UNIX

A Beginner’s Guide to
Unix, vi and X-Windows

Stephanie Wolf
Joint Institute of Computational Science
University of Tennessee
Knoxville, TN 37996–1301, USA

Last Revision: September 25, 1995
Author’s Email Address: jics@cs.utk.edu
Copyright (C) 1991, 1992, 1993, 1994, 1995 by the Computational Science Education Project

This electronic book is copyrighted, and protected by the copyright laws of the United States. This (and all associated documents in the system) must contain the above copyright notice. If this electronic book is used anywhere other than the project’s original system, CSEP must be notified in writing (email is acceptable) and the copyright notice must remain intact.
1 Unix

UNIX, written by Ken Thompson at Bell Laboratories in 1969, is the result of an attempt to create an operating system with an environment which would be simple for users. “The UNIX kernel, the heart of the operating system” was written in C by Thompson and Dennis Ritchie four years later in order to produce a portable version of the operating system. Portable means the code was not specifically written for one type of machine only.[Abrahams]

Some UNIX architecture terms which may be useful to know are kernel, shell, and user interface. The kernel, or main operating system, schedules the Central Processing Unit, or C.P.U., and allocates hardware resources. The shell calls different system utilities in response to a user’s request. The user interface is the method of communication between the user and the operating system. This may include keyboard, mouse, or monitor screen.

There are various flavors of Unix available. The most common ones are based on SVR4 (System 5 Release 4) from AT&T (now Unix Systems Laboratory). For example, Sun Solaris 2.x, SGI IRIX, IBM AIX and HPUX are based on SVR4. Another version of Unix is 4.3 BSD (Berkeley Software Distribution) on which the Sun OS 4 operating system was based. This guide is written as a generic platform independent introduction to SVR4 Unix. Some important differences from BSD Unix are noted. However, because there are often small differences between the Unix operating systems on different computer systems, the definitive reference should always be the online manual pages (see Section 1.7).

1.1 Logging In:

You may login to a Unix system in a number of ways. For example, you may login at the console of a workstation, or through an X terminal, or using rlogin, rsh or telnet from another computer on the network.

A workstation is a type of computer capable of operating in a windowing environment (in which the user can do several things at once in different windows on the screen attached to the computer). A workstation is often used by only one user (who sits at the screen). Unlike an X terminal (see below), however, a person may login to the workstation from another computer in the network and run programs on that workstation.

An X terminal, on the other hand, is a windowing display device whose only purpose is to connect the user sitting at the screen to other computers on the network in a windowing environment.

Consult your local computer administrator for how to use a windowing environment on your display device. (Some workstations automatically start you in a windowing environment whereas on others you may have to initiate the windowing environment yourself using, for example, the xinit or startx commands.) See Section 4 for an example of using an X window system at the University of Tennessee Computer Science Department.

If the computer you wish to work with has a dark screen, do not panic. The screen saver has been activated after keys have not been touched for a certain amount of time. (The screen saver keeps images from being permanently written into the screen.) Simply press
return and the login prompt should appear. At this prompt, type in the username which you were given and press return. Then, the password prompt will appear and you simply type in your password followed by return. This password will be given to you the first time you login. (You should change your password after the first time to something which is easy for you to remember so you do not have to write it down anywhere.)

As you type in your password, the characters you type WILL NOT appear on the screen.

If you type the password incorrectly or take too much time attempting to type the password, Login Incorrect will appear on the screen. You will then be given another chance to type in both your username and password.

Please remember that Unix is case-sensitive. Your password should be typed in exactly as written on the sheet of paper (for first login) or exactly as you typed it in when you changed your password with upper and lower case letters, numbers, and symbols the same as before.

1.2 Changing your Password:

Usually this is done using the yppasswd command. After entering this command, you are prompted for the current password, which you wish to change, the new password, and the new password again for confirmation. On some systems, local modifications have been made to the yppasswd command to force users to choose complicated passwords not containing easily guessed words or to force a mixture of upper and lower case characters and numbers. You should thus consult your local systems administrator for site-specific instructions on how to change passwords on your system.

None of the characters entered when you are specifying the old and new passwords will be shown on the screen.

If everything goes fine, the screen will say Password successfully changed. (or something similar) and after you logout, the new password will be effective.

Otherwise some error message will occur. If this happens, try again.

Passwords must be at least 6 characters long and only the first 8 characters will be used by the system. You can use upper and lower-case letters, numbers, and other symbols. You cannot use the space character in your password. Try to use a password that IS NOT a dictionary word. Those types of passwords are easily “cracked”. Instead, use a combination of symbols, numbers, and/or upper- and lower-case letters that will remind you of a phrase, someone, or something. Below are a few examples.

Examples

MhalBaa  (Mary had a little Lamb)
ChtCht!!  (Chitty Chitty Bang Bang)
BudMan#1  (Budman is number one)
D,P!dttg  (Daddy, Please don’t take the girl)
+ictd2lv  (Addicted to Love)
Songs, artists, places, favorite people, wise sayings, pets, etc. are some types of ideas. Be creative - but not so creative that you do not remember your password!

