# AFS Locker Maintenance Guide

## AFS Locker Maintenance Guide

On this page:


- Introduction
  - About AFS Permissions
  - Getting Help
- About Lockers
  - What should be stored in lockers?
  - What are Some Limitations to Lockers?
  - What are the Different Types of Lockers?
- Maintaining a Course Locker
  - Quota
    - Determining your quota usage.
    - Restoring deleted files
  - Access Control
    - Making good use of groups
  - Managing Executables
    - Introduction to binary directories
    - The `arch` directory
    - Compatibility symlinks
    - Sysname values
    - Configuring executables
    - Source code
    - Removing binaries for obsolete platforms
    - Dealing with operating system upgrades
- Technical Information
  - AFS
    - About AFS
    - Groups and AFS
    - The @sys link
  - 4.2.2 $ATHENA_SYS and $ATHENA_SYS_COMPAT
    - 4.2.3 athdir
    - 4.2.4 Using a shared directory for scripts
- Course Locker Re-use Policy
  - FAQ


## Introduction

Lockers are used for many different purposes in the Athena Computing Environment and many different people are responsible for maintaining, upgrading, and supporting the contents of these lockers. This document is aimed at members of the community who maintain a locker, such as a course locker, an ASA activity locker, a departmental or lab locker, or a personal locker.

This document will explain, among other things, how to configure a locker to serve a website, how to install software into a locker, and other best practices for locker maintenance.

### About AFS Permissions

This document assumes that you are already familiar with the basics of AFS permissions. At a minimum, you should understand the syntax of the `fs` command (e.g. `fs sa`, `fs la`). Please be sure you understand the content in the "How do permissions work in AFS?" article before continuing.

### Getting Help

This document does not cover every situation and issue surrounding lockers. If you run into problems, try searching this knowledge base for your issue. Many common problems are documented in other articles in the knowledge base.

One of the most common problems is running out of space. Be sure to check your quota usage to make sure you're staying below your quota. If you need more space, [contact User Accounts]

# About Lockers

## What should be stored in lockers?

- **Software**: You can install software specific to your course, activity, or group into a locker, so that others can type **`add locker`** and run software, similar to how you can run IS&T-installed software such as **matlab**.
- **Datasets** or **Code Repositories**: When teaching a course, you can put your datasets or code repositories in a locker for easy access by the students. Putting it in a locker allows you to update it over the course of the semester if you need to make changes without asking students to re-download the data. **Git**, **Subversion**, and other common repositories can be used.
- **Information and Metadata**: Your lab or group could store common information such as procedures or checklists in your locker, or templates for certain kinds of documents. The ASA requires a specific directory in student activity lockers for use with the ASA database. You can also configure part of your locker to be restricted to members of your group only, where you might store things like emergency contact information for your group's officers or your lab's emergency contacts.
- **Website**: You can host a basic website out of your locker, with static pages and images, or you can make use of SIPB's Scripts service to host more advanced content, such as a blog or wiki.

## What are Some Limitations to Lockers?

Networked file space is always more limited than what you may have on your own computer; we have limited resources and creating lockers above a certain size is problematic (for technical reasons we need to divide larger allocations into separate AFS volumes). Because of this, lockers are not ideal for storing space-intensive data such as sound and movies.

In addition to complying with MIT's Student Information Policy restrictions, you should be very careful about using a locker to store data which you consider sensitive, such as solutions for upcoming problem sets, or your group member's personal contact information. Make sure you understand access control settings before putting anything in the locker which shouldn't be public.

## What are the Different Types of Lockers?

A locker's type identifies it to the operations staff and also determines where it lives in the file system hierarchy. The following table describes other common locker types.

| Type | Description | Location in AFS hierarchy |
|---|---|---|
| system | system files | /afs/athena/system |
| software | commercial software | /afs/athena/software |
| user | home directories | /afs/athena/user |
| project | Athena-related projects | /afs/athena/project |
| dept | Departmental leased disk space | /afs/athena/dept |
| org | Organizations within MIT.<br>org lockers are typically used for web pages | /afs/athena/org |
| course | course locker allocations | /afs/athena/course |
| urop | urop projects | /afs/athena/course/urop |
| activity | Recognized MIT activities | /afs/athena/activity |

# Maintaining a Course Locker

## Quota

Each volume in a course locker has been assigned a quota - a limit to its size on disk. The following sections tell you how to view the quota usage and how to clean out old files that are wasting space. As of 2013, the default quota is 2 GB. If your locker has a quota that is smaller than 2GB, you should [contact User Accounts] for an increase.

