Fermilab

GU0001

UNIX at Fermilab

Release 3.0

November 26, 1997

Computing Division
Fermi National Accelerator Laboratory

Compiled by Anne Heavey

ABSTRACT

The *UNIX at Fermilab* manual has several goals: to help you get started as quickly and painlessly as possible in the UNIX environment here at Fermilab, to make you aware of the UNIX resources available at Fermilab, and to help you use UNIX easily and efficiently, whatever your particular needs.

We discuss the most commonly used UNIX concepts, commands and tools, and provide enough information to allow you to execute commands and perform tasks typically required by the Fermilab user community. We refer you to more specialized Fermilab documents as needed, and to commercially available sources of information for details on commands and features that will allow you to exploit the more sophisticated features of the UNIX operating system not covered in this document.

Revision Record

August 1990	Original Draft
March 1996	Total overhaul; then several minor revisions through September 1996
November 1997	Significant changes relative to 9/30/96 release (text changes *within* sections are not noted here). Chapter and section numbers correspond to new release:

Ch 1 (Introduction): new FNALU section (1.5)

Ch 2 (Getting Started): moved AFS information into AFS chapter (7)

Ch 3 (Information Resources): new Helpdesk section (3.5)

Ch 6 (File System): new tar section (6.3.7); moved AFS information into AFS chapter (7)

Ch 7 (AFS): new chapter

Ch 9 (Work Environment): added section on X-terminal support (9.5)

Ch 10 (UPS Products): new chapter (replacement)

Ch 11 (Editors): new information for the editor xemacs (11.3.2)

Ch 12 (Mail): new "Notice of Upcoming Changes"

Ch 14 (Batch): new chapter

Ch 15 (Tapes): condensed OCS coverage; added sections for FTT (15.4) and FMB (15.5)

Ch 16 (Sofware Dev): new section on the debugger GDB (16.11.3)

Ch 18 (Code Mgmt): condensed the UCM section (18.2)

Removed appendix on FUE Conformance Levels

Appx B (UPS Overview): new appendix

Appx F (mh and exmh Custom): new section on auto reply to mail (F.3)

Acknowledgments

In 1995-1996, the Fermilab Computing Division's VMS Migration Task Force was responsible for updating *UNIX at Fermilab* (from the June 1991 release) in time for the mass migration from FNALV. That task force included: Pushpa Bhat, Joel Butler, Lauri Loebel Carpenter, Mike Diesburg, Dave Fagan, Ken Fidler, Steve Hanson, Rob Harris, Joy Hathaway, Anne Heavey, Art Kreymer, Paul Lebrun, Frank Nagy, Judith Nicholls, Dane Skow, Mike Stolz, Roy Thatcher, Al Thomas, Matt Wicks, and Steve Wolbers.

Other contributors/reviewers included Liz Buckley-Geer, Chuck DeBaun, Lynn Garren, Randy Herber, Peter Kasper, Marc Mengel, Clyde Moseberry, Gene Oleynik, Ruth Pordes, Marilyn Schweitzer, Jonathan Streets, and Margaret Votava.

The *CERN UNIX User Guide*, after having lifted information from the early release of *UNIX at Fermilab*, was in turn used to provide much of that information back to us appropriately updated! Naturally, we lifted additional useful information from CERN's well-constructed manual. The CERN document *Shell Choice, A shell comparison* was also used.

This 1997 revision of *UNIX at Fermilab* includes contributions from many of the people listed above. Additional contributors/reviewers include Eileen Berman, Jim Fromm, Lisa Giacchetti, Jeff Kallenbach, John Marraffino, Laura Mengel, Liz Sexton-Kennedy, and Alan Wehmann.

We'd like to thank the readers who have taken the time to send us suggestions about the manual.

Table of Contents

Appendix A. VMS Migration for the Impatient

So, you've decided you're ready to convert (or you've run up against a deadline!), but you don't know the first thing about UNIX. Here's enough information to get you moved over. You can use the rest of the manual to learn about UNIX afterwards.

See the introductory remarks to Chapter 12 before continuing.

Read this entire appendix before running any of the shown commands!!

Keep these typeface conventions in mind as you read the commands:

`typewriter-bold`	In text, used to indicate commands and prompts. In command formats, indicates what the user types "as is".
bold-italic	In command formats, indicates variables for which the user must make context-specific substitutions.

A.1 The Two Necessary Commands

To convert to UNIX, you need to move your mail and your files over from your VMS node (***vmsnode***). To accomplish this while preserving the mail folder and directory structures you had set up on VMS, enter the following two commands exactly as shown from your new UNIX node. (In case you really haven't read any part of this manual yet, the `%` represents the UNIX prompt.)

For your mail, use the command:

`%` **`setup mh ; fvms2mh -vms`** *vmsnode*

Note that ***vmsnode*** must be an individual node, not a cluster alias. For your files, the command is:

`%` **`mkdir vms ; rcp -r`** *vmsnode*`:'[...]*.*'` **`vms`**

A.2 OK, What's the Catch?

You're right, there's a little more to it. You must have selected a UNIX host to which you will copy your mail and other files. On the FNALU cluster, we suggest that general users choose the FSUI02 node. You must have learned to log in to UNIX, edit and manage your files, and must have selected a mail reader (presumed to be **mh**, **exmh** or **pine**).

In addition:

- You must have **rcp** access from your UNIX to your VMS account. This takes two steps:

1 Create a `.RHOSTS` file in your VMS `SYS$LOGIN` directory containing one line for each UNIX host that needs automatic access to your VMS account. If your username is different on any machine, include a second line with your username. For example, the contents might look like:

```
fsgi02.fnal.gov
fsgi02 aheavey
fdei01.fnal.gov
fdei01 aheavey
```

2 Make sure your `SYS$LOGIN:LOGIN.COM` does not print anything out. To ensure this, include the following line at the top of the file:

```
IF ( F$MODE() .EQS. "OTHER" ) THEN EXIT !  'F$VERIFY(0)
```

If you can no longer log in to edit your files, send mail to *compdiv@fnal*, and someone can fix it for you.

- For the mail copy, be sure you are using **mailtools v2_2** or later, as installed on the FNALU nodes and most other public nodes at Fermilab. Earlier **mailtools** versions do not support the **`fvms2mh`** command name described here, and may have quota and cleanup problems. To see whether **mailtools v2_2** is current, type **`ups list -a mailtools`**. If it is not current, select it explicitly by typing **`setup mailtools v2_2`**.
- The utility **fvms2mh** brings your mail folders over in **MH** format. If you plan to use **pine** but want to leave your folders in **MH** format, **pine** must be configured to recognize them. To do this, go to **pine**'s SETUP CONFIGURATION menu. There are many options available. Your configuration list should include at least the following:

```
inbox-path          = #mh/inbox
user-domain         = fnal.gov
folder-collections  = mail/[]
                      #mh/[]
[X]  enable-aggregate-command-set
[X]  enable-alternate-editor-cmd
[X]  quell-user-lookup-in-passwd-file
```

- If you choose to use **pine** exclusively, you will want your folders in the **pine** format. First complete the configuration above (with the exception of `inbox-path`). You can move entire folders between the **pine** and **MH** collections as follows:

1 While in **pine**, go to the `FOLDER INDEX` screen for the desired folder.

2 Enter **`;`** (semi-colon) to issue the **pine** Select command.

3 Enter **`A`** to select All messages.

4 Enter **`A`** to Apply the command.

5 Enter **`S`** for Save.

6 Use <Ctrl-n> or <Ctrl-p> as necessary to choose the destination folder collection, then enter the new destination folder name. Or just type in a folder collection and name. Respond `Y` to create the new folder.

A.3 Whoa! Too Fast!

If converting in two easy steps doesn't suit your style, you can take somewhat smaller steps.

1) You should probably first forward your current new mail to UNIX, and get used to the **exmh** or **pine** mail readers.

2) Once you've stopped receiving VMS mail for a while, run the first command in section A.1. As a yet more conservative option, check out the "semi-automatic" mail conversion method in Appendix H.

3) Then, when you're happy with the UNIX copy of your mail, delete the original VMS mail files. You may wish to back them up or archive them before deleting.

4) Now to convert your files, run the second command in section A.1 *exactly as shown*. Changes to any of the wildcarding is likely to do nasty things (for instance, moving the oldest rather than most recent version of each file, or dumping everything into a single directory rather than keeping your directory hierarchy intact).

Appendix B. UNIX Product Support (UPS) Overview

In this appendix we discuss the Fermilab product support structure, **UPS**. We recommend that you read and understand this material before performing any of the tasks described in Chapter 10.

The information in this appendix has been taken from *UNIX Product Methodology at Fermilab: Guide to Using UPS v3 and UPD v2 for Product Maintenance, Installation, Distribution and Development* (GU0014), but covered here in much less detail, appropriate to the different audience. We refer you to that document if you need further information.

> **Notice of Upcoming Changes**
>
> **UPS** and **UPD** are currently undergoing redevelopment with a significantly different design. The new versions and accompanying documentation are due for release in the first half of 1998.

B.1 Introduction

The methodology and infrastructure for product support and distribution under FUE is provided via a software support toolkit called UNIX Product Support (**UPS**). **UPS** was developed with the goal of providing a uniform and consistent interface for the management, distribution, installation and use of all the UNIX software it makes available.

UPS consists of two parts:

- one or more databases
- a set of procedures/programs to manipulate database

A **UPS** database is a directory containing an ASCII file for each individual product. This file contains information about and pointers to all of the installed copies of the product. Product maintainers use **UPS** commands to add, delete and modify product information in these files (these activities are beyond the scope of this document). Products supported under this structure are called **UPS** products. A **UPS** product is comprised of one or more applications, each of which is invoked by its associated command or commands[1].

UNIX Product Distribution (**UPD**) is a companion product to **UPS**, and provides the functionality for distributing products, generally stored as tar files, from a remote host to a local system. A convenient menu interface is provided that includes the **UPD** and **UPS** functionality necessary to complete an entire process of listing available product instances on the distribution node, copying a product instance from the distribution node, installing it on the local node, and declaring it to the local **UPS** database. The **UPD** interface also can be used to list the files in or extract a file from a remote tar file.

---

1. Most applications have a single invoking command. However some applications have a set of commands, each of which invokes a different function. See section 10.2.4.

B.2 The UPS Environment

On a FUE-compliant system the standard default login files (also known as *Fermi files*[1]) set your environment to facilitate **UPS** operations, among other things. Regarding shells, **UPS** supports both UNIX shell families, C (Berkeley) and Bourne[2]. At login time, the environment variable *UPS_SHELL* gets set to indicate the family of the shell in use. Any shell within that family can then be used for **UPS** operations.

The Fermi files are designed for interactive use only and will break if they are called from a non-interactive script. You need to explicitly include the following actions in the order shown in any script that you will use to setup products (i.e. in which you use the **`setup`** command to access one or more products; **`setup`** is described in section 10.2.2):

For the C shell family:

```
source /usr/local/etc/setpath.csh
source /usr/local/etc/setups.csh
```

For the Bourne shell family:

```
. /usr/local/etc/setpath.sh
. /usr/local/etc/setups.sh
```

The `setpath` script sets a reasonable default starting path. The `setups` script (read "set **UPS**") sets the **UPS** environment variables described below[3].

The login files also set the alias for **`setup`**. **Note that if you change shells or processes, this alias is no longer available to you.** In this case you need to manually re-source the `setups` script in your new shell.

If you have made any changes to your login files, make sure that the following lines are still included before using **UPS**:

For the C shell family:

`.cshrc`	includes	`source /usr/local/etc/fermi.cshrc`
`.login`	includes	`source /usr/local/etc/fermi.login`

For the Bourne shell family:

`.profile`	includes	`. /usr/local/etc/fermi.profile`
`.shrc`	includes	`. /usr/local/etc/fermi.shrc`

Three environment variables get set by the setups script at login:

PRODUCTS	If only one **UPS** database is defined, this points to it; if two or more are defined, this variable can be set as the space-separated list of **UPS** databases. The order of the databases in this list reflects the order of precedence for accessing products.
UPS_DIR	This points to the top level directory (called the product root directory) of the current instance of **UPS** (read sections B.8 through B.10 -- they're short! -- to understand the term "current instance").
UPS_SHELL	As mentioned above, this indicates the shell family in use; it is set to either `sh` or `csh`.

1. See Chapter 9 and Appendix C for descriptions and listings of the Fermi files.
2. The C shell family includes **csh** and **tcsh**; the Bourne shell family includes **sh**, **ksh**, **bash** and **zsh**. **zsh** is not supported at Fermilab.
3. These scripts are included in the FUE package **systools**.

You can use the environment variable *PRODUCTS* to list all the available products. For example, if it is set to the following list of **UPS** databases:

```
/home/dcdsv0_lv0/aheavey/ups_play/ups_database/declared
/usr/products/ups_database/declared
/afs/fnal/products/ups_database/declared
```

then the command:

% ls $PRODUCTS

will yield information similar to the following (edited for brevity)[1]:

```
/afs/fnal/products/ups_database/declared:
acrobat         frame           lund            tcl
admintools      fslib           mad             tclx
afsemu          ftcl            mailtools       tex
...text removed...
flint           ktevana         softwindows     xpdf
fmb             lapack          stdhep          zephyr
fmss            libgpp.obs      systools

/home/dcdsv0_lv0/aheavey/ups_play/ups_database/declared:
cedit     exmh      fermitpu  glimpse   juke      www

/usr/products/ups_database/declared:
altuascvs    fulib        lund         qq           tk           www
chekker      funkern      mbone        req          tpu          xemacs
conv2html    futil        mgui         sdsscvs      uas_build    ximagetools
...text removed...
flint        html         ntp          tcl          upp
fmb          ispell       ocs          tclx         ups
fonts        juke         olscvs       teleserver   webservers
```

When you setup a product, the environment variable *{PRODUCT}_DIR* gets defined, where *{PRODUCT}* is the name of the product in upper case, for example *WWW_DIR*. This variable points to the root directory of the particular copy of the product that you setup; for example after running:

% setup www v2_7

the variable *WWW_DIR* is defined as something like `/{ups_dirs}/products/SunOS+5/www/v2_7`. You can then run a command like the following:

% ls -al $WWW_DIR

1. Notice that the **ls** command does not list the databases in the order of precedence according to the *PRODUCTS* variable, but rather, alphabetically.

which returns a list of the files and subdirectories in the product root directory for **www** version v2_7, for example:

```
total 40
drwxr-xr-x   7 products g150        2048 Jun 19 18:54 .
drwxrwxrwx   5 root     root        2048 Sep  9 23:16 ..
-rw-r--r--   1 products g150         165 Jun 18 11:31 BUILD_INFO
-rw-r--r--   1 products g150        4703 Oct 16  1995 Makefile
drwxr-xr-x   2 products g150        2048 Aug  9 18:19 bin
drwxr-xr-x   2 products g150        2048 May  5  1996 etc
drwxr-xr-x   2 products g150        2048 May  5  1996 include
drwxr-xr-x   4 products g150        2048 May  5  1996 lib
drwxr-xr-x   2 products g150        2048 May  5  1996 ups
```

B.3 UPS Products

Products distributed and managed by the **UPS** system on a distribution or user node are often called ***UPS** products*. **UPS** products can be maintained on different disks, and/or in different directories. To be included under the **UPS** umbrella on a node or cluster, products must be installed (preferably in an area specially designated for products, although **UPS** does not require this) and declared to a **UPS** database. Product declaration can be done in **UPD** (see section 10.3.1) or via the **UPS** command **`ups declare`** (not covered in this manual).

