# Table of Contents

## Basic UNIX Concepts
- Major UNIX Concepts and Feature .................................................. 4
  - KERNEL ......................................................................................... 4
  - SHELL ......................................................................................... 4
  - UTILITIES .................................................................................. 4

## Logging to UNIX System
- Logging In and Out .......................................................................... 5
  - Local terminal ............................................................................... 5
  - Logging in using the telnet utility ................................................ 5
- Initialization Files and Terminal setup .............................................. 6
  - .cshrc ......................................................................................... 6
  - .login ......................................................................................... 6
  - .logout ......................................................................................... 7

## Getting Help With UNIX
- The man Pages ................................................................................ 8

## File Management
- File Management Commands .......................................................... 10
  - Examples: ...................................................................................... 11
- Directory Management Commands .................................................. 12
  - Examples: ...................................................................................... 12

## File Security
- Security Issues ................................................................................. 15
- User Classes and Permissions ......................................................... 15
  - Example ....................................................................................... 16
- Setting File Permissions ................................................................... 16

## The Visual Display (vi) Editor
- Basic vi Editor Concepts .................................................................. 18
  - Invoking vi ................................................................................... 18
  - STATUS LINE ............................................................................... 18
  - WORDWRAP ............................................................................... 19
- Two Basic Editor Modes .................................................................. 19
  - INPUT MODE .............................................................................. 19
  - COMMAND MODE ....................................................................... 19
- Other Basic Commands .................................................................... 20
  - Moving Around ........................................................................... 20
  - Scrolling ...................................................................................... 21
Basic UNIX Concepts

Major UNIX Concepts and Feature

UNIX is a multi-user computer operating system used for electronic mail, networking, numerous programs, text processing, computation and scientific utilities. UNIX was originally developed at AT&T Bell Laboratories.

The UNIX operating system can be installed on almost all PCs, workstations, mini-computers, and mainframes with little or no modification.

Many people can use the same UNIX system simultaneously where UNIX supports background and foreground jobs, multiple windows/shells per user, and multiple login sessions.

UNIX includes programs that support TCP/IP networking protocols.

UNIX Files and Directories are set up in an inverted tree structure; users have the freedom to arrange directories and files as they wish.

Input and Output (I/O) from commands can be redirected to become input or output for another command. This allows the use of complex filtering and multiple command statements that use the results of previous commands.

In general UNIX consists of a kernel, a shell, and a large collection of utilities:

**KERNEL**

Manages hardware (processes, memory, disk, etc.)
The kernel is the core of the operating system. It is a virtual machine that hides the characteristics of the underlying hardware, providing a collection of services that is independent of any particular computer system. These services fall into four broad functional groups: processes, memory management, I/O, and timers.

**SHELL**

Interface between the kernel and the user.
The user communicates with UNIX through shells. A shell is a program, which provides the interface between the user and the operating system. Shells perform the following basic functions:

- Accept Commands
- Check SYNTAX of Statements
- Call UNIX Utilities
- Report on Program Results and Errors

**UTILITIES**

more then 220 built in programs and services (compilers, editors, etc.)
Logging to UNIX System

Logging In and Out

In order to login to a UNIX system, you need to know your Access ID and password. An access ID is an identifier that keeps your files separate from other users' files, a password is used to prevent others from using your access ID. The system administrator for the system you are working on assigns access IDs and passwords.

For more information on how to use your Access Id and password please visit: http://www.eng.wayne.edu/computing/connectservices/help_desk/faq/faq.htm

Suppose your access ID is: ai1111 and your password is: pass1. You type: ai1111 after the login prompt and: pass1 after the password prompt (Your password will not appear on the screen when typed) The system will tell you whether you have typed in the correct ID and password. Following are examples of how to login to UNIX from different systems. In all cases, the procedure to logout of a UNIX system is to simply type the command: exit.

