Fermilab

GU0008

FNALU User's Guide

Lisa Giachetti
Steve Hanson
Judy Nicholls

Computing Division
Fermi National Accelerator Laboratory

February 1997

Revision Record

August1993. First draft.
August, 1994 - Second Draft, First Full Release
Jan. 1995 - Third draft, Reflecting cluster hardware changes
Feb. 1995 - Fourth draft, some corrections and elucidations
May 1995 - Fifth draft, additional information, applications chapter added, hardware changes
November 1995 - Further additions, applications chapter.
February 1996 - Still more corrections, hardware changes, additional batch information
February 1997 - Replaced batch/LoadLeveler with batch/LSF.

Conventions:

bold	In command formats, indicates what you type as is. In text, the names of commands, variables, files, etc., to distinguish them from normal text.
italic	In command formats, indicates variables for which context-specific substitutions should be made.
`typewriter`	Used for examples, contents of files, output of commands.

Chapter 1

Introduction

FNALU, a computing resource managed by the Computing Division at Fermilab, provides UNIX computing services to the laboratory. It provides batch and interactive computing on IBM, Sun, and Silicon Graphics platforms. It also provides interactive computing on a Digital Equipment Alpha system running Digital UNIX.

1.1 Gaining Access

FNALU has two types of accounts. Grant Request Accounts are used by projects that will require significant computing resources, and require that the user request a computing grant by filing a Project Request Form. The Project Request form asks you to describe your project and estimate your CPU, disk, and tape requirements. As a result of knowing these in advance, we will be able to perform capacity planning based on them. Project request forms can be found in the FORMS folder of INFO on VAXes, from the FNAL.ANNOUNCE.FORMS newsgroup, or from the secretaries in the Computing Division.

Accounts are also available for users who will use minimal computing resources. Accounts which will require less than 50 Mbytes of disk space and will use less than 1% of the available CPU may request a non-grant account by filling out an account request form, which is also available in the FORMS folder of INFO on VAXes, from the fnal.announce.forms newsgroup, or from the secretaries in Computing DIvision.

1.2 Purpose

Although FNALU is a general-purpose platform, at this time we are not able to provide services such as text processing and other office automation functions, but focus on physics and engineering requirements. Future expansion plans provide for the installation of an ORACLE database server and clients, Text Processing software, and some other commercial software packages. Use of the machine should be consistent with the terms of the agreements, which will be reviewed annually.

The purposes of FNALU are primarily:

- Development machine for the Clustered Large UNIX Batch System (CLUBS)
- To support work groups in a distributed environment as a software repository/server and code development platform.
- A platform on which proprietary products reside which can be used by users whose usage patterns don't warrant purchase of the product and/or a platform on which to run it, and by users who wish to try out a product before purchasing it for their own platform.
- A platform on which experiments and individuals can "try out" UNIX before purchasing their own platforms.
- A home for users with relatively low requirements whose projects are most logically performed on a UNIX platform but who are unable to justify the purchase of a UNIX system.

- The platform from which division products are distributed, including both the distribution of products from the UNIX Kits area and use of software products directly by client systems participating in the AFS file system

1.3 Hardware

The FNALU system is composed of seperate interactive, batch and fileserving nodes. In general only the interactive and batch nodes should be visible to the user. Currently, the IBM nodes consist of an IBM RS/6000 model 590 as an interactive node (**fibi01**) and an additional IBM RS/6000 model 560 as a batch node (**fibb01**). The Silicon Graphics machines consist of a 4D/420 (**fsgi01**) which is a low end interactive system and a 20 CPU Challenge XL (**fsgi02**) that serves both interactive and batch functions. There are two 4 cpu SUN systems in fnalu. **fsui01** is a SPARCserver 20 used for interactive use and f**sui02**, a SUN ULTRA Enterprise.used for both batch and interactive. The fleet of primarily interactive systems is rounded out with 1, 4 cpu Digital UNIX Alpha system (**fdei01**) which is used for both interactive and batch processing.

There is also a set of strictly batch nodes (no interactive logins allowed) which have traditionally been referred to as the CLUBS systems. These nodes come in two flavors of UNIX, IRIX and AIX and utilize the Computing Division supported batch system, **LSF**. LSf is documented in chapter 4 of this document. The node names for these machines are **fncl00-fncl02, fncl04-fncl07 and fncl09-fncl11**. Figure 1.0 at the end of this chapter has a diagram of the FNALU hardware configuration.

1.4 System Software

FNALU's primary file system is AFS, which we believe will provide some benefits over the normal UNIX file system. AFS is designed explicitly to serve as a distributed file system both within a local area network (LAN) and a wide area network (WAN).

All systems participating in the AFS file system have the same view of the file system. All files for a set of machines known as a cell exist under **/afs/*cellname*** The cellname for the FNALU cluster is **fnal.gov**, thus all files are located under **/afs/fnal.gov**. There is also a symbolic link to a shorthand version of the cell name, which is simply **/afs/fnal.** Other cells are also accessible such as the CERN cell which is located under **/afs/cern.ch**. AFS has different file system semantics than either the standard UNIX file system or the VMS file system. In addition, AFS introduces increased security via Kerberos-style authentication. The change in semantics and the Kerberos-style security introduce some new commands as well as a new way of thinking about issues related to the file system. Some of these issues have been addressed more fully in Chapter 3 of this manual