On the distribution node, all the constituent programs and/or data files of a particular copy of a **UPS** product (collectively called an *instance*, see section B.8) are grouped into a directory tree which has been packed up into a single binary file for distribution, generally a tar file. The structure of the directory tree is not dictated by **UPS**, but generally it includes (at least) areas for the executables (`bin`), for files accessed by **UPS** tools (`ups`), and for documentation (e.g., `ups/toman`, `docs`, `html`). During product installation on a user node, this file can be unwound using **tar** (optionally with **UPD** as an interface) and installed into its product root directory.

B.4 UPS Databases

A **UPS** database is, simply put, a directory. Within the directory, there is a *product file* for each individual product that is accessible via **UPS** on the system. A product file, described in section B.5, contains identifying information for, and pointers to, each of the installed copies of a product. **UPS** commands refer to the database directory via the environment variable *PRODUCTS*, described in section B.2.

The environment variable *PRODUCTS* can be set to point to one or to several directories, thus allowing support for multiple databases. This allows users to maintain one or more private databases in addition to or instead of the common one(s). Creating additional databases is discussed in GU0014. In **UPS** operations, the databases are searched in the order they appear in *PRODUCTS*.

B.5 UPS Product Files

The information that **UPS** needs in order to identify, locate and retrieve a product and its requirements (requirements are described in section B.11) resides in an ASCII *product file* specific to that product in the **UPS** database. The name of each product file is simply the product's name (see the list of product files obtained by the command **`ls $PRODUCTS`** in section B.2). The file identifies the product, the UPS database version, the declared instances of the product and the defined chains. Instance is defined in section B.8, and chain in section B.10.

B.6 Product Versions

UPS supports multiple concurrent versions of software products. Each version of a product is installed and accessed independently of other versions. When a new version of a product becomes available, an existing directory tree is not replaced with another; rather, a branch is added for the new version.

Maintaining multiple, self-contained versions of a product on a single system is often necessary and/or desirable for several reasons. For example, in critical situations like data acquisition, if something goes wrong, you don't want to lose time. **UPS** allows you to back out of a new software version completely and assuredly, and immediately start up a previous tried-and-true version. You can also overlap development and production use of a product, and support many users sharing workstation resources.

B.7 UNIX Operating System Flavors

B.7.1 What is "Flavor"?

Many programs require separate compilation for the different UNIX operating systems. In **UPS**, we maintain different directory trees for the separate compilations (and related files) of the same product. We therefore need to distinguish between OS-dependent compilations. To indicate the OS dependency, we use the term *flavor*. This additional term allows us to maintain the same product name and version across the different operating systems and releases, which is desirable since the same program source files are used in the separate compilations.

On a given system, several different copies of a product may exist. When you use **UPS** commands to manipulate or use a product, the system needs to have enough information to select the appropriate one. Many **UPS** commands support a **`-f`** option allowing you to specify flavor-related information.

B.7.2 Simple Flavors

Simple flavors (as opposed to extended flavors) are the UNIX operating system names, as returned by the command **`funame -s`** (for example IRIX or SunOS). This is sufficient for products which can run on multiple releases of a single OS. For products which have no compiled programs, and are thus operating system-independent, a special flavor of NULL is used.

B.7.3 Extended Flavors

In addition to differentiation by OS, some products require separate compilations for different releases of the same operating system (e.g., IRIX+5 and IRIX+6). For this purpose, **UPS** supports *extended flavors*, which are designations that include both OS (simple flavor, e.g., IRIX) and OS release (e.g., +5). This allows a mixed cluster (e.g., IRIX+5 and IRIX+6) to share a single **UPS** database.

Extended flavors may further include information about options used by the product developer at compilation time (e.g., **`+debug`**).

Extended flavor is designed to accomplish two goals:

- identify the most appropriate compilation of a product for the given operating system and release
- allow the product developer to specify additional compilation options such as **`debug`**, **`optimized`**, etc.; the user can then select and run one of these compiled instances of a product for special purposes

B.8 Instances

Each installed, declared copy of a product in **UPS** is called an *instance* of the product. Each instance has a unique combination of product name, version and (extended) flavor. The concept of *chains*, discussed in section B.10, allows users to easily access the appropriate instance, without needing to remember its version number and (extended) flavor.

B.9 Flavor Specification

When a **UPS** command is issued, the system must determine which product instance or instances to act upon. For some **UPS** commands multiple instances can be retrieved, as is often the case for **`ups list`**, for example. The product name and version are generally specified unambiguously, but the extended flavor can be specified such that it requires some interpretation by the system. This allows **UPS** to do some of the instance selection work for you. In fact for some operations you can choose not to specify flavor at all and let **UPS** determine it completely. If you need to specify the flavor (generally you shouldn't) use the **`-f`** option:

`%` ***ups_command*** **`-f`** ***flavor*** `[`***other options***`]` ***product***

The system will then only retrieve an instance exactly matching ***flavor***.

B.10 Chains

We mentioned earlier that **UPS** supports multiple versions of software products on a machine. End users do not find it convenient to specify product version numbers each time they setup a product. This is especially true if product setups are needed at login. Most users want to run the latest, tested, approved version of products without having to keep track of the version numbers.

To allow users to specify the version of a product according to its *status* rather than by its version number, **UPS** supports *chains* to product versions. A chain is a **UPS** database entry (i.e. a clause in a product file) that points to a particular instance declaration in the database (an instance clause in the same product file). It must match the (extended) flavor and version of the instance declaration exactly. It "attaches" a chain name to a product instance, thereby tagging the product instance according to its status.

Five statuses, or chains, have been defined for use: current, new, test, old, and development.

Chain	Option	Usage
current	**`-c`**	default instance recommended for general use
new	**`-n`**	tested instance that is not yet current
test	**`-t`**	instance installed for testing

Chain	Option	Usage
development	`-d`	instance under development
old	`-o`	older instance that was previously current

In **UPS** commands, the command line option associated with the desired chain (status) is used to specify the product instance to retrieve. Using chains is optional, but recommended. Both chained and unchained instances of a product may be declared to **UPS**; the user can still retrieve any instance, chained or not, by using its product version number.

B.11 Product Dependencies (Use and Build Requirements)

Many **UPS** products rely on other products being installed, declared and setup for proper functioning or for use of special features. These are generally referred to as *product dependencies* or *use requirements*[1]. The coupling of products with their dependencies facilitates product setup. You need only setup a single **UPS** product to access any and all of the products listed as its dependencies. Each of the dependencies can be setup in its own right, as well.

Requirements can exist across databases. The databases must be included in *$PRODUCTS* for this to work. When you declare a requirement, **UPS** searches the databases in *$PRODUCTS* and uses the first instance of the required product it finds; you cannot specify the database in which a requirement resides.

Multiple levels of requirements are possible, and use requirements are setup recursively by default. As an example, the mail product **exmh** has several use requirements, one of which is **www**. **www** in turn has use requirements, one of which is **ghostview**. When you setup **exmh**, you also implicitly setup all its use requirements including **www**, and all the next level use requirements including **ghostview**, and so on.

B.12 Notes on Setup and Unsetup

The **`setup`** command was discussed in section 10.2.2. As simple as it is to use in everyday situations, **`setup`** contains some subtleties. It is equipped with a host of options that allow you to specify various parameters. We refer you to the man pages for specifics; here we attempt to provide a summary/clarification of the available options. You can specify:

- which chained instance of the product to setup
- which use or build dependencies to setup along with main product (you cannot setup both)
- an option string to pass to the setup script via the environment variable *UPS_OPTIONS*
- enable any optional functionality in the setup script.

By default, the use requirements are setup recursively along with the main product unless the **`-j`** option is used to restrict the setup to *just* the specified product and none of its dependencies. **`setup`** also comes equipped with a **`-b`** option that allows you to setup the build requirements *instead of* the use requirements. Build requirements are not setup recursively; in fact they are not generally distributed with the products. Most users never use **`-j`** or **`-b`**.

1. There are also such things as *build requirements*. These need to be in place only for building the main product, and are usually omitted on end-user systems.

The last two features are rarely if ever implemented, and require knowledge about the setup script by the user. They are included here for completeness only.

When you unsetup a product, any use-dependent products get deactivated (unsetup) automatically at the same time unless the **-j** or the **-b** option is used:

-j unsetup *just* the specified product and none of its dependencies

-b unsetup the build requirements of the product (useful only when the product was setup with the **-b** option)

The man pages list other options, but in practice, unsetup scripts aren't coded to exploit them.

Appendix C. Fermi Login Files

This appendix contains file listings of the FUE-customized default login files (the “Fermi Files”) used to set up your UNIX environment.[1] If you are on a FUE-compliant system, you are supplied with a copy of each file in your home directory (except for the `fermi.*` and `setup.*` files which are executed directly from `/usr/local/etc`). The files are reprinted here in their entirety except for the copyright disclaimers.

C.1 C Shell Family

C.1.1 .cshrc

The default FUE `.cshrc` file is found in `/usr/local/etc/stdcshrc`.

```
# .cshrc default settings for all users

# @(#) stdcshrc 1.10 Delta: 93/08/19 14:51:27 Extraction 94/07/06 15:16:15 @(#)

# execute fermi.cshrc first
if ( -r /usr/local/etc/fermi.cshrc ) then
        source /usr/local/etc/fermi.cshrc
endif

# Place any items that you want executed even for non-interactive use here

#skip if not interactive shell
if ( $?USER == 0 || $?prompt == 0 ) exit


set noclobber            #prevent overwrite when redirecting output
set ignoreeof            #prevent accidental logouts

# Have mail point to fermimail which is a Berkely mail version. Some packages
# expect mail to be a System V version of mail. If you are having problems
# of this type, you may need to remove the alias.
alias mail          fermimail       #Fermi recommended Mail

#Define various aliases; user selects desired alias by removng the # sign
#alias a            alias
#alias killit       kill -9         #guarantees that a process is killed
#alias h            'history | tail'
#alias ll           ls -l
#alias la           ls -a           #see hidden files
#alias lf           ls -CF          #check file TYPE (exe, dir ..)
#alias rmi          rm -i           #confirm before deletion
#alias home         cd              #HOME
```

1. The `setpath.*` and `setups.*` files reprinted here are part of the FUE product **funkern v5.0**. All other files are part of FUE’s **systools v4.3**.

```
#alias side         'cd ../\!*'     #side
#alias down         'cd \!*'        #down
#alias up           cd ..           #up
#alias cpi          cp -i           #no overwrite of output file
#alias cd          'cd \!*;echo $cwd'
#alias mvi          mv -i           #confirm before moving
#Next alias replaces standard info command on SGI platforms
#alias info         Info            #get list of info articles
#
#VMS type commands
#
#alias dir          ls -l
#alias copy         cp
#alias rename       mv
```

☞ `.cshrc` executes `/usr/local/etc/fermi.cshrc`.

C.1.2 fermi.cshrc

The default FUE `fermi.cshrc` file is found in `/usr/local/etc/fermi.cshrc`.

```
# @(#) fermi.cshrc 4.32 Delta: 94/09/29 22:22:46 Extraction 94/09/29 22:34:39 @(
#)

# fermi.cshrc settings for all users; will be called by the default .cshrc

set MACH_ID=`/usr/local/bin/funame -n`
set MACH_TYPE=`/usr/local/bin/funame -m`
set MACH_OS=`/usr/local/bin/funame -s`

#Determine if this is the first time the fermi.cshrc file has been executed
#on this machine
if ($?FERMICSHRC) then
        if ( $FERMICSHRC == "$MACH_ID" ) then
                set firsttime=0
        else
                set firsttime=1
                setenv FERMICSHRC "$MACH_ID"
        endif
else
        setenv FERMICSHRC "$MACH_ID"
        set firsttime=1
endif

#Execute the things that are only done the first time through
if ( $firsttime == "1" ) then
        # set path
        if ( -r /usr/local/etc/setpath.csh ) then
                source /usr/local/etc/setpath.csh
        endif
        #
        # Establish MANPATH
        #
        setenv MANPATH /usr/products/catman:/usr/products/man:/afs/fnal/products
/catman:/afs/fnal/products/man:/usr/catman:/usr/man:/usr/share/catman:/usr/share
/man:/usr/local/catman:/usr/local/man

endif

#Execute the items done each time (basically only aliases)

#
# Setup UPS
```

```
#
if ( -r /usr/local/etc/setups.csh ) then
        source /usr/local/etc/setups.csh
endif

switch ($MACH_OS)
case IRIX:
        alias fermimail '/usr/sbin/Mail'        # Berkely mail
breaksw
case SunOS:
        alias fermimail '/usr/ucb/Mail'         # Berkely mail
        alias man       'man -F'
breaksw
case AIX:
        alias fermimail '/usr/ucb/Mail'                 #Berkely Mail
breaksw
case ULTRIX:
        alias fermimail '/usr/ucb/Mail'                 # Berkely Mail
breaksw
case OSF1:
        alias fermimail '/usr/bin/Mail'                 # Berkely Mail
breaksw
case HP-UX:
        alias fermimail '/usr/bin/mailx'        #Berkely Mail
breaksw
default
        if ( -x /usr/ucb/Mail ) then
                alias fermimail '/usr/ucb/Mail'         #Berkely Mail
                endif
        if ( -x /usr/sbin/Mail ) then
                alias fermimail '/usr/sbin/Mail'        #Berkely Mail
        endif
        if ( -x /usr/bsd/Mail ) then
                alias fermimail '/usr/bsd/Mail'         #Berkely Mail
        endif
        if ( -x /usr/bin/mailx ) then
                alias fermimail '/usr/bin/mailx'        #Berkely Mail
        endif
        if ( -x /usr/bin/Mail ) then
                alias fermimail '/usr/bin/Mail'        #Berkely Mail
        endif
breaksw
endsw
#
# Check for existance of local.cshsrc file. This is the file admins should
# put in any changes, additions, etc. so that they don't need to re-edit this
# file every release.
#
if ( -r /usr/local/etc/local.cshrc ) then
        source /usr/local/etc/local.cshrc
endif
```

☞ `fermi.cshrc` calls `/usr/local/etc/setpath.csh` to set the shell variable *path,* and it executes `/usr/local/etc/setups.csh` to set up **ups**.

