An Inessential Guide to Athena [*]

(*Revision* : 1.32)

The Student Information Processing Board

January 11, 2011

Contents

Purpose and Conventions

The Student Information Processing Board (SIPB) compiled this document to provide a catalog of the less-publicized commands and capabilities of the Athena system for people who have learned the basics of Unix. If you are a complete newcomer to the system, we suggest that you begin with the Athena documents. These documents are available at Copy Tech in the basement of building 11 or the Student Center.

File names throughout this document are indicated in *italics*.

Program names, commands to be typed to the shell, and text that is returned by a command are in `typewriter text`.

Document names are in `typewriter text`.

Commands for GNU Emacs and other interactive programs are in **boldface**.

A C- prefix indicates Control character. For instance, C-x means that you hold down the Control key and hit the "x" key. A M- prefix indicates a Meta character, so for M-a you would hold down the "Alt" (or "Compose Character" on some keyboards) key and type "a".

If you have any questions, please feel free to ask us by coming to the SIPB office (W20-575), emailing us (*sipb@mit.edu*), calling us (253-7788) or zephyring us (class "sipb", instance "iathena").

SIPB and `/mit/sipb`: who? what? when? where? why? how?

SIPB (pronounced "Sip-bee") is the Student Information Processing Board, a student group concerned with computing at MIT. We administer several machines and an AFS cell, maintain a large body of software on Athena for your convenience, provide Usenet access to Athena, are available for telephone (253-7788) or in-person (W20–557) consulting at almost any time of day or night, have one-of-a-kind meetings Monday evenings at 7:30 pm, write documentation (like this guide), hack, and generally have a good time. We also act as an advocate for student computer users and student computer access on campus.

SIPB maintains a locker, `/mit/sipb`. To access the programs and files in it, type `add sipb` at the Athena prompt. This locker is the place where SIPB-written or modified programs and documentation can be found. The source code for most of the programs in `/mit/sipb` can be found in the `/mit/sipb/src` subdirectory. We hope that people will find the programs useful and the source code instructive.

Athena provides Intel stations running Linux for use by the community. All SIPB programs are compiled for use on these platforms.

Another locker which SIPB offers is `/mit/outland`. It is *not* supported, and while it contains many nifty programs, you should not depend on any of them, as they may not necessarily be available or completely debugged. We encourage you to look at it despite this, since it contains many neat programs that SIPB doesn't have the resources to officially support.

There are many other `add`able lockers with interesting and useful programs in them; it is not possible to catalog all of them here. We can only suggest you ask your friends who might know where the interesting "hacks" are stored, or ask us if you have specific questions. However, there is a short list of some useful lockers presented later in this document.

Problems or questions with SIPB programs should be sent to `bug-sipb@mit.edu`, except for problems with Usenet news programs, which should be sent to `usenet@mit.edu`.

Reading Mail and Bulletin Boards On Athena

New Athena users since 2009 use Outlook Web App for email. Documentation is provided by IS&T at `http://ist.mit.edu/services/email`.

Evolution and Pine

Ximian Evolution is a personal productivity suite available on all Athena workstations. It is also the officially supported mail reader for Athena. It can read the emails you have stored on your server and in your MH folders, filter your email and offers a personal calendar, task manager, and contact manager. To start it just click on the mail icon in your GNOME taskbar.

From `http://www.washington.edu/pine/`:

> Pine - a **P**rogram for **I**nternet **N**ews & **E**mail - is a tool for reading, sending, and managing electronic messages. Pine was developed by Computing & Communications at the University of Washington. Though originally designed for inexperienced email users, Pine has evolved to support many advanced features, and an ever-growing number of configuration and personal-preference options.

Pine is an easy to use mail client with comprehensive online help. Many of the keystokes used in the editor are similiar to GNU Emacs (C-c cancels, C-x sends, C-v pages down). In any screen, the most commonly-used commands appear at the bottom of the screen. For other commands at any time, press `o`. In any screen (except the editor) the **?** key will bring up a context-sensitive help page. You can **e**xit this menu at any time.

Pine can filter your incoming mail into seperate mail folders. From the **m**ain menu, enter **s**etup, select **r**ules, and select **f**ilters. Now, you can **a**dd a new filter.

Other Mail Readers

Mozilla's Thunderbird mail reader takes a bit of work to configure properly for your Athena account. To use it you will need to open the Mail & Newsgroups window and follow the dialog to fill in your account information. You will need to have your Mozilla certificates already working when you first try to read your mail, and you will need to know the name of your IMAP server, which you can find out by typing the command `chpobox`.

Gnus is an emacs-based mail and news reader. It is best not used for reading mail, but it is highly useful as a news reader. To start it, open an emacs window and type `M-x gnus`. One other client available is `mutt`.

The Discuss Bulletin Board System

Athena has a legacy networked conferencing system called Discuss that is used by many people in the community to, well, discuss things. This system is also used to archive many of the email lists used by the MIT community, so you will likely find yourself needing to have a look at the system. Several Discuss clients exist, including the shell client `discuss` and the graphical client `xdsc`, and the archives can also be seen from a Web gateway.[1] For more information on Discuss, there is the SIPB-maintained document Using Discuss[2].