Most UNIX systems are connected to a network of some type. In this section we will deal only with logging in to local systems - that is, systems that you work on directly - not-remote systems on a network. There are more then one way to login to UNIX.

Local terminal

1. Log onto the workstation by entering your access id and password
2. Using your mouse, right click on the desktop
3. Left click on Tools
4. Click on Terminal. The UNIX terminal will open.
5. You can type the program command you want to run.

Logging in using the telnet utility

Telnet is a program that will allow you to log into UNIX system from almost anywhere you have internet access. Most computers are equipped with Telnet utility.

Telnet is a utility, which opens a terminal connection from one machine to another via TCP/IP networking protocols. If the remote machine supports the telnet-TCP/IP protocols, then you can use telnet to connect to it. The machine does not need to be running UNIX.
To log onto Engineering UNIX system from a computer connected to the Internet (Using Windows OS), simply do the following:

1. From Window’s **Start menu**, select **Run**
2. Type in the flowing: **telnet Server name**  
   e.g. gigastar.eng.wayne.edu
3. Press **OK**. Telnet will open in a new window and connect to Wayne State Engineering UNIX system.
4. Enter in your user name and password as prompted.
5. **TERM = (vt100)** will appear. YOU MUST PRESS ENTER TO CONTINUE.

It should appear as follows.

```
SunOS 5.8
AccessID: ak2944
AccessID Password:
Last login: Fri Mar  1 11:34:46 from pm592-08.dialip.
Sun Microsystems Inc. SunOS 5.8 Generic February 2000
Fri Mar  1 11:47:39 EST 2002
/export/home/other/private/student/ak2944
No match
TERM = (vt100) ← Press enter at this point
```

**Initialization Files and Terminal setup**

Dot (. ) files are files users create in modifying their environment to reflect the needs and requirements of their specific application. These files have special names: cshrc, login, exrc, mailrc, sunview, and defaults. The following files are discussed in this section: .cshrc, .login, and .logout.

**.cshrc**

The filename .cshrc is an abbreviation of: C shell run command. csh is the C-shell command interpreter, and rc implies that the .cshrc file is read when a C-shell is started. The primary purpose of the .cshrc file is to setup or initialize variables used by the Cshell. Examples of these are: search paths, user prompts, and aliases (an abbreviated name for a command set).

**.login**

The .login file is only read in once. It is used to set up terminal characteristics and environment variables. Environment variables are different from shell variables in that they are automatically passed on to subsequent shells and programs. Examples of global
environment variables are: special pathnames, program variables, and default variables for some UNIX utilities.

For Example: To use Hypermesh remotely, a path needs to be established to the Display Environment Settings. This is done with a Display environment variable. Once set in the. Login file, this variable never needs to be defined again.

Server_name %setenv DISPLAY Computer name:0

N.B. The Computer Name is the IP address of the Computer.

To find your IP address you can type who at the command prompt (your IP_address should be located next to your access_id).

zimbabwe %who

aj1171 pts/45 April 01 15:45 141.217.24.39
aj3353 pts/4 April 01 12:45 gigastar

.logout

The, logout file is read when a user logs off the system. This file is recommended for the setup of customized logout messages and housekeeping functions.

Requirements of an initialization file:

- Filenames start with a period, for example .cshrc
- To list initialization files, enter ls -a (they are invisible to the standard ls command)
- Initialization files must be located in the users home directory.

To summarize, use:

- cshrc to set up C-shell variable and aliases
- login (read only once) to set up terminal characteristics and environment variables
- logout (read only once) to set up system messages and housekeeping functions

N.B. There are also other initialization files like. profile If you are using Korn or Born Shell.
Getting Help With UNIX

The man Pages

The on-line help facility on UNIX systems is called the Manual Pages or man pages for short. The man pages consist of a complete copy of all documentation pertaining to the current version of the operating system.