1.3 Logging Out:

In general, the logging off instructions are specific to the type of display device, and you should consult your local computer administrator for how to do this on the workstation or X terminal you are using.

If you are logged in to remote computers, either via telnet, rlogin or rsh, whether you are using a windowing environment or not, you should always terminate all remote connections before logging off. This is done as follows.

First “kill” all processes or jobs. (See ps or jobs for information on “killing” processes or jobs.) Then “kill” all connections to remote computers.

To “kill” the connection and logout of a remote computer, simply type exit in the window which you used to rlogin, telnet, or rsh to the remote computer.

After you have killed all the processes, jobs and connections to remote computers, follow your local site-dependent instructions for terminating your X-Windows session if you are using an X-Windows environment. If you have followed the procedure correctly, a new login prompt should appear. If this doesn’t happen, seek help.

Note: If you are using a workstation or X terminal, you should not turn it off when you are finished unless your local computer administrator indicates otherwise.

1.4 Unix Directories:

All directories on the system are subdirectories of the root directory. The root directory is physically located on the hard drive of the machine onto which you are logged in. Every directory can contain files and/or subdirectories. Any path which begins at the root starts with / (or slash). A path can lead to directories physically located on other computers. The network system makes this transparent to the user. Any path which begins at your home directory begins with ~/ (or a tilde followed by a slash). Any path which begins with ~username (or tilde username) starts in that particular username’s home directory.

UNIX Directory Examples

/banana/consumers/monkeys would refer to the monkeys subdirectory of the consumers subdirectory of the bananas subdirectory of the root directory.

/~maspar/lab5/parallel.f would refer to the parallel.f file in the lab5 subdirectory of the maspar subdirectory of your home directory.

/~brubble/advice/friends/FRED/bowling would refer to the bowling file or subdirectory of the FRED subdirectory of the friends subdirectory of the advice subdirectory of brubble’s home directory.
1.5 Handling Files and Directories

**mkdir**  make directory

**cd**  change directory

**pwd**  print working directory

**rmdir**  remove an empty directory

**ls**  list files

**chmod**  change or modify protections on a file or directory

**mv**  move a file or subdirectory from one directory to another,  
or change the name of a file or directory

**cp**  copy a file

**rm**  remove a file

**lpr**  print a file to a particular printer

1.5.1 **mkdir**

Creates a directory under the current working directory.

*Example*

`mkdir lab1` makes a subdirectory called `lab1` under the current working directory.

1.5.2 **cd**

Changes directories. If followed by a directory name, the window changes to that working directory. If not followed by anything, `cd` switches back to your `home` directory. `cd ..` allows you to move up to the parent directory of the current working directory.

*Examples*

`cd ~jics/MP2_June95/lab3` would change directory to the `lab3` subdirectory of the MP2_June95 subdirectory of the `jics` home directory.

`cd ..` would then take you from there to the MP2_June95 subdirectory of the `jics` home directory.

`cd ..` would then take you to the `jics` home directory.

`cd` will bring you back to your `home` directory from whichever directory you were working in.

1.5.3 **pwd**

Prints the current working directory on screen.

1.5.4 **rmdir**

Removes a directory, but only if it is empty.
**Example**

`rmdir temp` removes the subdirectory `temp` from the current working directory.

### 1.5.5 `ls`

Lists the files and directories under the current working directory.

`ls -F` lists files and directories under the current working directory and gives more information about the files and directories. Any listed names which are followed by a `/` are directories. Any listed names followed by a `*` are executable files. Any listed names followed by an `@` are symbolically linked files or directories (located elsewhere).

**Example**

Suppose you are in your home directory and you type `ls -F`, and this is what appears:

```
a.out*  jics/  matmult.m  notHERE@  work.lsp*
data*   mult.f   notes/   README
```

Two subdirectories are `jics` and `notes`. Three executable files are `a.out`, `data`, and `work.lsp`. `notHERE` is a file or directory symbolically linked to your current directory, but located elsewhere.

Notice also that names of files should indicate what the file contains. Program files should be followed with an extension indicating which language was used. For example, use `.m` for MPL on the MP-2, `.f` for FORTRAN, and `.c` for C, `.fcm` for CMFortran on the CM-5, and `.tex` for LATEX text files, etc.

`ls -la` produces a long listing (which includes the permissions) of all the files in the current working directory. Even the files beginning with a “.” (or dot-files) are included in that listing.

Protections are read in this manner. If this is a directory, there is a `d` in the first spot. The next three spots are the permissions for you, the user, these are `r`, `w`, and `x`. The next three are for your group. (Your group may be “graduate”, “faculty”, “guest”, or some other group defined by your local system administrator.) These permissions are also `r`, `w`, `x` similar to the first three. The last are also `r`, `w`, `x`, but this time, for all other users. If the letter appears in its position the permission is allowed, if a `-` occurs instead, the permission is not allowed. Use the `chmod` command to change permissions.

### 1.5.6 `chmod`

Can be used to change protections in this manner: user, group, other + (to add the permission) or - (to take it away) read, write, execute.

**Examples**

`chmod ug+r READTHIS` gives the user and your group the permission to read the file in the current working directory named `READTHIS`. 