Usenet

Usenet (Users' Network) is a bulletin board shared among many computer systems around the world. Usenet is a logical network, sitting on top of several physical networks, including UUCP, BITNET, and the Internet. Sites on Usenet include many universities, private companies, and research organizations. Usenet predates the Web and has its own protocol (NNTP), its own set of reading and posting software packages, and most importantly, its own conventions and organization. SIPB maintains the Usenet server `news.mit.edu` and several programs through which people can connect to it. These include `trn`, `xrn`, and Gnus (see above). Another way to reach Usenet is to visit the Google Groups site, which provides a Web gateway to Usenet. For more information, see SIPB's guide to Usenet.

[1]`http://www.diswww.mit.edu`

[2]`http://www.mit.edu/afs/sipb/project/www/discuss/discuss.html`

Instant Messaging at MIT: Zephyr et al

What is Zephyr?

Zephyr is an instant messaging environment used around MIT and several other campuses. It is older than most of the incoming class of 2006. Zephyr is turned on for you by default. When a message arrives for you, a window is created in the upper-left corner of your screen(by default), and the message is displayed there. Once you have read the message, you move the mouse cursor into the window and click on any button, and (poof!) the window goes away. Additional information about Zephyr can be found in `http://sipb.mit.edu/doc/zephyr/`.

Several Zephyr clients exist. The one turned on by default is the Zephyr windowgram client, `zwgc.` The most recent version of `pidgin` also handles Zephyr accounts. Most people use `barnowl`, a monolithic client in the `barnowl` locker.

How can I tell if someone's logged in?

There are two ways to find this out. If you just want to know where Jack Florey is now, you can type

```
% zlocate jflorey
```

and Zephyr will reply with his current location (e.g., `M11-116-1`), if he's logged in and not hidden, along with the date and time he logged in there.

If you want to be told whenever a friend logs in or out, you can use `znol`, the "Zephyr Notify On Login" program. You simply create a file in your account called `.anyone`, and put your friends' names in it, one to a line, like this:

```
jtkirk
spock
lmccoy
uhura
jrand
```

Then, each time you log in, type `znol`. (You can add this to your `.startup.X` file, too.) You'll get a list of everyone in your `.anyone` file who's currently logged in; then, whenever someone logs in or out, you'll get a Zephyr notice. (A GUI version of `znol` called `xzul` exists in the `outland` locker.)

Actually, that's not quite true. The person has to have told Zephyr, "Whenever I log in, announce my arrival." This is done by typing

```
% zctl set exposure net-announced
```

once. (It is automatic after that.) There are several other choices besides `net-announced`. The first of these choices is `none`. This disables Zephyr entirely. Next is `opstaff`, which allows you to receive important/emergency messages about the status of Project Athena. Another, `realm-visible`, will let people `zlocate` you, but it will not send messages when you log in and out. The default is `realm-visible`.

***AFS*: The Andrew Filesystem**

Any particular method for handling files of data is called a *filesystem.* Athena currently uses *AFS*, the *Andrew Filesystem*[3], as the filesystem for all user home directories and most of the other lockers.

AFS is a *distributed* filesystem. This means that the files stored using *AFS* are located on a remote server, so you can access them from any Athena workstation. In fact, the owner can make the files available from any computer running *AFS* in the whole world.

You don't have to attach *AFS* filesystems[4]; they are always available from the workstation under the directory */afs.* Another major difference between *AFS* and other distributed filesystems is that groups of *AFS* files can be replicated onto multiple servers in such a way that makes the failure of one server unnoticeable to a user. In addition, files and directory structures are cached locally to minimize network traffic to the server. This makes access to *AFS* files very fast after the first time a file is read.

***AFS* Volume? Why do I care how loud my files are?**

No, no, no; that's not what *AFS* volume means. In order to understand *AFS*, it is important to know some basic terminology. Following is a list of terms to provide you with some important background for when you use *AFS*:

Volume: An *AFS* volume is a collection of files and directories that are grouped together as one unit. A volume can be backed up, renamed, moved from one server to another, replicated, created, or destroyed as a unit.

User: A user is someone who is registered with *AFS*. You must be registered with *AFS* if you wish to be able to own files. The process of telling *AFS* who are you is called *authentication.*

Group: A group is a list of users. Currently, it is not possible for a group to contain other groups.

Cell: An *AFS* cell is essentially a domain of authority. It is currently the largest unit used in *AFS*. Each cell has a list of administrators, a list of users, a list of groups, and a list of volumes. Each of these lists is private to the cell. This means that an administrator in one cell may or may not have any access to another cell. A user in one cell may not even exist in another. A cell is also a domain of authentication. If you are the user *qjb* in the *athena.mit.edu* cell, you may not have any access at all to files belonging to a user named *qjb* in the *andrew.cmu.edu* cell. Likewise, someone who has administrative capabilities in the *sipb.mit.edu* cell may not be able to touch the *athena.mit.edu* cell.