Many questions that arise about using UNIX can be answered by using the man pages. To access the man pages for information on UNIX commands, simply type: man command at the system prompt, where command is the name of the command you are interested in. Typing man passwd, for instance, would list the man pages available pertaining to the password command at the Engineering Computer Center at Wayne State University.

The man pages are also used for general information about the systems and software available. To obtain a hardcopy of the content of a man page, you can use the following command statement:

`gigastar %man command | lpr -P lp`

Which will print the man page to your local printer. The vertical bar in this statement (|) is the pipe command, lpr is the UNIX print utility, -P is the option to select the printer, and lp is the name of the printer. (These commands will be discussed later in the course.) You can also obtain a concise summary of help information available for a particular command with the -k option of the man command.

For example:

`gigastar %man -k copy`

This command will list all commands associated with the copy command and give a brief description of each command listed.

A man page will look something like this when it is printed on your screen:

`gigastar %man mkdir`

User Commands

NAME
mkdir - make directories
SYNOPSIS

 mkdir(1)
mkdir [ -m mode ] [ -p ] dir...

DESCRIPTION
The mkdir command creates the named directories in mode 777 (possibly altered by the file mode creation mask umask(1)).
Standard entries in a directory (for instance, the files ".", for the directory itself, and "..", for its parent) are made automatically. mkdir cannot create these entries by name. Creation of a directory requires write permission in the parent directory.
File Management

Proper file management is necessary to maintain good data organization. UNIX file structures are hierarchical in nature. Each user is normally assigned a home directory with the same name as the user ID. It is helpful to think of the analogy of a file cabinet...the whole cabinet can represent the home directory, the drawers represent subdirectories within the home directory, and files within the drawers are analogous to files within each sub-directory.

For example:

The home Directory of user aj1171 will be
/export/home/ece/private/student/aj1171

File Management Commands

There are numerous file management commands that are useful in the UNIX environment. The basic commands will be explained in this section.

Summary of File Management Commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>touch</td>
<td>Creates a file</td>
</tr>
<tr>
<td>mv</td>
<td>Renames a file</td>
</tr>
<tr>
<td>cp</td>
<td>Copies a file</td>
</tr>
<tr>
<td>rm</td>
<td>Removes a file</td>
</tr>
<tr>
<td>more</td>
<td>Lists a file and pause after each page</td>
</tr>
<tr>
<td>cat</td>
<td>Lists a complete file</td>
</tr>
<tr>
<td>head</td>
<td>Lists a number of lines starting at the top of the file</td>
</tr>
<tr>
<td>tail</td>
<td>Lists a number of lines starting at the bottom of the file</td>
</tr>
</tbody>
</table>
Examples:

**touch**

`gigastar %touch apple`
Will creates a file called apple in the current directory.

**mv**

`gigastar %mv apple orange`
Will rename the file apple to orange.

**cp**

`gigastar %cp orange grape`
Will copy the file orange to another file named grape (if the file grape exists, then it will be overwritten).

**rm**

`gigastar %rm orange`
Will delete the file called orange.

**cat**

`gigastar %cat paper`
Will list the contents of file paper.

**more**

`gigastar %more paper`
Will list the contents of the file paper, pausing at the end of each page.

**head**

`gigastar %head -20 paper`
Will list the first 20 lines of the file called paper.

`gigastar %head -2 lesson1`
Will list the first two lines of the file called lesson1.

**tail**

`gigastar %tail -15 paper`
Will list the last 15 lines of the file called paper.

