AFS on Athena

AFS on Athena

On this page:

About This Document

This document assumes that you are familiar with the command line on Athena and the concept of directory hierarchies. These concepts are covered in Working With the Command Line

The AFS Hierarchy

A file system organizes the files and directories on a computer system. AFS's file structure and hierarchy depend on *cells* and *volumes.*

An AFS *cell* is a collection of filespace managed by some administrative domain (e.g., a company, a university department, or any defined group of users). There are many AFS cells around the world, and each cell can have files and volumes on many different file servers. Athena's AFS cell consists of all the directories under /afs/athena.mit.edu. Another AFS cell at MIT of interest to Athena users is /afs/sipb.mit.edu, which contains files maintained by SIPB.

An AFS *volume* is a collection of files and part of a cell. At MIT, there is often a 1-to-1 relationship between a volume and a *locker* (explained in the next section).

A path in AFS might look something like this: **/afs/athena.mit.edu/user/j/o/joeuser**. Let's look at each component of the path:

- **/afs**: All files in AFS are under the /afs path.
- **athena.mit.edu**: The directory immediately below /afs is the name of a cell. In this case, most files you'll access on Athena are in the athena.mit.edu cell.
- **user**: The next directory tells us the type of volume. In this case, it's a user volume (a home directory)
- **j**: Because of the large number of users, user home directories are further broken down by the first letter of the username.
- **o**: User home directories are further broken down by the second letter of the username
- **joeuser**: joeuser's home directory

Lockers

As you can imagine, typing the full AFS pathname to access a file would quickly become tedious. Fortunately, you don't have to – you can simply access the desired locker instead. Lockers are not part of the AFS software, but are used to access files in AFS. Lockers are available under the **/mit** directory on your workstation. Your home directory in AFS is also a locker.

Lockers are accessed simply by changing your working directory to **/mit/*lockername***. For example:

```
joeuser@athena:~$ cd /mit/joeuser
```

Note: Athena 9.4 users will need to attach the locker first with the **attach** command.

The two paths (/mit/joeuser and /afs/athena.mit.edu/user/j/o/joeuser) refer to the same location in AFS, but using lockers is the easiest way to access files in AFS.

Given a locker name, you can obtain the full AFS path in one of two ways:

- By changing your current directory to the locker and using the `pwd -P` command:
```
joeuser@athena:~$ cd /mit/joeuser
joeuser@athena:/mit/joeuser$ pwd -P
/afs/athena.mit.edu/user/j/o/joeuser
```
- By looking up the locker in Hesiod:
```
joeuser@athena:~$ hesinfo joeuser filsys
AFS /afs/athena.mit.edu/user/j/o/joeuser w /mit/joeuser
```

The second field in the Hesiod lookup will be the full AFS path of the locker.

Permissions and ACLs

In most UNIX-like operating systems, permissions can be set on a per-file basis as well as a per-directory basis. That is, you could have two files in the same directory with different permissions, or you could specify permissions for the entire directory. When using AFS, however, permissions can be specified on a per-directory basis only. This means that if you have a file you want to share with a friend, you need to give your friend the ability to read all files in that directory. (Since you probably don't want to give your friend the ability to read all files in your home directory, you'll want to create a new directory for sharing files.)

Every directory in AFS has an *access control list* (ACL). The ACL for a given directory is a list of users and groups, paired with their rights. The owner of a directory (and anyone who has administer rights, as explained below) can set and manipulate the *ACL* for the directory with the **fs** command.

Permissions in AFS are specified with some combination of 7 "rights"that enable users to:

- **r:** Read the contents of files in the directory.
- **l:** Look up status information about the files in the directory (e.g., list the filenames in the directory and look at the directory's access control list). This does not imply read access, but if you don't have lookup access, no other form of access (other than administer) can be used.
- **i:** Insert files or subdirectories into the directory (i.e., create new files or move existing files into the directory). This does not imply ability to modify these same files (w).
- **d:** Delete files or subdirectories from the directory.
- **w:** Write or edit the contents of files in the directory. This only allows changing existing files; it does not imply delete (d) or insert (i) access. Write access also gives **chmod** access to files.
- **k:** Set an advisory lock on a file. This is used mainly by application programs and is not useful to most users; see **man flock** for more information.
- **a:** Administer or change the rights on the access control list. This does not immediately imply any other kind of access. The owner of a volume always has implicit administer rights. The owner can give administer rights to other users, who can then also change the rights on the ACL. (Thus, be careful about giving administer rights to other users!)