C.1.3 setpath.csh

The default FUE `setpath.csh` file is found in `/usr/local/etc/setpath.csh`.

```
# @(#) setpath.csh 1.7 Delta: 95/03/22 22:13:29 Extraction 95/03/22 22:13:53 @(#
)

if ($?HOME == 0 ) then
        setenv HOME /
```

```
endif

set path=""
foreach DIR ( /usr/sbin \
        /opt/SUNWspro/bin \
        /usr/ccs/bin \
        /usr/lang \
        /usr/bsd \
        /bin \
        /usr/bin \
        /usr/lbin \
        /usr/ucb \
        /etc \
        /usr/etc \
        /usr/afsws/bin \
        /usr/openwin/bin \
        /usr/bin/X11 \
        /usr/kinet/bin \
        /usr/local/bin \
        /usr/bin/mh \
        /usr/sccs )
        if ( -d $DIR ) then
                set path=($path $DIR)
        endif
end
#
# Add your HOME/bin and bin."flavor"
# Put your Sun executables in a bin named bin.SunOS,
#     your IRIX executables in bin.IRIX,
#     your AIX     "         " bin.AIX,
#  ...etc...
#
# Do not do this if you are root, because //bin is your
# $HOME/bin, and do not put current directin in root's path
if ($?MACH_OS == 0) then
        set MACH_OS=`funame -s`
endif
if ( "$HOME" != "/" ) then
        foreach DIR ( $HOME/bin.$MACH_OS $HOME/bin )
                if ( -d $DIR ) then
                        set path=($DIR $path)
                endif
        end
        set path=($path .)
endif
```

C.1.4 setups.csh

The default FUE `setups.csh` file is found in `/usr/local/etc/setups.csh`.

```
#Set environment for UPS
if (! -r ~/.noupsproducts) then #start if .noupsproducts
        set MACH_OS=`funame -s`
        set setprod=no
        if ( $?PRODUCTS == 0 ) then
                set setprod=yes
        else
                 set fchar=`echo $PRODUCTS | cut -c1`
                 if ( "x/" != x"$fchar" ) then
                        set setprod=yes
                 endif
        endif
        if ( $setprod == "yes" ) then #start if $PRODUCTS already set
           if ( $?UPS_EXTRA_DIR == 0 ) then
```

```
              set UPS_EXTRA_DIR=""
          endif
               unsetenv PRODUCTS
               # Add other directories areas that might be product databases
               # to the foreach list
               foreach PROD ( $UPS_EXTRA_DIR \
                       /`funame -n`/products/ups_database/declared \
                       `logdir products`/ups_database/declared \
                       /usr/products/ups_database/declared \
                       /afs/fnal/products/ups_database/declared)
                       if (-d $PROD) then
                               if ($?PRODUCTS) then
                                       set exists=`echo $PRODUCTS | grep -c $PR
OD`
                                       if ( $exists == "0" ) then
                                               setenv PRODUCTS "$PRODUCTS $PROD
"
                                       endif
                                else
                                       setenv PRODUCTS $PROD
                                endif
                       endif
               end
       endif #end if $PRODUCTS already set

       if ($?PRODUCTS) then #set if $PRODUCTS set
               set setups=no
               if ( $?UPS_DIR == 0 ) then
                       set setups=yes
               else
                       set fchar=`echo $UPS_DIR | cut -c1`
                       if ( "x/" != x"$fchar" ) then
                               set setups=yes
                       endif
               endif
               if ( $setups == "yes" ) then #start if $UPS_DIR already set
                       set setups=no
                       set full=${MACH_OS}+`funame -r`
                       foreach PROD ( $PRODUCTS )
                       if ( -r $PROD/ups ) then
                               if ( `grep -c "current.$MACH_OS" $PROD/ups` == 0
 ) then
                                       continue
                               endif
                               foreach i ( `grep "current.$MACH_OS" $PROD/ups|a
wk '{print $2}'|sort -r` )
                                       if ( $full =~ $i* ) then
                                               set flavor=$i
                                               break
                                       endif
                               end
                               if ( $?flavor == 0 ) then
                                       continue
                               endif
#characater after ${flavor} and before " is tab (next 2 lines)
                               set CURRENT=`grep "current.${flavor}    " $PROD/
ups|awk '{print $3}'`
                               setenv UPS_DIR `grep "^Instance:.*${flavor}
" ${PROD}/ups|grep \"${CURRENT}\"|awk '{print $4}'`
                               set fchar=`echo $UPS_DIR | cut -c1`
                               if ( "x/" == x"$fchar" ) then
                                       set setups=yes
                                       break
                               endif
                       endif
```

```
                        end
                        if ($?UPS_DIR) then
                                if ( -r $UPS_DIR/ups/ups_init ) then
                                        source $UPS_DIR/ups/ups_init
                                        setup ups
                                endif
                        endif
                endif #end if $UPS_DIR already set
#Even though UPS_DIR may have already been setup, aliases aren't always
# passed along, thus we need to always do the source of ups_init and
# the setup.csh file. We can't simply do a setup of ups, because then you
# get the current version of ups instead of the version of UPS when you
# entered this process. The setup of ups relies on UPS_PROD_VERSION being
# set (although not in a serious way). This process assumes the version is
# the last component of the $UPS_DIR directory. This isn't guaranteed to be
# correct, but will work ok for this situation.
                if ($?UPS_DIR) then #start if $UPS_DIR is set
                        if ( -r $UPS_DIR/ups/ups_init ) then
                                setenv UPS_PROD_VERSION `basename $UPS_DIR`
                                source $UPS_DIR/ups/ups_init
                                if ( -r $UPS_DIR/ups/setup.csh ) then
                                        source $UPS_DIR/ups/setup.csh
                                endif
                                unsetenv UPS_PROD_VERSION
                        else
                                echo '$UPS_DIR set, but $UPS_DIR/ups/ups_init do
esnt exist, ups environment not setup'
                        endif
                else
                        echo 'Unable to set $UPS_DIR, ups environment not setup'
                endif #end if $UPS_DIR set
        else
                echo 'Unable to set $PRODUCTS, ups environment not setup'
        endif #end if $PRODUCTS set

endif #end if .noupsproducts
```

C.1.5 .login

The default FUE `.login` file is found in `/usr/local/etc/stdlogin`.

```
# .login default settings for all users

#@(#) stdlogin 1.6 Delta: 92/11/05 14:17:45 Extraction 94/07/06 15:16:16 @(#)

#
# .login default settings for all users
#
# execute fermi.login first
#
if ( -r /usr/local/etc/fermi.login ) then
        source /usr/local/etc/fermi.login
endif
#
# defines alias for cedit, comment out if not wanted
#
if ( ! -e ~/.noupsproducts ) then
        setup cedit
endif
#
# Next line sets prompt to <machine_name>
#
set prompt="<$MACH_ID> "
```

```
#
# The standard fermi.profile does not attempt to include /usr/5bin
# (in SunOS) in your PATH.  /usr/5bin contains the UNIX System V version
# of several commands.  By including /usr/5bin first, you will get the
# System V version before the BSD version.  Uncomment one of the following
# if you so desire:
#
#set path=(/usr/5bin $path)
#set path=($path /usr/5bin)
#
# The savehist & history settings maybe expanded to 100
#
set savehist=20      #save last 20 commands for next session
set history=20       #retain the last 20 commands
#
# Uncomment the following to change...
#
#stty kill '^u'         # sets line kill to <ctrl-u>
#stty erase '^H'  # sets erase to backspace
#stty intr '^?'   # sets interrupt to delete
```

☞ The `.login` file executes the file `/usr/local/etc/fermi.login`.

C.1.6 fermi.login

The default FUE `fermi.login` file is found in `/usr/local/etc/fermi.login`.

```
# @(#) fermi.login 4.31 Delta: 94/09/29 22:31:20 Extraction 94/09/29 22:34:39 @(
#)

#
# fermi.login settings for all users; called by the default .login
# Set the umask so that newly created files and directories will be readable
# by others, but writable only by the user.
#
umask 022
#
# Following put in to handle NQS
#
if ( $?ENVIRONMENT ) then
        if ( $ENVIRONMENT == "BATCH" ) exit
endif
#
# Determine terminal type
#
set ttype=`echo $term |cut -c1`
switch (x$term)
case x:
case xunknown:
case xarpanet:
case xnetwork:
case xnet:
case xdialup:
case xdumb:
        set term=vt100
        stty erase '^?'
breaksw
default
        if ( -r /usr/lib/terminfo/$ttype/$term || -r /usr/share/lib/terminfo/$tt
ype/$term ) then
                switch ($term)
                case iris-ansi:
                case iris-ansi-net:
```

```
		case hp:
			stty erase '^h'
		breaksw
		default
			stty erase '^?'
		breaksw
		endsw
	else
		set bterm=`echo $term | cut -c1-3`
		if ( -r /usr/lib/terminfo/$ttype/$bterm || -r /usr/share/lib/ter
minfo/$ttype/$bterm ) then
			set term=$bterm
			stty erase '^?'
		else
			switch ($MACH_OS)
			case IRIX:
			case SunOS:
			case AIX:
				switch ($bterm)
				case vt2:
				case vt3:
					set term=vt220
					stty erase '^?'
				breaksw
				default
					set term=vt100
					stty erase '^?'
				breaksw
				endsw
			breaksw
			case ULTRIX:
				switch ($bterm)
				case vt3:
					set term=vt300
					stty erase '^?'
				breaksw
				case vt2:
					set term=vt200
					stty erase '^?'
				breaksw
				default
					set term=vt100
					stty erase '^?'
				breaksw
				endsw
			breaksw
			case OSF1:
				switch ($bterm)
				case vt2:
				case vt3:
					stty erase '^?'
				breaksw
				default
					set term=vt100
					stty erase '^?'
				breaksw
				endsw
			breaksw
			default
				set term=vt100
				stty erase '^?'
			breaksw
			endsw
		endif
	endif
```

```
breaksw
endsw
#
# Set DISPLAY
#
if ( ! $?DISPLAY ) then
        set TTYPORT=`tty`
        if ( $TTYPORT == /dev/console ) then
                setenv DISPLAY "localhost:0"
        else if ( $?REMOTEHOST ) then
                setenv DISPLAY "${REMOTEHOST}:0"
        else
                set TTYNAME=`echo $TTYPORT |cut -c6-`
                set REMOTEHOST=`who|grep "$TTYNAME"|awk '{print $6}'|sed 's/(//'
|sed 's/)//'`
                setenv DISPLAY "${REMOTEHOST}:0"
        endif
endif
#
# Who are you?
#
if ( $?LOGNAME ) then
        setenv LOGNAME "$LOGNAME"
else if ( $?USER ) then
        setenv LOGNAME "$USER"
else if ( $?REMOTEUSER ) then
        setenv LOGNAME "$REMOTEUSER"
endif
#
# The switch statement is to set the mail variable
# based on machine type.  Also motd and news are
# dependent on system type. Some systems automatically
# display motd (and thus shouldn't be repeated here.
# Others don't have news.
#
switch ($MACH_OS)
case IRIX:
        setenv TZ CST6CDT
        if ( `funame -r|cut -c1` == 3 ) then
                set mail=/usr/mail/$LOGNAME
                if ( { /bin/mail -e } ) then
                        echo 'You have mail.'
                endif
        endif
        if ( { test -d /usr/news } ) then
                /usr/bin/news
        endif
breaksw
case SunOS:
        setenv TZ CST6CDT
        set mail=/usr/spool/mail/$LOGNAME
breaksw
case AIX:
        setenv TZ CST6CDT
        /bin/news
        setenv MAIL "/usr/spool/mail/$LOGNAME"
        setenv MAILMSG "[YOU HAVE NEW MAIL]"
        if ( { test -s $MAIL } ) then
                echo 'You have mail.'
        endif
        setenv EMULATE none
breaksw
case ULTRIX:
        set mail=/usr/spool/mail/$USER
breaksw
```

```
case OSF1:
        /usr/bin/news
        set mail=/usr/spool/mail/$USER
breaksw
case HP-UX:
        setenv TZ CST6CDT
        cat -s /etc/motd
        /usr/bin/news
        set mail=/usr/mail/$LOGNAME
        if ( { /bin/mail -e } ) then
                echo 'You have mail.'
        endif
breaksw
case "":
        #If for some reason funame -s returns NULL, as opposed to returning
        #something that isn't recognized, do nothing
breaksw
default
        cat -s /etc/motd
        if ( -x /usr/bin/news) then
                /usr/bin/news
        endif
        if ( -x /bin/news) then
                /bin/news
        endif
        if ( -d /usr/spool/mail ) then
                set mail=/usr/spool/mail
        endif
        if (    -d /usr/mail ) then
                set mail=/usr/mail/$LOGNAME
        endif
        if ( -x /bin/mail ) then
                if ( { /bin/mail -e } ) then
                        echo 'You have mail.'
                endif
        endif
breaksw
endsw
#
# Establish PAGER
#
if ( -r /usr/local/bin/less ) then
        setenv PAGER /usr/local/bin/less
else
        setenv PAGER more
endif
#
# Common terminal characteristics
#
stty intr '^c'  # set interrupt key to <ctrl-c>
stty kill '^x'  # set kill key to <ctrl-x>
stty echoe                      # erase ERASEd characters
echo "Terminal Type is $TERM"
/usr/local/bin/Info -new
#
# Some systems the user doesn't own his tty device ( Sun OpenWindows) so
# redirect stderr
#
/bin/chmod 622 `tty` >& /dev/null
#
# Check for existance of local.cshsrc file. This is the file admins should
# put in any changes, additions, etc. so that they don't need to re-edit this
# file every release.
#
if ( -r /usr/local/etc/local.login ) then
```

```
        source /usr/local/etc/local.login
endif
```