If you have exhausted your initial 2GB allocation, you can contact Accounts for an increase. Please note that there are limitations on how much data can be stored in AFS, and Accounts can work with you to determine the best situation for your needs.

### Determining your quota usage.

The AFS command **fs lq** (File Server List Quota) command displays the quota allocated to and used by any locker. Since AFS quota is allocated in terms of volumes and not lockers, **fs lq** is not guaranteed to list the entire quota allocated to a locker. However, most course lockers consist of a single volume, making the numbers returned by **fs lq** accurate.

For more information on cleaning up your locker and reducing your quota usage, please see How can I clean up my locker and get below my quota?.

### Restoring deleted files

All AFS lockers have a special directory called `OldFiles` which contains a nightly snapshot of your locker. If you accidentally delete a file, you can restore it from `OldFiles` immediately. See What is the OldFiles directory? for more information.

Tip: The OldFiles directory is actually a link to your backup volume in AFS. It is possible to hide or remove this link. If you do not see the OldFiles directory in the top level of your locker, see What is the OldFiles directory? for instructions on restoring the link.

## Access Control

The term access control refers to the mechanism that controls which individuals or groups can access files and directories. In short, access control lists, or ACLs for short, determine who can read files, who can write files, and who cannot access files in a locker.

> **AFS Permissions and ACLs**
> AFS permissions are explained in detail in the article "How do permissions work in AFS?" You are strongly encouraged to read that document. This section explains best practices regarding access control in lockers, but assumes familiarity with the basics of AFS permissions.
>
> In particular, remember that permissions in AFS are **per-directory**: all files in a directory share the same permissions. This **cannot** be overridden with the `chmod` or `chown` commands.

### Making good use of groups

You are strongly encouraged to use groups as the primary level of access control, even if groups only have a single member on them. That ensures the best forward compatibility for your locker, so that when a staff member leaves MIT, or student graduates, you only need to change a single list rather than combing through your locker to find all entries with that username and changing them. Remember that groups can contain other groups.

Additionally, AFS has an internal limitation on the size of Access Control Lists, and using groups can help avoid that limitation.

#### Common Conventions

Below are some common conventions used in lockers. In the examples below, replace *`locker`* with the name of the locker.

| List | Purpose |
|---|---|
| *`locker`*`-admin` | A list of the "owners" of the locker. This list should have full (`rlidwka`) permissions on all directories of the locker. |
| *`locker`*`-www` | A list of people who update the website hosted in the locker. This list should have `write` permissions on the `www` subdirectory and its children. |
| *`course`*`-staff`<br>*`course`*<br>`-instructors`<br>*`course`*`-tas` | For course lockers, consider creating separate lists for the instructors and the TAs, if certain areas of the locker should be accessible to faculty only. |
| *`course`*<br>`-students` | For course lockers, you may also wish to restrict certain content to students enrolled in the course |
| *`activity`*<br>`-members` | For activity lockers, you may wish to restrict certain content to members of the group. |

Note: If you already have Moira lists or groups that mirror the membership of lists above, consider creating the above lists anyway, and adding your existing lists as members. That way, if you re-purpose one of your lists, you only need to change one thing. For example, if you have the list myclub-officers, consider creating myclub-admin anyway, and adding myclub-officers as a member of that list.

## Managing Executables

Many instructors want to make executable programs available to students via a course locker. Due to the nature of the Athena environment, this is slightly more complicated than just plunking a binary into the locker. The guidelines outlined here conform to the standards used by Athena developers and make the course locker easy to maintain and course software easy to use. The **lockers** man page is the definitive description of Athena locker organization conventions.