`gigastar %tail -4 ice`
Will list the last 4 lines of the file called ice.
Directory Management Commands

Summary of Directory Management Commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mkdir</td>
<td>Creates a directory</td>
</tr>
<tr>
<td>pwd</td>
<td>Prints the working directory</td>
</tr>
<tr>
<td>rmdir</td>
<td>Removes a directory</td>
</tr>
<tr>
<td>rm -r</td>
<td>Removes a directory, recursively</td>
</tr>
<tr>
<td>cd</td>
<td>Changes to a directory</td>
</tr>
<tr>
<td>cp -r</td>
<td>Copies a directory, recursively</td>
</tr>
<tr>
<td>mv</td>
<td>Moves a directory</td>
</tr>
<tr>
<td>ls</td>
<td>Lists the contents of a directory</td>
</tr>
</tbody>
</table>

Examples:

```
mkdir
gigastar %mkdir temp
Type "mkdir temp" at the command prompt to create a new directory called temp.
```

```
pwd
gigastar %pwd
Will print the current working directory.
```

```
rm -r
```
```
gigastar %rm -r notes
Will delete the directory called notes and ALL its contents (use with extreme caution).
```

```
cd
gigastar %cd fruit
will move you into the directory called fruit (the full pathname can also be used with the cd command).
```
```
gigastar %cd /export/home/ece/private/student/aj1171
will move you into the directory called /export/home/ece/private/student/aj1171
```
```
gigastar %cd
will move you into your home directory
```

```
cp
gigastar %cp month storage
Will copy the file called month to the file called storage.
```
gigastar %cp -r folder1 folder2
Will copy folder1 and its contents to folder2.

mv

gigastar %mv myfiles yourfiles
Will rename the directory myfiles to the name yourfiles (the directory contents are unaffected).


gigastar %mv plants greens
Will rename the directory plants to a directory named greens.

ls

gigastar %ls
Will display the contents of the current working directory.


gigastar %ls -a
Will show all hidden files.


gigastar %ls -la
It will return your file names including hidden files, and a forward slash will be inserted in front of subdirectories

For example:

yellow %ls -la

```
total 2820812
  drwxr-x--x   7 aj3353   23           512 Apr 11 00:17 ./
  drwxr-xr-x  391 root     other       6656 Apr  6 01:17 ../
  -rw-r--r--   1 aj3353   37            94 Mar 29 14:03 .aliasrc
  -rw-r--r--   1 aj3353   37          2977 Mar 29 14:02 .cshrc
  -rw-r--r--   1 aj3353   37          2977 Mar 29 14:02 .cshrc.bak
  drwxr-xr-x   13 aj3353  37           512 Jul 27  2001 .dt/
  drwxr-xr-x   2 aj3353  37           512 Apr 11 00:17 2001TAX/
  drwxr-xr-x   2 aj3353  37           512 Apr  6 01:46 EmailBackup/
  drwx------   2 aj3353  37           512 Mar 29 14:02 nsmail/
```
Explanation:

The output are formatted into columns where each column from right to left:

**Type**
- Subdirectory (d), Linked directory or file (l), Regular data file (-)

**Permissions**
Permissions allow or deny access to read, write or execute files or directories.

**Owner**
Name of the user who created the file or directory

**Owner group**
Group ID, which the user belong to.

**Size**
Size of directories or files in bytes

**Date/time**
Date and time of last modification to the file or directory

**Name**
Name of the subdirectory or file

Other commonly used options for the ls command are:
- `-d` list directory without its contents
- `-R` list contents of all directories in the current directory
- `-F` identifies executable files with an asterisk (*) and directories with backslash (/)
- `-a` list hidden files (dotfiles).
File Security

Security Issues

Security on a UNIX system is a very important issue for both system administrator (also called superuser) and individual users.

Under the UNIX operating system, both types of users can control access to directories and files. A particular class of users can be assigned to a group that allows certain types of access (called permissions) and the person to whom a directory has been assigned is called the owner of that directory. The owner can change permissions on his or her directories and files. Only the system administrator, however, can change the ownership of an assigned directory.

User Classes and Permissions

Users are grouped into three user classes as follows:

USER- User permissions pertain to the owner of the file or directory. The owner can change the permissions but is bound by them while they are in effect.

GROUP- The system administrator can group users so that the group shares a common name. This group name can be a number or single word that can be used to allow special permission for a group of users.