C.2 Bourne Shell Family

C.2.1 .profile

The default FUE `.profile` file is found in `/usr/local/etc/stdprofile`.

```
# @(#) stdprofile 1.6 Delta: 92/11/05 14:19:26 Extraction 94/07/06 15:16:17 @(#)

#
# .profile default settings for all users
#
# Execute fermi.profile first!
#
if [ -r /usr/local/etc/fermi.profile ]
then
  . /usr/local/etc/fermi.profile
fi
#
# PS1 sets prompt to <machine_name>
# PS2 sets secondary prompt to <more>
#
export PS1;             PS1="<${MACH_ID}> "
export PS2;             PS2="<more> "
#
# HISTSIZE is the number of commands retained in history
#
export HISTSIZE;        HISTSIZE=20       #retain the last 20 commands
#
# EDITOR is used as your default editor, emacs or vi typically
# VISUAL is used as your command line editor, emacs or vi typically
# To use the emacs version, comment out or remove the vi references
# and uncomment the emacs verisons...
#
#export EDITOR;         EDITOR=emacs
#export VISUAL;         VISUAL=emacs
#
export EDITOR;          EDITOR=vi
export VISUAL;          VISUAL=vi
#
# The standard fermi.profile does not attempt to include /usr/5bin
# (in SunOS) in your PATH.  /usr/5bin contains the UNIX System V version
# of several commands.  By including /usr/5bin first, you will get the
# System V version before the BSD version.  Uncomment one of the following
# if you so desire:
#
# PATH=/usr/5bin:$PATH
# PATH=$PATH:/usr/5bin
#
# ENV is the location of your shell alias file
#
export ENV;             ENV=~/.shrc
#
# Uncomment the following to change...
#
#stty kill ^U   # sets line kill to <ctrl>U
#stty erase  # sets erase to backspace
#stty intr ^?   # sets interrupt to delete
#
```

```
if [ "`basename $SHELL|sed 's/[0-9]//'`" = "sh" ] && [ -r $HOME/.shrc ]
then
  . $HOME/.shrc
fi
```

☞ This file calls `/usr/local/bin/fermi.profile`.

C.2.2 fermi.profile

The default FUE `fermi.profile` file is found in `/usr/local/etc/fermi.profile`.

```
# @(#) fermi.profile 1.33 Delta: 94/10/31 22:14:25 Extraction 94/11/02 16:56:08
@(#)

# fermi.profile settings for all users; called by the user bourne/korn .profile
# Set the umask so that newly created files and directories will be readable
# by others, but writable only by the user.
umask 022
MACH_ID="`/usr/local/bin/funame -n`"
MACH_TYPE="`/usr/local/bin/funame -m`"
MACH_OS="`/usr/local/bin/funame -s`"
#
# Set the PATH appropriately
#
if [ -r /usr/local/etc/setpath.sh ]
then
  . /usr/local/etc/setpath.sh
fi

#
# Set a respectable MANPATH
#
MANPATH=/usr/products/catman:/usr/products/man:/afs/fnal/products/catman:/afs/fn
al/products/man:/usr/catman:/usr/man:/usr/share/catman:/usr/share/man:/usr/local
/catman:/usr/local/man
#
# Following put in to handle NQS
# Only do rest if non-interactive session
#
if [ "$ENVIRONMENT" != "BATCH" ] && [ "x`logname`" != "x" ]
then
#
# Determine the terminal type
#
TTYPE=`echo $TERM | cut -c1`
case $TERM in
""|arpanet|net|network|dialup|unknown|dumb)
        TERM=vt100
        stty erase '^?'
;;
*)
        if [ -r /usr/lib/terminfo/$TTYPE/$TERM -o -r /usr/share/lib/terminfo/$TT
YPE/$TERM ]
        then
                case $TERM in
                iris-ansi*|hp)
                        stty erase '^h'
                ;;
                *)
                        stty erase '^?'
                ;;
                esac
        else
```

```
                BTERM=`echo $TERM | cut -c1-3`
                if [ -r /usr/lib/terminfo/$TTYPE/$BTERM -o -r /usr/share/lib/ter
minfo/$TTYPE/$BTERM ]
                then
                        TERM=$BTERM
                        stty erase '^?'
                else
                        case $MACH_OS in
                        IRIX|SunOS|AIX)
                                case $BTERM in
                                vt[2|3])
                                        TERM=vt220
                                        stty erase '^?'
                                ;;
                                *)
                                        TERM=vt100
                                        stty erase '^?'
                                ;;
                                esac
                        ;;
                        ULTRIX)
                                case $BTERM in
                                vt3)
                                        TERM=vt300
                                        stty erase '^?'
                                ;;
                                vt2)
                                        TERM=vt200
                                        stty erase '^?'
                                ;;
                                *)
                                        TERM=vt100
                                        stty erase '^?'
                                ;;
                                esac
                        ;;
                        OSF1)
                                case $BTERM in
                                vt[2|3])
                                        stty erase '^?'
                                ;;
                                *)
                                        TERM=vt100
                                        stty erase '^?'
                                ;;
                                esac
                        ;;
                        HP-UX)
                                case $BTERM in
                                vt3)
                                        TERM=vt320
                                        stty erase '^?'
                                ;;
                                vt2)
                                        TERM=vt200
                                        stty erase '^?'
                                ;;
                                *)
                                        TERM=vt100
                                        stty erase '^?'
                                ;;
                                esac

                        ;;
                        *)
```

```
                                TERM=vt100
                                stty erase '^?'
                        ;;
                        esac
                fi
        fi
;;
esac
#
# Set the DISPLAY
#
if [ x"$DISPLAY" = x ]
then
        TTYPORT=`tty`
        if [ "$TTYPORT" = "/dev/console" ]
        then
                DISPLAY="localhost:0"
        elif [ "$REMOTEHOST" = "" ]
        then
                TTYNAME=`echo $TTYPORT | cut -c6-`
                REMOTEHOST=`who|grep "$TTYNAME"|awk '{print $6}'|sed 's/(//'|sed
 's/)//'`
                DISPLAY="${REMOTEHOST}:0"
        else
                DISPLAY="${REMOTEHOST}:0"
        fi
fi
#
# Who are you?
#
if [ "x$LOGNAME" = "x" ]
then
        if [ "x$USER" = "x" ]
        then
        if [ "x$REMOTEUSER" != "x" ]
        then
                LOGNAME="$REMOTEUSER"
        fi
        else
                LOGNAME="$USER"
        fi
fi
#
# The case statement is to set the mail variable
# based on machine type.  Also motd and news are
# dependent on system type. Some systems automatically
# display motd (and thus shouldn't be repeated here.
# Others don't have news.
#
case $MACH_OS in
IRIX)
        TZ=CST6CDT
        if [ "`funame -r | cut -c1`" = "3" ]
        then
                MAIL=/usr/mail/$LOGNAME
                /bin/mail -e
                if [ $? = 0 ]
                then
                        echo 'You have mail.'
                fi
        fi
        if [ -d /usr/news ]
        then
                /usr/bin/news
        fi
```

```
;;
SunOS)
        TZ=CST6CDT
        MAIL=/usr/spool/mail/$LOGNAME
;;
AIX)
        TZ=CST6CDT
        /bin/news
        MAIL="/usr/spool/mail/$LOGNAME"
        export MAILMESG;MAILMESG="[YOU HAVE NEW MAIL]"
        if [ -s $MAIL ]
        then
                echo 'You have mail.'
        fi
        EMULATE=none
        export EMULATE
;;
ULTRIX)
        MAIL=/usr/spool/mail/$USER
;;
OSF1)
        /usr/bin/news
        MAIL=/usr/spool/mail/$USER
;;
HP_UX)
        TZ=CST6CDT
        cat -s /etc/motd
        MAIL=/usr/mail/$LOGNAME
        /bin/mail -e
        if [ $? = 0 ]
        then
                echo 'You have mail.'
        fi
        /usr/bin/news
;;
"")
        # Catch for the "funame returns null" bug
        :
;;
*)
        cat -s /etc/motd
        if [ -x /usr/bin/news ]
        then
                /usr/bin/news
        fi
        if [ -x /bin/news ]
        then
                /bin/news
        fi
        if [ -d /usr/spool/mail ]
        then
                MAIL=/usr/spool/mail
        fi
        if [    -d /usr/mail    ]
        then
                MAIL=/usr/mail/$LOGNAME
        fi
        if [ -x /bin/mail ]
        then
                /bin/mail -e
                if [ $? = 0 ]
                then
                        echo 'You have mail.'
                fi
        fi
```

```
;;
esac
#
#  Establish PAGER
#
if [ -x /usr/local/bin/less ]
then
        PAGER=/usr/local/bin/less
else
        PAGER=more
fi
#
# Common terminal characteristics
#
stty intr '^c'  # set interrupt key to <ctrl-c>
stty kill '^x'  # set kill key to <ctrl-x>
stty echoe                      # erase ERASEd characters
echo "Terminal Type is $TERM"
if [ -x /usr/local/bin/Info ]
then
        /usr/local/bin/Info -new
fi
# Some systems the user doesn't own his tty device ( Sun OpenWindows) so
# redirect stderr
/bin/chmod 622 `tty` >/dev/null 2>&1
export DISPLAY MAIL TZ TERM PATH PAGER MANPATH

fi      # This is end of if statement for interactive use.
#
# Check for existance of local.profile file. This is the file admins should
# put in any changes, additions, etc. so that they don't need to re-edit this
# file every release.
#
if [ -r /usr/local/etc/local.profile ]
then
  . /usr/local/etc/local.profile
fi
```

☞ `fermi.profile` executes `/usr/local/etc/setpath.sh` to set the *path* variable.

C.2.3 setpath.sh

The default FUE `setpath.sh` file is found in `/usr/local/etc/setpath.sh`.

```
# @(#) setpath.sh 1.9 Delta: 95/01/18 15:29:38 Extraction 95/03/22 22:13:53 @(#)

if [ "x"$HOME = "x" ]
then
        HOME=/
fi

PATH=""
for DIR in /usr/sbin \
                                         /opt/SUNWspro/bin \
                                         /usr/ccs/bin \
                                         /usr/lang \
                                         /usr/bsd \
                                         /bin \
                                         /usr/bin \
                                         /usr/lbin \
                                         /usr/ucb \
                                         /etc \
                                         /usr/etc \
                                         /usr/afsws/bin \
```

```
                                        /usr/openwin/bin \
                                        /usr/bin/X11 \
                                        /usr/kinet/bin \
                                        /usr/local/bin \
                                        /usr/bin/mh \
                                        /usr/sccs
do
        if [ -d ${DIR} ]
        then
                PATH=${PATH}${DIR}:
        fi
done
#
# zsh bug fix
#
PATH=`echo $PATH|/bin/sed 's/\.//g'`
if [ "x"$MACH_OS = "x" ]
then
        MACH_OS=`funame -s`
fi
# do not do this if you are root, because //bin is your $HOME/bin, and
# do not put current directory in root's path
if [ "$HOME" != "/" ]
then
        #
        # Add your HOME/bin and bin."flavor"
        # Put your Sun executables in a bin named bin.SunOS,
        #     your IRIX executables in bin.IRIX,
        #     your AIX     "         " bin.AIX,
        #  ...etc...
        #
        for DIR in ${HOME}/bin.${MACH_OS} $HOME/bin
        do
                if [ -d ${DIR} ]
                then
                        PATH=${DIR}:${PATH}
                fi
        done
        PATH=${PATH}.:
fi

export PATH
```

C.2.4 .shrc

The default FUE `.shrc` file is found in `/usr/local/etc/stdshrc`.

```
# .shrc default settings for all users

#execute fermi.shrc first
if [ -r /usr/local/etc/fermi.shrc ]
then
        . /usr/local/etc/fermi.shrc
fi

#Place items that you want run even when not interactive shell here

#skip if not interactive shell
if [ "x$USER" = "x" ] || [ "x$PS1" = "x" ]
then
        :
else
```

```
        if [ "`basename $SHELL|sed 's/[0-9]//'`" != "sh" ]
        then
          set -o noclobber            #prevent overwrite when redirecting output
          set -o ignoreeof            #prevent accidental logouts

  # Have mail point to fermimail which is a Berkely mail version. Some packages
  # expect mail to be a System V version of mail. If you are having problems
  # of this type, you may need to remove the alias.
        alias mail='fermimail'

  #Define various aliases; user selects desired alias by removng the # sign
  # ONLY IF YOUR SHELL SUPPORTS ALIASING!
          #alias        a='alias'
          #alias        logout='exit'
          #alias        killit='kill -9'        # guarantees that a process is k
illed
          #alias        ll='ls -l'              # generate a long listing
          #alias        la='ls -a'              # see hidden files
          #alias        lf='ls -CF'             # check file TYPE (exe, dir ..)
          #alias        rmi='rm -i'             # confirm before deletion
          #alias        cpi='cp -i'             # no overwrite of output file
          #alias        mvi='mv -i'             # confirm before moving
          #alias        more='less'             # when you explicitly "more" a f
ile

  #Next alias replaces standard info command on SGI platforms
          #alias        info='Info'            #get list of info articles

        #VMS type commands
          #alias        dir='ls -l'
          #alias        copy='cp'
          #alias        rename='mv'

        else
  # Have mail point to fermimail which is a Berkely mail version. Some packages
  # expect mail to be a System V version of mail. If you are having problems
  # of this type, you may need to remove the alias.
          mail ()
          {
            fermimail "$@"
          }
  #Define various functions; user selects desired functions by removng the # sig
n
  #Provided for shells that do not support aliasing.
          #killit ()
          #{
            #  kill -9  # guarantees that a process is killed
          #}
          #ll ()
          #{
            #  ls -l    "$@"     # generate a long listing
          #}
          #la ()
          #{
            #  ls -a    "$@"     # see hidden files
          #}
          #lf ()
          #{
            #  ls -CF "$@"      # display file type, columnwise
          #}
          #rmi ()
          #{
            #  rm -i "$@"       # confirm before delete
          #}
```

```
          #cpi ()
          #{
            #  cp -i “$@”       # no overwrite of output file (see also set nocl
obber)
          #}
          #mvi ()
          #{
            #  mv -i “$@”       # confirm before moving
          #}
          #more ()
          #{
            #  less “$@”    # when you explicitly “more” a file
          #}

  #Next function replaces standard info command on SGI platforms
          #info ()
          #{
            #  Info “$@”        # get list of info articles
          #}

  #VMS type commands
          #dir ()
          #{
            #  ls -l “$@”
          #}
          #copy ()
          #{
            #  cp “$@”
          #}
          #rename ()
          #{
            #  mv “$@”
          #}
        fi
fi
```

The `.shrc` file calls `/usr/local/etc/fermi.shrc`.