## Introduction to binary directories

When deploying software in lockers, it is important to consider what portions of the software are *architecture-depedent* and which are *architecture-independent*. For example, compiled C code is architecture-dependent: A program compiled on Solaris will not run on Linux. However, man pages or HTML documentation is architecture-independent. There are also grey areas: Your software may use a data format that is byte-order dependent, or that uses a different database subsystem on different platforms. Similarly, if your software consists entirely of Python or Perl scripts, it may be architecture-independent.

As of 2009, Athena only has one supported platform (Linux), however we still make use of the architecture-dependent infrastructure to differentiate between different versions (e.g. Ubuntu 10.04 vs Ubuntu 12.04) which may have different dynamic library versions, etc.

If you wish to simply ignore the architecture-dependece issue, you can configure your locker to assume everything (including documentation) is architecture-dependent, and install one copy for every architecture you wish to support.

## The `arch` directory

Note: If your locker dates from 1997 or earlier, it may have other entries at the top level, such as `linuxbin`, `sun4bin`, etc. **That syntax is deprecated and should not be used for new software installations.**

Every locker that provides software should have a directory at the top level called `arch`, and with in that directory, you should have a directory for each Athena "sysname" you wish to support. Consult the table below for different values of Athena sysnames. For the purposes of this document, we will work with the sysname `amd64_ubuntu1204`, which refers to the current (2013-2014) version of Athena.

Within each sysname directory, you will typically have a `bin` and `lib` directory, as binaries and libraries are the most common architecture-dependent things.

## Compatibility symlinks

Many locker maintainers choose to create compatibility symlinks at the top level of their directory, so that their locker somewhat resembles what one might find under `/usr` or `/usr/local` on a workstation.

| Link | Target |
|---|---|
| bin | arch/@sys/bin |
| lib | arch/@sys/lib |

If you choose to make everything architecture-dependent, you might also have:

| | |
|---|---|
| share | arch/@sys/share |
| man | arch/@sys/man |

## Sysname values

- **Current Values (Debathena)**

| Directory Name | Platform and Operating System |
|---|---|
| arch/amd64_ubuntu1204/bin | Linux, Ubuntu 12.04 |
| arch/amd64_ubuntu1004/bin | Linux, Ubuntu 10.04 (previous version) |
| arch/amd64_deb60/bin | Linux, Debian 6.0 |

- **Previous Values**

| Directory Name | Platform and Operating System |
|---|---|
| arch/sun4x_58/bin | Sun, Solaris 8 (a.k.a. 2.8) |
| arch/sgi_65/bin | SGI, Irix 6.5 (no change) |

| | |
|---|---|
| arch/i386_linux24/bin | Linux, Red Hat 7.1 |
| arch/sun4x_57/bin | Sun, Solaris 7 (a.k.a. 2.7) |
| arch/sgi_65/bin | SGI, Irix 6.5 |
| arch/i386_linux22/bin | Linux, Red Hat 6.2 |
| arch/i386_linux3/bin | Linux, Red Hat 5.2 (SIPB, desupported 6/2001) |
| arch/i386_nbsd1/bin | NetBSD (SIPB, desupported 6/2001) |
| arch/sun4x_56/bin | Sun, Solaris 2.6 |
| arch/sgi_65/bin | SGI, Irix 6.5 |
| arch/sun4x_55/bin | Sun, Solaris 2.5 |
| arch/sun4m_54/bin | Sun, Solaris 2.4 |
| arch/sgi_63/bin | SGI O2, Irix 6.3 |
| arch/sgi_62/bin | SGI Indy, Irix 6.2 |
| arch/sgi_53/bin | SGI Indy, Irix 5.3 |
| arch/pmax_u14/bin | Digital DECstation, Ultrix 4.3 |
| arch/rs_aix32/bin | IBM RISC/6000, AIX 3.2 |

## Configuring executables

There are two cases to consider when configuring executables. In the first case, users run the program without any special environment settings, or else are expected to set up the environment for themselves. The second case are programs for which you want to set up the environment for students before invoking the executable itself.

The first case is simple; just put the executables into the directories as described above and tell users to use **add** as described above.