Example
chmod o-rwx MINE.m takes away the permission of all others to read, write, and execute the file MINE.m in the current working directory.

1.5.7 mv

Move a file or subdirectory from one directory to another, or change the name of the file or directory.

Examples

mv file1.f junk moves file1.f from your current working directory to a junk directory if the junk directory exists. Otherwise, file1.f is renamed junk under the current directory.
mv file1.f matmult.f renames the file file1.f as matmult.f in your current working directory.
mv jicslabs oldlabs moves your jicslabs directory to be a subdirectory of the directory oldlabs if that directory exists. Otherwise, the jicslabs directory is renamed oldlabs.

1.5.8 cp

Copy one file into another location.

Examples

cp jics/MP2_June94/lab1/halftone.f ~/lab1/halftone.f copies halftone.f from the lab1 subdirectory of the MP2_June94 subdirectory of the jics home directory to a file called halftone.f in the lab1 subdirectory of your own home directory.
cp jics/MP2_June94/lab1/* (Please notice the dot here!) copies everything in the lab1 subdirectory of the MP2_June94 subdirectory of the jics home directory into your current working directory. In this command, you should notice the “.” This is VERY important!
cp trial1.f trial2.f copies trial1.f in your current working directory to trial2.f in your current working directory.
cp trial1.f lab4 copies the file trial1.f into the subdirectory called lab4.
cp -r jics/CM5 /CM5+ copies the entire CM5 subdirectory of the jics home directory recursively into the CM5 subdirectory of your home directory.

1.5.9 rm

Removes or deletes a file. WARNING: once removed, a file is GONE FOREVER.

Examples

rm work.f removes the file work.f from your current working directory.
rm *.f  removes all files ending in .f
rm -i a*  removes all files beginning with “a”, but first inquires whether you wish
to remove a file before actually removing that file. (An answer of “y” or “n” is
expected).
rm -ir mydirectory  removes every file and subdirectory under and including my-
directory recursively. Before each file or directory is removed, the user is asked
whether that file or directory should be removed.

1.5.10 lpr

Prints a file to a your default printer.
lpr -Pprintername  Allows one to print a file to the specified printer printername.
Note: the lpr command used without the -Pprinter option will print a file to your
default printer. To learn which printer this would be, type echo $PRINTER or echo
$LPDEST (which are the most common names for environment variables which contain
the name of the default printer).

Examples
lp -d jics helpme  prints the helpme file in the current working directory to the
  printer named jics.
lpr -Pjics helpme  prints the helpme file in the current working directory to the
  printer named jics.

1.6 Viewing and Modifying Files

more, less, cat  allows a file to be read without making changes
diff  line by line comparison of two files
vi  opens up a file and places you in command mode of the vi
  editor
view  opens a file, places you in command mode of the vi editor,
  but will not allow you to save changes to file in the ordinary
  manner

1.6.1 more, less, cat

Allows you to read a file without making changes.

Examples
more README  shows the README file one screen at a time. Press return
to continue line by line. Press the space bar to continue screen by screen, or press
q to quit. d will scroll down half a screen length. The prompt will return when
the whole file has been shown.
less workfile.c  shows workfile.c in the same manner as the more command described above, but you can also back up half a screen with u or back up a whole screen with b. Other “vi” commands are available to move around within the text, search for a string, etc.
cat info.tex  shows the file info to the screen all at once scrolling off the top if there is not enough room in the window to hold the entire file. The prompt will return after the end of the file.

1.6.2  diff

Allows you to compare two files line by line. This could be useful in comparing data files.

Example
diff data1 data2  will compare files data1 and data2 of your current working directory line by line.

1.6.3  vi

The command vi filename will open the filename file and place you in the command mode of the vi editor. vi is ONE widely utilized text editor available on UNIX systems. There are others, such as emacs, which will not be discussed within this document.

Examples
  vi parallel.m  opens a file called parallel.m. If one does not already exist, a new file named parallel.m is created and opened. You are now in command mode of the vi editor.
  vi -r matrix1.f  will allow you to recover most or all of the work you had finished on the matrix1.f file in the current working subdirectory if you were working on this matrix1.f file when the system crashed.

More details on vi are given in Section 3 of this guide.

1.6.4  view

Opens a file and places you in command mode similar to that of vi, but will not allow you to overwrite a file, or save changes you may make to the file as you normally would in vi while view-ing the file.

Example
  view readonly.c  opens a file called readonly.c. You are now in command mode of the vi editor. However, you can only READ the file; you cannot make changes and save them in the same manner as in vi.
1.7 Obtaining Information

- **man**: Gives information on subject, command, or topic.
  - **man -k**: Gives information on where one would find man pages about a topic.

  *Examples*
  - `man cd`: Will list information on the `cd` command.
  - `man -t ftp`: Will print the man pages information on the file transfer protocol to your default printer.
  - `man -k print`: Will list information about which man pages include the word “print”.

- **who, w**: Lists who is logged on the computer.
  - **whoami**: Prints your own username to the screen.