C.2.5 fermi.shrc

The default FUE `fermi.shrc` file is found in `/usr/local/etc/fermi.shrc`.

```
# @(#) fermi.shrc 1.10 Delta: 94/09/29 22:22:52 Extraction 94/09/29 22:34:43 @(#
)

# fermi.shrc settings for all users; will be called by the default .profile

#
# Setup UPS
#
if [ -r /usr/local/etc/setups.sh ]
then
        . /usr/local/etc/setups.sh
fi

#Do machine dependent functions:
case $MACH_OS in
IRIX)
  fermimail ()
  {
    /usr/sbin/Mail “$@”
  }
;;
SunOS)
```

```
  fermimail ()
  {
    /usr/ucb/Mail "$@"
  }
  man ()
  {
        /usr/bin/man -F "$@"
  }
;;
AIX)
  fermimail ()
  {
    /usr/ucb/Mail "$@"
  }
;;
ULTRIX)
  fermimail ()
  {
    /usr/ucb/Mail "$@"
  }
;;
OSF1)
  fermimail ()
  {
    /usr/bin/Mail "$@"
  }
;;
HP-UX)
  fermimail ()
  {
    /usr/bin/mailx "$@"
  }
;;
*)
  if [ -x /usr/ucb/Mail ]
  then
    fermimail ()
    {
      /usr/ucb/Mail "$@"
    }
  elif [ -x /usr/sbin/Mail ]
  then
    fermimail ()
    {
      /usr/sbin/Mail "$@"
    }
  elif [ -x /usr/bsd/Mail ]
  then
    fermimail ()
    {
      /usr/bsd/Mail "$@"
    }
  elif [ -x /usr/bin/mailx ]
  then
    fermimail ()
    {
      /usr/bin/mailx "$@"
    }
  elif [ -x /usr/bin/Mail ]
  then
    fermimail ()
    {
      /usr/bin/Mail "$@"
    }
  fi
```

```
;;
esac
#
# Check for existance of local.shsrc file. This is the file admins should
# put in any changes, additions, etc. so that they don't need to re-edit this
# file every release.
#
if [ -r /usr/local/etc/local.shrc ]
then
  . /usr/local/etc/local.shrc
fi
```

☞ `fermi.shrc` executes `/usr/local/etc/setups.sh`.

C.2.6 setups.sh

The default FUE `setups.sh` file is found in `/usr/local/etc/setups.sh`.

```
# @(#) setups.sh 2.2 Delta: 95/01/09 15:48:34 Extraction 95/03/22 22:13:54 @(#)

#Set environment for UPS
if [ ! -r $HOME/.noupsproducts ] #start if .noupsproducts
then
        MACH_OS=`funame -s`
        fchar=`echo ${PRODUCTS:-}|cut -c1`
        if [ "x$fchar" != "x/" ] #start check if $PRODUCTS already set
        then
                prodset=no
                unset PRODUCTS
                # Add any other directories that are UPS databases in the
                # for list
                for PROD in $UPS_EXTRA_DIR \
                    /`funame -n`/products/ups_database/declared \
                         `logdir products`/ups_database/declared \
                         /usr/products/ups_database/declared \
                         /afs/fnal/products/ups_database/declared
                do
                        if [ -d $PROD ]
                        then
                                if [ "$prodset" = "no" ]
                                then
                                        PRODUCTS=$PROD
                                else
                                        exists=`echo $PRODUCTS | grep -c $PROD`
                                        if [ $exists = "0" ]
                                        then
                                                PRODUCTS="$PRODUCTS $PROD"
                                        fi
                                fi
                                prodset=yes
                        fi
                done
                export PRODUCTS
        else
                prodset=yes
        fi #end check if $PRODUCTS already set
        if [ "$prodset" = "yes" ] #start if $PRODUCTS is set
        then
                fchar=`echo ${UPS_DIR:-}|cut -c1`
                if [ "x$fchar" != "x/" ] #start if $UPS_DIR already set
                then
                        setups=no
                        full=${MACH_OS}+`funame -r`
```

```
			for PROD in $PRODUCTS
			do
			if [ -f $PROD/ups ]
			then
				if [ `grep -c "current.$MACH_OS" $PROD/ups` -eq
0 ]
				then
					continue
				fi
				flavor=x
				for i in `grep "current.$MACH_OS" $PROD/ups|awk
'{print $2}'|sort -r`
				do
					case "$full" in
					$i* )	flavor=$i
						break;;
					*) ;;
					esac
				done
				if [ $flavor = x ]
				then
					continue
				fi
#character after ${flavor} and before quote is tab (next two lines)
				CURRENT=`grep "current.${flavor}	" $PROD/
ups|awk '{print $3}'`
				UPS_DIR=`grep "^Instance:.*${flavor}	" ${PROD
}/ups |grep \"${CURRENT}\"|awk '{print $4}'`
			fi
			fchar=`echo ${UPS_DIR:-}|cut -c1`
			if [ "x$fchar" = "x/" ]
			then
				setups=yes
				break
			fi
			done
			if [ "$setups" = "yes" ]
			then
				export UPS_DIR
				if [ -f $UPS_DIR/ups/ups_init.sh ]
				then
					. $UPS_DIR/ups/ups_init.sh
					setup ups
				fi
			fi
		else
			setups=yes
		fi #end if $UPS_DIR already set
#Even though UPS_DIR may have already been setup, aliases aren't always
# passed along, thus we need to always do the source of ups_init and
# the setup.sh file. We can't simply do a setup of ups, because then you
# get the current version of ups instead of the version of UPS when you
# entered this process. The setup of ups relies on UPS_PROD_VERSION being
# set (although not in a serious way). This process assumes the version is
# the last component of the $UPS_DIR directory. This isn't guaranteed to be
# correct, but will work ok for this situation.
		if [ "$setups" = "yes" ]
		then
			if [ -f $UPS_DIR/ups/ups_init.sh ]
			then
				. $UPS_DIR/ups/ups_init.sh
				UPS_PROD_VERSION=`basename $UPS_DIR`
				if [ -f $UPS_DIR/ups/setup.sh ]
				then
					. $UPS_DIR/ups/setup.sh
```

```
                                fi
                                unset UPS_PROD_VERSION
                        else
                                echo "\$UPS_DIR set, but \$UPS_DIR/ups/ups_init.
sh doesn't exist. No ups environment setup."
                        fi
                else
                        echo "Unable to set \$UPS_DIR, ups environment not setup
"
                fi
        else
                        echo "Unable to set \$PRODUCTS, ups environment not setup"
        fi #end if $PRODUCTS is set
fi #end if .noupsproducts
```

Appendix D. awk's Programming Model

This appendix is adapted from a section of the same name in the book *sed & awk*, published by O'Reilly & Associates. It describes the generic structure and organization of an **awk** program.

An **awk** program consists of what is called a *main input loop*. A loop is a routine that is executed over and over again until some condition exists that terminates it. You don't write this loop, it is given; it exists as the framework within which the code that you do write will be executed. The main input loop in **awk** is a routine that reads one line of input from a file and makes it available for processing. The actions you write to do the processing assume that there is a line of input available. In another programming language, you would have to create the main input loop as part of your program.

The main input loop is executed as many times as there are lines of input. This loop does not execute until there is a line of input. It terminates when there is no more input to be read.

awk allows you to write two special routines that can be executed before any input is read and after all input is read. These are the procedures associated with the BEGIN and END rules, respectively. In other words, you can do some pre-processing before the main input loop is ever executed using the BEGIN procedure, and you can do some post-processing with the END procedure after the main input loop has been terminated. The BEGIN and END procedures are optional and they do not need to be defined.

You can think of an **awk** script as having potentially three major parts: what happens before, during, and after processing the input. The "what happens during processing" part is where most of the work gets done. Inside the main loop, your instructions are written as a series of pattern/action procedures. A pattern is a rule for testing the input line to determine whether or not the action should be applied to it. The actions can be quite complex, consisting of statements, functions, and expressions.

Appendix E. VMS to UNIX Command Reference

This appendix is intended as a convenient UNIX command reference for the migrating VMS user. The UNIX equivalents of commonly used VMS commands are listed in tabular format. No information on syntax, options, or arguments is presented here, however some of the listed commands are described elsewhere in *UNIX at Fermilab.*

Commands which are Fermilab-specific or AFS-specific are indicated as such.

E.1 UNIX Equivalents for Many VMS Commands

Symbols used in table:

These symbols are not part of any UNIX command with which they appear.

*	for systems running AFS, the command may not work as documented
**	your system may be configured to use an alternate Fermilab command to perform this action
+	AFS-specific commands
#	local Fermilab commands

Actions	VMS	UNIX
Actions on files and directories		
create directory	CREATE/DIR	**mkdir**
change working directory	SET DEFAULT	**cd**
display working directory	SHOW DEFAULT	**pwd**
display list of files	DIRECTORY DIRECTORY/FULL	**ls** **ls -l**
display list of subdirectories only		**ls -l|grep '^d'** or **find * -type d \\ -prune -print**
display contents of file	TYPE	**cat**
display file with pauses	TYPE/PAGE	**more, less**

Actions	VMS	UNIX
display first few lines of file	TYPE/PAGE	**head**
display last few lines of file	TYPE/TAIL	**tail**
copy a single file	COPY	**cp**
copy all files oldfile.* to newfile.*	COPY OLDFILE.* NEWFILE.*	Create shell script; see section E.2.
find file	DIRECTORY	**find**
find file containing a specified expression or string	SEARCH	**grep** **egrep** **fgrep**
compare files	DIFF	**diff** **cmp**
rename file	RENAME	**mv**
rename all files oldfile.* to newfile.*	RENAME OLDFILE.* NEWFILE.*	Create shell script; see section E.2.
delete file or directory	DELETE	**rm** **rmdir**
merge files	COPY or APPEND	**cat**
three-way file merge	MERGE	**merge**
sort or merge files	SORT	**sort**
assign (link) files	ASSIGN DEFINE SET FILE/ENTER	**ln** **setenv**
deassign files (remove link)	DEASSIGN	**rm**
convert and copy file	EXCHANGE	**dd**
update last modified date of file	OPEN/APPEND	**touch**
print file	PRINT	**lpr** **flpr** #
format text file	RUNOFF	**nroff** **troff** **groff** #
octal, decimal, hex, or ascii dump of file	DUMP	**od**
change file protection	SET FILE/PROT	**chmod** *

Actions	VMS	UNIX
change file ownership	SET FILE/OWNER	**chown** * **chgrp** *
list ACL settings on an AFS directory		**fs listacl** +
set ACL settings on an AFS directory		**fs setacl** +
General Commands		
get help	HELP	**man** **apropos**
logout	LOGOUT	**logout** **exit**
change password	SET PASSWORD	**passwd** * **kpasswd** + **yppasswd**[a]
obtain AFS (Kerberos) authentication		**klog** +
discard AFS tokens		**unlog** +
display logged-in users	SHOW USERS	**who** **finger**
display environment settings	SHOW LOGICAL	**alias** **setenv** **printenv** **set**
display date and time	SHOW TIME	**date**
display free disk space	SHOW DEVICE	**df** **du**
display disk quota (On AFS, shows AFS volume quota; see section 7.5.)	SHOW QUOTA	**quota -v** **fs listquota** ***pathname*** **+**[b]
write to screen	WRITE SYS$OUTPUT	**echo**
stop process	STOP	**kill**
display print queue	SHOW QUEUE	**lpq** **flpq** #
display print entries	SHOW ENTRY	**lpq** **flpq** #
delete print entry	DELETE/ENTRY	**flpk** #

Actions	VMS	UNIX
display processes	SHOW SYSTEM	**ps**
tape archiver[c]	BACKUP	**tar**
change terminal settings	SET TERMINAL	**setenv LINES** **setenv COLUMNS** **stty** (for Solaris)
talk to another user	PHONE	**talk**
disable messages	SET NOBROADCAST	**mesg n**
connect to remote node	SET HOST	**telnet** **rlogin**
copy file to a remote node	COPY FTP	**rcp** **ftp**
Commands for Executables, Procedures, and Batch Jobs		
compile FORTRAN program	FORTRAN	**f77**
link FORTRAN program	LINK	**f77**
see the entry point names in a library	LIBRARY/LIST/NAMES	**odump -c**[d]
execute program	RUN	***prog_name***
debug program	RUN/DEBUG	**dbx**
create or manipulate library	LIBRARY	**ar**
execute a command procedure (shell script)	*@prog_name*	***prog_name***
submit a batch job	SUBMIT/QUEUE	**qsub** **fbatch_submit** #[e]
check batch queue	SHOW QUEUE	**qstat** **fbatch_q** #
stop batch job	DELETE/ENTRY	**qdel** **kill** **fbatch_cancel** #

a. for use with NIS (see section 2.7)
b. For normal Berkeley-style systems, the UNIX command **quota -v** indicates how much of your disk quota you have available. On an AFS system, the quota is based not on individual users, but rather on AFS volume. So, if you have a shared volume, it just tracks the quota of the entire volume. For more information, enter **man fs_listquota** (the underscore is required).
c. See section E.3 for information on VMS backup save-sets.
d. under IRIX and OSF1

e. **fbatch** is a FUE interface used with the **LSF** job scheduling product, described in Chapter 14.

E.2 Shell Scripts for Copying/Renaming Multiple Files

In the above table, the UNIX command equivalents for copying/renaming a set of files require short shell scripts. In this section, we give you the script formulas.

