

Introduction to Using *RA* and the Unix Operating System at
Mississippi State University
for use in ST 8353 *Statistical Computations*

Jane L. Harvill

Fall, 2003

Contents

1	Introduction	3
1.1	Creating a RA account	3
1.2	Accessing and Logging Into Your RA Account	3
1.3	Logging Out of Your RA Account	3
1.4	Changing Your RA Password	3
2	Unix Overview	3
2.1	Unix File Structure	4
2.2	Unix Commands for File Management	4
2.2.1	Creating, Viewing, and Removing Files	4
2.2.2	Moving Between and Within Directories	5
2.3	Help in Unix	5
3	Editing Files	5
3.1	The vi Editor	6
3.2	Emacs	6
4	E-Mail on RA	7
4.1	Basic E-Mail Program	7
4.2	The PINE E-Mail Program	7
5	Space Management	8
5.1	Archiving (and Unarchiving) Files	8
5.2	Compressing (and Decompressing) Files	8
6	Some Other Commands	8
6.1	The <code>grep</code> Command	8
6.2	File Modes and Changing File Modes	9

1 Introduction

This document is intended to serve only as an introduction to RA and the Unix operating system. There is much more to using RA than what can be addressed here. For more information on a number of topics, please refer MSU's Information Technology Services web page at

www.its.msstate.edu/Services/Support/Unix/.

The links labeled **Unix Commands** and **XWin32** should provide especially helpful. The (PDF) document at the **Unix Commands** link explains some of the basic commands of the Unix operating system. The software *XWin32* is free through MSU and allows you access to your RA Unix account on your PC.

1.1 Creating a RA account

Before you can use the Unix system known as RA you need a username and password. These are created when you create a RA account. The easiest way to create an account on RA is by using the internet. You can get access to the internet in the the Mathematics & Statistics Department Computer Lab. Open a web browsing program such as *Netscape* or *Internet Explorer*, and go to the MSU ITS web site at <http://www.its.msstate.edu/>.

Click on the **Accounts** link on the right of the page in the "Frequently Requested Services" box. Then click on **Create a New RA Account** and follow the directions.

1.2 Accessing and Logging Into Your RA Account

Two of the most widely available methods at MSU for accessing RA are *telnet* and *X-Win32*. *X-Win32* is the preferred method. If *X-Win32* is not available, RA can be accessed using a standard telnet program. In either case, to login to your RA account, at the **login:** prompt, type your username and strike the **Enter** key. Then at the **Password:** prompt, type your password, and strike the **Enter** key.

If the message "**login incorrect**" appears, then a mistake was made when you typed your username or password. Try again. If you get the same error a second time, and you're sure you've typed everything correctly, make sure the **Caps Lock** is not on. Unix is case sensitive. In other words, an "A" isn't the same as an "a".

1.3 Logging Out of Your RA Account

When you are finished with your session on RA, you must logout. This is accomplished by typing **logout** or **exit**, and striking the **Enter** key. If this does not work, try typing **<Ctrl>-d**; that is, hold down the **Ctrl** key and the **d** key simultaneously.

1.4 Changing Your RA Password

To change your password on RA, issue the command **passwd**. You will be prompted to enter your current password. Enter the password used to login at the beginning of this session. If you type your current password incorrectly, the error message

```
passwd(SYSTEM): Sorry, wrong passwd
Permission denied
```

will appear. If you correctly type in the old password, you will then be prompted to type in your new password. The next prompt will ask you to re-type the new password for confirmation. If you make a mistake so that the new password does not match the confirmation, the message **Mismatch - password unchanged**. will appear and your password will remain unchanged. Try again if you still desire to change your password.

2 Unix Overview

Unix is the name of the operating system running on RA. Roughly speaking, an *operating system* is the set of commands that provides the ability to access files on that system. The idea of "accessing" files is much deeper than it may appear. The surface interpretation of "accessing" a set of files may be something as simple as viewing the list of files that resides in a particular directory. Accessing a file can also be as complicated as issuing a command that runs a program associated with the file. The word "accessing" (in the definition of operating system) also refers to the ability to find and run an executable file. A Unix executable file will **not**, in general, run on a PC running a *Windows* operating system, and vice versa. Subsection 2.2 contains a partial listing of commands that will help you in creating new files and directories, removing files and directories, and moving between and within directories. Subsection 2.3 explains how to use the help feature in Unix.