- **finger**: Points out information about someone.
  - **date**: Gives day, date, time, and year.
  - **mail**: Puts you into mail mode (q to quit mail).

1.7.1 **man**

Gives information on a subject or a command.

**man -k** gives information on where one would find man pages about a topic.

*Examples*
- `man cd`: Will list information on the `cd` command.
- `man -t ftp`: Will print the man pages information on the file transfer protocol to your default printer.
- `man -k print`: Will list information about which man pages include the word “print”.

1.7.2 **whoami**

Writes your username to the screen (in case you forget which username you logged on with or you find an active computer terminal and wish to know who is using it.)

1.7.3 **who, w**

Lists who is logged on the computer on which you type `who` or `w`. However, `w` gives more information, such as load average, which shows you who is “hogging” most of the C.P.U. time on the computer.

1.7.4 **finger**

Allows you to find out information about another computer user. Different systems may give different amounts of information through the **finger** command.

*Examples*
- `finger jeff`: Gives you information about anyone named Jeff on your system.
- `finger -m jsmith`: Gives you information about the user with the username jsmith.
- `finger jsmith@cs.utk.edu`: Gives you information about the user with the E-mail address jsmith@cs.utk.edu.
finger @pulsar gives information about users logged onto the pulsar computer on the system. This information may contain the username of each user logged into any window, the idle time for each window, and from which machine the user logged onto pulsar.

finger jsmith@pulsar.cs.utk.edu (where pulsar is a machine name) allows one to acquire slightly more information about jsmith on a specific machine, pulsar. Sometimes one may only be able to finger someone in a different department by using the machine name in the command.

1.7.5 date
Returns the day, date, time (in hours, minutes, and seconds), and year.

1.7.6 mail
Puts you in mail mode. You can press return to view mail messages. s filename will save, or append the mail message you just read to the file filename. d will delete the current message and show the next message. h will show you the current list of messages. Entering the message number followed by pressing return will allow you to read that particular message. q will quit, or take you out of mail. For more information on mail, you can type ? while in mail mode. Typing man mail from the original prompt, which is not inside the mail system, is another method for obtaining more information on mail.

There are other mailing software packages available such as mh and xmh. For more information on those, view their man pages.

1.8 Helpful Little Commands

    clear  clear a window
    history prints a history of the commands typed in a window
          ! repeats previous specified commands

1.8.1 clear
Clears the screen and brings the prompt to the top of the window.

1.8.2 history
Lists the most recent set of commands you have entered in that window. This command is useful for repeating previous commands and saves typing when used with the ! command described next.
Repeats the last command beginning with a certain letter or history number.

Examples

!ma  repeats the last command beginning with ma, (such as man -k fortran).
!42  repeats the command numbered 42 in the history list.
!!  repeats the last command you typed in that window.

1.9 Accessing and Using Remote Computers

xhost +  adds other computers to the access control list

setenv DISPLAY  allows other computers to open display on specified computer

ftp  file transfer protocol
rlogin, rsh  login remotely to another UNIX computer
telnet  login remotely to another UNIX computer

1.9.1 xhost +

Allows applications running on a remote machine and creating additional windows to appear on your workstation.

Examples

xhost + cranberry.cs.utk.edu  will allow applications running on cranberry.cs.utk.edu to appear on your workstation even if your workstation IS OUTSIDE of the cs.utk.edu (University of Tennessee Computer Science department) domain.

xhost + cranberry  will allow applications running on cranberry to appear on your workstation if your workstation IS INSIDE the same domain.

xhost +  will allow applications running on any computer to appear on your workstation.

1.9.2 setenv DISPLAY machine:0.0

When running X windows, this allows you to be working on a remote machine and view the window applications at your workstation. This is used together with xhost +.

Note: xhost + must be typed in a local window. The setenv DISPLAY machine:0.0 command must be typed in the window of the remote computer (after logging onto it).

Examples

From a window on a computer named pulsar, type

setenv DISPLAY cetus4e:0.0
if you are at workstation cetus4e and working on pulsar (both of which are in the same domain of the local network, e.g. cs.utk.edu at the University of Tennessee Computer Science Department) and want to be able to view your applications.

Note that if you are attempting to display from a remote computer which is not in the same local area of the network as your workstation, you will have to specify the full name of the remote workstation. For example:

From a pulsar window, type

```
setenv DISPLAY frog.phy.ornl.gov:0.0
```

if you are at workstation frog.phy.ornl.gov and want to be able to view your applications. Please remember to use this command in conjunction with the `xhost` + command listed above.

### 1.9.3 ftp

Places you in the **file transfer protocol**, which allows you to transfer files to and from remote computers. You get into ftp by typing

```
ftp computer_name
```

OR

```
ftp computer’s I.P. address
```

You will then be asked for your username and the password. Once you are in `ftp`, you can use some of the usual UNIX commands, such as `pwd`, `ls`, and `cd`. You may wish to obtain a file from a remote computer. To do this, type `get filename` OR simply type `get`. If you did not specify the filename, you will be asked for the name of the file on the remote computer and for what you wish to name it on the computer on which you are currently working. You can also type `put` to put a file from where you are working onto the remote computer. Again, you will be asked for the name of the file here and what you want to name it there. `quit` at the prompt will `quit` the `ftp` session.