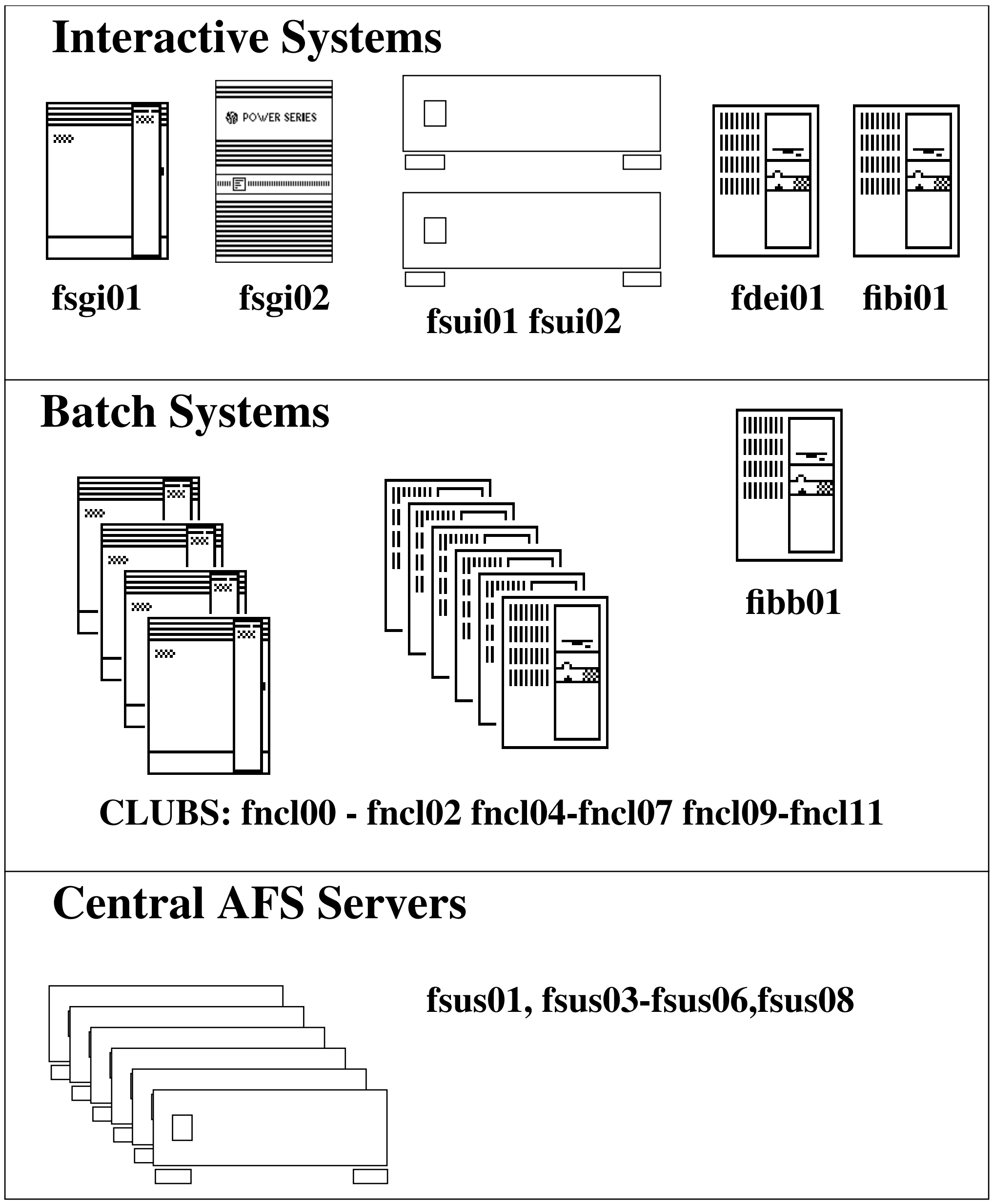
1.4.1 Application Software

Other than being served from AFS, the software installed on FNALU is similar to the other machines supported by the Computing Division. FUE is installed and products are handled in a way that should seem the same to users of the system.

1.5 Information

Information specifically about FNALU will be made available in the newsgroup **fnal.sys.fnalu.announce .** More general UNIX information will be posted in **fnal.announce.unix**.

0.1 Figure 1.0 FNALU Hardware Diagram



Chapter 1

Using the System

1.1 Introduction

As with all the FUE-compliant systems at Fermilab, *UNIX at Fermilab*, GU0001, should serve as a basic background on how to use UNIX, on the basic defaults at Fermilab, how products are supported, and background to I/O on UNIX. But, as was mentioned above, FNALU does not use the normal UNIX file system, but an enhancement known as AFS. This does result in a few differences between this and the other systems on site, and some of these differences are major. It is suggested that you read Chapter 3 of this manual, which gives further details on AFS.

References:

1) *UNIX at Fermilab*, GU0001

1.2 Logging In and Out

Since all of the nodes that are in FNALU are now running AFS natively, the process for logging on all of the systems is identical. Logging in appears similar to other UNIX systems, but there are some issues that you should be aware of:

1) In order to access the file system, you have to be authenticated to AFS. Note also that we are in the process of modifying the login procedure on FNALU systems so that you will receive both an AFS ticket and a general Kerberos ticket granting ticket on login. This will allow users to use general Kerberos basked utilities in addition to the AFS file system and utilities. This should normally be the case if you have properly entered your AFS password when logging in. Since we now *only* support the use of AFS passwords, you must use the **kpasswd** command to change your password(s) (see below).

2) AFS authentication does not expire with a UNIX logout. In order to destroy your Kerberos (AFS) *token*, you need to enter an **unlog** command. (You can put this in a logout script.).Unlogging will increase the security of your account and files since it will make it impossible for other users to “steal” a token that you left lying around after logout.

1.3 Home Directory

Your home directory is of the form:

/afs/fnal/files/home/room*?/username*

where *?* is a number and ***username*** is the login username. The home directories are arbitrarily assigned to a room number to keep the directory holding them from growing to an immense size.

1.4 Changing your Password

As you may have discovered already, there are a number of kinds of passwords possible on UNIX systems (e.g., the normal “UNIX” password and the yellow pages password), and AFS adds yet another: the AFS password. In order to minimize the confusion we have set up FNALU to only pay attention to AFS passwords. Although we previously provided a special password command called **fpasswd**, we now recommend that you simply use the kerberos password command (**kpasswd**) to change your FNALU password.

1.5 Backups

A snapshot of your home directory on fnalu is available to you on a backup volume mounted under **/afs/fnal/files/backup**. Therefore, if your current home directory volume is mounted at

`/afs/fnal/files/home/room1/hanson`

your backup snapshot is mounted at

`/afs/fnal/files/backup/home/room1/hanson`

If you accidentally delete a file or want to retrieve an old version, you may restore it yourself from the backup snapshot from the night before.

To restore a file, just copy it from your backup snapshot volume to your normal home directory. Backups are normally started about midnight so the snapshot of your file system will duplicate your directory at approximately midnight the night before.

Note, only one day of backup is available on-line. Files previous to that time will have to be restored from tape.

If you should happen to try to read files from your backup volume and can’t locate it, it is possible that it has never been mounted. Please contact fnalu-admin@fibi01 to resolve the problem.

1.6 Getting Help

As always, of course, you’re probably going to have some questions and problems that you can’t solve or understand yourself. If you need some help you should pursue one of the following to resolve your problem:

1.6.1 Help Desk

Problems of a general nature, systems down, or any other kind of problem - Please feel free to contact the help desk. They may not be able to resolve or answer all problems, but they are very good at getting in touch with someone who can for you. The help desk is only manned during business hours currently. You can contact them by

- Email to **helpdesk@fnal.fnal.gov**
- Phone to extension 2345.

1.6.2 USS Group

The USS group does the Operating System administration for most UNIX boxes at the lab, and all of the FNALU systems. You may contact USS directly if you feel that your problem is operating system related by sending email to

- **fnalu-admin@fnal.gov**

1.6.3 UAS Group

UAS is responsible for writing, porting, and maintain UNIX applications at FERMILAB. If you have problems that you believe to be specific to application software, you can contact UAS directly by sending email to

- **uas-group@fnal.gov**

Note however that in this case as well you can always feel free to contact the helpdesk.

1.6.4 Online Documentation

All of the platforms on FNALU have the standard UNIX man pages accessible. Man pages should be available for all (or almost all) of the commands on the systems. If you find a problem with this, please send email to **fnalu-admin@fnal.gov** and we'll try either to install a man page or explain how to access it.

Additionally, all of the systems have online documentation readers of various levels of success depending on the manufacturer. We attempt to keep these up to date. This section will contain more details in a later edition of the guide, but these are implemented in the standard fashion so if you know how to use them on another system the same technique should work here.

1.6.5 Web Servers

More and more documentation is being made available as time goes on in HTML format (or some other web-readable format) as time goes on. Some WWW pages to note:

- FERMILAB Home Page - **http://www.fnal.gov/**
- Computing Division - **http://www.fnal.gov/cd/**
- USS, UAS, and OSS - including FNALU and vendor information- **http://www-oss.fnal.gov:8000/**
- General Documentation - **http://fnala.fnal.gov:8000/lib/nprods**
- Physics Analysis Tools - **http://cdibm.fnal.gov/**
- FERMI UNIX Environment - **http://www-oss.fnal.gov:8000/fue/**
- Computing News - **http://www.fnal.gov/cd/CDN-issues.html/**

Chapter 2

AFS on FNALU

2.1 Why AFS?

AFS is a modern implementation of distributed file serving. It has several advantages over older file systems such as NFS. FERMILAB is investigating the use of AFS as a distributed file service model, and it was decided that the FNALU cluster would serve as a test of this model in a production environment. Briefly, the advantages of AFS in our environment are:

2.1.1 Performance

AFS institutes local file caching on the client machines. A system of callbacks is implemented to keep the caches on the client machines current with the copy of the file on the server. The use of caching should increase performance both from the viewpoint of the user and by decreasing network traffic.

2.1.2 Consistent file organization over distributed servers

AFS provides a consistent view of the served file tree across all systems that read the file system. The layout of the file tree is determined entirely on the server end of the relationship. Therefore, logging in to any machine on the internet, the AFS file tree for FERMILAB will look the same. The organization of this tree is totally conceptual, and the actual physical placement of the files is hidden from the user. There is no need to be aware of what server holds a particular file system.