2.1 Unix File Structure

Files on RA are stored in a structure that is similar to that of *Windows*. Each time you login to your RA account, you begin in the *root* or *home* directory. The root directory is analogous to the `C:\` drive *Windows*. Directories in RA are analogous to folders in *Windows*. In the root directory, you can create new files and new directories. You can create files and directories within directories, and so forth and so on. A directory that contains another directory is referred to as a *parent directory*; the directory it contains is referred to as the *child directory*, or simply the *directory*. The commands in Section 2 are for working with the files and file structures within your RA account.

2.2 Unix Commands for File Management

The commands described in Subsection 2.2.1 are for creating and removing directories and for removing files. The commands described in Subsection 2.2.2 explain how to move between and within directories. You have an allotted amount of drive space on RA. To display the amount of disk space issue the command

```
quota -v
```

The `-v` in this command is called a *flag*. Most Unix commands have various flags for specifying different options associated with that command. The `mkdir` and `rmdir` commands in Subsection 2.2.1 are two examples of how to use flags for variations on base commands.

2.2.1 Creating, Viewing, and Removing Files

- To make a directory, use the command

```
mkdir directory.name
```

where *directory.name* is replaced by the name you desire for the newly created directory.

- The command

```
mkdir -p parent.dir/directory.name
```

ensures that the parent directory named in *parent.dir* exists. If it does not, the directory *parent.dir* is created and the new child directory *directory.name* is created within *parent.dir*. (Note the forward slash separating the directory names.)

- To remove an empty directory, use the command

```
rmdir directory.name
```

where *directory.name* is replaced by the name of the directory that is to be removed.

- If the directory to be removed is not empty, use the command

```
rmdir -r directory.name
```

The *flag* `-r` means “recursively.” This will remove the directory specified by *directory.name* as well as all other files and directories within that directory.

- The command

```
rmdir -p parent.dir/directory.name
```

removes the directory *parent.dir* if it becomes empty after the child directory *directory.name* is removed. (Note the forward slash separating the directory names.)

- To remove a file, use the command

```
rm filename
```

where *filename* is replaced by the name of the file that is to be removed.

- To move files and directories from one place to another use the command

```
mv filename1 filename2
mv directory1 directory2
mv filename directory
```

- To view the contents of a file issue the command

```
more filename
```

where *filename* is replaced by the name of the file to be viewed. To move to the next page of the displayed file, use the spacebar. While the file is being viewed, to search for a pattern within the file type */pattern*. The quit viewing the file, type `q`.

- To view the end of a file, use the command

```
tail filename
```

The command

```
tail -nN filename
```

will display the last *N* lines of *filename*.

2.2.2 Moving Between and Within Directories

- The command

```
cd dir.name
```

moves you down one level from the current to the directory specified by *dir.name*. If the directory specified by *dir.name* is not in the current directory you will get the error message

```
cd: dir.name:No such file or directory
```

This message does not mean the directory *dir.name* does not exist in your account. It only means that *dir.name* does not exist in the current directory.

- To move up one level of the directory tree, use the command

```
cd ..
```

- To move to another directory using a relative pathname use the command

```
cd ../directory.name
```

This moves you up one level in the directory tree and then moves you to the subdirectory *directory.name*.

- To move directly to your root or home directory from any other directory, simply issue the command

```
cd
```

- To see a listing of the files and directory that exist in the current directory issue the command

```
ls
```

You can also add directory names and wildcard characters `*` to get more specific listings.

- To see which directory you are currently in use the command

```
pwd
```

`pwd` stands for pathway of working directory.

2.3 Help in Unix

The command

```
man
```

gives you access to an on-line manual containing a complete description of every command available on the Unix system. The `man` command can also provide you with on-line descriptions of commands specified by name; or for all commands whose description contains any of a set of keywords. To view the manual page for a command, enter

```
man command.name
```

where *command.name* is replaced by the command for which you desire help.

The content of every `man` page is organized under a number of headings. To check what a command does issue the command

```
whatis command.name
```

where *command.name* is replaced by the command for which you desire help. This will display the description for the command that is given in its online manual page.

The command

```
man -k keyword
```

displays an on-line synopsis of each command that has *keyword* in its description. Enclose a phrase in single quotes to search for it.

The command

```
apropos
```

locates commands by keyword lookup. It displays the `man` page name, section number, and a short description for each `man` page whose name line contains the keyword.