ACL: ACL stands for Access Control List. When referring to access control lists, some people say the initials ("A-C-L") and others pronounce the acronym ("ackle"). An access control

[3]*AFS* was developed at Carnegie Mellon's *Project Andrew*, which is a large-scale distributed workstation experiment similar to *Project Athena*, the predecessor of today's *Athena.* It has since become a commercial package sold by *Transarc Corporation.*

[4]You might want to use the `attach` command for Athena file systems anyway – this is discussed later.

list pairs users and groups with operations that they are allowed (or not allowed) to perform. Although access control lists are used for much more than just *AFS*, only *AFS* access control lists will be discussed here. A typical access control list would say something like, "The user *potato* is allowed to create files in this directory. The user *soup* is allowed to modify existing files and create new files in this directory. Anyone else is allowed only to read files in this directory." *AFS* access control lists and how to use them are explained in more detail later in this document.

How do I use *AFS*?

In general, you won't even notice whether files you are using come from *AFS* or not. The time when it really does make a difference, however, is when you want to change file protections, and this is critical since your Athena home directory is in *AFS* space.

If you wish to list or change protections, you will need to become somewhat familiar with the program `fs`. This program allows you to get various kinds of information about *AFS* files and directories and to change and look up protections on directories. Although it is not necessary for you to understand standard UNIX file protections in order to understand *AFS* file protections, it will probably help you if you do. Standard UNIX file protections are explained in the **File Protection** section of this document.

The `fs` command has a built-in help feature. Typing `fs help` will give you a lot of information on how to use various subcommands. All the commands that you will probably want to use, though, are described here.

Authenticating to *AFS*

Before *AFS* can know who you are, you have to be *authenticated* to the cell. Authentication is just the name given to confirming your identity. There are several ways to authenticate to *AFS*. The most common way will be through the `attach` command. If you attach a filesystem that is in *AFS*, you will be authenticated automatically to it. Should you have to reauthenticate for some reason, you can just reattach whatever it is that you are trying to use.

`attach` authenticates to *AFS* by using a program called `aklog`. Generally, typing `aklog` with no arguments will reauthenticate you to whatever you are using. To be safe, however, you can specify the path of the file you need to reach. For example, if you are working in the *potato* locker, you can type `aklog /mit/potato` to reauthenticate.

File protections under *AFS*

AFS offers considerably more flexibility in directory protections than regular UNIX. Unfortunately, though, protections currently can be set on files only in a limited way. Protections in *AFS* are enforced with access control lists instead of with bit fields as in regular UNIX. This means that in *AFS* it is possible, for example, to give one set of users read access to a directory, one set of users write access, and another set no access at all.

There are seven types of access that can be granted:

Read (r): Read access on a directory implies permission to read the contents of all the files in that directory. It has no implications about access to subdirectories.

Lookup (l): With lookup access on a directory, it is possible to look at the directory's access control list and to list the contents of the directory (*i.e.*, what files and directories are in it). It does not imply read access to the files. **If you do not have lookup access to a directory, no other form of access can be used.**

Insert (i): Insert access on a directory implies permission to create files or subdirectories in the directory. It does not imply the ability to modify the files that are created, however. Insert access without write access is useful mainly for the case when you want to allow someone to create subdirectories in a given directory but not to modify files that are already there.

Delete (d): Delete access on a directory implies the ability to remove files or empty subdirectories from the directory. Like insert, delete access does not imply write access.

Write (w): Write access on a directory grants permission to modify files and subdirectories within a directory. It implies neither insert nor delete access to the directory.

Lock (k): A user with lock access on a directory can put advisory locks on files within the directory. This is typically useful only to applications programmers.

Administer (a): With administer access on a directory, it is possible to change the access control list of the directory. Administer access does not imply any other kind of access, and like all other types, is useless without lookup access. Administer access, contrary to some people's intuition, cannot be "forced" onto subdirectories. It only applies to the directory on which it is set, not that directory's subdirectories.

In addition, the *user* portion of a files UNIX mode bits (described in the **File Protections** section of this document) apply to all users, not just the owner of a file. That means that if a file's *user* mode bits show the file to be unreadable, the file will not be readable by anyone regardless of the access control list set on its parent directory. Likewise, if the *user* mode bits are set to give *read*, *write*, and *execute* access, then access to the file is determined completely by the *AFS* access control list on the directory. The *group* and *other* bits as well as the *UNIX group* of the file are currently ignored in *AFS*[5]. If you didn't understand that, ignore it.

The owner of a volume has automatic administer access to all directories in the volume. The owner of a directory has automatic administer rights to that directory.

Access can be granted to individual users or to groups. There are two special groups: *system:anyuser* and *system:authuser*. *system:anyuser* means all users whether or not they are known

[5]Well, actually, the *UNIX group* field and *group* mode bits are used in a limited way for *setgid* executables, but this is beyond the scope of this document.

to the cell. *system:authuser* means any user who is registered with the cell and has been authenticated (*i.e.*, anyone who has told *AFS* who they are).