2.1.3 Improved Authentication of users

AFS uses a modified version of the Kerberos authentication scheme from Project Athena. This allows much higher security and more powerful authentication of users. However, it does impose some potential problems on the user, which will be covered later.

2.1.4 Improved Server Management

Several features and procedures in AFS provide an improvement on the traditional UNIX procedures. AFS file systems are organized out of a (typically) large number of logical volumes. Each of these volumes has a separate ACL associated with it, which allows finer setting of permissions than the normal UNIX file protection mechanism. Additional ACL's can also be placed on any directory in the AFS tree, whether it is at the top of a volume or not. Quotas may also be set on volumes, to control the amount of disk space a user will use in, for example, a home directory.

2.1.5 Replicated and Backup Volumes

A very useful feature of AFS is the concept of replicated volumes. In addition to a normal read-write volume, individual volumes may be replicated in another location. Typically the replicated volumes would reside on a separate file system and normally also on a separate file server machine. This allows for redundancy in the case of a system failure. The replicated volumes are read-only volumes. Thus, updating files in the read/write volume doesn't automatically update the files in the read-only replicas. The volume must be manually released to the replicas. Replication only works well on volumes that are not frequently updated. Typically these would be volumes of site-wide executables like the products directories, etc.

Additionally, AFS has a concept of backup volumes. These backup volumes are similar to read-only replicas. They don't take up any file space, but are mirrors of the original volume. They only keep track of copies of files that have changed in the original volume. Thus, the backup volume is a snapshot in time of what the file system looked like when the backup was created. These volumes are used either to back up the file system (the backup volumes are made, and are then backed up, presenting an exact snapshot of what the file system looked like when the volumes were created) and to create backup volumes for users. Thus the user could always go to a backup of, for example, their home directory volume from the day before and retrieve a file. On FNALU, we create backup volumes nightly for every read/write volume, and do the backup from those snapshot backups.

2.1.6 Server function redundancy

AFS also allows for a fair amount of server process redundancy, so that if any individual server goes out, the entire cell will remain running. We currently have 3 primary file servers in FNALU, all of which provide full server services.Thus, the loss of one file server will still allow access to files that are still available on any of the file servers. Thus, volumes that are replicated on at least two file servers should **always** be available within the cell. Note, however, that a client system must be able to contact a quorum of the volume location servers. Therefore, systems that cannot contact at least two of the central AFS file servers will not be able to write files on the remote file server, but will be able to read all of the files on file servers that are contactable.

2.2 User Issues Presented by AFS

A number of user education issues are raised by AFS, as well as some actual problems that will be caused for certain classes of users. This chapter will attempt to address some of these issues and provide some solutions where they are known.

2.2.1 Command formats

The various AFS commands have a (reasonably) consistent format. There are several commands that the average AFS user may need from time to time. Each of the major AFS commands deals with one of the subsystems (actually, one of the server daemons) in AFS. The primary commands are:

`fs` - which is the file system command. This command is used to deal with file system issues, such as checking permissions on a directory or volume, checking quota, making mount points, finding where a volume is mounted in the file tree, etc. To obtain help on using fs, you can simply type

fs help

basically this same help mechanism works on all of the AFS commands (well, most of them, anyway). To get help for a particular fs command (for example, fs listacl) you may type something like

fs listacl -help

`kas` - the Kerberos authentication server - This is used to query or interact with the Kerberos system - typically this would be to check tickets granted, set a user password, etc. This is less likely to be of a lot of use to the typical user.

`pts` - The protection server. This is used for things like checking group membership, adding a member to a group, adding a user to the protection server, etc.

`vos` - The volume server. This is primarily used by the administrator. It is used to look at the system volumes, move volumes, etc. It isn't usually available on client machines.

`klog` - Used to obtain kerberos authentication. Normally users on this system won't need to klog as they will be authenticated by the Kerberized login program.

`knfs` - Used to enable authenticated access from an NFS client mounting the AFS tree through the NFS translator. This is only likely too be useful to people who don't have AFS installed on their machines. Currently this is going to be the only way to access the AFS files from an SGI machine, for example. It requires logging in to the NFS translator (currently fsut01) and authenticating there. Then upon returning to the NFS client the user will be authenticated to access files. The NFS security is considerably more lax as a practical matter.

`tokens` - Used to check the tokens the user is holding.

`unlog` - used to discard all tokens. It is best to unlog before logging out.

Reference cards for AFS which list the most common AFS commands and their syntax are available by request from the Helpdesk staff.

2.2.2 Authentication Issues

On most machines in the FNALU cluster, users are granted Kerberos and AFS authentication at the time they log in. Providing a user name and matching password is considered proof of the user's identity. Because a modified login procedure is used on these machines, the process of logging in provides the user with an AFS ticket that identifies the user as legitimate. This authentication is passed on to all processes that are spawned from the login process. Thus, any normal sort of operation that will be done interactively by the user will be authenticated, and will have access to files in the AFS tree that have permission turned on for the login name of the user.

Tickets that are granted by the Kerberos server have a limited lifetime. Currently the default tickets granted by AFS are good for 25 hours. Thus, the user will have access to files on the system for 24 hours after logging in. This policy may change, but for the time being the implication for the user is that file permissions will mysteriously disappear after 25 hours. This will obviously cause some problems for long-running processes, batch processes, at jobs, and some other classes of work. We will be exploring solutions to these problems, which we hope will be more creative than simply granting extremely long-lived tickets. Two current solutions are being investigated. The FNCLUB and FNALU systems use a program called fbatch, which allows batch jobs to run by having the user provide a password to AFS at the time that the job is submitted. This password is squirreled away with the job, so that the batch job can reauthenticate itself at the time the job is submitted. Note, however, that if your token expires during the run of the job, it will fail. Heavy batch users on CLUBS may ask to have their default token expiration time extended to 72 hours, which should cover any possible problems. We will additionally be experimenting with several other solutions such as the reauth daemon, which periodically reauthenticates a running process. You may see what tokens are currently associated with your login shell by typing the **tokens** command. If it lists no tokens, you are currently unprivileged, and will only have access to files accessible to the **system:anyuser** special user.

There are several things worth noting about the process of obtaining a token. One of the confusing aspects here is that users may potentially have *two* different passwords - one for the AFS/Kerberos system, and one for the traditional UNIX login. In general we are setting up users on FNALU to have ONLY a Kerberos password, and to have their normal UNIX password disabled. The thing to look out for here is that changing your password with the normal *passwd* command will change your UNIX password, which is probably not what you want. If you then log in with the changed password, you will be authenticated in UNIX but not in AFS. Thus, although you will be logged in, you will probably not be able to access your own files. The normal procedure for changing your password on these systems will be to use the **kpasswd** command, which will change the Kerberos password.

Finally - logging out doesn't destroy your Kerberos token. Your token will live for the default lifetime. Before logging out, you should issue the **unlog** command. It may be helpful to put unlog into the logout script of whatever shell you are using.

To see what token or tokens you currently hold, you may use the **tokens** command. Commands are covered in detail later in this paper.

One practical issue raised by the Kerberos environment involves the use of the Berkeley networking programs such as rlogin, rsh, and rcp. Normally systems are *equivalenced* to enable the use of these protocols. The equivalencing of the machines implies that once a user is logged into one system they may login to the equivalenced machines without providing further proof of identity, such as a password. This won't fit in with the Kerberos authentication system. We have installed token-passing versions of a lot of software on the FNALU systems in an attempt to make life less painful, but they have at the same time changed the behavior of these programs from what might be expected.

Token-passing is available for rsh and rcp. The general effect of this is that if a user has authenticated into the *fnal.gov* cell, it is possible to use rcp and rsh more or less normally between machines - the token will be passed along with the request and the network communications will be authenticated. Thus the user can rcp or rsh and be authenticated on the remote machine automatically. This is set up with the usual UNIX method of using /etc/hosts.equiv or .rhosts to equivalence systems.