For more information on `ftp`, you can type `man ftp`.

**Examples**

```
ftp pulsar.cs.utk.edu  OR
ftp 128.169.92.22  will open a ftp session on pulsar, where 128.169.92.22 is the IP
number address for pulsar.cs.utk.edu.
```

### 1.9.4 rlogin, telnet, or rsh

These commands allow access to a remote machine. `rlogin` and `telnet` allow you to login remotely to another UNIX machine. The `rsh` command allows you to start a shell or execute a command on the remote system.

The difference between `rlogin` and `telnet` is that if you have an appropriately configured `.rhosts` file on the remote system, you do not have to specify a username and password to login using `rlogin`. With the `telnet` command you always have to specify both a username and a password to login. You need to have an `.rhosts` file on the remote system to use
the `rsh` command on the remote system to execute a command. For more information on configuring an `.rhosts` file type `man rhosts`.

**Examples**

```
rlogin pulsar.cs.utk.edu  allows you to login to pulsar in the University of Tennessee Computer Science Department.
rlogin pulsar  allows you to login to the machine named pulsar within your domain.
telnet pulsar  also allows you to login to pulsar if you are connecting from another computer in the University of Tennessee Computer Science Department.
rsh pulsar who  allows you to execute the command `who` on pulsar. After the command is executed, you DO NOT remain logged onto pulsar.
```

### 1.10 Checking Job and Processes Before Exiting

```
ps  report your processes and their i.d. numbers
jobs (4.3 BSD only) lists current jobs and their i.d. numbers
exit “kill” a window or close a connection with a machine
```

#### 1.10.1 ps

```
ps -f(SVR4), or ps -ux(4.3 BSD) list processes with their i.d. numbers (under PID), % of CPU and memory used, status (Running, Stopped briefly, Idly Waiting), time started, and amount of time spent on the process, and the command issued. The syntax you should use depends on which UNIX system you are using. (You can type `man ps` for more information).
```

```
ps -aux lists all processes on the computer. Note: you can only kill your own processes.
```

To kill an idle process, type `kill -9 PID#`

#### 1.10.2 jobs

Lists jobs and their i.d. numbers. If the prompt is returned without any additional information, you have no jobs running.

A job can be created by placing an `&` after a command, which makes the command executable in a background mode. For example: `xwebster&` allows one to keep the window where the command has been typed active, even while the `xwebster` process is also still active.

```
<CONTROL> Z stops a running process, but does not kill it. bg will restart a stopped job and put it in the background, giving control to the window you were in. fg will restart a stopped job and put it in the foreground, giving control to the process. <CONTROL> C will then kill a job if necessary.
```

Another way to kill a job is to first find the number of the job you wish to kill by typing `jobs`. Then, to kill, say `job number 5`, which was listed, type `kill %5`.
Example
Suppose you type `jobs` and this appears:

1. Stopped xwebster +
2. Running xcalc
3. Running mosaic

To kill `xwebster`, job #1 listed above, type `kill %1`.

1.10.3 exit

The command `exit` “kills” windows or closes connections with remote machines or exits a shell. Always `exit` all remote machines (after all processes and jobs on those machines have been “killed”) before you log out of your workstation.

2 Compiling in Unix

2.0.4 Notes on Compiling

into machine language, or object code. Compare the computer to a foreigner. The computer speaks and understands its own language, but the computer does not understand your language. Most people do not speak or understand the computer’s language. Therefore, we program in a higher level language which we can understand, and then use a translator, a compiler, to translate what we have written into a form the computer can understand. This version of your code is called object code. The object code will be in `filename.o`, where `filename` is the name of the file you compiled (minus the appropriate extension).

After the object code has been created, the compiler then links the code with the necessary library routines. Often library routines are required in a program. Math functions are often in a library which need to be linked with your program before you can run it.

The result is an executable file, which can be run on a computer.

If there are any syntax errors in your code, the compiler will detect them as the compiler attempts to generate object code. When errors are found, they are listed on the screen. Some compilers document errors better than others. Some compilers state specifically what is wrong, some state where the problem is, while others just state that there are problems.

To compile, type `compiler_name compiler_option(s) filename(s)`.

Remember the `filename` should include the appropriate extension. For example: Fortran files should end in `.f`, MPL files on the MasPar should end in `.m`, C files should end in `.c`, Pascal files should end in `.p`, and LISP files should end in `.lsp`. On the CM5, CMFortran files should end in `.fcm` and C* files should end in `.cs` as well.

2.0.5 Unix Compiler Options

- `-o newfile` Name the executable file `newfile` (rather than `a.out` if this option is NOT used).
-c  Compile only (suppressing linking), and produce a .o file for each source file being compiled.
-O, -O1, -O2, etc.  Optimize code at different levels - Note: At most levels of optimization, debugging is not allowed.
-g  compile and produce information useful to a debugger, (such as dbx, dbxtool, or xdbx), or a performance profiling package.
Note:  In some environments, the -g flag or the -O flag may be automatically implied.