The most efficient way to copy all files `oldfile.*` to `newfile.*` in UNIX is to create a short shell script. Its contents are shown for both the Bourne and the C shell families.

sh, **ksh**, or **bash**:

```
for file in oldfile.*
do
   newname=`echo $file | sed -e 's/oldfile/newfile/'`
   mv $file $newname
done
```

csh or **tcsh**:

```
#!/usr/local/bin/tcsh
foreach file (oldfile.*)
   set newname=`echo $file | sed -e 's/oldfile/newfile/'`
   mv $file $newname
end
```

Note that the shell variable *newname* in each case is set to the output of the command that follows the equals sign (**=**). This entire command string must be enclosed in back quotes (**`**...**`**). The argument for the **-e** option in the **sed** command is enclosed in normal single quotes (apostrophes). **sed** is not covered in this manual; most texts discuss it, and the book *sed & awk* published by O'Reilly & Associates covers it thoroughly.

E.3 Unpacking VMS Backup Save-sets

If you made VMS backup save-sets, the UNIX utility **vmsbackup** is a tool that lets you unpack the save-set, converting the files to UNIX format. **vmsbackup** is a separate **UPS** product which must be installed first (see section 10.3.1). The default operation of the program is to go through an entire VMS-generated tape, extract each file, and write it to disk. Several options are provided; we refer you to the man pages for a discussion of them.

Once **vmsbackup** is installed on your machine, you need to run **setup vmsbackup** before using it.

☞ You cannot make save-sets with this utility; you can only unpack *existing* save-sets.

Appendix F. mh and exmh Customization

This appendix contains information for further customizing the **mh** and/or **exmh** mail readers. Given the multitude of customizable features in these mail readers, we cannot provide you with a comprehensive treatment of the subject here. The information presented in Chapter 12 and this appendix should be sufficient for most users.

We refer you to the on-line help and to the O'Reilly book, *MH & xmh - Email for Users & Programmers* by Jerry Peek, for further information.

F.1 Forwarding and Notification

The `.forward` file is a general mail forwarding file, and is used no matter which mail package you choose to use. It tells the system what to do with your mail; whether to leave it in the system spool area, forward it to the spool area on a different node, forward it to a different system entirely, or (in some cases) deliver it to your preferred local mailbox automatically. It may also be used as a means of notifying you that mail has arrived.

Note that comment lines are generally not permitted in the `.forward` file. Some OS versions and versions of **sendmail** (an underlying, transparent internet mail forwarding tool) do allow comment lines in `.forward`. The comment lines must be preceded by a pound sign (`#`).

F.1.1 Forwarding Address

The first line of your `.forward` file tells the system where to put your mail. On your chosen mail node, it should look like:

```
\{username}@{node}.fnal.gov
```

where `{username}` is your username, and `{node}` is the name of the node where you plan on reading mail (e.g., `\joe@fsui02.fnal.gov`). The backslash prevents infinite loops in a "clustered" environment where more than one node sees the same `.forward` file in your login area.[1]

On all other systems, we recommend that you forward your mail to FNAL, the lab's central mail server. For these systems, the first line of each `.forward` file will look like:

```
{username}@fnal.gov
```

e.g., `joe@fnal.gov`. Note that this doesn't have the backslash or the node name. The backslash is not necessary if you are forwarding to a node that does not share the same login area. See sections 2.6 and 2.7 for a discussion of shared login areas, or what we call UNIX *clusters*.

1. If you choose to use *unattended autoincorporation*, which delivers your mail directly into your login area rather than leaving it in the system spool area, the first line of your `.forward` file will look significantly different from this example. Unattended autoincorporation is covered in section F.4.

F.1.2 Mail Notification

Another use of the `.forward` file is to allow you to customize the type of notification you receive when new mail arrives. Usually, the second line of your `.forward` file will control if and how you receive notification.

FUE provides minimal notification of incoming mail by default, however most people like to receive notification of individual messages as they come in. If you want to scan your new mail when you log in, include the commands **`inc`** and **`scan +inbox unseen`** to your `.login` or `.profile` file.

The use of the **MH** program **rcvtty** provides notification of new mail whether or not you have a mail reader active. To set this up, you need to edit your `.forward` file.

Basic Configuration

Even if you don't know what **biff/comsat** is, you will need to determine if it is running on your system. How do you tell? Follow these instructions and you should get one concise notification message per incoming mail message if it is not running. If you get two notifications, one of which is quite verbose, then **biff/comsat** is running, and you'll need to change your configuration as described below.

If you're on a cluster where some nodes run **biff/comsat** and some don't, set up notification according to your chosen mail node. This procedure assumes that **MH** is installed and configured on all nodes within the cluster.

First, you need to make sure you have set up the proper file access permission. Activate tty user execute permission by using the command:

```
% chmod u+x `tty`
```

where the quotes around **`tty`** are both single back quotes.

You can include this in your `.login` or `.profile` file to run it automatically on all sessions. Be sure to include it *after* the call to `fermi.login` or `fermi.profile` since these files set it differently. Or, if you prefer to have notification only in one window, just run this command in that window.

The second line of your `.forward` file should look like:

```
"| /usr/local/products/mh/current/lib/rcvtty -biff"
```

When mail arrives, **rcvtty** should send a notice containing the time, sender, subject, and message sample to all user tty sessions with proper file access permission in the default format:

```
time:  SENDERNAME   subject << first line of text (truncated)
```

If this type of notification is what you get, skip to "Customizing rcvtty Output", below. If instead you get a multiline notice, **biff/comsat** is running, and you'll have to make a couple of changes.

Some systems run **biff** with **comsat** for notification purposes. **biff/comsat** and **rcvtty** (without the `biff` option) can work together, however this yields double notification (both **comsat**'s verbose notice and **rcvtty**'s single line notice) because the `r-w-x` permissions (see section 6.6.1) need to be different for **biff/comsat** and **rcvtty**.

To get **rcvtty** to work properly on systems running **biff/comsat** follow these steps:

1) Enter **`biff n`** to deactivate **comsat** (best to include in `.login` or `.profile` file).

2) In `.forward`, replace:

"| /usr/local/products/mh/current/lib/rcvtty -biff"

with:

"| /usr/local/products/mh/current/lib/rcvtty"

3) Enter **`ls -lL `tty``** to see your default tty setting. If `o+w` (that is, if *write* permission for *other*; see section 6.6.1) is not set, use the command **`chmod o+w `tty``** to set it (again, best to include in `.login` or `.profile` file).

Customizing rcvtty Output

You can reformat the output of **rcvtty** if you like. To do this you need to create a form file describing the syntax of the output you want to see, and you need to modify the **`rcvtty`** command. For example, if you wish to use the same format as `scan-form` (see section F.2.5) the form file used for incorporating or scanning mail, the line in `.forward` would look like this (shown in the format used when **biff/comsat** is *not* running):

"| /usr/local/products/mh/current/lib/rcvtty -biff -form scan-form"

F.2 Files Used to Customize mh and exmh

Your personal configuration for **MH** is primarily defined via the `.mh_profile` file. It defines, among other things, where mail is stored and what editor to use. Some additional files are available if you wish to customize further. These were listed in section 12.4.

Each **MH** mail composition command, (**`comp`**, **`repl`**, **`forw`** for **mh**; Send, Reply, Forward for **exmh**) uses its own template file to define the message header format:

- **`comp`** or Send uses the `components` file
- **`repl`** or Reply uses `replcomps`
- **`forw`** or Forward uses `forwcomps`

If you don't have these template files in your `$HOME/Mail` directory the system default files in the **MH** library are used.

Options for incorporating mail were discussed in Chapter 12. The `scan-form` and `inc-form` files provide the format of the display when you scan and incorporate mail. You can control how local delivery is performed by creating a `.maildelivery` file.

F.2.1 .mh_profile

It also allows you to specify options to **MH** commands. If you have run **`setup mh`** or **`setup exmh`**, you already have a simple one in your home directory. You can tailor your `.mh_profile` file to suit your taste. Following are some of the extra lines you can add:

`Path: Mail`	Directory for your **MH** transactions.
`Unseen-Sequence: unseen`	Name of the unseen sequence of messages.
`Draft-folder: drafts`	Name of the default folder for drafted, unsent mail.
`Editor: emacs`	Sets up **emacs** as editor for **`comp`**, **`repl`**, **`forw`**, **`dist`** (use any editor you want).[1]
`rmmproc: /bin/rm`	Causes mail to be truly deleted when requested (see section 12.3.8). **Recommended!**

1. **exmh** ignores any editor defined here (defining an editor is optional for **MH**).

`Aliasfile: mh_alias`	File containing **MH** aliases (distribution lists).
`Scan: -form scan-form`	Provides format of display when you scan mail.
`inc: -form inc-form`	Provides format of display when you incorporate mail.
`Signature: John Peoples`	Your signature. Appears in `From:` header field.

The `Path` is taken to be relative to your `$HOME` directory. The files/directories used for `Draft-folder`, `Aliasfile`, `Scan`, and `inc` are assumed to be relative to the directory specified by `Path`.

F.2.2 components

The default template file for composing (sending) a message is called `components` and looks like:

```
To:
cc:
Subject:
-------
```

You can add other fields to the header by making your own `components` file. For example, an enhanced `components` file may include the additional fields shown below:

```
Reply-to: {username}@fnal.gov
To:
cc:
Bcc:
Fcc: +yourfolder
Subject:
-------
```

The `Reply-to:` field should give your FNAL address. It is not needed if you want the reply to come to the node from which you are sending the message. The **`repl`** command in **MH** (and some other mail agents as well) will route replies to this address rather than the `From:` address.

The “blind carbon copy” `Bcc:` field can be used to send copies of the message to addresses that are not seen by the people specified in the `To:` and `cc:` fields. This is like forwarding a copy of the message to someone else after sending it, except that here you can do it at the time you send the original message. `Bcc:` can also use distribution lists or aliases.

You can send a copy to yourself by specifying a folder for your copies in the `Fcc:` field. *Fcc* stands for folder copy. This copy goes directly to the specified folder, and only to that folder.

F.2.3 replcomps

The `replcomps` file controls the header in replies. The default file is given below:

```
%(lit)%(formataddr %<{reply-to}%?{from}%?{sender}%?{return-path}%>)\
%<(nonnull)%(void(width))%(putaddr To: )\n%>\
%(lit)%(formataddr{to})%(formataddr{cc})%(formataddr(me))\
%<(nonnull)%(void(width))%(putaddr cc: )\n%>\
%<{fcc}Fcc: %{fcc}\n%>\
%<{subject}Subject: Re: %{subject}\n%>\
%<{date}In-reply-to: Your message of "\
%<(nodate{date})%{date}%|%(pretty{date})%>."%<{message-id}
                 %{message-id}%>\n%>\
--------
```

The first two lines in the file build the `To:` address field for your reply draft. It takes the first available address by searching the `Reply to:`, `From:`, `Sender:` and `Return-path:` fields, in that order. Lines 3 and 4 build a `cc:` address field by taking all the `To:` and `cc:` addresses from the original message and including your address as well. It then builds an `Fcc:` field by testing a **`-fcc`** switch on the command line or in the `.mh_profile`. The `subject:` field is taken from the subject in the incoming mail preceded by `Re:`. The last three lines make up the field that says "*In-reply-to: Your message of ...*" with the date and message number at the end. The default reply header takes the form:

```
To:
cc:
Subject:
In-reply-to: Your message of "Day, date time"
             <message-id>
--------
```

You can customize this formatting by modifying your own `replcomps` file (note that no spaces are allowed after the backslash (****) on each line). You will most likely need to reference the O'Reilly **MH** text for this task.

As you now know, `replcomps` controls the format of the message header. You can also customize the format of the message body for replies via a file called `.mh_filter` in your home directory. The O'Reilly **MH** book describes the different possible ways of setting this up. The contents of the default `.mh_filter` is:

```
body:component="> ",compwidth=0
```

F.2.4 forwcomps

The `forwcomps` file is used as the template header for forwarded messages. This file may look very similar to the `components` file, in fact the default files are identical. If you create your own `components` file and you want to use it for both sending and forwarding, include the following line in your `.mh_profile` file:

```
forw: -form components
```

You would only need a separate `forwcomps` file if you wanted the headers on forwarded and original mail to be different.

F.2.5 scan-form and inc-form

These files provide the format of the display when you scan and incorporate mail. If you aren't satisfied with the default scan format, we recommend that you simply set up a `scan-form` file, and use it for both situations (set both `Scan:` and `inc:` to `scan-form` in `.mh_profile`). The `scan-form` file's format is also used by **rcvtty** to provide messages. An example of a simple `scan-form` file showing incoming mail only and set up in a VMS-style format is shown below:

```
%4(msg)\
%21(addr{from})\
%02(mday{date})-%3(month{date})-%4(year{date})\
%02(hour{date}):%02(min{date})\
%{subject}
```

This example produces output in the format:

`msg# sender_address  date  time  subject`

Some sample output follows:

```
18 helpdesk              13-Nov-1995 16:30 1017699 Problem Report
19 lauri                 14-Nov-1995 10:56 VY0124 damaged
20 STUART@fnald.fnal.gov 14-Nov-1995 11:28 Problems with an fnaldb tape drive
21 wolbers@fsui01.fnal.g 14-Nov-1995 15:31 Department Meeting
```

An slightly more complicated example is shown below. It allows display of both incoming and outgoing messages:

```
%4(msg)\
%<(cur)+%| %>\
%<{forwarded}F%| %>\
%<{replied}R%| %>\
%<(mymbox{from}) <- %19(friendly{to}) %| -> %20(addr{from})%> \
 %02(mday{date})-%03(month{date})-%4(year{date})\
 %02(hour{date}):%02(min{date})\
 %{subject}
```

This example produces output in the format:

`msg# (arrow) (sender or recipient address)  date  time  subject`

The arrow indicates whether the mail is from you (<–) or to you (–>). If it is from you, the address field shows who you sent it to. Sample output looks like:

```
 258    -> bhat                       14-Oct-1995 12:15 batch job 341: <run_small> D
 260    <- joe@fnal                   13-Oct-1995 17:24 Program and Working Groups -
 261    <- lhn@sld.slac.stanford.edu  13-Oct-1995 16:31 e+w- Workshop, November 16-1
 262    <- martha@fnal                13-Oct-1995 17:24 re: revised mail chapter
 263    -> STROVINK@d0sfa             16-Oct-1995 09:21 Re: FYI re directions in NN
```

F.2.6 .maildelivery

The `.maildelivery` file controls how local delivery is performed. This file can be quite simple, or very complex. Within the file you can configure options to:

- restrict notification to mail received only from certain senders
- control notification format and content

• specify further processing such as delivery to one or more specific folders

• start up certain processes depending on information in the mail message header

☞ Note that if you use the **mh** command **`inc`** (at the shell prompt) instead of incorporating within **exmh**, your mail will not be sorted. It will be filed into your `inbox` folder. The `.maildelivery` file is only used when you 1) incorporate within **exmh** (described in section 12.4.3) and 2) check the Presort box in the Incorporate Mail window under Preferences. The `.maildelivery` file is also used with unattended autoincorporation (see section F.4).