In the second case, the program installed in the binary directory will actually be some sort of startup script which sets the environment and then invokes the real binary. The following example shows a startup scripts that sets an environment variable, attaches an extra locker, and then invokes a binary. Obviously, the startup script cannot share the same name as the binary. In cases like that, the custom is to change the binary name by appending ".real" to it, or by calling the startup script "launch_whatever".

```
$ cat /mit/mylocker/arch/amd64_ubuntu1204/bin/launch_myprogram
#!/bin/sh
export TEMPDIR=/var/tmp
add sipb
exec $(dirname $0)/myprogram "$@"
```

## Source code

If you have source code for a program or programs developed here at MIT or elsewhere, we recommend that you install it in a subdirectory called `src`.

## Removing binaries for obsolete platforms

With the passage of time, some platforms become obsolete and are no longer supported in the Athena environment. As of 2014, the only Athena platform is Linux. Directories for Sun, SGI, and other platforms can safely be removed.

## Dealing with operating system upgrades

Because the sysname changes with every release, you must make sure you test your locker after operating system upgrades, particularly if you rely on it for classwork or other everyday use.
We recommend adding yourself to the release-announce mailing list to receive announcements of upgrades, new sysnames, and other relevant information in timely fashion. This list is very low traffic, and receives only 2-3 e-mails per academic year.

Whenever an operating system is upgraded, you should make sure that all your binaries run correctly. At the very least, this requires that you create a new link in your arch subdirectory for the new operating system version and run each program on a workstation running the new release. At the most, it will require recompiling software for which you have sources or possibly getting a new version from the vendor.

- If all of your binaries work correctly under the new operating system, you can just make the new directory a symbolic link to the old one.

```
$ cd /mit/mylocker/arch
$ ln -s amd64_ubuntu1204 amd64_ubuntu1404
```

However, this may prevent you from installing new software specifically for the new release without also breaking the old release. Instead, you are strongly encouraged to create a new binary directory, and create symlinks for each piece of software that is known to work in the new release.

```
$ cd /mit/mylocker/arch
$ mkdir -p amd64_ubuntu1404/bin
$ cd amd64_ubunt1404/bin
$ ln -s ../../amd64_ubuntu1204/bin/program1 program1
$ ln -s ../../amd64_ubuntu1204/bin/program1 program1
```

# Technical Information

## AFS

### About AFS

For a thorough discussion of AFS, see Athena publication AFS on Athena. This section covers only those topics applicable to course lockers and leaves out many details.

AFS is organized by units called cells. Within each cell, quota is allocated in terms of volumes. Most of the entities we call lockers consist of a single volume, though it is possible to have multi-volume lockers. Volumes are mounted in the AFS file system hierarchy which begins at /afs. Mount points appear to be directories in the afs hierarchy. Everything in the Athena cell is located under /afs/athena.mit.edu (/afs/athena for short), and all course volumes are under /afs/athena/course. Thus, you do not have to attach a locker to gain access to its contents, though we strongly recommend this for many reasons.

AFS supports different types of volumes: read-write volumes, read-only volumes, and backup volumes. The directory OldFiles in your home directory or course locker is simply a mount point to the backup volume for the volume comprising the locker. (At least, the volume if the locker consists of a single volume.) AFS manages quotas on a per-volume basis rather than a per-user basis, so anyone with write access to a volume can use up its quota allocation.

### Groups and AFS

AFS does not use Athena groups directly but rather maintains its own database of groups and group members. When you create a new group or make changes to an existing group, there is an automatic propagation of the changes into the corresponding AFS group. AFS groups are used to set access control lists and are identified by a system: prefix. Examples of AFS groups are system:facdev, system:2.14, and system:anyuser. system:anyuser is a special group that corresponds to anyone using AFS whether or not he or she has an MIT user id, and system:authuser is anyone who has authenticated themselves locally in the Athena cell. This corresponds roughly to anyone on Athena.