However, the issue with rlogin is more complicated. When equivalencing between two machines is enabled, the rlogin protocol will not ask for a password. This is very convenient on normal UNIX systems. However, in AFS systems, the lack of a provided password makes the remote user untrusted. Thus, if two systems are equivalenced, rlogin will work and no password will be requested, but the user will **`not`** be authenticated at login time. Therefore, the choices are either to use telnet instead, which will automatically authenticate the user, or to use rlogin and then to immediately authenticate using **pagsh** and **klog.**

Likewise, certain other programs in the normal FERMI environment are likely to break in this environment. One known program is zftp. It should not be too difficult to convert these programs to do authentication through the AFS libraries rather than the normal UNIX authentication routines, but unfortunately these libraries have only recently been documented by Transarc.

One final note - There is also an AFS-NFS translator available. The entire FERMI AFS tree is exported from **fsut01** as an AFS mountable file system. Thus, the AFS files may be mounted on any system that works as an NFS client (supposedly including VMS systems that have NFS client software). However, anyone accessing the files will essentially be granted permissions as system:anyuser (that is, only files that are readable by system:anyuser will be readable). Users may authenticate with the AFS/NFS translator by logging into fsut01 and using the **`knfs`** command. Typical use would be as follows:

```
knfs pogo.fnal.gov 1231
```

assuming that the NFS client is node pogo and the users UID is 1231. The NFS authentication is pretty weak in some ways. The NFS translator may be removed at some time in the future, but it does currently provide a path for accessing AFS files on those machines that don't have AFS software available, such as SGI machines.A set of shell scripts has been written that make authentication less painful in this environment. They simply replicate the actions of the normal AFS authentication programs (such as klog, unlog, tokens, etc.) by doing the appropriate NFS authentication actions on the file server. Thus a user can log in to one of the translator client systems and use the klog command on the NFS client, which will authenticate the user to the cell and set up NFS permissions to the user on the client system. We consider the use of the translator to be a stopgap measure which we do not want to encourage however. We now have native AFS available for all of the platforms supported at FERMILAB except for IRIX 4 - if you are using IRIX 4 and wish to participate in the AFS cell, we'd suggest upgrading to IRIX 5.3. None of the FNALU or CLUBS systems require the use of the NFS translator any more.

2.3 File Permissions

This is a confusing issue, so try to follow along. File protection in AFS is quite different than in a normal UNIX file system. A normal UNIX file or directory has nine major mode bits. They grant read, write, and execute permission for the owner, group, and world users respectively. This is shown in the output from ls -l as follows:

```
d-rw-r--r-- 1 hanson 1222 Oct 24 1991 README
-rwxr--r-- 1 hanson 311 Oct 31 1991 doit
-rw-r--r-- 1 hanson 1040384 Oct 31 1991 patchtar
drwxrwxr-x 2 hanson 2048 Oct 31 1991 shells
drwxrwxr-x 4 hanson 2048 Oct 24 1991 tree
```

a. First field is the type of the element (- for file, d for directory, l for symbolic link, etc.)

b. The next three fields list the "user" read, write and execute permissions.

c. The three fields after that list the read write and execute permissions pertaining to the group owning the file.

d. Finally, the last 3 fields list the permissions for all other users on the system.

In AFS, you can use the standard chmod command just as you would in a standard UNIX file system. However, the behavior of the file will be different. Although all of the bits may be changed or examined on a file in the AFS file system, only the owner bits are actually used in AFS. They apply to *all* users of the file. Thus to turn off write access to a particular file, you will need to turn off the *owner* write bit of the file. This will inhibit writing to that file for all users, including the owner. In the current AFS implementation, file level permissions are only changeable for the owner bits.

Other permissions are done with ACL's, which take effect on the directory level only. ACL's can be set for various users and groups. The permissions granted in the ACL are:

2.3.1 l - lookup rights

Lookup permissions allow the possessor to

- Issue an ls command on files in the directory
- Examine the ACL for the directory
- Access the directory's own subdirectories, which are protected by their own ACL's

2.3.2 i - insert rights

Insert permissions allow the user to

- add new files to the directory, either by creating new files or by copying files in.

2.3.3 d - delete rights

Delete permissions allow the user to

- remove files or move them to other directories

2.3.4 a - administer rights

Administer permissions allow the holder to

- change the ACL for a directory. Users always have this right for their home directories, even if they accidentally remove this acl themselves.

2.3.5 r - read rights

Read rights allow the holder to

- look at the contents of the directory. For files in this directory the possessor can read the data in the file.

2.3.6 w - write rights

Write rights allow the holder to

- Modify the contents of the files in the directory and to change their UNIX mode bits with chmod.

2.3.7 k - lock rights

Lock rights allow the user to

- run programs that need to “flock” files in this directory.

2.3.8 Combination Rights

Rights may also be referred to by special names that refer to commonly-assigned combinations of rights.

- Write - all rights except administer (that is, *rlidwk* rights)
- read - read and lookup only
- all - all seven rights (*rlidwka*)
- none - no rights - this removes the user’s entry from the ACL entirely.

Users may examine the ACL setting on a particular directory with the

```
fs listacl
```

command. Rights are set with the

```
fs setacl
```

command.

Please note that a reasonable set of permissions will be set up on each volume by the system administrators when the volume is created. In general users will be granted *all* permissions on their home directories. ACL’s are by default inherited by child directories when they are created.

Additionally, home directories will be created with the permissions of system:authuser set to rl. System:authuser is a special system user name that is assigned to any user who is authenticated on the server. Thus, by default any files in your home directory and directories within your directory will be readable by any other user who has authenticated with our AFS server (including, potentially, offsite users who have accounts with us). If you want to read-protect directories within your home directory, or your home directory itself, you will have to do so by removing the ACL for system:authuser. Note that if you turn off rl permission to your home directory for system:anyuser, you will find that the system can no longer read your .forward file, etc. We are working on moving toward a more sophisticated model in which your home directory can be protected more stringently, either by having a public directory under your home directory, into which things like.forward may be simlinked, or by using an authenticated mail system such as AMDS.

2.3.9 Some Practical Implications of AFS ACL's and examples

In general the implementation of security in our AFS cell is based on the notion that sharing of information is more common than trying to protect it, so in most cases the preference in how ACL's is set is to have the least reasonable security be the default. As currently implemented, all user home directories have their ACL set so that system:anyuser has read and lookup permission. The practical implication of this is that *anyone* on the internet running an AFS client can read your files as the default system is set up. Home directories are only *writable* by their owners (that is, the owner has rldiwka permission), but the world can read them. For many things, this is probably fine, but you should be aware of it and take pains to protect any files you don't want to be publicly readable. In general, you can protect any individual directory by changing the ACL to turn off permission for system:anyuser. As an example, if I had a directory in my home area in which I kept the secret formula for the hair-restoring tonic I just perfected, and I wanted to keep it private, I'd use a command something like:

fs setacl $HOME/hairtonic system:anyuser none

which would keep the world from reading it. You can do this on any directory which you want to keep the outside world from exploring. Note that you should also look at the ACL for the directory to make sure that it isn't readable by anyone else that you want to protect it from.

Likewise, you may find the default permission is too stringent - you may have a common work area which you want to share with someone else, so that they may write into it at will. Say you are in a collaborative effort with your friend, whose account name is mrchips. You could allow him full permission in your shared directory (called *shared* in this example) by typing in

fs setacl $HOME/shared mrchips all

which would give mrchips all privs in the directory. Note that permissions for a directory are the **union** of all of the ACL's, so if an individual user matches more than one ACL, he would get the highest privilege that applies.