Simple Case: Send All Incoming Mail to inbox Folder

If you simply want to send all incoming mail to your `inbox` folder, your `.maildelivery` file needs only to consist of the following one line:

```
* - ^ A "/usr/local/products/mh/current/lib/rcvstore +inbox"
```

If you want to do something more sophisticated, continue reading this section.

File Format

Lines starting with `#` are comments. Fields are separated by one or more spaces. Each record in the file has the following fields in this order:

header field	the message header field that you're sorting on (e.g., `from`, `to`, `subject`)
match pattern	the text string found in the header field that you want to match (enclosed in quotes if it includes spaces). If the header field argument is `default` or `*`, this should be `-` (a dash).
action	what to do if pattern is matched. Set to `file` or `>` to send the matched message to a file. Set to `qpipe` or `^` to directly execute a command without invoking a Bourne shell. Set to `pipe` or `|` to invoke a Bourne shell to execute the command. If your command has special characters in it, you will need to use `pipe` or `|`, otherwise `qpipe` or `^` is more efficient.
result	a code indicating whether to set matched message to "delivered", or not. `A`: if field and pattern matched, action will be performed, and if successful, then set message to "delivered". `R`: same as A, but don't set message to "delivered" so that further entries in `.maildelivery` can act on it. `?`: perform the action if message not yet set to "delivered" by a previous entry. If action succeeds, set message to "delivered".
file or command	depending on action, this is either a file (if action says "send message to a file"), or a command (if action says "execute a command"). Always include full pathname.

☞ Note that some **MH** systems expect this file to have the mode 644 (**`u+rw`**, **`g+r`**, **`o+r`**).

Examples

The following examples should provide you enough information to modify your `.maildelivery` file to suit your tastes. We'll start with an annotated version of the simple `.maildelivery` file discussed above in which all your incoming mail gets placed in `inbox`:

```
# execute rcvstore to put all as-yet-undelivered incoming mail into inbox (sets messages
# to "delivered")
#hdr  pattern  action  result  command
 *    -        ^       A       "/usr/local/products/mh/current/lib/rcvstore +inbox"
```

Here is a slightly more complex example. The comments explain what is being done in each line:

```
# File mail with fsui in the To: line into file $HOME/fsui.log
To fsui file A fsui.log

# Pipe messages from daemon@fnck to the (local) program err-message-archive
From daemon@fnck pipe A err-message-archive

# Put anything with Sender: address "mh-users" into $HOME/mh.log if not yet filed
Sender mh-users file ? mh.log

# If the address is "uas-admin", send an acknowledgment copy back
addr uas-admin ^ A "/bin/resendR-r $(reply-to)"

# Destroy anything from daemon@fnck!
From daemon@fnck destroy A -

# For anything not matched yet, execute rcvstore to put into inbox folder
* - ^ ? "/usr/local/products/mh/current/lib/rcvstore +inbox"
```

If you plan to create a significantly different set of criteria, you may need to see the man pages for **slocal** or the O'Reilly **MH** text referenced at the start of this appendix.

F.3 Automatic Reply to Incoming Mail

There is a handy UNIX program called **vacation** that you can use to send an automatic reply to emails that arrive while you're away. To enable **vacation**, enter simply:

`% vacation`

```
This program can be used to answer your mail automatically when you go away on vacation.
You need to create a message file in
/afs/fnal.gov/files/home/room3/aheavey/.vacation.msg first.
Please use your editor (emacs) to edit this file.
```

The editor invoked is determined by either the *VISUAL* or *EDITOR* environment variable. The default otherwise is **vi**. The default message is displayed in the editor window for you to edit. When you finish, the system returns:

```
You have a message file in /afs/fnal.gov/files/home/room3/aheavey/.vacation.msg.
Would you like to see it?
```

If you answer **`y`**, the file contents gets typed to the screen, and you are given the opportunity to re-edit the file. If you choose not to re-edit it, the system goes directly to:

```
You have a .forward file in your home directory containing:
(file contents displayed here)
Would you like to remove it and disable the vacation feature?
```

Respond **n**. At this point the dialog ends, and you need to go to your `.forward` file and add the line:

```
\{username}, "|/usr/bin/vacation {username}"
```

where `{username}` is your username, for example:

```
\aheavey, "|/usr/bin/vacation aheavey"
```

Now the message in `.vacation.msg` will get sent in reply to all subsequent incoming email messages.

To disable **vacation** when you return, simply remove the above line from your `.forward` file[1].

Run **`man vacation`** to get the full documentation.

F.4 Unattended Autoincorporation

It is possible, though not recommended in a *clustered* environment (see section 2.7 about clusters), to use the `.forward` file to automatically deliver your mail from the system mail spool area directly into your mail area, whether or not you are reading mail, or whether or not you are even logged in! This is called *unattended autoincorporation*. It has a number of attractive features:

- since mail doesn't really stop in the system spool area, but rather is delivered directly to your `inbox` file, you are no longer restricted to reading and receiving mail on one particular node of a cluster
- reduces the startup time (and de-iconizing time) of **exmh** because it doesn't have to "catch up" on lots of spooled messages
- mail can be automatically filed to specified folders as it is being delivered (otherwise only possible for **exmh**, not for the line-mode **mh**)

However, unattended autoincorporation also has a number of drawbacks and caveats which make it a less attractive option:

- in a clustered environment where several separate nodes (hence several separate mail spool areas) see the same mail destination `inbox` file, file locking can become a serious problem. It can happen that a deadlock results in *no* mail being delivered while an overload of processes attempt to deliver mail in vain. This problem is especially serious for people who receive a lot of mail, and is exacerbated by the use of **rcvtty** (mail notification; see section F.1.2).
- **exmh** does not update the visual display as new mail gets autoincorporated. Therefore, while you may have saved the time of incorporating at startup, you will need to manually Rescan all Folders (under the More... button in the middle window) in order to see that you have new mail (and in which folders) when you are running **exmh**.

Because of the detrimental effect that unattended autoincorporation can have on a system, and the fact that when it breaks, mail is not delivered, **we do not recommend its use in most circumstances**. In particular, we believe that it should not be used in clustered environments. It is documented here for completeness only, and should not be taken as an endorsement.

F.4.1 In Standard UNIX Environment

To configure unattended autoincorporation on a node:

1) create a `.maildelivery` file describing your mail sorting preferences (see section F.2.6)

---

1. Some OS versions and **sendmail** versions support comments in `.forward`. You can test to see if yours does. If it does, you can leave the vacation line in the file, and just comment it out with a pound sign (#) when you don't need it.

2) remove the line of your `.forward` (generally the first line) which contains something similar to `\{username}@{node}.fnal.gov`

3) replace it with `"| /usr/local/products/mh/current/lib/slocal -user {username}"` (include the double quotes but not the braces)

where `{username}` is your login id.

F.4.2 In AFS Environment

Proper ACL settings (AFS directory permissions, discussed in section 7.6.2) must be made for autoincorporation in **MH**. The easiest way to set your mail folder ACLs is to run the utility **fixmailperms** from your `$HOME` directory. This allows the mail folders that you've designated for receiving mail to work with **slocal**, and it sets *all* your mail folders to be readable only by your account and the system administrator's. **fixmailperms** is part of the **UPS** product **mailtools**. To see a brief explanation, enter:

```
% fixmailperms -help
```

The command is entered as:

% fixmailperms [*folder_1 folder_2* **. . .** *folder_n***]**

The folder specifications are taken as relative to your `Mail` subdirectory. You need to specify all the folders into which you plan to automatically incorporate mail, and only those folders.

If you do not specify a minimum of one folder when you execute **fixmailperms**, unattended autoincorporation will not work.

For example, if you have a simple setup in which all incoming mail gets sent to your `inbox` folder, you'd enter:

```
% fixmailperms inbox
```

If this is satisfactory to you, skip to the next section. If instead you choose to set your ACLs individually (and not set your folders private), these are the ACLs you will need:

`system:anyuser rl`	for `$HOME`
`system:anyuser l`	for `Mail`
`system:anyuser li`	for `Mail/{folder}`[1]

Go to the directory whose ACL you want to change, and use the command syntax:

% fs setacl . system:anyuser ***xy***

where ***xy*** is **`rl`**, **`l`**, or **`li`**, depending on the directory. Run **`fs listacl`** to see the current setting.

1. This folder should be your *inbox* or equivalent, e.g., `Mail/inbox`. You may need to set this for several folders, depending on the complexity of your autoincorporation set up.

Appendix G. mh Command Reference

This appendix provides an alphabetical reference to the subset of **mh** commands discussed in Chapter 12.

Remember to use the man pages if you need help on any of these commands.

Command	**Description**
% **comp [-e *editor*]**	Compose a message. This brings up a **vi** (or specified editor) session. Then save the message file, and exit the editor (see box below for more information).

Emergency **vi** exit: hit the Escape key (or <Ctrl-[>) and then enter:

:x	to save and exit, or
:q!	to quit without saving

At the `What Now?` prompt, you can enter a carriage return to get a list of options. Normally you will enter one of these options:

send	to send the message
quit	to save the message in `Mail/drafts` but not send it
quit -delete	to quit and not save

To send a message in which you want to include a file, compose the message, and use the "include file" command for **vi**:

:r *filename*

% **folder +*new_folder_name***	Create a new folder.
% **forw [*n*]**	Forward the current message (or the specified message).
% **inc**	Incorporate new incoming message(s).
% **next**	Display the next message.
% **pick -help**	Display a list of pick options.
% **pick *options***	Select messages by content, date, sender, recipient, etc.
% **prev**	Display the previous message.
% **refile +*folder* [*n m* ...]**	Move the current message (or specified messages) from the current folder to a different folder.
% **repl [*n*]**	Reply to the current (or specified) message. By default, everyone who got the original message gets a copy (the sender, plus everyone in `To:` and `cc:`). Respond to the `What Now?` prompt.
% **repl_inc [*n*]**	Like **repl**, but include the original message. Respond to the `What Now?` prompt.

% **rmf +*folder***	Delete the specified folder.
% **rmm [*n m* ...]**	Delete the current (or specified) message(s).
% **scan [+*folder_name*]**	Display message headers from current folder (or ***folder_name***, and set this folder to current).
% **send *file***	Send ***file*** to all the destinations defined, namely `To:` at a minimum, and `cc:`, if it's filled it in. (If you change your header to include `Bcc:` (blind copy), `Fcc:` (folder copy), it will send it to addresses in these categories as well.)
% **setup mh**	Setup the **mh** product.
% **show +*folder_name***	Display the current message in ***folder_name*** and set this folder to current.
% **show [*n*]**	Display the current (or specified) message.
% **show *n* | flpr -q *queue***	Print message number ***n*** on ***queue***.
% **show *n* > *filename***	Extract current (or specified) message into a file.

% **show -showproc mhl *n* | flpr -q *queue***

Print and format the message a little differently so that it's easier to read (and nicer for printing), and pipe it to **more**. Then send it to the printer.

% **show -showproc pr *m n* ... | flpr -q *queue***

Print messages and separate them onto successive pages. Then send it to the printer.

Appendix H. Mail Conversion from VMS

This appendix is intended to guide you through a mail conversion process from VMS to the UNIX **MH** (Message Handling) system. Two options for conversion are presented.

See the introductory remarks to Chapter 12 before continuing.

H.1 Preparation for Conversion

Moving your VMS mail folders to UNIX is a process initiated from the UNIX system. It uses **ftp** (see section 13.1.1) or the **rcp** and **rsh** commands (see sections 13.1.2 and 13.1.3). Therefore, before you start, you need to set up two files:

- on your VMS machine (the remote node for this operation), a file named `.RHOSTS` that includes records for the UNIX node you're using as your mail destination. Section 13.1.3 describes the contents of this file, appropriate for both UNIX and VMS.
- `.mh_profile` on your UNIX mail node. To create this automatically, run **setup mh** or **setup exmh**. The file will contain the line `Path: Mail`, and that's all you need.

Make sure that v2_2 of **mailtools** is current. Enter:

```
% ups list -a mailtools
```

to see which version is current. If it is not v2_2 or higher, select the right version explicitly by typing:

```
% setup mailtools v2_2
```

If you are moving from FNALV, you should encounter no disk quota problem. The `USR$SCRATCH` area on FNALV is used to hold temporary files. If that is not set up on your VMS machine, you need to ensure that you have enough disk space on your UNIX system and enough disk quota on the VMS node to hold duplicate copies of all of your mail. For VMS, we suggest the commands:

```
$ SHOW QUOTA
```

to see how much space you have available, and

```
$ DIR/SIZE=ALL/GRAND *.MAI;*
```

(from the directory containing your mail) to see how much space you'll need.

For the UNIX side, as usual, it's not so cut and dried. You may be able to use **quota -v** or **df** to provide you the information; check the man pages on your system. On some systems (for instance FNALU running AFS), individual quotas may not be established.

H.2 Choosing the Process to Use

Now you're ready to convert your VMS mail folders to UNIX **MH** mail folders. You can use a "semi-automatic" process, or a totally automatic one.

For either process, the mail messages will be read and extracted into individual filenames in folders for use in **MH** or **exmh**. The filenames are incremented numbers starting at one (1), or the next highest available number for an existing **MH** folder. The **MH** folder name will be the same as the name of the mail folder on the VMS node (unless you rename it during a semi-automatic transfer); e.g., folder `SMITH` will become `/Mail/smith`.

The script used in both processes is `fvms2mh`. For full documentation on the **`fvms2mh`** command enter:

```
% fvms2mh -h |less
```

As mentioned above for those of you for whom `USR$SCRATCH:[username]` is not available or lacks sufficient quota, you need to have enough space on VMS for duplicate copies of your mail while you're doing the conversion. If you use the automatic method, you cannot use any scratch space (there is no "standard" way of assigning scratch space, therefore the automatic method has no way of knowing where it is). However, if you use the semi-automatic method, you can use scratch space for the EXTRACT/ALL second copy, the copy you bring over to your UNIX node.