For more information on compiling options type, man compiler_name, where commonly available compilers are cc, gcc or xlc for C programs and f77 or xlf for Fortran programs.

Examples
cc -o parallel -O2 parallel.c  Calls the C compiler, cc, to compile parallel.c and writes the compiled, executable version of this file to parallel. The code is optimized to level 2.
f77 -o newfile -g parallel.f  Calls the Fortran compiler, f77, to compile parallel.f, writing the executable version of this file to newfile. Information for debugging is produced as well.

    cc -c subr1.c
    cc -c subr2.f
    f77 -o test mainprog.f subr1.o subr2.o

First, compile subr1.c with the C compiler, cc, without linking. This will produce an object file, subr1.o. Next, compile subr2.f with the Fortran compiler, f77, without linking. This will produce an object file, subr2.o. Finally, compile mainprog.f, with the Fortran compiler, f77, linking the libraries as well as subr1.o and subr2.o. The executable code produced will be written into test. This is useful when a program calls subroutines in both C and Fortran. If changes are only made to one subroutine, compiling subroutines separately and then compiling the main program with the object files can save time.

3  The vi Editor

vi is one of the most commonly utilized visual text editors available on UNIX systems. Please remember when using commands that vi is case-sensitive, so whether a letter is upper-case or lower-case DOES matter.

3.1  Two Modes of the vi Editor

Before you begin working with the vi editor, you should know that there are two modes in vi - the command mode and the insert mode.
**Command** mode is used to move through text, search for words, save a file, etc. This mode covers everything except inserting code. **Insert** mode is only for inserting text into a file.

Using i, I, a, A, cw, o, O, or R will place you into insert mode from command mode. These commands will be described further in Section 3.2.2 of this guide.

**IMPORTANT!!** Use the escape key to return to command mode from insert mode. If you are not sure which mode you are in, hit the escape key to be sure to be in command mode again.

If the keys you are pressing are not achieving the results they should, you may be in the wrong mode, or you may have accidentally pressed the Caps Lock key. The little green light on the Caps Lock key should **not** be lit (if such a light exists on your keyboard).

Sometimes you may hit escape to switch to command mode, and find that the words **INSERT MODE** still appear at the bottom left corner of the window (as it does when in INSERT MODE with some setups). If this happens, press escape again to be sure you are actually in command mode. If the words still appear in the lower left corner of the window, you may want to “refresh” the screen by pressing the control and 1 keys at the same time. (That is the lower-case letter “l” in case you were not certain.) Now, if you are not in insert mode, the words stating otherwise will disappear from your screen.

**IMPORTANT!!** Do NOT use the arrow keys to move around your text while in insert mode!! Strange things will happen!! This can be frustrating! In time, you will learn to escape from insert mode **FIRST**, before moving around in your file to make any necessary corrections.

## 3.2 Useful vi Commands

### 3.2.1 Moving Around Within Text in Command Mode

| ←, ↓, ↑, and → | Move through text: left, down, up, and right |
| <Control> f | Move forward one screen |
| <Control> b | Move backward one screen |
| <Control> d | Move half a screen down |
| <Control> u | Move half a screen up |
| G | Move to end of file |
| 1G | Move to beginning of file |
| 65G | Move to line 65 in the file |
| $ | Move to end of line |
| 0 (zero) | Move to beginning of line |
| /myword | Forward to any existing occurrence of myword |
| ?myword | Backwards to any existing occurrence of myword |
| n | Next occurrence of myword in the search direction |
| N | Next occurrence of myword in opposite direction as the search |
3.2.2 Inserting Text

All of these commands will put you in insert or append mode. Enter the text and then press escape to get back into command mode.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>insert text before cursor</td>
</tr>
<tr>
<td>a</td>
<td>insert text after cursor (append it)</td>
</tr>
<tr>
<td>I</td>
<td>insert text at beginning of current line</td>
</tr>
<tr>
<td>A</td>
<td>insert text after end of current line</td>
</tr>
<tr>
<td>o</td>
<td>open a line below current line and insert text</td>
</tr>
<tr>
<td>O</td>
<td>open a line above current line and insert text</td>
</tr>
<tr>
<td>ESCAPE</td>
<td>to switch back to command mode from insert mode</td>
</tr>
</tbody>
</table>

3.2.3 Deleting Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>delete one character</td>
</tr>
<tr>
<td>7x</td>
<td>delete 7 characters</td>
</tr>
<tr>
<td>dw</td>
<td>delete one word</td>
</tr>
<tr>
<td>10dw</td>
<td>delete 10 words</td>
</tr>
<tr>
<td>dd</td>
<td>delete one line</td>
</tr>
<tr>
<td>25dd</td>
<td>delete 25 lines</td>
</tr>
<tr>
<td>D</td>
<td>delete from cursor to the end of that line</td>
</tr>
</tbody>
</table>

Note: all the commands above store the deleted text in a temporary buffer. The contents of the buffer can be retrieved back into the file using the P and p commands described in the “moving text” section below.