Other groups can be one of two types: *system* groups or *user* groups. System groups have names that start with *system:*. They are controlled by the system administrators. User groups start with <*user*>*:*, where <*user*> is the name of the user who created the group. It is possible for any user to create his or her own groups in *AFS*. Information about how to do this appears below.

An access control list on a directory contains a list of users and groups and the operations they are allowed to perform. To find out what the access control list on a directory is, you can use the command `fs listacl <directory>` where <*directory*> is the name of the directory you are interested in. Note that `fs la` is short for `fs listacl`.

For example:

```
athena% attach sipb
sipb: AFS mounted /afs/sipb.mit.edu/project/sipb on /mit/sipb (read-only)
athena% fs la /mit/sipb
Access list for /mit/sipb is
Normal rights:
  system:gsipbbin rlidwka
  system:anyuser rl
```

This means that members of the system-controlled group *system:gsipbbin* have full access to */mit/sipb* and all other users have read and lookup access only.

Here is another example:

```
athena% attach qjb
qjb: AFS mounted /afs/sipb.mit.edu/user/qjb on /mit/qjb (read-write)
athena% fs la /mit/qjb
Access list for /mit/qjb is
Normal rights:
  qjb:plorq rla
  qjb:all rlidwka
  system:anyuser rl
```

This means that the user-controlled group *qjb:all* has all access to */mit/qjb*, the user-controlled group *qjb:plorq* has read, lookup, and administer access to */mit/qjb*, and everyone else has read and lookup access only.

Where to go for more

The SIPB also publishes a guide called **Inessential AFS**. This document contains everything you see here and some additional information about changing access control list, quotas, and other *AFS* programs. Good luck!

Lost Contact With File Server

At some point in your dealings with Athena, you will almost certainly encounter a message saying something like:

```
afs:  Lost contact with file server 18.70.0.210 in cell athena.mit.edu
```

(The number may be different.) This is disconcerting if you've never seen it before. But DON'T PANIC.

What This Means To You

(The number may be different.) This means that your workstation has lost contact with a *file server* that your workstation is using. You may be unable to use some programs (depending on what file server you've lost, this can be anything from your background to *idraw* to *ls*); you may be unable to save or retrieve your files. If you've gotten this message, can't save your files, and only want to know how to save and/or print your files, you can skip to the end. If you want to know why it's happening and what you can do, read the next section first.

What Did All That Mean?

Files in the Athena system are kept in *lockers* on *file servers*. A locker is basically a chunk of space on a disk somewhere, generally in building E40 or W20 or 11; a file server is a workstation, exactly like the workstations you'll log in on, but with rather more disk space. If a file server goes down – for maintainance, or there's a power outage, or terrorists take over building E40 and refuse to turn the machines on until President Vest sings the theme to Gilligan's Island – you will not be able to access the files on that file server until it comes back up. Shortly after a servers comes back up you will see a message saying this:

```
afs:  file server 18.70.0.210 in cell athena.mit.edu is back up
```

If you have attached a locker (the sipb locker, for instance, or the graphics locker) and the file server that contains that locker goes down, you will get a "lost contact" message in your console window. Some lockers are spread out over more than one file server so that if one goes down you will still be able to access the information. For more information about this, you can look in the *AFS* section of *Inessential Athena*.

So What Did I Lose?

The first thing you want to do will probably be to find out which locker you've just lost. Suppose you have just received the message:

```
afs:  Lost contact with file server 18.70.0.210 in cell athena.mit.edu
```

The number 18.70.0.210 is the internet address of the machine; the command `hostinfo` will tell you what the name of the machine is. So typing `hostinfo 18.70.0.210` will give you this:

```
Desired host: 18.70.0.210
Official name: ROSEBUD.MIT.EDU
Host address: 18.70.0.210
```

This tells you that the name of the machine is *rosebud*. If you want to know whether a particular locker is on that server, you can use the command `lookup` (in the consult locker). Thus `lookup sorokin` will return:

```
Filesystem sorokin (AFS) is located on:

    ROSEMARY.MIT.EDU (read-write)
```

which says that sorokin's homedir is still accessible. `lookup sipb` will return:

```
Filesystem (AFS) is located on:

    ROSEBUD.MIT.EDU (read-write)
    ROSEBUD.MIT.EDU (read-only)
    RONALD-ANN.MIT.EDU (read-only, alternate)
```

This tells you that sipb is located on rosebud. Handily, there is another file server containing the same data (ronald-ann), and so you probably shouldn't be worried. However, you may want to look at *all* the filesystems you have attached. The command for this (also in the consult locker) is `wherefs`. It will give you something like this:

```
sorokin : AFS filesystem on host ROSEMARY.MIT.EDU
sipb : AFS filesystem on host ROSEBUD.MIT.EDU
gnu : AFS filesystem on host ARTEMIS.MIT.EDU
consult : AFS filesystem on host CHICKADEE.MIT.EDU
graphics : AFS filesystem on host KITE.MIT.EDU
```

There may be lockers listed that you don't recognize. The reason for this is that when you attach a locker, it is connected to the workstation. Therefore, a locker attached by the previous user may show up in your listing. You can ignore it.