Lastly, note that the fs setacl command only changes the permission for an individual directory. If you have a full directory hierarchy on which you want to change permissions, you'll have to resort to doing a command that traverses down the tree and changes all the directories as it goes. The *find* command is very useful for this. As an extension of the above example, if you had a big directory hierarchy you wanted to allow mrchips full access to, you would execute the following command instead of the one above

find $HOME/shared -type d -exec fs setacl {} mrchips all \;

which would traverse down from the shared directory, changing all of the ACL's on directories as it goes.

2.3.10 Protecting your home directory

You may find that you want to keep people from reading the top level of your home directory. This is not as simple as it may sound. Although you would think that you could just change the ACL for $HOME and make it unreadable by system:anyuser, this has several bad effects:

- If you should happen to manage to log in unauthenticated, you won't be able to enter your home directory, and may not be able to log in at all, depending on what platform you are on.

- This will have the side effect of making your dot files in your home directory unreadable by unauthenticated system processes, which in many cases will break their functionality. As a simple example, if you make your home directory ACL not give rl permission to system:anyuser, the sendmail program will not be able to read your .forward file, and your mail forwarding will break. Thus all mail sent to an individual node will be delivered there rather than forwarded on.

It is usually easiest to make your home directory world readable and to make sure your private files are kept in better-protected subdirectories. However, if you really wish to do this, you must make sure that any files you can identify that *must* be world readable, such as your .forward files, remain accessible. You would to this as follows:

- Every AFS home directory is created with a subdirectory called *public*. You first need to take your files that must be publicly accessible and move them into the public subdirectory.
- You should then create a symbolic link from your home directory to the publicly-available directory.
- When all the files are moved and linked, you can then shut off the read permission on your home directory. Note that you *must* leave lookup permission turned on or programs won't be able to find the file in the public directory through the simlinked. Also note that it may be difficult to determine which of the files in your home directory *must* be publicly readable to preserve the behavior of your environment, so you must be careful if you are setting up your home directory to be more secure.

As a simple example, if the only file you could identify that must be publicly readable is your .forward file, you could do the following:

cd

mv .forward public/.forward

ln -s public/.forward .forward

fs setacl . system:anyuser l

2.3.11 Quota issues

Currently we are creating non-grant-request home directories on the FNALU systems with 50 Mbyte quotas. Quota change requests must go through the ACCESS office. In the future quota will be set according to the size that is approved when a project petitions for use of FNALU. You can check quota on a volume by using the

```
fs listquota
```

command. Likewise, quota may be set on a volume (if you have permission to set quota on the volume) with the

```
fs setquota
```

command.

2.4 AFS in the FUE/PRODUCTS/UPS world

I will give here a few practical examples of how AFS permissions have been implemented in the products area on FNALU to show how the file permissions might work in a real-world situation we are familiar with. This will also serve as a guide for those people who are maintaining UPS products on the FNALU cluster.

2.4.1 File layout

The layout of the files in the products area is the same as it is on other multi-architecture clusters at the lab, with the exception of the fact that the products tree begins at

```
/afs/fnal/products
```

rather than /usr/products. If you are unfamiliar with the multi-architecture layout, it is very similar to the normal single architecture layout of files except that each separate platform has a directory under the products directory which holds the actual products code. The database directory is shared among platforms. Thus, there are directories for

```
/afs/fnal/products/AIX
/afs/fnal/products/IRIX
/afs/fnal/products/SunOS
```

etc.

A design decision was made to allow product developers to work within individual products directories logged in as themselves. Thus, each of the products has a separate AFS volume. The ACL for each volume has been set so that the product developers have full permissions on the volume. Any authenticated user on the system may read the contents of the volume, as well as do lookups. Products also has full permissions on the directory. Thus, as an example, the permissions on /afs/fnal/products/AIX/bufio are set as shown:

```
pogo% fs listacl bufio
Access list for bufio is
Normal rights:
 products:bufio rlidwka
 system:administrators rlidwka
 system:authuser rl
 products rlidwka
```

The quotas on each directory are set approximately to allow three revisions of the product and to allow some extra space. On the same directory, the quota is shown as:

```
pogo% fs listquota bufio
Volume Name Quota Used % Used Partition
products.AIX.bufio 3500 872 25% 13%
```

To be able to declare products the installer must be able to write into the **/afs/fnal/products/ups_database** directory. The permissions on this directory are set so that all permissions are granted to a group called products:upsdatabase -

```
  •pogo% fs listacl ups_database
Access list for ups_database is
Normal rights:
 products:upsdatabase rlidwka
 system:administrators rlidwka
 system:authuser rl
 products rlidwka
```

The products:upsdatabase group has as its members ALL of the products maintainers. Thus any of the products maintainers may make a product current.

```
pogo% pts membership products:upsdatabase
Members of products:upsdatabase (id: -288) are:
 edel
 kreymer
 lebrun
 trang
 janetc
 b92799
 marafino
 garren
 jeffk
 sandrac
 stein
```

2.5 Currently Open Issues and Problems

2.5.1 File System Permissions are Too Lax

This was mentioned above. We hope to have a more sophisticated scheme in place to allow home directories to not be world readable some time in February, 95.

2.6 User Documentation

There is not a large availability of user documentation for AFS. Many of the on-line manual pages are somewhat out of date, or are confusingly named. Laminated reference cards are available from the help desk for the common AFS commands.

2.7 Availability of AFS on the SGI platform

AFS software has recently become available for SGI systems. However, it is only available for IRIX 5.2 and later. Therefore, systems running IRIX 5 may participate in the AFS cell without having to use the AFS translator. Systems running IRIX 4 that will be participating in the AFS cell should upgrade to IRIX 5.2 or later.

2.8 Oddities of the AFS file system

There are a few minor issues with AFS that are not immediately apparent and that should be explained so that if you run across them you won't be too shocked or confused.

2.8.1 File ownership is almost meaningless

Since the ownership bits on file permissions apply to ALL users who have access allowed by the ACL, the ownership of the file is nearly unimportant. Programs which specifically test for ownership of a file will of course care about this. But the ownership of a file means nothing as far as file *access* is concerned.

2.8.2 Hard links are only legal within a volume

This may cause some problems with some Makefiles, etc. Usually doing a symbolic link instead is an acceptable workaround.

2.8.3 Problems with find, etc.

You should be *very careful* about anything that traverses the file system on a machine that has /afs mounted. Please be aware that a find on your system starting at / will traverse the *whole* AFS file tree - including all the other sites that we have mounted on our cell. This is particularly a problem on some workstations like Solaris 1 Suns that by default run a nightly cron job that traverses the whole file system. Also note that using things like pruning NFS mounted file systems in your find won't help - find can't tell the difference between local files and AFS files.

2.8.4 User confusion over file permissions

Just to bring this up one more time. Users who expect to have the file system permissions on AFS files act anything like they do in a normal UNIX file system are going to be mighty disappointed or confused. The ACL on a directory is the overall governing permission on a file. The *only* effect of the UNIX file permissions is to serve as a mask where the owner permission bits will affect the permissions of all users who are granted permission by the ACL.

2.8.5 Batch, at, and cron processes

Users should be aware of the fact that there will be problems with running programs with batch, at, cron, or any other program that will spawn a job that isn't part of the user's login process group. You can certainly schedule a job to run with at or cron, but the job won't run with authentication, so it will likely not be able to write output anywhere in the AFS file space (or even read data in some cases). Computing division currently provides a rather awkward means to execute authenticated cron jobs, which you can do by making a special arrangement through the USS group. If you have a strong need for this, please send email to *uss-group@fnal.gov.* However, we are working on a more elegant solution that will allow users to schedule cron jobs themselves, which we would expect will be available in 1996.Likewise, be aware of the fact that being root on your computer grants you no special permissions in AFS file space, and in fact will be the same as being any other user. There is no such thing as being "authenticated as root".