3.2.4 Moving Text

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy</td>
<td>yanks one line into a buffer without deleting that line</td>
</tr>
<tr>
<td>3yy</td>
<td>yanks three lines into a buffer without deleting those lines</td>
</tr>
<tr>
<td>p</td>
<td>puts line(s) from buffer into file after the current line</td>
</tr>
<tr>
<td>P</td>
<td>puts line(s) from buffer into file before the current line</td>
</tr>
<tr>
<td>J</td>
<td>joins the line below the current line with the current line</td>
</tr>
</tbody>
</table>

You would use the yy and p commands when wanting to copy text from one part of the file to another. You would use the dd and p commands when you want to actually move text from one part of the file to another.
3.2.5 Changing Text

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>change the case of a letter and advance the cursor</td>
</tr>
<tr>
<td>5~</td>
<td>changes the case of 5 letters and advances the cursor five spaces</td>
</tr>
<tr>
<td>r</td>
<td>replaces a character</td>
</tr>
<tr>
<td>R</td>
<td>replaces everything until you press &lt;ESCAPE&gt;</td>
</tr>
<tr>
<td>cw</td>
<td>changes a word to everything typed between cw and &lt;ESCAPE&gt;</td>
</tr>
</tbody>
</table>

3.2.6 Saving and Quitting

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:w</td>
<td>writes the file (saves it) while remaining in the file</td>
</tr>
<tr>
<td>:w new_file_name</td>
<td>saves the current file under a new name</td>
</tr>
<tr>
<td>:q</td>
<td>quit when you have NOT changed a file</td>
</tr>
<tr>
<td>:q!</td>
<td>really quit – without saving ANY changes (use with care!)</td>
</tr>
</tbody>
</table>

You should use the :w command often to save your file when you are working on it. This will help protect you in the event of a system failure. This method will also allow you to save code in one window before compiling the file in another window. If there is an error, you can make changes and save with the :w command in the code window without having to quit vi repeatedly in order to compile. Then you can compile again in the other window.

Also, you will generally use :wq when you wish to end your vi session, saving all changes you made to the file.

3.3 Error Recovery

<table>
<thead>
<tr>
<th>Key Combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>undoes the last step you did</td>
</tr>
<tr>
<td>U</td>
<td>restores current line</td>
</tr>
<tr>
<td>&lt;CONTROL&gt; q</td>
<td>escape the “no-key-does-anything stuck-state.”</td>
</tr>
<tr>
<td>&lt;CONTROL&gt; l</td>
<td>refreshes the screen.</td>
</tr>
</tbody>
</table>

When in doubt, press <ESCAPE> to be certain you are in command mode. If strange things are happening, make sure the Caps Lock is not on.

Remember to use

view to view a file you do NOT wish to change.
:q to quit a file you opened and do NOT wish to change, or to close a file you did not mean to create (and into which you did not insert text).
:q! to quit a file you unintentionally created, and into which you inserted text. (Be careful with this command so that you do not lose changes you wished to keep!)

3.4 Final Note on vi

The best way to learn anything, including vi, is PRACTICE, PRACTICE, PRACTICE!! So you may try creating some text files to practice using these new commands and become
familiar with the two modes and how to switch from one to another. *vi* does get easier after a while, once you become familiar with it - believe it or not.

4 X-Windows

*Please note: This discussion applies only to the environment created for JICS users on the University of Tennessee Computer Science Department. This will dictate the manner in which the windows and other items are situated on your screen. This setup will also determine how the mouse buttons will react in different instances. We present this here as an example of how an X windowing environment would be used. For details about the configuration of the X windows at other sites, consult the local computer systems administrator.*

X is an environment which allows great flexibility through the use of windows. For example, one could be remotely logged on one or more machines in different windows while working on the local computer in one or more windows. One could edit code in one window, compile in another, and execute the code in a third window.

Windows can also be used for software packages, demos, or graphs.

The mouse is the main user interface for creating windows, placing windows, and manipulating windows in all the ways listed below.

There are several possible window managers which may be available to the user. The windowing environment on which the following descriptions are based, is managed by the “tab window manager” (twm) through which windows are moved, resized, iconified, closed, etc.. The window manager may be used to access certain utilities and functions (defined in the .twmrc file) through the mouse buttons. Some of these window operations are described below.

4.0.1 Creating New Windows

One way to create new windows is to type `xterm &` in a window. (The & allows one to keep both windows active instead of freezing the window in which the command was typed until the `xterm` process is killed.)

Another way to create a new window is more specific to this system. First move the mouse to the dark gray, background area and press the right mouse button. A menu should appear with a selection of possible computers from which to open a window. While you keep the button down, move the mouse down until the type of window you wish to create is highlighted. Local places a window local to the computer you are using. There are also options available for opening a window logged in to *duncan*, *cranberry* (for CM-5 work), and *pulsar* (for MASPAR work).

When the “ghost” outline of the window appears on the screen, use the mouse to move it where you will want the window to be. Then click any mouse button to actually position the window there.
4.0.2 Moving Windows

Windows can be moved rather easily. Simply move the mouse so the cursor appears in the horizontal bar at the top of the window. Click the left mouse button and hold the button down while moving the mouse so that the window moves where you want it. Releasing the button places the window in its new position.