So What Should I Worry About?

First of all, servers will often come back up fairly quickly. If you have the time and something to do, it may be worthwhile to just wait.

If the locker on the lost server is not one you're using, or is replicated elsewhere, you can pretty much ignore the message.

If you need a program on that server, you can try to find out how long the server will be down or whether that program exists anywhere else. (Try asking on instance *help*; see the section on zephyr for details on instances.) There may be nothing you can do but go home.

But I Can't Save My Files!

First of all, don't assume that not being able to save your files means that you've lost your fileserver. Make sure that this *is* the problem.

If it is the case, you can still save your files. You just need to save them somewhere else. In emacs, the command *C-x C-w* will present you with a pathname, and will ask you to type in the name of the file you want to save to. Delete all of the pathname that it gives you, and type `/usr/tmp/`*filename*. In EZ, simply choose the *Save As* option. You have now saved your file in a temporary file on the hard drive of your workstation. This file may hang around for up to three days. You can mail it to yourself, with the command

```
/bin/mail username < filename
```

DON'T INCORPORATE IT NOW. WAIT UNTIL YOU HAVE YOUR HOME DIRECTORY BACK.

Now, disaster averted, you can wait for the fileserver to come back up, or you can come back later. While the server is down, you will not be able to access any of your files.

Useful Lockers

The following is a partial list of useful lockers. To access them, type `add` ***`filesystem`***. Many of them have a file named README in their top-level directory, which will have more information about the filesystem, its layout, and whom to contact for more information. The What Runs Where page[6] also lists approximately all of the useful software available for Athena.

gnu contains a wide array of software written and distributed by the Free Software Foundation,[7] such as gcc, a C/C++/Objective C compiler; gdb, a source-level debugger; gnuchess, a chess program; bison, a replacement for YACC. These programs are available in varying degrees for various platforms. Send any problems or questions to *gnu@mit.edu*.

consult contains software written and/or maintained by Athena's On Line Consultants. The *consult* locker contains programs for the current Athena platforms. Read */mit/consult/README* for instructions to use or learn about software contained in *consult*. Send any problems or questions to *bug-consult@mit.edu*.

games is a random collection of games and puzzles. Currently, the games locker supports games for Linux, Solaris, and SGI platforms to varying degrees, in addition to games to be run within Emacs. The locker also contains source code for most of the games that are installed. Look at the */mit/games/POLICY* file in the games locker for rules about submissions. Send any problems or questions to *bug-games@mit.edu*.

info contains up-to-date information about the status of Athena software and hardware, such as Athena policies. Comments should be sent to *bugs@mit.edu*.

watchmaker is maintained by the so-called "watchmakers,"[8] a group of student systems programmers at Athena. **BEWARE!** Most of the programs in the locker are esoteric and only useful to experienced systems hackers, but some of the programs are of general interest. Report any problems or bugs to *watchmakers@mit.edu*.

graphics The graphics locker contains various graphics-related programs useful for viewing and manipulating graphics. See the **Graphics** section of this guide for more information.

postscript is a collection of interesting and useful PostScript programs and utilities. See the *discuss* meeting on postscript_hacks for information about what is currently stored in the filesystem[9]. PostScript is a page description language developed by Adobe Systems Inc. and is available on all of Athena's printers. Problems should be sent to *postscript@mit.edu*.

rfc contains all the current Internet Engineering Task Force (IETF) Requests For Comment (RFC's). These server are documents on "standards". It also has an index, */mit/rfc/rfc-index.txt*, which you should definitely read, because the proposals are listed only by number.

[6]`http://web.mit.edu/acs/www/whereruns.html`

[7]`http://www.gnu.org`

[8]Read *The Mote in God's Eye* by Larry Niven and Jerry Pournelle for an explanation of this term.

[9]Discuss is a conferencing system; more details are in the **Discuss** section or in the **Using Discuss** document, distributed by the SIPB

dotfiles The dotfiles locker contains a number of extensively modified dotfiles. Many of these may be useful, and there are a number of neat hacks that you can play with; but be *sure* you know what you are doing before you hack your own dotfiles. Read the README file in */mit/dotfiles/README* for an explanation of how the locker is set up.

Some other neat lockers are *shakespeare, bible* and *weather*. They're just about exactly what you think they are. Look around in them; ask your friends for their suggestions; see what you can find on your own.