### The @sys link

AFS supports a special string which can be used in path names and dynamically expands to a special value according to the machine type. This @sys string is sometimes used as a user convenience link in configuring binaries or libraries. For example, you can make the file path /mit/locker/bin evaluate to the correct binary directory for each supported platform. In other words, /mit/locker/bin can point to the correct SGI binary directory if the user is on an SGI, or the Sun binary directory if he/she is using a Sun and so on. The command to set up this link is shown in Example 15.

```
athena% cd /mit/locker
athena% ln -s arch/@sys/bin bin
```

This will only work if you configure the locker according to the conventions outlined in this document in Section 3.3 , and should only be used for convenience. The **add** command will not find a directory named simply bin! You must follow the conventions outlined in Section 4.2

**Note**: the string "@sys" should never be used literally, except in making symlinks as above. See Section 4.2.3 on the **athdir** command for references in scripts, makefiles, etc.

## 4.2.2 $ATHENA_SYS and $ATHENA_SYS_COMPAT

The environment variable $ATHENA_SYS is set in the system-wide dotfiles at login time; it is used by commands such as **add** and **athrun** to

locate the appropriate files for the system on which the user is currently working. The value of $ATHENA_SYS matches the AFS value for the @sys link and can also be determined according to the output of the command machtype -S:

```
athena% machtype -S
sun4m_53
```

When writing your own shell scripts, makefiles, etc., it is recommended that you use **athdir** to locate machine-dependent files (as described in the next section) rather than $ATHENA_SYS; **athdir** is more reliable in case of new operating system releases and changed locker conventions.

The environment variable $ATHENA_SYS_COMPAT was introduced to help lockers which have not yet been updated for a new operating system continue to function until the maintainers have updated their arch directory structure. The system-wide dotfiles set $ATHENA_SYS_COMPAT to a fallback list of @sys values which are known to be generally compatible with the current system; if the correct @sys value isn't supported by a locker, commands such as **add** and **athdir** will attempt to use a value from this fallback list instead. On releases prior to Athena 9.0, the user may see a message like this:

```
athena% add badlocker
add: warning: using compatibility for /mit/badlocker
```

**Note**: Compatibility is intended as a mechanism to keep lockers working temporarily; it is not to be relied on as a substitute for updating binary directories when operating systems are upgraded. See Section 3.3.5 Dealing with Operating Systems Upgrades for recommended procedures.

### 4.2.3 athdir

Along with the environment variable $ATHENA_SYS and the new locker conventions, Athena introduced the **athdir** command. **athdir** can be used to find any type of machine dependent directory in a locker. One common use of **athdir** is to locate the machine-specific binary directory in a locker as in Example 16. As this example shows, **athdir** recognizes both the old and new style directory configurations.

**athdir** can be used in Makefiles and startup scripts as shown in Example 17. **athdir** takes many options as described on its man page.

```
Example 16 The athdir command
-----------------------------
athena% attach frame; athdir /mit/frame
/mit/frame/arch/sun4m_53/bin
athena% attach sipb; athdir /mit/sipb
/mit/sipb/sun4bin
```

```
Example 17 athdir in a Makefile
-------------------------------
LOCKER = "/mit/25.578"
BINDIR = `athdir $LOCKER`
INCLUDES = `athdir $LOCKER include`
LIBS = `athdir $LOCKER lib`
```

### 4.2.4 Using a shared directory for scripts

If you have many startup scripts or other programs which are actually scripts and therefore can be shared between different machine types, you may want to install them into a shared directory. You will still need links in the machine-specific directories, but installing into a single location makes it easier to maintain and update the script. The recommended configuration for using a shared directory is

```
/mit/locker/arch/share/bin
```

For each shared script you must create a soft link from the machine-specific binary directory to the shared installation location. Example 18 shows you how to do this for the sun4m_53 case (Solaris 2.3):

```
Example 18  Sharing a startup script
------------------------------------
athena% attach 25.678
athena% cd /mit/25.678
athena% mkdir -p arch/share/bin
athena% cd arch/share/bin
athena% emacs myscript &
create the script here
athena% chmod +x myscript
athena% cd ../../sun4m_53/bin
athena% ln -s /mit/25.678/arch/share/bin/myscript
```

## Course Locker Re-use Policy

Course lockers are re-used from semester to semester. Locker owners/maintainers are advised that any files they leave behind in a course locker at the end of the semester may be deleted or re-used by course staff who later inherit the locker. At the end of the semester you are responsible for saving any files that you wish to keep, and for deleting anything which you do not wish to share with subsequent course staff.