These rights have been aliased into commonly used groups of rights that can be referred to with the following shorthand notation:

- **read** expands to **rl**: read and look-up rights
- **write** expands to **rlidwk**: all rights but administer
- **all** expands to **rlidwka**: all rights
- **none**: used to clear access

Looking at Permissions

To list the ACL for a directory, use the **fs la** command:

```
fs la directory
```

In the above example, *directory* is the directory for which you want to see the ACL. If you don't specify a directory, the ACL for the current directory is displayed. The command will return a list of users and groups along with their associated rights. (A name with a colon in it is a group. See About Groups for more information.) For example:

```
joeuser@athena:~$ fs la
Access list for . is
Normal rights:
```

```
system:expunge ld
system:anyuser l
joeuser rlidwka
```

In the example above, no directory was specified, so the ACL for the current directory (indicated by the **.**) was used. The permissions are:

- The group system:expunge has list and delete (ld) access by default. (This is a special group described later in this document.)
- The group system:anyuser has list (l) access. (Like system:expunge, this is a special group.)
- The owner (joeuser) has full access (rlidwka).

Setting Permissions

As a directory owner, you can set permission rights for users to access your directories. To assign access rights to a directory, use the **fs sa** command:

```
fs sa directory user-or-group rights
```

In the above example, *directory* is the directory whose ACL you want to change; *user-or-group* is the name of a user or a group (group names are prefaced with the word "system" and a colon); and *rights* is some combination of the letters "rlidwka" or the words "read", "write", "all", or "none".

For example, if you were working with a friend, you might want to create a subdirectory (e.g. "group_project) and give that friend the ability to edit files in that directory:

```
joeuser@athena:~$ mkdir group_project
joeuser@athena:~$ fs sa group_project jruser write
joeuser@athena:~$
```

You won't receive any confirmation of your change; you will simply see a new shell prompt. We recommend always double-checking by running **fs la** again on the directory to verify that the ACL is set correctly. It's easy to make a typo and assign permissions to the wrong user or group.

Note: Changing the ACL for a directory only affects that directory. Any subdirectories within that directory will not be affected. It is possible to change the ACL for a directory and all its subdirectories as discussed in: How do I change AFS permissions for all subdirectories of a directory (recursively)?.

Removing Permissions

To remove a user or group from an ACL, set that user or group's rights to the word **none**. For example, to revert the change from the previous example:

```
joeuser@athena:~$ fs sa group_project jruser none
```

This clears the rights of the user or group from being explicitly specified in the directory's ACL. However, this does not necessarily preclude access to the directory--a user could be a member of another group that still has access to the directory, and the user would therefore have access.

About Groups

If you are only working with one or two other users, adding each of them by hand is easy to do. For more people, or a group of people whose membership is likely to change regularly, using a **group** is the best choice.

A group is a Moira list that has been designated as a group. Any moira list you control can be designated as a group, using WebMoira or the **listmaint** or **blanche** commands. (When you update a group – with **listmaint**, **blanche** or the web interface – the change takes effect immediately for AFS purposes such as updating access control lists.)

When setting permissions, all groups must be prefaced by the word "system" and a colon (:). For example, if you wanted to give the group "myfriends" permission to read files out of the directory "project", you would use the following commmand:

```
joeuser@athena:~$ fs sa project system:myfriends read
```

Had you left out the "system:", you would be telling AFS to give "read" permissions to the user "myfriends" instead. Since that user doesn't exist, you would get an error message.

Special Groups

In addition to groups that you specify, there are some special groups which exist in AFS:

- **system:authuser:** Any user with valid Kerberos tickets in the same cell (e.g., under athena.mit.edu). For all practical purposes, this is all users at MIT.
- **system:anyuser:** Any user, including AFS users not at MIT. You should **NEVER** assign "write" permissions to system:anyuser.
- **system:expunge:** The process which runs automatically on your fileserver to remove old deleted files permanently. This group is given **ld** access to your directories by default, so that the process can look up the old deleted files and remove them.