Kerberos: Athena's Watchdog

Kerberos; also spelled *Cerberus. "n. The watch dog of Hades, whose duty it was to guard the entrance–against whom or what does not clearly appear;...is known to have had three heads..."*
-Ambrose Bierce, *The Enlarged Devil's Dictionary*

Kerberos is the system designed by MIT to provide security on the Athena network. This document will not provide you with a complete description of the mechanisms and the math used by *Kerberos*; it will explain the necessity for *Kerberos*, and will quickly summarize what *Kerberos* does. For more information, you can look at the Kerberos section of the Athena Technical Plan, the PostScript file */mit/kerberos/doc/techplan/techplan.PS.*

Why *Kerberos* is needed. *Kerberos* has two purposes: security and authentication. On most computer systems, a password is used to prove a user's identity; on a distributed network system, like Athena, this password must be transmitted over the network, from the workstation being used, to any other machines containing files or programs the user wants access to. Because this password is the one secret piece of information that identifies a user, anyone knowing a user's password can effectively *be* that user on Athena, reading their files, sending mail as that user, etc. **Please note that THE ELECTRONIC COMMUNICATION PRIVACY ACT of 1988 does make this a Federal crime** punishable by all kinds of nasty stuff you don't want to be punished by. No kidding.

Obviously, it is therefore necessary to prevent anyone from intercepting or eavesdropping on the transmitted password. In addition, it is necessary to provide a means of *authenticating* users: any time a user requests a *service,* such as mail, they must prove their identity. This is done with *Kerberos*, and this is why you get your mail and no one else's.

A Few Handy Definitions - We have now used two pieces of jargon in one paragraph, and at this point you will probably want to learn the meanings of a few terms that will be used in this section.

user: A person using a computer system. A user, through her workstation, may make a series of requests to several servers. This user, we assume, would like to avoid retyping her password every time she makes such a request.

service: Very simply, a service is a program or set of programs, running on a computer which is accessible over the network. A user will request a service for the workstation which she is using; the service will want to be sure that the service is really being used by that user.

principal: A principal is some entity which can prove its own identity and verify the identity of other principals. Each **user** and each **service** registered with *Kerberos* is a principal, since *Kerberos* provides the authentication services required.

ticket: Once a user has proved her identity to *Kerberos* with her password, *Kerberos* sends a block of encoded data, called a ticket, to the user. It is this ticket that is used to prove a user's identity to a service. Tickets are stored in the */tmp/* directory and are erased upon logout. Tickets will expire after 10 hours by default, though they can be set to live longer than that.

authenticator: When a user tries to use a *service*, her workstation sends an block of data called an *authenticator*, built from the *Kerberos* ticket and containing a timestamp and the name of the workstation, to that service. The service decodes it, verifies that the user is who she says she is, and then lets that user, at that workstation, use the service.

This is the basics of how *Kerberos* works – for the most part, you'll see none of it, which is a nice feature. For a longer and more technical explantion, you can look in:

```
/mit/kerberos/doc/techplan
```

Once registered with *Kerberos*, tickets are obtained by the login program every time you log into a workstation. You can also manually obtain new tickets (which you usually do only if your old ones have expired, 10 hours after you log in) by running the program `renew`. You can also use `kinit`, which prompts for a username, requests an initial ticket from *Kerberos*, and then asks for your password. If you are not registered with *Kerberos*, it will print `Principal unknown (Kerberos)`. Unless you mistype your username, this should not happen. To correct this, or any other errors, contact a Consultant or the Athena Accounts Administrator by using `olc`.

Graphics on Athena

Quick Work

There will be times when you will need to make a quick image file for some purpose: to make a poster, decorate a web page, make a graphic for a course related paper, or other purposes. There are several tools on Athena that will help you do that. Most reside in the *graphics* locker. If a tool is mentioned and its locker is not named, presume that it lives in the *graphics*.

For a quick and dirty drawing, you have your choice of `xpaint`, `gimp`, as well as the Applix and Open Office suites, both of which have drawing components. All these are available from your Athena menu bar. For a more schematic drawing, you can use `xfig`, `tgif`, in the `sipb` locker, `dia`, in its own `dia` locker, or `xcircuit` in the `xcircuit` locker.

To edit an image file for cropping, resizing, color corrections, or other quick tasks, there is the `xv` utility in the `graphics` locker. Just type `add graphics; xv filename` and when the window showing the image pops up, right-click on it to see what xv can do for you.

To grab a screenshot in the Athena GNOME environment, jsut press the PrintScreen button. A dialog will show up to let you save your screen in the PNG format. You can also use the utilities `xdpr`, and `xv` for the same purpose, as well as the GIMP.

To make file conversions at the command line there is a whole slew of utilities in the `graphics` locker. If the `convert` utility (on the Suns it is known as `imconvert`) can't do it for you, then just type the suffix of your input graphics file name (for example, `athena% png`) and then the tab key, and a whole list of utilities should pop up. For more information, type `add graphics; man convert` and `man mogrify`.

To make animated image files, the `graphics` locker containes two very useful utilities, `gifsicle` and `animate`. Both have manual pages with instructions for using them.

The GIMP

The GIMP is your Serious image editing tool on Athena. It slices. It dices. Just type `gimp &` and have at it. Tutorials and guides for using it are found at `http://www.gimp.org`.

