsd
```

If you get output of the form (note the `/usr/vice/etc/afsd`):

```
root 305 1 0 Sep 30 ? 14:10 /usr/vice/etc/afsd -stat 2800 -dcache 2400
-daemons 5 -volumes 128
root 306 1 0 Sep 30 ? 5:47 /usr/vice/etc/afsd -stat 2800 -dcache 2400
-daemons 5 -volumes 128
```

then it is installed and running on your machine. If you do not get output lines like this, AFS is not installed. However it is still possible that you have access to the AFS file system via a translator<sup>1</sup> service. We discuss translator mode in section 8.8.2 *AFS in Translator Mode*. To check, enter the command:

```
% df |grep afs
```

If no output is returned, AFS is not running on your machine in any capacity. If output is returned and the line begins with a node name preceding `/afs`, for example:

```
nodename:/afs 144000000 0 144000000 0%
```

then AFS is running in translator mode. (If the output line begins with `/afs`, then AFS is actually installed on your machine.)

## 8.3 Issues Related to Login and File Access

---

### 8.3.1 Authentication in AFS

In order to access AFS files, you must be *authenticated* to AFS. Once you are authenticated, AFS issues an *AFS token*<sup>2</sup>. This token allows you to access the AFS file system; without one, you can't. The strong authentication implementation (Kerberos V5) is integrated with AFS. Thus, if your machine is part of the strengthened realm *and* it runs AFS, then when you log on and get Kerberos V5 authentication, you also automatically get an AFS token.

---

1. In translator mode, a "translator machine" runs AFS and exports the AFS file tree to other systems via NSF.

2. An AFS token is also known as a *Kerberos token*. Kerberos V4 is integrated into AFS; it is independent of the Kerberos V5 that is the heart and soul of our strong authentication project, described in section 1.1 *Computer Security at Fermilab*.

As long as you remain logged on, the Kerberos token “lives” for a period of time equal to the renewable lifetime of the Kerberos ticket, seven days.

The token is passed to all subprocesses of the login process. All normal UNIX interactive operations are therefore automatically authenticated, and access is granted to files in the AFS tree, provided you have the appropriate permissions (AFS permissions are covered in section 8.6 *File and Directory Permissions*). The Fermilab standard batch interface **fbatch** provides for token renewal at job execution time, since you can’t control when your batch jobs actually run.

Situations occasionally arise in which you are not automatically authenticated (e.g., some remote login methods) or you lose your token (e.g., you remain logged in for more than seven days). When this happens and you need to obtain a new token, take the following steps:

- First reauthenticate on your local machine to Kerberos.
- Next, forward your tickets to open remote sessions via the **k5push** command. This command is described in **Strong Authentication at Fermilab section 9.2.6 Update Tickets on Remote Terminal Sessions**.
- If your local machine runs AFS, you’re done.
- If not, ensure that your connection to the remote AFS session is encrypted. On that session, enter the command **aklog**. **aklog** prompts you for your AFS password (different from your Kerberos password) and obtains a Kerberos token, thus granting AFS authentication and access to files.



A few notes:

- 1) Running **pagsh**<sup>1</sup> first is much more secure than just running **klog**. It ensures that the token is associated with your **pagsh** process, and thus with all processes you spawn. **klog** by itself gets a token associated with your UID, which is not always unique. This could potentially allow another user to share the token, which is undesirable.
- 2) You cannot enter the commands on one line in the format **pagsh;klog**. **pagsh** starts a new **sh** shell, and **klog** needs to be run at the new shell prompt on the next line.
- 3) **pagsh** changes your shell to **sh**, so you will need to run your preferred shell afterwards (e.g., enter **tcsh**, **bash**, or **ksh** on the command line). You may also then want to source your **.login** and **.cshrc** or your **.profile** and **.shrc** scripts to ensure that your FUE environment is back to normal.

---

1. The **pagsh** command creates a new command shell (owned by the issuer of the command) and associates a new process authentication group (PAG) with the shell and the user. A PAG is a number guaranteed to identify the issuer of commands in the new shell uniquely to the local Cache Manager. The PAG is used, instead of the issuer's UNIX UID, to identify the issuer in the credential structure that the Cache Manager creates to track each user.



There are Kerberos authentication problems with running programs that spawn jobs external to your login process group. **cron** sometimes falls into this category (it is described in section 6.5.3 *Scheduling Jobs: at and cron*). You can run the job, but it will not run with authentication, and most likely will not be able to write into `/afs` space. From your Kerberized machine, use **kcron** (see the **Strong Authentication at Fermilab** manual, section 10.3 *Automated Processes*).



Be aware that being logged on as *root* grants you no special permissions in `/afs` file space; there is no such thing as being “authenticated as root”.

## 8.3.2 Managing your Token

### View Active Tokens

To see what tokens you currently hold, you can issue the command:

```
% tokens
```

The output should look similar to this:

```
Tokens held by the Cache Manager:
```

```
User's (AFS ID xxxx) tokens for afs@fnal.gov [Expires Feb 11  
12:57]
```

If the output shows no tokens, then you only have access to (usually a very limited number of) files designated as accessible to the special user *system:anyuser* (a pre-defined AFS protection group; see section 8.7 *AFS Protection Groups*). As its name implies, this designation includes anyone who can access the system (e.g., a user with a standard UNIX password but no AFS password).

### Get Back an Expired Token

See the notes in section 8.3.1 *Authentication in AFS* regarding this operation.

### Destroy a Token



Logging out does not destroy your token; it remains “live” for up to 26 hours afterwards. This is a security risk. Prior to logging out, we advise that you issue the command:

```
% kdestroy
```

This destroys your Kerberos tickets and your AFS tokens. If you create a `.logout` file (see section 9.8 *Tailoring Your Environment*), you should include this command in it.



## 8.4 AFS File System Commands

---

AFS provides a command (with many options) that allows you to address file system issues such as checking permissions, checking quota, making mount points, finding where a volume (see section 8.5 *AFS Volumes and Quota*) is mounted in the file tree, and so on. The AFS file system (**fs**) command is entered in the format:

```
% fs <main_option> <options> <arguments>
```

Many of the options can be abbreviated, and option flags can often be omitted from the command. To get a list of the main options of the **fs** command, enter:

```
% fs help
```

Here is an edited output listing showing only a few of the main options:

```
fs: Commands are:
listacl      list access control list
listquota    list volume quota
lsmount      list mount point
quota        show volume quota usage
rmmount      remove mount point
setacl       set access control list
setquota     set volume quota
whereis      list file's location
whichcell    list file's cell
```

To get usage information on a particular **fs** main option, enter:

```
% fs <main_option> -help
```

For example:

```
% fs setacl -help
```

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



A complete command reference can be found at the IBM site under <http://www-3.ibm.com/software/stormgmt/afs/manuals/Library/index.html>.

## 8.5 AFS Volumes and Quota

---

UNIX divides disks into partitions. AFS further divides partitions into subsections called *volumes*. A volume houses a subtree of related files and directories. Normally, volumes are considerably smaller than traditional file

systems. For example, each user's home directory would normally be stored in a separate volume. Large sub-directories are further sub-divided. You do not need to know which file server houses any volume. AFS locates volumes automatically.

For information on disk areas in AFS space available to Fermilab users, see <http://www.fnal.gov/cd/forms/afsdisk.html>. To examine the quota on a volume within AFS, the **fs listquota** command may be used. You can request information on several directories at a time. For example the following command requests information on the current working directory (.) and on another one specified via the environment variable \$UAFWWW:

```
% fs listquota . $UAFWWW
```

Volume Name	Quota	Used	% Used	Partition	
room.aheavey	130000	126024	97%<<	75%	<<WARNING
files.reports.UNIX	2000000	77370	4%	63%	

The output includes the name of the volume containing the specified directory(ies), the quota size, amount used, percent used, and the percent of space used on the partition containing the volume. You might also get a warning! All sizes are in kilobytes.

## 8.6 File and Directory Permissions

---

### 8.6.1 File Permissions

File permissions work quite differently from those in standard UNIX, which are described in section 7.6.1 *File Access Permissions*. In AFS, you can use the **chmod** command just as you would in a standard UNIX file system, however it behaves differently.



Although in AFS all the permission bits on a file may be examined or changed, **only the owner bits are actually used in AFS, and they apply to all users of the file** (as permitted by users' ACL settings; see below). To turn off write access to a particular file by all users, including the owner, you just need to turn off the owner write bit of the file (see section 7.6.1 *File Access Permissions*), e.g.,

```
% chmod 555 <filename>
```

or

```
% chmod a-w <filename>
```

### 8.6.2 Directory Permissions via Access Control Lists

## (ACLs)

All other AFS permissions are done with Access Control Lists (ACLs) which take effect at the directory level only. Every directory has its own ACL that defines who can access the directory and its files. Each entry in an ACL consists of a username or an AFS protection group paired with a set of permissions (e.g., read, write). An AFS protection group is simply a list of usernames grouped to share a set of permissions in one or more ACLs. If a user is in two or more ACL entries (e.g., is a member of two groups listed in the ACL) with different permissions assigned, the user gets the union of the permissions.

The permissions granted in a directory's ACL represent the *maximum* permissions. If a file in the directory has more restrictive permissions set, the user is limited by the restrictions on the file. If a file has more lenient permissions set, the user is limited by his ACL entry.

ACL rights include:

- l** lookup rights (allows user to issue an **ls** command on files in the directory, examine the directory's ACL, and access the directory's subdirectories which are protected by their own ACLs)
- i** insert rights (allows user to create new files or copy files into the directory)
- d** delete rights (allows user to remove files or move them to other directories)
- a** administrator rights (allows user to change the ACL for a directory; note that you always have this right for your home directory even if you accidentally remove this ACL.)
- r** read rights (allows user to look at the directory's contents and to read the data in the files contained in the directory)
- w** write rights (allows user to modify the contents of the files in the directory and to change the UNIX mode bits with the command **chmod**)
- k** lock rights (allows user to run programs that need to *flock* files in this directory, i.e., to apply or remove an advisory lock on an open file; see the man pages for **flock**)

Rights may also be referred to by special names that designate commonly-assigned combinations of rights. These are called *combination rights*. The defined combination rights are:

- write** all rights but **a** (i.e. **lidrwk**)
- read** **l** and **r** rights only
- all** all rights (i.e. **lidarwk**)

**none** no rights; this removes the group's or user's entry from the ACL entirely

Combination rights can be used in commands, as shown in the examples below.



A couple of notes:

- In general, users are granted all permissions on their home directories.
- When a child directory is created, it inherits the parent directory's ACL. The child directory's ACL can then be changed. Users of the child directory must have at least lookup rights (**l**) on the parent directory.

## Examining a Directory's ACL

You can examine a directory's ACL rights with the command:

```
% fs listacl /path/to/directory
```

This returns a listing of all the users/groups (groups are defined in section 8.7 *AFS Protection Groups*) that have any permissions on the directory, and what the permissions are. The directory path can be absolute (starting from root) or relative to the current working directory. For example, if you run the command:

```
% fs listacl /afs/fnal.gov/files/wwwdocs/cd/webwg/tools
```

The system returns information in the format:

```
Access list for /afs/fnal.gov/files/wwwdocs/cd/webwg/tools is
Normal rights:
  lauram:www_cd_webwg_tools rlidwk
  nicholls:www_cd rlidwk
  hanson:newsmachine rlidwka
  nicholls:wwwdocs rlidwka
  system:administrators rlidwka
  system:anyuser rl
```

The group `lauram:www_cd_webwg_tools` has read, list, insert, delete, write, and lock permissions in this directory (all but administer permissions), i.e., the group has *write* rights. Any member of that group has these permissions in this directory.

## Adding/Changing/Deleting a Directory's ACL

You can modify a directory's ACL for a particular AFS group or for an individual using the `fs setacl` command. The `fs setacl` command only changes the ACL for a single directory, not for a directory tree. The command syntax is:

```
% fs setacl -dir /path/to/directory -acl <group> \  
  <permission(s)>
```

where **<group>** is either a group or an individual username. When it is a group, it must be entered in the format **<group\_owner>:<group\_name>**.



We recommend that you generally define ACL entries for groups rather than individuals; it is much easier to maintain. When you need to add or remove permissions for an individual, it is easier to add/remove the user from one or more groups than to track down every directory whose ACL includes that user.

The directory path in the command can be absolute (starting from root) or relative to the current working directory. Any pre-existing permissions for the group or individual are invalidated; the specified permissions collectively become the new set of permissions. The permissions apply to all members of the specified group.

For example, in order to modify the ACL for the current directory (.) to include only read and lookup rights for any user (including unauthenticated users), enter:

```
% fs setacl -dir . -acl system:anyuser rl
```

or, using combination rights syntax:

```
% fs setacl -dir . -acl system:anyuser read
```

The group `system:anyuser` is described in section 8.7 *AFS Protection Groups*.



A note for Web page providers: set the permissions for `system:anyuser` to `rl` on directories containing files that you want to make accessible via a Web browser.

To remove all permissions in an ACL for a particular group (or individual), issue the `fs setacl` command with no permissions, e.g.,

```
% fs setacl -dir /path/to/directory -acl <group> ""
```

or, using combination rights syntax:

```
% fs setacl -dir /path/to/directory -acl <group> none
```

## 8.7 AFS Protection Groups

---

An AFS protection group is a list of usernames grouped to share a set of permissions on one or more directories. Any user can include any existing protection group in any ACL within your AFS cell. A protection group is designated in the format:<sup>1</sup>

```
<group_owner>:<group_name>
```

1. You may encounter groups that do not have an owner prefix; these are special groups created by the system administrators.

AFS provides three predefined protection groups:

`system:anyuser` This is similar to world permissions in UNIX. Any AFS user (anywhere in the world, and not necessarily authenticated) can access files or directories, according to the permissions granted (e.g., read, write).

`system:authuser` This is a more restrictive version of `system:anyuser`. Only users who have authenticated within the local cell (`/afs/fnal.gov` at Fermilab) may access files, according to the permissions granted (e.g., read, write).

`system:administrators`

This group includes only the few people in the `/afs/fnal.gov` cell authorized to administer AFS.

As determined by your project's `/afs` area manager(s), you may need to manage, and possibly create, protection groups.

Groups can be owned by other groups or by individual userids. Group members often are not allowed to add or remove other members of the group. If a group is owned by a group, then all the members of the *owner group* can by default add or remove other members from the *owned group*. This can avoid problems when key individuals are unavailable. Having one group consisting of a few key individuals, and using this group as the owner for all your other groups is a nice, neat way to organize your groups. Find out from your `/afs` area manager how group ownership and permissions are assigned within your project or on your system.

AFS provides the `pts` command (protection server) for group-related tasks. Like the `fs` command, `pts` has several main options. Issue the command `pts help` to list the main options (the list shown here has been abbreviated to contain only the options we discuss in this section):

```
pts: Commands are:
adduser          add a user to a group
chown            change ownership of a group
creategroup      create a new group
delete           delete a user or group from database
examine          examine an entry
listowned        list groups owned by an entry or zero id gets orphaned groups
membership       list membership of a user or group
removeuser       remove a user from a group
setfields        set fields for an entry
```



A complete command reference can be found at the IBM site under <http://www-3.ibm.com/software/stormgmt/afs/manuals/Library/index.html>.

## 8.7.1 Permissions for Performing Group-Related Tasks

Group characteristics (e.g., membership, ownership) can only be seen and/or modified according to the permissions set on the group. Here we present a brief explanation; more detailed information can be found at the IBM site under

<http://www-3.ibm.com/software/stormgmt/afs/manuals/Library/index.html>.

Every group has a set of five access flags, which represent permissions for performing sensitive tasks regarding (1) status, (2) ownership (listing owners), (3) membership (listing members), (4) adding members, and (5) removing members. There is a **pts** main option associated with each of these tasks:

status (s)	<b>pts examine</b>
owned (o)	<b>pts listowned</b>
membership (m)	<b>pts membership</b>
add (a)	<b>pts adduser</b>
remove (r)	<b>pts removeuser</b>

Each flag has one of three possible values: its first letter in lowercase, its first letter in uppercase, or a hyphen. The value determines which users can issue the corresponding command option for the group as follows:

lowercase letter (s, o, etc.)	all members of the group
uppercase (S, O, etc.)	all users (i.e., <code>system:anyuser</code> )
hyphen (-)	group owner and members of <code>system:administrators</code> only

As an example, we'll issue a **pts examine** command and examine its output:

```
% pts examine lisa:uss-group
Name: lisa:uss-group, id: -316, owner: lisa, creator: hanson,
membership: 14, flags: S-M--, group quota: 0.
```

The permissions information is contained in the `flags` entry. The flags `S-M--` are the default flags when a group is created (all users can check status and membership information, only group owner and administrators can verify ownership and add/remove group members).



If you can't successfully issue one of the **pts** command options, check the access flags! Of course if you can't issue **pts examine** to check the flags, then you don't have status permissions for the group.

## 8.7.2 Listing Information about Groups

### List Members of a Group

To list the members in a group, use the command:

```
% pts membership <group>
```

For example:

```
% pts membership lauram:www_cd_webwg_tools
```

returns the output:

```
Members of lauram:www_cd_webwg_tools (id: -454) are:
  nicholls
  hathaway
  stolz
  george
  lauram
  dwalsh
  nelly
```

### List Groups in which an Individual is a Member

To list the groups to which an individual belongs, again use **pts membership**, but with the user's id as the argument:

```
% pts membership <username>
```

For example:

```
% pts membership aheavey
```

```
Groups aheavey (id: 6302) is a member of:
  nicholls:www_reports
```

### List Groups Owned by Group or Individual

To show what groups a particular group or user owns, issue the command:

```
% pts listowned <group>
```

where **<group>** is actually either the owner group or an individual owner username. If you try to list groups owned by someone other than yourself, you may find that you do not have permission to do so.

Here are a couple of examples. To check groups owned by the *group* `nicholls:wwwdocs`, issue the command:

```
% pts listowned nicholls:wwwdocs
```

Output is returned in the format:

```
Groups owned by nicholls:wwwdocs (id: -306) are:
```

```
nicholls:www_cd_support
nicholls:www_cd_mgmt
nicholls:www_faw_events
nicholls:www_orgs_folkclub
nicholls:www_directorate
nicholls:www_cd_ups
nicholls:www_cd_webwg
```

To check groups owned by the *individual user* lauram, issue the command:

```
% pts listowned lauram
```

Output is returned in the format:

```
Groups owned by lauram (id: 1866) are:
  lauram:wwwmachine
  lauram:expwwwmachine
  lauram:expwwwadm
```

## Show Group Ownership

To find a group's owner, use the command:

```
% pts examine -name <group>
```

This is helpful to determine if a group is owned by an individual or a group. For example, to find the owner of the group `nicholls:www_reports`, run the command:

```
% pts examine nicholls:www_reports

Name: nicholls:www_reports, id: -378, owner: nicholls:wwwdocs, creator: hanson,
membership: 5, flags: S-M--, group quota: 0.
```

Its output in the entry `owner` indicates that it is owned by a group (`nicholls:wwwdocs`), not by the individual `nicholls`.

## 8.7.3 Modifying Group Characteristics

### Change the Owner of a Group



Note: It is best to change the owner of the group *before* you run `fs setacl` to add directory permissions for the owned group.

You can change ownership of a group using the command:

```
% pts chown -name <owned_group> -owner <owner_group>
```

Let's take for example the group `owner1:groupname1`, where `owner1` is an individual. We want to change its ownership to a group. The group we want to own it is designated `owner2:groupname2`. We issue the command:

```
% pts chown -name owner1:groupname1 -owner owner2:groupname2
```

The owned group is now designated `owner2:groupname1`. Notice that it takes its owner designation from the owner group, and maintains its former group name. Here's a more real-life example for clarity:

```
% pts chown -name lauram:www_cd_webwg_tools -owner \  
nicholls:wwwdocs
```

The old `lauram:www_cd_webwg_tools` is now designated `nicholls:www_cd_webwg_tools`.

You can change a group's ownership to itself (and set the group's access flags appropriately if needed) to allow all members of the group to add/remove other members and perform other administrative tasks. To change the group's ownership to itself, issue the `pts chown` command with the same group as both arguments:

```
% pts chown -name nicholls:wwwdocs -owner nicholls:wwwdocs
```

The group designation `<group_owner>:<group_name>` does not change. If you need to reset the group's access flags, see the documentation on `pts setfields` at

[http://www-3.ibm.com/software/stormgmt/afs/manuals/Library/unix/en\\_US/HTML/AdminRef/auarf225.htm#HDRPTS\\_SETFIELDS](http://www-3.ibm.com/software/stormgmt/afs/manuals/Library/unix/en_US/HTML/AdminRef/auarf225.htm#HDRPTS_SETFIELDS).



Note that there is a potentially confusing consequence of the way the group names change. All groups *look like* they're owned by individuals. You can always issue the command:

```
% pts examine -name <group>
```

to determine if the owner is an individual or a group, as shown under **Show Group Ownership** in section 8.7.2 *Listing Information about Groups*.

## Add a Member

To add a member, use the command:

```
% pts adduser -user <username> -group <group>
```

For example:

```
% pts adduser -user nelly -group lauram:www_cd_webwg_tools
```

The new member (`nelly`) must have an account on the system/cluster that mounts the AFS files he or she needs to access. The new member must reobtain an AFS token in order for the group membership to take effect.

## Remove a Member

To remove a member from a group, use the command:

```
% pts removeuser -user <username> -group <group>
```

## Create a Group



Check with your /afs area manager before creating new groups. As groups proliferate, system management can become more difficult.

To create a new AFS protection group, use the command:

```
% pts creategroup -name <group>
```

or, leaving off the **-name** option flag for simplicity:

```
% pts creategroup <group>
```

Always enter a group in the format `<group_owner>:<group_name>`; don't enter only the `<group_owner>` portion. By default, the group owner is yourself.<sup>1</sup>

As an example, user *lauram* could run the command:

```
% pts creategroup lauram:www_cd_webwg
```

## Remove a Group

To remove a group, use the command:

```
% pts delete -nameorid <group>
```

For example:

```
% pts delete -nameorid lauram:www_cd_webwg_tools
```

## 8.8 Implications of ACLs

---

Kerberos aside, the implementation of security in our Fermilab AFS cell is based on the notion that sharing information is more important than trying to protect it. Therefore, in most cases, the default has been to set ACLs to have the least security that is still reasonable. As currently implemented, all user home directories come with their ACL set so that `system:anyuser` has **r1** (read and lookup) permissions. A `Mail` subdirectory (used by the **MH** mail readers) is provided with more secure permissions.

The practical implication of this is that *anyone* on the internet running an AFS client can read your files, unless you change the ACL. (The AFS client for Windows bypasses Kerberos authentication.) Home directories are *writable* only by their owners (that is, the owner has **rldiwka** permission), but the

---

1. There is an option (**-owner**) to set the owner to another individual or a group, but we recommend that you just use **chown** afterwards as described in section 8.7.3 *Modifying Group Characteristics*.

world can read them. This is probably fine in many cases, but you should be aware of it and protect your files as you see fit, according to the guidelines presented below.

## 8.8.1 Protecting your Subdirectories

You can protect any single directory by changing its ACL to turn off permission for `system:anyuser` as well as for other users or groups that should be denied permissions. For example, if you use the mail reader **pine**, you may want to protect the message subdirectory `mail`. To make it completely inaccessible by `system:anyuser`, you'd issue this command:

```
% fs setacl $HOME/mail system:anyuser none
```

On the other hand, if you need to allow others to write into any of your directories, the default permission is too constraining. Say you are in a collaborative effort with user *mrchips*. You could allow him full permission in your `$HOME/shared` directory by issuing the command:

```
% fs setacl $HOME/shared mrchips all
```



Recall from section 8.6.2 *Directory Permissions via Access Control Lists (ACLs)* that if a user is in two or more groups that have different permissions on a directory, the user gets the union of the permissions.

Also, recall from section 8.6.2 that the `fs setacl` command only changes the permission for a single directory. If you have a directory hierarchy on which you want to change permissions, you'll have to use a UNIX command that traverses down the tree and changes all the directories as it goes. The `find` command can be used (see section 7.4.1 *Find a File: find*), but it must be used judiciously in the AFS environment!



**The `find` command is not recommended for inexperienced UNIX users** (see section 8.10.2 *AFS and the UNIX Command “find”*).

As an extension of the above example, say you had a directory hierarchy under `$HOME/shared` to which you wanted to allow *mrchips* full access. The `find` command could be used instead of `fs setacl`, as follows:

```
% find $HOME/shared -type d -print -exec fs setacl -dir {} -acl\  
mrchips all \;
```

This would traverse down from the `$HOME/shared` directory, changing the ACL for each of the directories it finds. The `-print` argument causes the system to print out all the directories the command encounters, allowing you to monitor the progress.

## Protecting your Home Directory

We strongly recommend that you make your home directory world readable, and simply keep your private files in protected subdirectories. That said, ...



... we do *not* recommend that you set the ACL on your home directory such that `system:anyuser` has no permissions (i.e., combination rights **none**) in order to keep your top level directory private. There are at least a couple of undesirable consequences:

- If you ever managed to log in unauthenticated, you wouldn't be able to enter your home directory. In fact you might not be able to log in at all, depending on the UNIX operating system.
- The dot files in your home directory (e.g., `.login`, `.profile`) would be unreadable by unauthenticated system processes, which could cause them to break. For example, if `system:anyuser` does not have at least `r1` permissions on your home directory, the **sendmail** program will not be able to read your `.forward` file, and your mail forwarding will break.

If for some reason you really want to protect your home directory, you can do so to the extent that only `1` (lookup) permission is granted. However, you must make sure that any files that *must* be world readable, such as your `.forward` file, remain accessible. **Be aware that it is not always obvious which files must remain world readable in order to preserve the behavior of your environment.** You can protect your home directory as follows (Proceed with caution!):



- 1) Every AFS home directory is created with a subdirectory called `public`. Move the files that must remain world readable into this directory.
- 2) For each file moved into `public`, create a symbolic link in your home directory to the file in the `public` directory. Use the same filename.
- 3) *After* all the necessary files are moved and linked, *then* shut off all permissions except `1` (lookup) on your home directory.

Note that you *must* leave the `1` permission turned on or programs won't be able to find the file in `public` via the symbolic link.

Here is a sample session, assuming the only file that must remain world readable is `.forward` (there would actually be many files). It would be run from the user's home directory:

```
% mv .forward public/.forward
% ln -s public/.forward .forward
% fs setacl . system:anyuser 1
```

## 8.8.2 AFS in Translator Mode

Translator mode is not terribly reliable, and we don't recommend it.

If your machine is accessing AFS via a translator node, you do not get authenticated when you log in, and in fact you can't run the **aklog** program discussed in section 8.3.1 *Authentication in AFS*. You cannot access your AFS login area. You only have access to directories for which an ACL is defined for `system:anyuser`. You have access to files in those directories according to the ACL entry for `system:anyuser` and the file owner bits, as usual.

At Fermilab most of the **UPS** products (see section 1.3 *The Fermi UNIX Environment (FUE) and Product Support*) are set with *read* permissions (**r1**) for `system:anyuser`, thus allowing access to products maintained in the `/afs` products area from a machine running in translator mode. This is not true for products that are site-licensed, which are made accessible only to users on site.

A system admin can replace the **aklog** program with the following script:

```
#!/bin/sh
/usr/krb5/bin/rsh -F fsut01 /usr/afsws/bin/knfs/ `hostname`
```

where `fsut01` is the translator. The user can do this to get an NFS translator token for their host.

## 8.9 File Locking in AFS

---

The file locking mechanism in AFS does not really follow POSIX<sup>1</sup> semantics. There are a few issues to mention:

- Files may only be locked as a whole; regions of a file may not be locked.
- File locking only works properly and reliably from a single system. If a file is locked from one client and an attempt is made to access the file from another client, the error `EWOULDBLOCK` is returned.
- There is no deadlock prevention in AFS, so deadlock situations can occur with file locking.
- Any program that attempts to use byte-range file locking in AFS will get a message from the cache manager warning that other users may be accessing the same file. Usually these messages can be safely ignored.

---

1. IEEE's Portable Application Standards Committee (PASC) is the group that has and continues to develop the POSIX family of standards. POSIX stands for Portable Operating System Interface (X). It is the IEEE's version of UNIX. For more information, see <http://www.pasc.org/>.

Generally we don't recommend including applications that depend on file locking in the AFS file space. Contact the help desk at <http://csdserver1.fnal.gov/HelpDesk/cd/> for more information or for resolution of a problem.

## 8.10 Frequently Asked Questions

---

### 8.10.1 Lost Access to Files

#### Why can't I access files I'm supposed to be able to edit?

First see what permissions you have by checking:

- which groups have `rlwidk` permissions on the directory (use `fs listacl <directory>`)
- that you are in at least one of these groups (use `pts membership <username>`)
- that the owner of the file has the standard `rw` UNIX permissions (see section 7.6.1 *File Access Permissions*)



Your AFS authentication may have expired. Remember that AFS authentication lasts only as long as the maximum renewable Kerberos authentication (six days at Fermilab). If your authentication has expired, you will not have access to your files. You can reauthenticate to AFS by reauthenticating to Kerberos.

### 8.10.2 AFS and the UNIX Command “find”

#### Why shouldn't I use “find” in AFS space?

You should be *very careful* about using any command that traverses the file system on a machine that has `/afs` mounted. Be aware that a `find` on your system starting at root (`/`) will traverse the *whole* AFS file tree, including all the other AFS sites mounted on our cell. This is particularly problematic on some workstations, like Solaris 1 Suns, which by default run a nightly `cron` job that traverses the whole file system. Also note that the `-mount` and `-xdev` options (e.g., `find / -mount ... -print`) won't recognize an `/afs` file system boundary; `find` can't tell the difference between local files and AFS files. The `find` command is discussed in section 7.4.1 *Find a File: find*.

## 8.10.3 Error Messages

### What does it mean if I get an error message like this:

```
afs: Waiting for busy volume 536870945 in cell fnal.gov
```

This is an error message from AFS that indicates that you are trying to access a volume that is busy. There may be a number of perfectly normal reasons for this. It generally means that either your volume is in the process of being cloned for a nightly backup, or one of the system administrators is in the process of moving your volume to a different disk because the one you are on is filling up. Normally the process that generated the error will just hang harmlessly for a few minutes until the process that locked the volume finishes. If this goes on for more than 20 minutes or so, contact the help desk and inquire about what is going on.

## 8.10.4 Retrieving Old Files

### What if I need to retrieve yesterday's copy of a file?

Daily backups of the entire Fermilab cell are available from `/afs/fnal.gov/files/backup/`. For example if your home area is `/afs/fnal.gov/files/home/room3/joe`, you should be able to find yesterday's files in `/afs/fnal.gov/files/backup/home/room3/joe`. Only backed-up files show up there. The backups are done at 12:45 a.m. seven days a week.

If you cannot locate your files, contact the help desk to request that your backup volume be mounted so that you can access it.

## 8.10.5 Link Failure

### Why did my link fail?

Hard links can be used only within an AFS volume, not across volumes. Generally, you should use symbolic links. Links are discussed in section 7.3.5  
*Reference a file: ln.*