2.8.6 File Locking

The file locking mechanism in AFS does not really follow POSIX semantics. Files may only be locked as a whole. Regions of a file may not be locked. In addition, file locking really only works as expected from a single system. If a file is locked from one client and attempts are made to access the file from another client, **EWOULDBLOCK** is returned. There is also no deadlock prevention, so it is entirely possible that deadlock situations will occur with file locking in AFS. Generally we don't recommend the use of applications that depend on file locking functioning in the AFS file space. Programs that attempt to use byte-range file locking in AFS will receive a warning message from the cache manager, indicating that if you use this application you should be certain that other users are not accessing the same file. Usually these messages can be safely ignored. If you have a concern about these issues, you should contact the USS group for resolution.

Chapter 3

Batch

The batch system used on FNALU is **LSF**[1] (Load Sharing Facility) developed by Platform Computing. **LSF** is a general purpose distributed computing system that unites a group of computers into a single system to make better use of the resources on a network.

There is one batch cluster defined on FNALU, called **fnalu_batch**. Any host defined as a batch server within **fnalu_batch** is eligible to run batch jobs. Previous batch systems were comprised of two batch clusters, FNALU and CLUBS. This distinction does not exist today. Some functionality provided by the old CLUBS system (spacall for example) is still only available on the former CLUBS hosts (fnclxx), however there is no reason why this could not be available all of the hosts in the batch cluster. **fnalu_batch** is comprised of IRIX, AIX, SunOS, and DEC/ALPHA hosts.

For a complete description of the fbatch product, refer to the fbatch users guide (PU0152)

3.1 Batch Jobs

A batch job is any UNIX executable that is submitted to LSF. Control information is passed to LSF via command line arguments. The most common arguments supplied when submitting a job are the name of the queue to submit the job to, the resources needed to run the job, and the name of the executable (required).

When a job is submitted, LSF will determine the best host to run the job on. This determination is made by looking at the resource requirements specified for the job, and the load of the hosts that contain the resource requirements specified. If no resource requirements are specified, the job will run on a host that has the same model and type.

LSF monitors the load of each host in the batch cluster. When a host becomes overloaded LSF will begin suspending batch jobs. When the load thresholds fall below a certain point, LSF will then begin resuming jobs. LSF maintains a batch priority for each job, which is based on the queue that the job was submitted to. There is also a nice value associated with each queue. Jobs submitted to a queue are automatically reniced.

The batch system is comprised of three types of hosts:

- Interactive only (LSF client host)
- Batch only (LSF server host where users can not login)
- Batch + Interactive (LSF server host where users can login)

The batch system consists of several queues that you can submit your job to. In general, the name of the queue reflects the amount of CPU time that a job submitted to this queue can run. For example, the 30min queue will run jobs that need 30 minutes or less of CPU time. Some queues are suffixed with _disk. These queues provide your job with scratch disk space for the duration of the job. These queues are currently setup to run only on the fnclxx hosts.

1. The batch system on FNALU was previously based on LoadLeveler

3.2 fbatch Utility

fbatch utility was developed to allow users to execute the LSF commands on the **fnalu_batch** cluster. **fbatch** supports the following commands:

fbatch_acct - provide LSF accounting information.

fbatch_bot - move job to bottom of queue

fbatch_cal - get information about calendars in the lsbatch system

fbatch_caladd - add,modify, or delete a calendar

fbatch_caldel - add,modify, or delete a calendar

fbatch_calhist - display history of modifications to caledars

fbatch_calmod - add,modify, or delete a calendar

fbatch_chkpnt - checkpoint one or more unfinished batch jobs

fbatch_del - delete to one or more unfinished jobs

fbatch_hist - display history of batch jobs

fbatch_hostinfo - display a summary about hosts and their resources

fbatch_hosts - display information about hosts in an LSF cluster

fbatch_hpart - get info about host partitions in the lsbatch system

fbatch_jobs - get info about running jobs in the lsbatch system

fbatch_kill - send a signal to one or more unfinished jobs

fbatch_load - get load information on batch hosts

fbatch_mgroup - display the membership of lsbatch user or host groups

fbatch_mig - migrate a batch job to another host

fbatch_modify - modify attributes of a running job

fbatch_params - display LSF system parameters

fbatch_peek - display LSF system parameters

fbatch_queues - look at stdout/stderr of unfinished job

fbatch_restart - restart a terminated job that has been checkpointed

fbatch_resume - resume an unfinished batch jobs

fbatch_stop - stop an unfinished batch jobs

You need to use `% setup fbatch` to access **fbatch** commands. When you setup **fbatch**, your $MANPATH environment variable will be set to point to the man pages for these commands. Each command has a man page which describes the command format and gives a full description of the command. This section will describe the more commonly used commands.

3.3 Submitting Jobs

The **fbatch_sub** command is used to submit a job to the batch system. The most common arguments used are **-q** (queue name), **-R** (resource requirements), **-o** (stdout redirection), **-e** (stderr redirection), and **-N** (notify). To see which resources and hosts are defined in the cluster, you can issue the **fbatch_hostinfo** command.

```
fbatch_hostinfo
```

```
HOST_NAME      type    model cpuf ncpus maxmem maxswp server RESOURCES
fsgi02          SGI R4400Ch2 84.0    20   511M  2755M    Yes (irix any fsgi02)
fsui02       SUNSOL ULTRA167 93.0     4   320M   870M    Yes (sparc any fsui02 sun)
fibb01          AIX      560 39.0     1   192M   400M    Yes (aix any fibb01)
fncl09          AIX      370 49.0     1   128M  1136M    Yes (fncl09 aix any)
fncl10          AIX      370 49.0     1   128M  1136M    Yes (aix any fncl10)
fncl06          SGI  R4400Cr 63.0     1   143M   738M    Yes (irix any fncl06)
fncl07          SGI  R4400Cr 63.0     1   143M   639M    Yes (irix any fncl07)
fncl00          AIX      370 49.0     1   128M   592M    Yes (aix any fncl00)
fncl01          AIX      370 49.0     1   128M   368M    Yes (aix any fncl01)
fncl02          AIX      370 49.0     1   128M   368M    Yes (aix any fncl02)
fncl04          SGI  R4400Cr 63.0     1   143M   663M    Yes (irix any fncl04)
fncl05          SGI  R4400Cr 63.0     1   143M   663M    Yes (irix any fncl05)
fncl11          AIX      370 49.0     1   128M   368M    Yes (aix any fncl11)
fdei01        ALPHA 2100A500 136.0     4   637M  1499M    Yes (dec osf1 alpha fdei01)
fibi01          AIX      560 39.0     -      -      -     No (aix any fibi01)
fsgi01          SGI    4D420 30.0     -      -      -     No (any fsgi01)
```

For complete information on `fbatch_sub`, see the `fbatch_sub(1)` man page.

EXAMPLE: Submit a script called myjob to the `4hr` queue to an IRIX host. Redirect the stdout to myjob.out, and stderr to myjob.err.

```
fbatch_sub -q 4hr -o myjob.out -e myjob.err -R "irix" -N myjob
```

fbatch will prompt you for your AFS password. When your job begins, you will receive a renewed AFS token on the execution host.

3.4 Monitoring Jobs

fbatch provides several commands to monitor your job. **fbatch_jobs** will produce a listing of running jobs. If no options are supplied, then the list will contain only your running jobs. To see all running jobs, use the **-u all** option:

```
fbatch_jobs -u all
```

To see the stdout and stderr of a running job, use the fbatch_peek command:

```
fbatch_peek 1001
```

This command will display the stdout and stderr of job 1001.

History information about a job is available using the **fbatch_hist** command.