Moira: Athena glue-all, or Your Fate

As you may have gathered by this time, you, as a user of Project Athena, are aided by a fairly large number of services. You are served by *NFS* and *AFS* which bring you your files, by *Kerberos* which allows you to convince Athena that you are who you say you are, by *Hesiod* which allows you to look up information about many things, by *Zephyr* which allows you to locate users and communicate with them electronically, by the post offices which allow you to send mail – the list goes on. It would be virtually impossible for a single person to keep all of these services under control and consistent with each other. This is why we have *Moira*, the Service Management System. *Moira* is an alternate spelling for *moera*, which means fate. The three aspects of fate were Clotho, who spun the thread of life; Lachesis, who wove the thread into the tapestry of life setting a person's destiny; and Atropos, who cut the threads at the appropriate time. Similarly, *Moira*, the system, creates user accounts, controls the user's home placement and quota, and deactivates user accounts[10].

Moira is a very large database containing most of the information that is necessary to keep Athena running smoothly. Although *Moira* is not needed for someone to log in and use the system (unlike *Kerberos*, which must be available all the time), without it, nobody would be able to change anything about any service. This means you couldn't add anyone to a mailing list, create a new filesystem, or any number of similar operations.

Even though *Moira* controls a vast array of things and can perform several tasks, there are only two things that you will probably ever need to use *Moira* for: changing or inspecting lists and updating finger information. Both of these are explained in this section.

Things Not To Do

It is possible for you to put yourself into a situation that you won't be able to get out of. For example, you can change the administrator of a list you own to someone else. If you do this, **you will not be able to reclaim the list without help from an administrator**. When you make a change, especially a transfer of authority, think about whether you'll be able to undo it. If you really get messed up, you can call the user accounts consultant at x3-1325.

How to manage lists

To *Moira*, a list is simply, well, a list. The items on an *Moira* list are called *members*. Each member of an *Moira* list has two attributes: a type and a name. A *Moira* list can have several types of members. The only ones useful to most users are users, other lists, and arbitrary strings such as electronic mail addresses.

Lists can serve several functions. The two most common are UNIX groups and mailing lists. Your list is a *Unix AFS group*. You use it to control file access. For more information about AFS

[10]Until the beginning of 1989, *Moira* was known as *SMS*, which stands for *Service Management System.* Since so many other things were called *SMS*, and since almost everything else at Athena had a greek name, it was decided that the Service Management System should be renamed.

groups, look at the manual pages for the AFS `fs` utility, and `chmod` (change mode) by typing `add afsuser; man fs` and `man chmod`. You can also find information in this guide in SIPB's *Inessential AFS* guide and in the OLC answers repository (`olc answers`).

The main tool that you will use to manipulate lists is `listmaint`. `Listmaint` is a fairly self-explanatory, menu-based program. To run `listmaint`, just type `listmaint`. Here is what the top-level menu of `listmaint` looks like:

```
athena% listmaint


                                   List Menu
 1. (show)         Display information about a list.
 2. (add)          Create new List.
 3. (update)       Update characteristics of a list.
 4. (delete)       Delete a List.
 5. (query_remove) Interactively remove an item from all lists.
 6. (members)      Member Menu - Change/Show Members of a List..
 7. (list_info)    List Info Menu.
 8. (quotas)       Quota Menu.
 9. (help)         Print Help.
 t. (toggle)       Toggle logging on and off.
 q. (quit)         Quit.
 t. (toggle)       Toggle logging on and off.
 q. (quit)         Quit.
Command:
```

To add someone to your list, select *members* from `listmaint`'s top-level menu. This will ask you for the name of a list and then put you in a submenu entitled `Change/Display membership of '`*list*`'`, where *list* is the list you specified. As you may expect, you will want to select *add* from this menu. You will be asked what type of member you wish to add. The answer to this question should be *user*. You will then be asked for the name of the user you wish to add. At this prompt, enter the user's username. You will be told whether the operation completed successfully.

Using `listmaint` is rather straight-forward. Once you get the hang of things, you will probably be able to figure out how to do mostly anything you want to.

You can also use `blanche` instead of listmaint. `blanche` can perform a number of the operations listmaint performs, and for those operations is much quicker. For a quick list of the functions `blanche` can perform, simply type `man blanche`. For more detailed information, read the manpages for all of the above.

Getting information through moira

In Unix, it is possible to `finger` a user to find out more information about him or her. To do this, just type `finger user`. You can change this information about yourself with the `chfn` command.

The use of the command is very straight forward. Just type `chfn`. You will then be told what your current information is and prompted for new information. The new information that you entered will be available to people who use `finger user` within a day or so.

You can also find `finger user@mitdir` to get the "official" information about a person; this service is maintained by Distributed Computing and Network Services and any changes of information must go through the Registrar's Office (for students) or Personnel (for staff). You cannot change this finger information yourself.

What are you waiting for?

One last point. Because of the way *Moira* works, the changes you make won't take effect right away. If you add someone to your list, *Moira* will know right away, but the rest of the world won't know for between 15 minutes and three hours or so. You may remember that your account wasn't ready until the day after you registered. In future versions of *Moira*, it may be possible to do this updating without delay.

But wait! There's more

There are more *Moira* programs than just `listmaint` and `chfn`. The program `moira` is the master *Moira* client. You could also have changed any of this information from the `moira` program, but it would have been slightly more complicated to do so.