When an existing course locker is reused in later terms, we try to notify the people already on the ACLs as a courtesy, but it is up to new course staff to follow up with previous course staff regarding the disposition of any existing files. We are happy to assist you with archiving existing files in an inherited locker, and clearing out or archiving your own files at the end of the term.

## FAQ

1. What does the error "add: warning: using compatibility for /mit/xxx" mean?

   This means that the **add** command can't locate files according to the current @sys value for your platform, and is attempting to use a fallback value supplied by $ATHENA_SYS_COMPAT. Most likely, there has been an operating system upgrade, and the locker needs to have its arch directory structure updated; see 3.3.5 Dealing with Operating Systems Upgrades for help.

2. Why doesn't add find my Solaris binaries?

   If the **add** command is returning an error such as "Command not found", your locker is probably not configured correctly. Under Athena 9.0, all Solaris binaries should be put into a directory named according to the convention /mit/locker/arch/sun4x_58/bin.

3. My Linux binaries worked on the old SIPB Linux-Athena, but not on newer versions; why?

   The IS Linux-Athena ports began with Red Hat 6.2 (Summer 2000); Red Hat's backward compatibility is not as good as we're used to on other platforms, so you may have to recompile programs built under Red Hat 4.x. Specifically:
   - Curses-using programs must be recompiled or they will try to look in /usr/lib/terminfo, which does not exist in Red Hat 6.x.
   - Programs which print dates must be recompiled or they may try to look in /usr/lib/zoneinfo, which does not exist in in Red Hat 6.x.

4. Where should I put executables for platform <X>?

   If you are setting up a new locker, the best thing is to follow the new conventions for naming binary directories; see the appropriate table

5. Where should I put my web pages?

   New course lockers are created with a directory named www at the top level; place your web pages in this directory and they will be world-readable.

6. What's the URL for the locker's web page?

   The URL will follow the naming scheme:
   `http://web.mit.edu/locker/www/`
   if you name your top level page index.html and place it in the www directory of your locker. (See also the Academic Web Page Creation Guide.)

7. Is there any way to restrict access to my web pages to MIT?

   Yes, there is. For more information, see the Protecting Content

8. How can I get more quota for my locker?

   First make sure that you have removed any files and directories which are no longer needed. Athena does not have unlimited resources! Once you have done this, send an email request to accounts@mit.edu stating what locker needs the quota increase, how much more you

need, and why you need it. We will listen to each request, but because of resource limitations we cannot grant every one.

9. Last year the TA stored some files in his home directory. He left MIT; how can I get those files back?

   If you are fortunate, the TA's directory is still online. You can find out if this is the case by typing the command
   `hesinfo username filsys`
   If this command does not return an error, the directory is still online. In this case, send email to accounts@mit.edu, telling us what files you need to access.
   If the TA's directory has been deleted, we will need to restore from tape. To do this, send email to accounts@mit.edu giving us the TA's username and files you need to recover.

10. Last year students used a program in my course locker. Now suddenly it says "Command not found" when I use **add**. What happened?

    Are you using the recommended configuration for binary directories? There was probably a new Athena release and an operating system upgrade. You may be able to fix the problem simply by creating a symbolic link, but contact the Faculty Liaisons for more information. You may also want to read Section 4.2 The New Binary Directory Convention and Section 3.3.5 Dealing with Operating Systems Upgrades.

11. How can I find out what files recently changed in my locker?

    The UNIX find command is useful for finding recently modified files. Go to the top level of your locker and type the command
    `find . -mtime -<n> -print`
    where <n> is the number of days since the files changed.

    You can also get a good idea of what changed recently by using the **ls -ltr** command:
    `ls -ltr | tail -20`
    **ls -ltr** lists the files and directories in reverse order according to modification date. By piping this command into **tail**, it will list only the last 20 modified files or directories.

Back to top