3.5 Controlling batch jobs

fbatch provides commands to suspend, resume, kill, move jobs within a queue, and modify the resources of a job submitted to the batch system. Below are examples on the usage of these commands:

Suspend Job 1022:

```
fbatch_suspend 1022
```

Resume job 1022:

```
fbatch_resume 1022
```

Cancel job 1022:

```
fbatch_kill 1022
```

Send signal 2 to job 1022.

```
fbatch_kill -s 2 1022
```

Move a job to the bottom of the queue:

```
fbatch_bot 1022
```

Move a job to the 2nd position in a queues:

```
fbatch_top 1022 2
```

Change the resource requirements of a job:

```
fbatch_modify -R “aix” 1022
```

The **fbatch_bot** and **fbatch_top** commands move jobs within queues relative to the users own jobs. You can not move your job ahead of another users job with these commands.

3.6 Queue Definitions

The general purpose queues defined on the **fnalu_batch** are:

30min, 30min_disk

4hr, 4hr_disk

12hr, 12hr_disk

1day, 1day_disk

4day

The names of the queues reflect the amount of CPU time a job is allowed to use. The *_disk queues provide scratch disk space for your job. The path of this scratch disk space is stored in the environment variable $CLUBS_WORK_DIR. This scratch disk space is created by the space allocator before the job starts and removed when the job ends. The space allocator is currently only installed on the fnclxx hosts (the old CLUBS hosts), but may be ported to the FNALU nodes in the future. Thus, when submitting to the *_disk queues, your job will only run on hosts with the space allocator installed.

Each queue has a batch priority specified with it as well as a nice value. In general, the shorter queues have a higher batch priority and a lower nice value (meaning a higher UNIX priority) than the longer running queues. The batch priority is used by LSF to determine scheduling and suspending order.

3.6.1 Related Software Components

This section describes other software components that can be used with the batch system.

needfile

The **needfile** system provides two functions. If the user’s data does not yet exist in the HSM, **needfile** tells the system to copy the data there from its original medium. **needfile** then provides access to this data for the user

This data remains in the HSM based on a least-recently-used policy. There is no guarantee how long data will remain in the HSM.

tcache

The **tcache** system is a set of tools that allows users to save bulk data for later retrieval. The data is stored in the HSM. Whereas **needfile** is a read-only system, **tcache** is a facility to read and write data to the HSM with a specified retention period.

spacall

The space allocator provides scratch disk storage for a users job. The **spacall** utility is invoked by submitting a job to the *_disk queues under **fbatch**. When a job is submitted to these queues, scratch disk space will be allocated to the users job. The path of this space is stored in the environement variable **$CLUBS_WORKDIR.**

Chapter 4

Using Tapes on FNALU

4.1 Introduction

Tape mounts on the FNALU system are requested with the **OCS** system. OCS is documented thoroughly in a separate manual, called **OCS Reference Guide**, which is available via WWW as http://dcdsv0.fnal.gov:8000/hanson/FNALU/ocs.ps and as a Postscript file suitable for viewing or printing by issuing a `setup ocs` command and looking in the doc directory beneath $OCS_DIR. An OCS overview is also available at http://???????. OCS allows users on the system to request that tape mounts be done by the operators.

OCS may be accessed either through a series of text commands, or an X interface. The OCS system may also be used by linking your programs to the library of OCS routines. To access the full OCS package, you must first set up the software by entering

setup ocs

which configures your environment to use the ocs package. After doing this, the complete documentation for OCS is available in $OCS_DIR/doc. $OCS_DIR/examples will contain an example of using the OCS system with C, FORTRAN, and Bourne shell scripts. These executables may be run to demonstrate OCS features.

4.2 xocs

The X interface program may then, for example, be started by making certain your X environment (permissions and DISPLAY variable, for example) is set properly, and entering

xocs

which will display the xocs display, as shown above: The **xocs** command allows you to do most of the same manipulations that you can do with the text commands, but works with an intuitive GUI interface.

4.3 OCS text-oriented commands

In addition to the GUI interface, there are many text-based commands in the OCS package, some of which are described in this section.

4.3.1 ocs_allocate

The **ocs_allocate** command allows users to allocate a tape drive for their own use. Options are:

Table 1: ocs_allocate options

Letter	Description	Default
-d	device type	any
-T	preferred device	none
-t	device group	any

Table 1: ocs_allocate options

Letter	Description	Default
-h	host group	localhost
-m	must get	no

Some examples of the ocs_allocate command:

- Allocate a specific tape, fail if not available.

ocs_allocate -T fnalu22 -m

- Allocate a tapedrive on the local host, preferably drive fnalu22

ocs_allocate -T fnalu22

- Allocate a tape drive from device group “CDF_drives”

ocs_allocate -t CDF_drives -h any

- Allocate an 8505 tape drive on my local host

ocs_allocate -d 8505

4.3.2 ocs_deallocate

The **ocs_deallocate** command is used to deallocate tapes that have been previously allocated. Options are:

ocs-deallocate specific options

Letter	Description	Default
-a	all drives owned on localhost	none
-A	All drives owned on all hosts	none
-t	Tape Drive Name	None

Some examples of **ocs_deallocate**

- Deallocate drive fnalu22

ocs_deallocate -t fnalu22

- Deallocate all tape drives owned by me on the local host

ocs_deallocate -a

- Deallocate all tape drives on all hosts

ocs_deallocate -A

4.3.3 ocs_cancel

The **ocs_cancel** command is used to cancel a pending mount request. An example of this command would be:

- Cancel a pending mount request for drive fnalu22

ocs_cancel -t fnalu22

4.3.4 ocs_request

The **ocs_request** command is used to request a tape mount, and also to send special instructions to the operators along with tape mount requests. Some examples:

- Request tape xyz123 be mounted read-only on fnalu22

ocs_rquest -t fnalu22 -v xyz123 -r

- Request tape xyz123 be mounted on fnalu23, write-enabled:

ocs_request -t fnalu23 -v xyz123 -w

- Same as above, but send a special message to operators:

ocs_request -t fnalu23 -v xyz123 -w -o "Tape mismounted: make sure write-protected"

4.3.5 ocs_tape

The **ocs_tape** command is used for various tape functions. The options that may be used with it are as follows:

Table 2: ocs_tape options

Option	Description	Default
-t	device group	any
-d	device type	any
-a	allocation	any
-s	status	any

Some examples:

- List all tape drives on all FNALU hosts, using the long format:

ocs_tape -o long -h any

- List all unallocated tape drives on the local host, using the short format:

ocs_tape -o short -a unallocated

- List all allocated 8505 tape drives in the default (long) format:

ocs_tape -a allocated -d 8505

• List all broken tape drives on the local host:

ocs_tape -s broken

• List all working tape drives in the group “CDF_devices”

ocs_tape -s working -t CDF_devices -h any

4.3.6 ocs_setdev

The **ocs_setdev** command is used to set device characteristics of tape devices, and only works on AIX systems currently. It may be used to set device characteristics on remote nodes. However, the user *must* have the device allocated before setting its characteristics.

Some examples:

• Set device characteristics to variable block with no compression:

ocs_setdev -t fnalu22 -nv

• Set device characteristics to fixed block (4k) with compression:

ocs_setdev -t fnalu22 -f 4096 -c

4.3.7 ocs_devfile

ocs_devfile is used to get correct special file for a particular set of device characteristics. The options for this command are:

Table 3: ocs_devfile options

Option	Description	Default
-t	Tape Drive	none
-s	Byte Swapping	no swap
-f	Fixed-length records	variable
-r	Auto-rewind on close	no rewind
-n	Auto-retension on open	no retension
-d	8200 or 8500	8500

4.3.8 ocs_broken

This command is used to mark a tape broken (you will be prompted for a reason):

ocs_broken -t fnalu22

4.3.9 ocs_dismount

Rewind and unload the tape from the drive:

ocs_dismount -t fnalu22

Rewind a tape drive without unloading

ocs_dismount -t fnalu22 -o nounload

4.3.10 ocs_pending

Determine if a mount is pending:

ocs_pending -t fnalu22

4.3.11 ocs_check_tape

Verify that a tape has been properly mounted, without checking that the label is correct:

ocs_check_tape -t fnalu22 -r

Verify that a tape has been properly mounted, with label verification

ocs_check_tape -f fnalu -w -v xyz123

Chapter 5

FNALU Applications

5.1 Applications Unusual to FNALU

There are a number of applications unusual to FNALU, or that work somewhat differently here than on other systems at the lab. This section is not intended to replace documentation for these software packages, but is intended solely to give you enough information to start them up, and to give pointers to where you can find out more.