To find more information on these other programs, refer to `moira` locker, which contains binaries and manual pages for the various moira clients. These programs are listed in the file README in */mit/moira*.

Accessing Athena Remotely

Why would I want to login to Athena remotely?

Presumably, you use computers outside of Athena clusters. You might want to take advantage of some of the features Athena offers at one of these computers, whether it be email, zephyr, files in your homedir or any number of things. There exist a number of applications which give you some of these features (you may, for example, check your email with Eudora, or use a zephyr client on your home machine), but sometimes you want to connect to Athena directly.

What is SSH?

SSH, or Secure Shell, is a protocol for establishing a secure connection to a machine elsewhere on a network. It is encrypted, which means that the data transmitted over the network cannot be read by anyone trying to pick up information (for example, passwords). SSH actually creates a window on your computer which is almost equivalent to an xterm on the remote computer; so connecting to an Athena machine with SSH gives you the equivalent of the window which pops up when you first start Athena. To understand why it woks like that, here is a bit of history. Back in the day, there were not that many computers; rather than every person in a cluster having their own computer, they had their own *terminal*. Each terminal was composed of a monitor and a keyboard, and each terminal gave a user a different means of interacting with the same computer. When you SSH to a server, the window you get is an emulation of such a hardware terminal; when using that window you might as well be sitting at that computer.

OK, so where should I SSH to? Athena offers a number of dialup servers; connecting to `athena.dialup.mit.edu` will redirect you to one of them. (They tend to be named after famous hacks pulled at the Institute, if you're wondering where the names come from.)

SSH clients If you are running some variant of linux, your SSH client is called `ssh`. You shouldn't have any trouble. If you are running Windows or MacOS, MIT offers at `http://web.mit.edu/software/` SSH clients; the recommended choices are SecureCRT for Window and NiftyTelnet SSH for MacOS. You need MIT certificates to download these clients.

Forwarding mail

You may have other email accounts that you check more often than your Athena account. If you'd like to check all your mail from a single account, you can set up forwarding. All you need to do is run the command `chpobox` to set up forwarding. Read the `man chpobox` manpage for more details.

Web interfaces

You may sometimes be in a situation in which you would want to either check mail or connect directly to Athena without installing additional software. It is possible to do both from a browser.

Webmail To check your email from a browser, you will need a JavaScript-enabled, SSL-supporting browser and a MIT CA certificate. From the browser, go to `http://webmail.mit.edu`. The browser will ask you about your certificate; click `Next` until you reach `Finish`. If you do not have a MIT CA certificate, go to `http://web.mit.edu/is/help/cert` to obtain one.

Browser dialup MindTerm is a Java applet that offers a quick dialup Athena connection. Although MindTerm may run on other browsers, it is recommended that you use Netscape 4.06 or later. To use MindTerm, simply enter `http://athena.dialup.mit.edu/ssh.html` into your browser, and wait for the applet to start.

Debathena installer

It is possible to convert your own machine to an Athena workstation. The SIPB has created an installer which can be run on most computers supported by Ubuntu or Debian, and supports dual booting. IS&T and SIPB are collaborating on the maintenance of this service. More information is available at `http://debathena.mit.edu/`.

Acknowledgments and other randomness

The first edition of this document was produced and orchestrated by Jean Marie Diaz. Varying amounts of comments, criticism, text, and sympathy were provided by Simson L. Garfinkel, Henry Holtzman, John Kohl, Robert Krawitz, Henry Mensch, Ken Raeburn, Bill Sommerfeld, and Stan Zanarotti. Thanks, guys!

Second edition thanks to Rob French for RCSifying, tidying up, and generally letting Jean ignore this thing over IAP 1987, and to Ike Chuang, who valiantly volunteered to take over the *Guide* after Rob punted. (You haven't gotten away with anything, Ike, you'll get volunteered for it next time. . .).

The third edition encompassed revisions and additions reflecting Athena's migration from time-sharing to single-user workstations. John Kohl coordinated the conversion and edited the third edition. Ken Raeburn and Bill Sommerfeld (and other anonymous SIPB members and friends) helped edit early drafts. André DeHon, Mark Eichin, Jon Kamens, John Kohl, and Carol Smith contributed new sections.

The fourth edition, involving only minor changes, was revised to take into account Project Athena's version 6.0C software release. Jon Kamens edited the fourth edition. Andrew Greene and Emanuel ("Jay") Berkenbilt contributed new sections.

The fifth edition, which includes information relevant to Athena's version 6.3B software release and new sections on Xpix, AFS, and Andrew, was edited by Rob French (thus redeeming himself for punting on revision two). Many thanks to Jay Berkenbilt for writing the AFS section and revising some of the others, Seth Finkelstein and Barry Jaspan for writing the Xpix section, and Bill Cattey for writing the Andrew section.

This document was created with GNU Emacs and LaTeX, printed using the Times Roman font family (hacked to use Courier-Bold for typewriter text). The source for the document is on the *sipb* filesystem in the directory `/afs/sipb/project/doc/guide`.