Finally, many commands contain an abbreviated help utility that is easily accessed by typing *commandname* `--help`. For example, for quick help on the `mkdir` command, type

```
mkdir --help
```

at the Unix prompt.

3 Editing Files

Two universal Unix editors are *vi* and *emacs*. Partial listings of the on-line tutorials are reproduced below for your reference.

3.1 The vi Editor

`vi` (visual) is a display-oriented text editor based on an underlying line editor `ex`. The visual commands are described on this manual page; how to set options (like automatically numbering lines and automatically starting a new output line when you type carriage return) and all `ex` line editor commands are described on the `ex(1)` manual page.

When using `vi`, changes you make to the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file.

The view invocation is the same as `vi` except that the read only flag is set.

The `vedit` invocation is intended for beginners. It is the same as `vi` except that the report flag is set to 1, the showmode and novice flags are set, and magic is turned off. These defaults make it easier to learn how to use `vi`.

To invoke the `vi` editor, issue the command

```
vi filename
```

where *filename* is the name of the file to be edited. If you do not specify a filename, the `vi` editor opens. From here you can enter text and save the changes into a new file.

For more information on using `vi`, issue the Unix command `man vi` or connect to an excellent tutorial on the web through the Purdue University. A link to this site is provided on WebCT on the *Links to Technical Writing* page.

3.2 Emacs

Emacs is another editor universally available on Unix systems. It is an advanced, self-documenting, customizable, extensible real-time display editor Emacs. We say that Emacs is a display editor because normally the text being edited is visible on the screen and is updated automatically as you type your commands.

We call it a real-time editor because the display is updated very frequently, usually after each character or pair of characters you type. This minimizes the amount of information you must keep in your head as you edit.

We call Emacs advanced because it provides facilities that go beyond simple insertion and deletion: controlling subprocesses; automatic indentation of programs; viewing two or more files at once; editing formatted text; and dealing in terms of characters,

words, lines, sentences, paragraphs, and pages, as well as expressions and comments in several different programming languages.

Self-documenting means that at any time you can type a special character, Control-h, to find out what your options are. You can also use it to find out what any command does, or to find all the commands that pertain to a topic.

Customizable means that you can change the definitions of Emacs commands in little ways. Extensible means that you can go beyond simple customization and write entirely new commands, programs in the Lisp language to be run by Emacs's own Lisp interpreter. Emacs is an "on-line extensible" system, which means that it is divided into many functions that call each other, any of which can be redefined in the middle of an editing session. Almost any part of Emacs can be replaced without making a separate copy of all of Emacs. Most of the editing commands of Emacs are written in Lisp already; the few exceptions could have been written in Lisp but are written in C for efficiency. Although only a programmer can write an extension, anybody can use it afterward. If you want to learn Emacs Lisp programming, the authors of emacs recommend the *Introduction to Emacs Lisp* by Robert J. Chassell, also published by the Free Software Foundation.

When run under the X Window System, Emacs provides its own menus and convenient bindings to mouse buttons. But Emacs can provide many of the benefits of a window system on a text-only terminal. For instance, you can look at or edit several files at once, move text between files, and edit files while running shell commands.

To invoke the `emacs` editor, issue the command

```
emacs filename
```

where *filename* is the name of the file to be edited. If you do not specify a filename, the `emacs` editor opens. From here you can enter text and save the changes into a new file. If you are running *XWin-32*, the command `emacs` can be replaced by `xemacs`. This runs an enhanced version of the `emacs` programs that provides menus and point-and-click buttons.

The *GNU Emacs Manual* is available in a variety of file formats at the URL www.gnu.org/manual/emacs/emacs.html.

4 E-Mail on RA

Three programs for accessing e-mail on RA are `mail`, `PINE`, and *IMAP WebMail*. The universal Unix e-mail program is `mail`. It is described in Subsection 4.1. The `PINE` e-mail program is not a universal e-mail program, but it is more user-friendly than `mail`. `PINE` is described in Subsection 4.2. You should decide to use either `mail` or `PINE`. Moving between the two is not recommended. The *IMAP WebMail* system is set up for users of `PINE` and requires an internet browser such as *Netscape* or *Internet Explorer*.