H.3 Using the Semi-Automatic Process

This process won't take long unless you have many mail folders on VMS to convert. You can see and control all the steps using this method. Perform the following the steps:

1) Log in to the VMS node where your mail is kept.

2) For each folder in your MAIL file (and for each MAIL file that you maintain), enter:

`MAIL> SET FOLDER` *foldername*

`MAIL> EXTRACT/ALL` *tempfile*

3) Log in to the UNIX node where you intend to store your mail and use **ftp** or **rcp** to copy over to your `$HOME` directory the ***tempfile*** containing the extracted VMS mail messages. Sections 13.1.1 and 13.1.2 describe these file transfer utilities.

Do not create a `$HOME/Mail` directory and copy files directly there! Step 4 takes care of this.

4) Now you're ready to convert the copied mail files to **MH** mail folders. If for some reason you haven't run either **`setup mh`** or **`setup exmh`** on UNIX, do so now. Then, for each ***tempfile*** you copied over, enter:

`% fvms2mh` *tempfile*

H.4 Using the Automatic Process

This process simply automates all the steps of the semi-automatic process.

The `fvms2mh` script invoked in its simplest form:

`% fvms2mh -vms` *vmsnode*

performs the “EXTRACT/ALL” command on each of the VMS mail folders in your `MAIL.MAI` file in your default mail directory. Each folder is extracted into a separate file on the VMS node. Each file is then copied to the UNIX node and converted into **MH** format.

☞ Note that ***vmsnode*** must be an individual node, not a cluster alias.

If you have other mail files, you’ll need to convert them separately. For example, if all of your mail files are in the same (default) mail directory on ***vmsnode***, you can enter the command as:

`%` **`fvms2mh –vms`** *vmsnode* **`–file "\*"`**

We use the UNIX quoting syntax here to pass the asterisk wildcard to VMS so that all mail files are handled.

If your login id on VMS is different from your UNIX id, you’ll need to add the **`–l`** switch:

`%` **`fvms2mh –vms`** *vmsnode* **`–l`** *vms_login_id*

Appendix I. Programming Examples

This appendix contains examples of programs that illustrate information presented in Chapters 16 and 17.

I.1 Interfacing C and FORTRAN

The following C/FORTRAN program illustrates some of the points discussed in section 16.9. Our executable is called `cfort`. We assume use of the **make** utility and Makefiles in this example. These are described in chapter 17.

The Makefile:

```
SHELL=/bin/sh
cfort : cf.o fc.o ; f77 -o cfort cf.o fc.o
fc.o  : fc.f      ; f77 -c fc.f
cf.o  : cf.c      ; cc  -c cf.c
```

The FORTRAN routine:

```
      PROGRAM CFORT

*   Test C/FORTRAN mixed programming techniques

*   Place COMMON variables in order REAL*8 , REAL/INTEGER , INTEGER*16 , BYTE
      COMMON /MIXED/ DCOMM, FCOMM, ICOMM
      REAL*8       DCOMM
      REAL         FCOMM
      INTEGER      ICOMM(2)

*   Arguments used in ICF function call
      INTEGER      INTE(2)
      REAL         FLO
      CHARACTER*10 STR

*   Initialize

      DCOMM    = 1.
      FCOMM    = 2.
      ICOMM(1) = 3
      ICOMM(2) = 30
      INTE(1)  = 4
      INTE(2)  = 40
      FLO      = 5.
      STR      = 'From f77'

C   Write initial values

      WRITE (*,*) ' '
      WRITE (*,*) ' / / / From FORTRAN / / / '
      WRITE (*,*) ' DCOMM  = ', DCOMM
      WRITE (*,*) ' FCOMM  = ', FCOMM
```

```
      WRITE (*,*) ` ICOMM  = `, ICOMM
      WRITE (*,*) ` INTE   = `, INTE
      WRITE (*,*) ` FLO    = `, FLO
      WRITE (*,*) ` STR    = `, STR
      WRITE (*,*) ` `

C   Use C function icf_, which prints, changes, and reprints COMMON and
C   Argument values

      IRET = ICF ( INTE , FLO , STR )

C   Write values after call to ICF
      WRITE (*,*) ` / / / Back in FORTRAN / / / `
      WRITE (*,*) ` DCOMM  = `, DCOMM
      WRITE (*,*) ` FCOMM  = `, FCOMM
      WRITE (*,*) ` ICOMM  = `, ICOMM
      WRITE (*,*) ` INTE   = `, INTE
      WRITE (*,*) ` FLO    = `, FLO
      WRITE (*,*) ` STR    = `, STR
      WRITE (*,*) ` IRET   = `, IRET
      WRITE (*,*) ` `

      STOP
      END
```

The C Routine (`icf_`):

```
           /*  Test of mixed C/FORTRAN features    */

int icf_( int* inte , float* flo , char* str , int lenstr ) {
extern struct {
    double dcomm ;
    float  fcomm ;
    int    icomm[2] ;
    } mixed_ ;

printf ( "  / / / Into C / / /\n\
  DCOMM  = %f\n\
  FCOMM  = %f\n\
  ICOMM  = %d %d\n\
  INTE   = %d %d\n\
  FLO    = %f\n\
  STR    = '%.10s'\n\
  LENSTR = %d\n\n"
  ,  mixed_.dcomm , mixed_.fcomm , mixed_.icomm[0] , mixed_.icomm[1]
  , *inte , *(inte+1) , *flo , str , lenstr ) ;

mixed_.dcomm    = 10000. ;
mixed_.fcomm    = 20000. ;
mixed_.icomm[0] = 30000  ;
mixed_.icomm[1] = 330000 ;
*inte           = 40000  ;
*(inte+1)       = 440000 ;
*flo            = 50000. ;

strcpy ( str , "Out of C" ) ;

printf ( "  / / / Out of C / / /\n\
  DCOMM  = %f\n\
  FCOMM  = %f\n\
  ICOMM  = %d %d\n\
  INTE   = %d\n\
  FLO    = %f\n\
  STR    = '%.10s'\n\n"
```

```
  ,  mixed_.dcomm , mixed_.fcomm , mixed_.icomm[0] , mixed_.icomm[1]
  , *inte , *flo , str ) ;

return ( lenstr ) ;

}
```

Program Execution (see section 16.10):

```
/ / / From FORTRAN / / /
DCOMM  =    1.000000000000000
FCOMM  =    2.000000
ICOMM  =            3          30
INTE   =            4          40
FLO    =    5.000000
STR    = From f77

/ / / Into C / / /
DCOMM  = 1.000000
FCOMM  = 2.000000
ICOMM  = 3 30
INTE   = 4 40
FLO    = 5.000000
STR    = `From f77  `
LENSTR = 10

/ / / Out of C / / /
DCOMM  = 10000.000000
FCOMM  = 20000.000000
ICOMM  = 30000 330000
INTE   = 40000
FLO    = 50000.000000
STR    = `Out of C'

/ / / Back in FORTRAN / / /
DCOMM  =    10000.00000000000
FCOMM  =    20000.00
ICOMM  =        30000      330000
INTE   =        40000      440000
FLO    =    50000.00
STR    = Out of C
IRET   =           10
```

I.2 Makefiles and the make Process

I.2.1 A Simple make Process

Our “Hello World” example is quite simple, and illustrates many of the features we have discussed in Chapter 17. First we’ll run a file listing to see the files that are involved:

```
-rw-r--r--   1 aheavey  g020         515 Dec 20 16:27 Makefile
-rw-r--r--   1 aheavey  g020          91 Dec 20 16:27 mcprog.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model1.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model2.f
```

The contents of the FORTRAN source files that will be used as required files follows:

```
<fsui01> cat mcprog.f
      WRITE (*,*) ` Hello world `
      CALL MODEL1
      CALL MODEL2
      STOP
```

```
      END
```

```
<fsui01> cat model1.f
      SUBROUTINE MODEL1
      WRITE (*,*) ' Hello from MODEL 1 '
      RETURN
      END
```

```
<fsui01> cat model2.f
      SUBROUTINE MODEL2
      WRITE (*,*) ' Hello from MODEL 2 '
      RETURN
      END
```

Our Makefile illustrates the use of macro definitions, iterative target defintions, the use of a library in addition to other required files, comment lines, and macro expansion (described in the man pages). It provides two levels of "housekeeping" at the end, *clean* to remove just the "stray" files, and *clobber*, which gets rid of all resulting files thus allowing you to run **make** again under the same initial conditions.

```
<fsui01> cat Makefile
# use /bin/sh within make
SHELL = /bin/sh

#   Standard FORTRAN command and flags
F77    = f77
FFLAGS = -O2

# Build the 'mcprog' program from mcprog.f and libmc.a
mcprog   : mcprog.f   libmc.a ; $(F77) $(FFLAGS) -o mcprog mcprog.f libmc.a

# Move any missing modules into libmc.a
libmc.a  : model1.o  model2.o ; ar -r libmc.a $(?)

# Compile missing or out-of-date modules
model1.o : model1.f ;     $(F77) $(FFLAGS) -c $(?)

model2.o : model2.f ;     $(F77) $(FFLAGS) -c $(?)

clean    : ; rm -f *.o

clobber : ; rm -f *.o ; rm -f mcprog ; rm -f libmc.a
```

Run **make** without specifying a target, and it will use the first target it encounters, `mcprog`. Notice that **make** displays its progress on the screen, creating the library `libmc.a` prior to building the executable `mcprog`:

```
<fsui01> make
f77 -O2 -c model1.f model1.f
model1.f:
        model1:
model1.f:
        model1:
f77 -O2 -c model2.f model2.f
model2.f:
        model2:
model2.f:
        model2:
ar -r libmc.a model1.o model2.o
ar: creating libmc.a
f77 -O2 -o mcprog mcprog.f libmc.a
mcprog.f:
 MAIN:
```

Run another file listing to see that our target and some intermediate files were created:

```
-rwxr-xr-x   1 aheavey  g020       13452 Dec 20 16:34 mcprog*
-rw-r--r--   1 aheavey  g020        2712 Dec 20 16:34 libmc.a
-rw-r--r--   1 aheavey  g020        1248 Dec 20 16:34 model2.o
-rw-r--r--   1 aheavey  g020        1248 Dec 20 16:34 model1.o
-rw-r--r--   1 aheavey  g020          91 Dec 20 16:27 mcprog.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model2.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model1.f
-rw-r--r--   1 aheavey  g020         515 Dec 20 16:27 Makefile
```

Run the executable, `mcprog`:

```
<fsui01> mcprog
  Hello world
  Hello from MODEL 1
  Hello from MODEL 2
```

Run **`make clean`** to remove the object files:

```
<fsui01> make clean
rm -f *.o
```

A directory listing shows the object files were deleted:

```
-rwxr-xr-x   1 aheavey  g020       13452 Dec 20 16:34 mcprog*
-rw-r--r--   1 aheavey  g020        2712 Dec 20 16:34 libmc.a
-rw-r--r--   1 aheavey  g020          91 Dec 20 16:27 mcprog.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model2.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model1.f
-rw-r--r--   1 aheavey  g020         515 Dec 20 16:27 Makefile
```

Finally, run **`make clobber`** to remove all the created files:

```
<fsui01> make clobber
rm -f *.o ; rm -f mcprog ; rm -f libmc.a
```

... and check the final directory listing, which should match the original one:

```
-rw-r--r--   1 aheavey  g020         515 Dec 20 16:27 Makefile
-rw-r--r--   1 aheavey  g020          91 Dec 20 16:27 mcprog.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model1.f
-rw-r--r--   1 aheavey  g020          88 Dec 20 16:27 model2.f
```

I.2.2 A Physics Makefile

The following is an example physics Makefile with FORTRAN and C sources.

```
###############
#   MACROS    #
###############

# Always force the shell to be /bin/sh, for portability

SHELL=/bin/sh

# some variables come from the environment existing when make is run

#    CRNLIB       (from 'setup cern')
#    HISTO_DIR    (from 'setup histo')
#    MCFAST_DIR   (a group library from 'source ~bphyslib/mcfast/mcenv')
#    MCFIO_DIR    (a group library from 'source ~bphyslib/mcfast/mcenv')

# macros for directories containing INCLUDE files:

INC1 = $(MCFAST_DIR)/inc/event
```

```
INC2 = $(STDHEP_INC)

# macro for MCFAST object library directory

LIB = $(MCFAST_DIR)/lib.IRIX

# macro listing all USER object routines

OBJS  = \
  usr_analysis.o \
  usr_before_trigger.o \
  assign_off_mass.o \
  costh.o \
  fin_init.o \
  find_b_psiks.o

# macros for object libraries from products HISTO and CERN

HISTLIB = $(HISTO_DIR)/lib/libFHistoHB.a
PACKLIB = $(CRNLIB)/libpacklib.a
MATHLIB = $(CRNLIB)/libmathlib.a
KERNLIB = $(CRNLIB)/libkernlib.a

# macro for the SGI exception handler library

FP_EXCEPTION = /usr/lib/libfpe.a

# macro for all linked libraries

LOADLIBS= \
  $(LIB)/libgen.a      \
  $(LIB)/libdatafile.a \
  $(LIB)/libio.a       \
  $(LIB)/libutil.a     \
  $(MCFIO_DIR)/lib.IRIX/libFmcfio.a \
  $(HISTLIB) \
  $(PACKLIB) \
  $(MATHLIB) \
  $(KERNLIB) \
  $(FP_EXCEPTION)

# macros for compiler option flags

CFLAGS = -g
FFLAGS = -g -I$(INC1) -I$(INC2)

###############
#   TARGETS   #
###############

# f77 is used to run the link editor here.
# FORTRAN and C compilation is done via the rules defined below.

mcfast: $(OBJS) $(LOADLIBS); f77 -o mcfast $(OBJS) $(LOADLIBS)

###############
#  SUFFIXES   #
###############

# You must list all suffixes used in this Makefile's rules

.SUFFIXES : .c .F .f .o

###############
#    RULES    #
```

```
###############

# RULES for building *.o files from *.f or *.c

#     The $? internal macro expands to the list of *.f or *.c files which are
#     newer than their corresponding *.o file.

#     Macros CFLAGS and FFLAGS are defined above with compiler options

.f.o: ; f77 -c $(FFLAGS) $?

.c.o: ; cc  -c $(CFLAGS) $?
```

Index

Symbols

Hidden Files

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z