OTHERS- All remaining users on the system are considered others and are bound by the permissions assigned to this class.

There are three types of permissions that can be assigned to a file or directory:

READ- list contents of directories using ls, list contents of files using more, cat, head, tail

WRITE- change contents of files using the vi editor create and remove files using rm, vi editor copy files to a directory using cp.

EXECUTE- execute programs in a directory, change directories using cd
A mode is a set of combined permissions assigned to a file or directory. For example, *rw* indicates that a user has read, write, and execute permissions. The permissions assigned to a particular user can be listed using the -l and -lg options of the ls command as follows:

**Example**

```
yellow % ls -la
```
```
total 2820812
  drwxr-x--x  7 aj3353   23           512 Apr 11 00:17 ./
  drwxr-xr-x 391 root other       6656 Apr  6 01:17 ../
  -rw-r--r--  1 aj3353   37            94 Mar 29 14:03 .aliasrc
  -rw-r--r--  1 aj3353   37           2977 Mar 29 14:02 .cshrc
  -rw-r--r--  1 aj3353   37           2977 Mar 29 14:02 .cshrc.bak
  drwxr-xr-x  13 aj3353   37           512 Jul 27  2001 .dt/
  -rw-r--r--  1 aj3353   37           1073 Mar 29 14:02 .login
  drwx------   5 aj3353   37           512 Oct  4  2001 .netscape/
  drwxr-xr-x  13 aj3353   37           512 Jul 27  2001 .dt/
  -rw-r--r--  1 aj3353   37           1073 Mar 29 14:02 .login
  drwx------   5 aj3353   37           512 Oct  4  2001 .netscape/
  drwxr-xr-x  13 aj3353   37           512 Jul 27  2001 .dt/
  -rw-r--r--  1 aj3353   37           1073 Mar 29 14:02 .login
  drwx------   5 aj3353   37           512 Oct  4  2001 .netscape/
  drwxr-xr-x  13 aj3353   37           512 Jul 27  2001 .dt/
  -rw-r--r--  1 aj3353   37           1073 Mar 29 14:02 .login
```

**Setting File Permissions**

- **Symbolic Notation**

The following syntax is used for symbolic notation:

<table>
<thead>
<tr>
<th>CLASS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>user (owner)</td>
</tr>
<tr>
<td>q</td>
<td>group</td>
</tr>
<tr>
<td>o</td>
<td>others</td>
</tr>
<tr>
<td>a</td>
<td>all</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>set permission</td>
</tr>
<tr>
<td>-</td>
<td>remove permission</td>
</tr>
<tr>
<td>+</td>
<td>add permission</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PERMISSION</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>read</td>
</tr>
<tr>
<td>w</td>
<td>write</td>
</tr>
</tbody>
</table>
The owner of each file and/or directory or the system administrator has the option to change the permissions using the chmod command. This command is in the form:

```
gigastar % chmod mode filename (or directory name)
```

- **Numeric Notation**
  The following notation is used for changing permissions by numeric notation:

<table>
<thead>
<tr>
<th>Permission Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>4</td>
</tr>
<tr>
<td>write</td>
<td>2</td>
</tr>
<tr>
<td>execute</td>
<td>1</td>
</tr>
<tr>
<td>none</td>
<td>0</td>
</tr>
</tbody>
</table>

Permissions are calculated by summing the octal values of the read, write, and execute privileges.

<table>
<thead>
<tr>
<th>Octal Notation</th>
<th>Permission modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>rwx --- ---</td>
</tr>
<tr>
<td>755</td>
<td>rwx r-x r-x</td>
</tr>
<tr>
<td>777</td>
<td>rwx rwx rwx</td>
</tr>
<tr>
<td>000</td>
<td>--- --- ---</td>
</tr>
</tbody>
</table>
The Visual Display (vi) Editor

The most widely used UNIX editor, vi is the one we will use in this course. In this section, you will be introduced to the basic commands needed to use vi for file editing and manipulation. We will cover the main commands used for inputting text; moving around on the screen (or page) and within a multi-page file; adding, changing, and deleting text; copying and moving text; and several other useful commands such as searching for text and undoing changes. The vi editor was originally developed at the University of California at Berkeley and was added to the version of UNIX that became System V. It evolved from the ex line editor because of the advantages of working with a full-screen editor. The term vi is derived from the concept of visual editing.