As far as is possible we've attempted to make **all** software products be set up with the normal FUE **setup** command. So, to, for example, use the **emacs** editor you would type in

```
setup emacs
```

```
emacs
```

which would set up the emacs system and then execute the editor. Most software on FNALU will set up in a similar fashion.

5.2 Framemaker

The Framemaker publishing tools are currently available on the Sun and SGI systems in FNALU. Framemaker version 4 is installed on the SGIs and Framemaker version 5 on the Sun. The syntax to start up framemaker is a little peculiar. In general you will only have to type

```
setup frame
```

```
maker
```

Since this is an X application, you should first be certain that you have your DISPLAY environment variable set up properly.

Note that Framemaker is a network licensed software product. We currently only hold licenses for five seats for Framemaker version 4 and five seats for Framemaker Version 5, so after the five licenses are used no one else can start the product up.You will receive a message when you start the product if that is the case. You must then wait until a license is freed or if you know someone who is using a license, you could ask them to relinquish it. In an attempt to keep people from hogging licenses indefinitely, the licenses are set so they are given up after 12 hours of idle time. If your license is given up that way, you will no longer be able to work in Frame. To get a license back, click on the info box on your frame startup window, and go to the license menu. There you can either give up your license if you so choose, or ask for one if yours has expired.

It is possible, however, that there will still be times of peak use where all the licenses are in use. Please try to be patient. Let us know if the number of licenses becomes an issue for your FNALU use, and we will either try to acquire more floating licenses, or may suggest that you might want to purchase a personal license instead, which is fairly inexpensive and ties a license to your own use permanently. You can even install personal licenses yourself once you've bought one. If you want further information on this please send email to **uss-help@fnal.gov** as you would with any other request.

5.3 DOS and Windows emulation services

FNALU provides limited emulation of Microsoft Windows for users to run Microsoft Office. Currently we support *only* Microsoft Office under Windows. The windows emulation is available only on the Sun interactive system, **fsui01**, and is somewhat complex to set up. Because of this, we are not providing full setup instructions here, but they are available as a postscript file in /afs/fnal/files/docs/wabionfnalu.ps. You can display windows applications to any reasonably configured X device, and expect performance to be similar to a 386DX PC, assuming that fsui01 isn't heavily loaded.

5.4 Explorer

We are currently installing the Explorer software product on FNALU, and it should be available on both fsui01 and fsgi02 by the beginning of March, 1996 You should be able to access it by executing

setup explorer

explorer

Please check in fnal.sys.fnalu.announce for information on the availability of Explorer on the FNALU cluster.

5.5 IMSL Libraries

The IMSL FORTRAN libraries will be available on node fsgi02 some time in March, 1996. Note that this is a license-managed product, and a limited number of programs linked against the IMSL libraries will be runnable at one time.

5.6 MACSYMA

Macsyma should also become available on FNALU some time in March 1996. Again, please check fnal.sys.fnalu.announce for details on release time.

5.7 Email

Email on FNALU is the same as email on any other FERMILAB system in most respects. You should read the UNIX at FERMILAB document for details. One difference is that FNALU will shortly have a mail server that speaks both IMAP and POP protocols, which can be used to store your email. In this way, you can keep all of your email in one place and access it from any UNIX or PC system, even if you are off-site. Mail clients speaking POP will only be able to use the mail server as their email inbox, but users with IMAP compatible clients will be able to keep both their inbox and mail folders on the IMAP server. The server node will be called **imap1.fnal.gov**, and should go into production in mid-March, 1996. More details will follow. Please watch fnal.sys.fnalu.announce for details. Computing Division plans to support the IMAP protocol on UNIX systems with the pine mail client, and POP protocol with exmh.

Chapter 6

FNALU FAQ

A lot of the same questions about using FNALU keep on cropping up. To make your life and the help desk person's life easier, this document is a list of frequently asked questions and answers. On the net, these documents are usually referred to as **FAQ**'s. If you have a problem on FNALU, please look here first.

6.1 AFS

One of the things that is different about using FNALU is the us of the AFS file system. Most of the questions about using FNALU are about AFS, because it is the one feature of FNALU that is different from the other systems at FERMILAB.

6.1.1 What are tokens?

In the AFS system, kerberos-like authentication is used. In this system, you are assigned a token at login time. This token applies to your **pagsh**, which is a group of processes that are forked from your login process. Therefore, you should only need to obtain your token once for a login session, and all of the programs you run in that login session will be authenticated by your initial login. On most of the systems here that actually run AFS, your login token is assigned automatically when you log in to the system, so you need do nothing special.

Tokens currently are set to expire in 25 hours, so if you stay logged in for a long time, you will lose your permissions to read and write files.

6.1.2 What do I do when my tokens expire?

You just have to reauthorize yourself by using the **klog** command. It will prompt you for your password. Once you've klogged, the new token will apply to all of your programs automatically.

6.1.3 How do I tell if my token is still valid?

If you type in the **tokens** command, it will list out all of the tokens you have that are valid, as well as when they expire.

6.1.4 Why don't my file permissions make any sense?

The file permissions in AFS are primarily controlled by the AFS ACL's, not by the permissions on individual files. The only part of the file permissions that matters in AFS is the owner bits, and these bits apply to **all** users of the file. ACL's, on the other hand, apply to the directory, not to individual files. So the control of access is by and large mediated by the ACL's on the directory the file is contained in.

To look at the ACL applying to the directory, use the command

fs listacl *directory*

where directory is the name of the directory you want to look at.

6.1.5 How do I get more information on AFS?

There are several choices. The help desk has a small quantity of laminated AFS reference cards that list the syntax of the most common AFS commands. There is a full set of AFS manuals in the Computing Division library. There are also online manual pages on the systems running as AFS clients

6.1.6 Why can't I read the AFS manual pages?

Manual pages are available on-line on the AFS systems, but you'll find that the names of the manual pages are a little less than clear. Since there are very few AFS commands, and they have a large number of options, the man page names are in the format of

command_option

For example, the man page for the **fs setacl** command is found by typing

man fs_setacl

This is a little confusing, but it works pretty well in practice.

6.1.7 Other Sources of AFS Information

Finally, if you have a way to read WWW pages, there is a WWW link to the CERN WWW help page for AFS (some of which doesn't apply to us) as well as a large number of other AFS documents. These are all accessible at url

http://dcdsv0.fnal.gov:8000/hanson/afs.html

More AFS information will be added to this html page as it is found.

6.1.8 What does it mean if I get an error message saying something like "afs: Waiting for busy volume 536870945 in cell fnal.gov"

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

6.2 Getting help

As on any of the other centrally-supported systems here, your first point of contact should usually be the help desk. Call them (ext. 2345) and they will log the call, and either answer your question or make sure that someone who can answer it will get back to you.

We prefer that **all** queries go through the help desk. However, you may also send email to the administrative alias for the FNALU cluster (**fnalu-admin@fibi01.fnal.gov**). However, please be aware that email queries will receive a lower priority than calls to the help desk. Your best bet is always to contact the help desk.

Off-hour queries should ALWAYS be called in to 2345, and will roll over to the operators if no one is manning the help desk. This is the only way you will receive service during off-hours for any sort of system problem. Off-hour calls should be restricted to actual problems with the system, rather than queries about how something on the system works. During off-hours, the responsible person who is on call will be paged.