4.0.3 Resizing Windows

Windows can be resized quickly as well. First, you click the left mouse button on the box in the upper right corner of the window. Hold the button down and move to whichever edge of the window you wish to move, and push or pull the edge in the desired direction. Once again, releasing the button places the window edge in its new position.

4.0.4 Scrolling Up or Down within a Window

On the left side of the window, there may be a vertical bar known as the scrolling bar. This bar allows you to view text which has scrolled off the top of the window. Just move the mouse so the cursor is in the vertical bar and press the right button to scroll up into the window. The left button will scroll back down towards the text you could see at first. (The middle mouse button allows one to scroll up and down more smoothly.) Pressing return inside the window should bring you back to the prompt.

4.0.5 Iconifying Windows

Iconifying windows can give you more space when you do not need to view a certain window for a while (but do not wish to “kill” it). Move the mouse so the cursor is at the box in the upper left corner of the window. Then, simply click the left mouse button and watch the window become a small box, or icon, with the window name inside of it.

To deiconify the window, click the left mouse button on the icon.

4.0.6 Useful Toys

Moving the mouse so that the cursor is in the dark gray background area and pressing the middle mouse button will show some “toys”. These are a clipboard, news (which is the same as xrn), and a calculator. The clipboard allows a person to keep memos or reminders for him- or her-self. news will allow one to search through material posted by many news groups. The calculator is fairly self-explanatory. The clipboard and news have a quit option. If you open a calculator, you can “kill” it later with the kill option under the left mouse button, or by pressing <CONTROL> C, or by typing :q as one would within vi.
4.0.7 Killing Windows

Windows can either be killed by typing **exit** at the prompt in each window you wish to “kill”, or you can move the mouse so the cursor is in the background dark gray area, press and hold the left mouse button, pulling the mouse down until **kill** is highlighted, release, and move the skull and crossbones to the window (or calculator) you wish to “kill” and click any mouse button. The window should then disappear. If this does not happen, try again.

4.0.8 Reminder on Opening X-Windows from Remote Hosts

When working on a remote computer (which you connected with using **rlogin** or **telnet**, for example) and wishing to run code which would create a window on the screen of the workstation on which you are working, you **must** set the name of the display device on the remote system using the **setenv DISPLAY** command, and you also **must** permit access to your display device to the remote system using the **xhost** command.

For example, if you are remotely logged onto the UNIX computer **crypt.phy.ornl.gov** from **pulsar.cs.utk.edu**, and wish to create a window on pulsar, then in a window in which you are logged onto crypt, type

```
setenv DISPLAY pulsar.cs.utk.edu:0.0
```

to set the display on the remote system. Then, in a local window on pulsar, type

```
xhost + crypt.phy.ornl.gov
```

to permit access of the remote system to your screen.

Now, if you run the windowing application on the remote system, the display will be made to your screen.

4.0.9 Final Note on X-Windows

X-Windows also becomes much easier with practice. So, practice X-Windows and practice **vi** and soon the computer system will be much more comfortable to you.

5 Other Material on UNIX, **vi**, X-Windows

If you are on campus at the University of Tennessee at Knoxville, you can find more information on these three topics, as well as **mail**, or **MH**, available outside the Hydra computer lab. These documents can be found in the hanging organizer by the printer. If the door to the Hydra lab is open, look behind the door for these colorful, informative sheets!!

Quick Reference Cards or Sheets on UNIX, **vi**, and E-mail are also available through the University of Tennessee Computing Center at the Stokely Management Center on the second floor (here at UTK).

Even if you are not at UTK, more information on topics or commands can be obtained by typing **man command name** on the computer. This command has been described in greater detail earlier in this document.
Other http addresses on the World Wide Web may also prove helpful. Here are two that we have found:

http://www.eecs.nwu.edu/unix.html
http://mathsun1.math.utk.edu/1/Computing/Unix
http://www.mhpcc.edu/training/vitecbids/UnixIntro/UnixIntro.html

Finally, there are also several books around which can prove to be quite instrumental in learning UNIX, vi, and X-Windows. A few of these are listed in the following bibliography.
Good luck in your new adventures!

6 Acknowledgments

Thanks to Christian Halloy for critiquing drafts of this guide.

Special thanks to Wallace Mayo and the CS102 staff of the University of Tennessee’s Computer Science Department for information presented in CS102 labs regarding UNIX, vi, and X-Windows.

Thanks also to Gail Wauford, Loretta Haack, Sue Smith, Mitch Kovarik, from the University of Tennessee Computing Center for information presented in the form of slides and quick reference guides and other material regarding UNIX, vi, and X-Windows.

Thanks to Mitch Kovarik from CSEP for additional suggestions as well as for converting this document into a form which could be accessed online through the World-Wide Web.

Thanks also to all the people and companies who have made Unix and its associated tools into such a powerful system.¹

References


¹ DEC and ULTRIX are registered trademarks of Digital Equipment. UNIX is a registered trademark of UNIX System Laboratories, Inc. The X Window System is a trademark of the Massachusetts Institute of Technology.