About system:anyuser

You'll notice that by default, your home directory has "list" (l) permissions assigned to system:anyuser. This is required for the web servers to display your personal website in your "www" directory. However, if you open up a web browser and visit your home directory (e.g. http://web.mit.edu/joeuser), you will see the names of files and directories in your account. These files cannot be accessed (you will receive a "Permission Denied" error if you click on them). If you are concerned about the names of your files being displayed, you can create an empty "index.html" file with the following command:

```
joeuser@athena:~$ touch ~/index.html
```

This will cause the web server to return a "Permission Denied" error instead of the directory list.

Common Errors

Occasionally, you may receive an error like the following when assigning permissions:

```
fs: Invalid argument, possible reasons include:
-File not in AFS
-Too many users on access control list
-Tried to add non-existent user to access control list
```

If you receive this error, ask the following:

- Did you type the name of the directory correctly?
- Did you type the arguments in the correct order? A common mistake is to transpose the user and directory.
- If you're trying to change permissions for a group, did you put "system:" before the group name?
- Is the group actually designated as a group?
- Do you in fact have too many (> 15) entries on the list for a directory? If so, consider making a group to contain those entries and adding that.

If you continue to receive the error after checking all these things, you should contact Athena User Accounts at x3-1325 or accounts@mit.edu.

Negative Rights

The ACL can actually be two lists for a directory: *Normal rights* give users or groups access to that directory; *Negative rights* are rights that a user or group explicitly does *not* have. The Negative rights list always takes precedence over the Normal rights list.

Note: Negative rights are *not* the same thing as removing a user's rights or setting a user's rights to "none". Do *not* attempt to use negative rights to simply clear a user from the access list.

To specify Negative rights, and thus ban a user or group from having those specific rights, use the **-negative** (**-n**) flag in the **fs sa** command:

```
fs sa directory user-or-group rights -negative
```

This prevents the specified user or group from having the specified access to the directory, even if they are explicitly or implicitly (by being a member of another group) given "Normal rights". Negative rights are removed from the list by specifying **none** for the rights, just as with normal rights:

```
fs sa directory user-or-group none -negative
```

Warning: Negative rights can be complicated and have unintended consequences. If you're not sure whether or not you should use negative rights, you can check with Athena User Accounts at x3-1325 or accounts@mit.edu. (For example: The group system:anyuser does not require authentication. If you assign Negative rights to a user, but give system:anyuser Normal rights, it's possible for an unauthenticated user to gain access to the directory with system:anyuser's rights)

Using Symbolic Links

Although permissions in AFS are on a per-directory basis, it is possible to emulate some aspects of per-file permissions using symbolic links. A symbolic link is simply a pointer to a file elsewhere in your account.

For example, suppose you had submitted a scientific paper that included a URL such as "http://web.mit.edu/joeuser/research.html" instead of "http://web.mit.edu/joeuser/www/research.html". Anyone trying to access that file would get a "Permission Denied" error, since your home directory is not publically accessible, but you don't want to make your entire home directory accessible to the world just to fix this one error.

Using a symbolic link, you can put the file in your "Public" directory, but still allow it to be accessed at the original location:

```
joeuser@athena:~$ mv research.html Public/
joeuser@athena:~$ ln -s Public/research.html research.html
```

This moves the file into your "Public" directory and creates a *symbolic link*. Thus, when a user tries to access http://web.mit.edu/joeuser/research.html, the server knows to look at the file in your Public directory (which is publically accessible) and use that instead.

Links allow you to have files appear in different directories or under different names, while avoiding the complications of making multiple copies. Making multiple copies of the files would take up extra disk space, and you would have to remember to edit each file when you wanted to change its contents.

About File Mode Bits (rwx)

AFS ACLs are privileges belonging to a user or group, giving permission to read, write, delete, or otherwise access any file in a given *directory*. In contrast, file *mode bits* – read (r), write (w), and execute (x) – are properties belonging to an individual *file*. Under AFS, only the first three (owner) mode bits are relevant, and they apply to anyone who has read or write access to the directory, not just the owner of the directory.

The default Athena setup allows most AFS-based users to ignore the mode bits, because read and write are set for the owner by default when the file is created, allowing users with read and/or write access to the directory to access that file.

In general, it is best to establish accessibility of a specific file in AFS by putting the files in a directory with the ACL you want rather than by toggling the file mode bits of the specific file.

Related Links

Athena at MIT
Athena On-line Help
Getting Started with Athena
Working with the Command Line
Athena Stock Answers