4.1 Basic E-Mail Program

Every Unix system has “the” mail program. It is command driven (as opposed to menu driven) and so is more difficult to use than `PINE` (which is menu driven). However, `PINE` does not exist on all Unix systems. It is best if you decide to use either `mail` or `PINE`. It is not recommended that you use both systems. To invoke the universal Unix e-mail program, issue the command `mail`. A `?` prompt will appear and wait for your command before any action is taken. The `mail` commands are

```
?          print this help message
#          display message number #
-          print previous
+          next (no delete)
! cmd     execute cmd
<CR>      next (no delete)
a          position at and read newly
           arrived mail
d [#]     delete message # (default
           current message)
dp        delete current message and
           print the next
dq        delete current message and
           exit
h a       display all headers
h d       display headers of
           letters scheduled for
           deletion
h [#]     display headers around #
           (default current
           message)
m user    mail (and delete) current
           message to user
n         next (no delete)
p         print (override any
```

```
warnings of binary
content)
P         override default 'brief'
           mode and display ALL
           header lines
q, ^D    quit
r [args] reply to (and delete)
           current letter via
           mail [args]
s [files] save (and delete) current
           message (default
           mbox)
u [#]    undelete message #
           (default current
           message)
w [files] save (and delete) current
           message without
           header
x        exit without changing mail
y [files] save (and delete) current
           message (default
           mbox)
```

4.2 The `PINE` E-Mail Program

`PINE` is a Unix based menu-driven e-mail client created by the University of Washington in Seattle and is available at MSU. `PINE` is not necessarily on Unix systems at other locations. However, since it is free for public use, it is quite popular. Directions on using `PINE` can be found on the internet at the MSU ITS web site.

IMAP WebMail E-Mail Program `IMAP WebMail Program` (`IMP` for short) is a Web-based e-mail program that MSU students, faculty, and staff can use as an alternate to `PINE` to access e-mail from their RA account. `IMP` is similar to other Web-based e-mail systems available on the internet, however, it is exclusive to MSU users. Advantages include on-campus support, increased security, and better performance for the MSU user community. `IMP`'s point-and-click interface allows users to access their RA e-mail from any computer that is connected to the web. This means users can sit at any machine anywhere in the world and check and send e-mail without having to perform any special configurations. `IMP`'s ease of use allows users at any skill level to perform any type of e-mail function. Enter

<http://webmail.msstate.edu>

in your browser to get `IMP`. Complete instructions

for getting started with IMP are available at the ITS web site.

5 Space Management

Oftentimes the occasion arises when somewhat large seldom used files exist on your account that are seldom used, but that you do not want to delete. In order to make the most efficient use of your allocated disk quota, you may want to compress and archive these files. *It is common practice to first archive a directory using the `tar` command, and then to compress that archived directory using the `gzip` command.* There are number of Unix compression and archiving utilities available. In the paragraphs that follow, I explain two: `gzip` and `tar`.

5.1 Archiving (and Unarchiving) Files

The `tar` command archives and extracts files to and from a file called a `tarfile`. The files that are added to the `tarfile` are not removed from their original location. To archive a file (or directory), issue the command

```
tar -cvf tarfile.tar filename1 filename2 ...
```

It is possible to change the `.tar` extension on the `tarfile` filename, but it is not recommended. In addition to `cvf` the `tar` command has a number of other flags. There are also a number of options you can use in working with `tar` for more control over what is and is not included in the `tar` function. Only the `c`, `v`, `f`, `r`, `u` and `x` flags of the `tar` command are described here. For more information on `tar`, refer to the `man` pages, or issue the command `tar --help`.

- `c` The `c` flag in the `tar` command means “create” a `tarfile`. By default, writing begins at the beginning of the `tarfile` instead of at the end (see flags `r` and `u`).
- `r` The `r` flag can be used in place of the `c` flag. In this case, the files named in `filename1 filename2`, etc. are written at the end of the `tarfile`.
- `u` The `u` flag can be used in place of the `c` and `r` flags. It causes the files named in `filename1 filename2`, etc. to be written at the end of the `tarfile` if they are not already in the `tarfile` or if they have been modified since last written to `tarfile`. An update can be rather slow.

- `x` This flag will extract or restore the contents of the `tarfile` to their original form. You can name specific files contained in the `tarfile` to be extracted without having to extract the entire set of files.
- `v` The `v` (verbose) flag in the `tar` command results in the name of each file added to the `tarfile` being written to the window while the files are being archived.
- `f` The `f` flag preceded the name to be assigned to the `tarfile`.

To unarchive a file (or directory), issue the command

```
tar -xvf tarfile.tar
```

5.2 Compressing (and Decompressing) Files

The command `gzip` is used to compress files. To compress a files, issue the command

```
gzip filename
```

The resulting compressed file will have the original `filename` with an added `.gz` extension. It is possible to change the `.gz` extension, but is not recommended. The `gzip` command has a number of flags.

Files can be restored to their original form using one of

```
gzip -d filename.gz
```

```
gunzip filename.gz
```

For more information on `gzip` and `gunzip`, refer to the `man` pages, or issue the command `gzip --help` at the Unix prompt.

6 Some Other Commands

Some other useful Unix commands are described below.

6.1 The `grep` Command

The `grep` command is used to search for a pattern within each file. There are a number of flags and options for the `grep` command. For more help on the `grep` command, refer to the `man` pages, or issue the command `grep --help` at the Unix prompt. The `grep` command takes the form

```
grep -e expression filename1 filename2 ...
```

For example, to search the files called `ra.tex` and `vi.tex` for the expression `then`, the command is

```
grep -e then ra.tex vi.tex
```

If I wanted to search for an expression containing special characters, or a space, such as `then return`, the same format is used except that the expression should be enclosed in double quotations. For example,

```
grep -e "then return" *.tex
```

results in the `grep` command searching for the expression `then return` in every file in the working directory with a `.tex` ending.

6.2 File Modes and Changing File Modes

Your RA account is set up so that you can allow (or disallow) other users to view your files. The permissions set on each file are called the *mode* of the file. You can view a file's mode using `ls -l`. For a directory containing the file `syllabus.tex` and the directory `supplements` the output of `ls -l` would look something like this

```
drw----- 1 harvill user ... supplements
-rw----- 1 harvill user ... syllabus.tex
```

The first column of this output contains letters and `-s`. Each of these letters imply something about the type and mode of the directory and file. *Note:* The `...` in the example replaces information about the size of the file, and the date and (military) time the file was last modified.

First notice that the directory `supplements` has a `d` as the first character in the first column. That `d` tells you that `supplements` is a directory. The first character in the first column for `syllabus.tex` is a `-` indicating that `syllabus.tex` is not a directory, it is a file.

The next three symbols in the first column for both `supplements` and `syllabus.tex` are `rw-`. These first three characters indicate what mode the file is in for the user `harvill`. The `r` means the user `harvill` has permission to *read* the contents of the file. The `w` means the user `harvill` has permission to *write* to the file. The `-` indicates a lack of permission for `harvill` to *execute* the file. If `harvill` did have permission to execute the file, the `-` would have been replaced by an `x`. The lack of permission to execute

a file is usually not a problem unless the file is an executable program file (compiled from some computer language).

The next group of three `-` indicate there are no **groups** with any of read, write, or execution permissions. The last set of three `-` indicate there is no **other** user with any of the three permissions.

At times, it is desirable to change permissions on a file (for example, if the file contains the HTML code for a web page to be viewed by the public). The command `chmod` changes the mode of a file. The options and flags on `chmod` are varied. Below are a few examples to show you how to use this command to add and remove permissions.

Suppose you wanted to *add* read and execute permission for others to the file `syllabus.tex`. Then the command would be

```
chmod o+rx syllabus.tex
```

In this command the `o` stands for others, the `+` indicates the permissions following the `+` are to be added to others. The `r` and `x` are for read and execute, respectively.

If you later decided to *remove* the read and execute permission for others from `syllabus.tex`, the command is

```
chmod o-rx syllabus.tex
```

Notice the `+` in the former command changes to a `-` in the latter.

Now suppose you wanted to add read, write, and execute permission for **groups**, **others**, and **users** for the directory `supplements`. The `chmod` command takes the form

```
chmod gou+rwx supplements
```

Note that this only allows groups, others, and users to view the listing of files contained in `supplements`. It doesn't allow anyone to view the files. If you wanted to recursively add permissions throughout the directory `supplements`, you can use the `-R` flag so that the form of the `chmod` command becomes

```
chmod gou+rwx -R supplements
```

This gives all users permission to read, write, and execute any file in the `supplements` directory.

One final note: if you want all users to have a set of permissions, you can replace the `gou` with an `a`.