Basic vi Editor Concepts

While the vi editor is very versatile when used as a text editor, it was not designed to function as a word processor by today’s standards. Although it does contain many advanced commands and features, they are beyond the scope of this course and will not be covered. In this section, you will learn to use vi for relatively simple tasks such as editing text and program files that you will need to do your work in UNIX.

Invoking vi

The form of the command to invoke the vi editor is:

```
gigastar % vi filename
```

You can open a file for editing by using the actual filename or the full pathname as in the following example:

```
gigastar % vi notes
```

```
gigastar % vi /home/user/smith/notes
```

If the file notes already exists, it is opened and read into the working Buffer space. If the file does not exist, it is created and given the name notes.

STATUS LINE

The line at the bottom of the screen is reserved for status information. This information includes the file name and length (in lines and characters).
WORDWRAP

The vi editor includes a feature called wordwrap. This means that when the cursor reaches the right margin of the screen, it will automatically move to the next line at the left margin.

Two Basic Editor Modes

The vi editor has two basic modes of operation: Input Mode and Command Mode.

INPUT MODE

The input mode is used for inserting text within an existing file, appending text to the end of an existing file, or building a new file. There are several commands you can use to enter the input mode; these will be discussed later in this section.

COMMAND MODE

Command mode is used for such functions as moving around in a document, deleting text, searching for text, saving text and leaving the editor. You are in command mode when you first invoke vi. Most commands in command mode do not require you to press <return> after entering a command; commands are accepted directly upon entry. This is different from the way most non-vi UNIX commands are processed.

Remember that UNIX is case-sensitive. Upper and lower case characters are interpreted differently. If you enter an invalid character or sequence of characters, the editor will signal you with an error beep or message.

Exit Commands:
Leaving Input Mode:
You can leave the input mode and return to the command mode by pressing the ^c(i.e. control c) key at any time.

Leaving Command Mode:
You can leave the command mode and enter either input mode or return to the UNIX command level at any time.
Inputting Text:
The vi editor has three basic types of input commands with two options within each:

- **Insert** (i or I)
- **append** (a or A)
- **open** (o or O)

### Summary of Command Options:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Inserts text in front of the cursor</td>
</tr>
<tr>
<td>I</td>
<td>Inserts text at the beginning of the current line</td>
</tr>
<tr>
<td>a</td>
<td>Adds text after the cursor</td>
</tr>
<tr>
<td>A</td>
<td>Adds text at the end of the current line</td>
</tr>
<tr>
<td>o</td>
<td>Opens a new line to add text below the current line</td>
</tr>
<tr>
<td>O</td>
<td>Opens a line to add text above the current line</td>
</tr>
</tbody>
</table>

When you have finished entering text, press the `<escape>` key at any time to return to the command mode. The basic sequence of operations for inputting text is: command mode --> input mode --> enter text --> exit back to command mode.

### Other Basic Commands

#### Moving Around

In order to edit text in a file, you must move the cursor to the location where the editing is to begin. You must be in command mode to use any of the movement commands described here. Following are the basic commands to move around on the screen:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>h or &lt;backspace&gt;</td>
<td>Moves cursor to the left one character</td>
</tr>
<tr>
<td>l or &lt;spacebar&gt;</td>
<td>Moves cursor to the right one character</td>
</tr>
<tr>
<td>k</td>
<td>Moves cursor up one line</td>
</tr>
<tr>
<td>j or &lt;return&gt;</td>
<td>Moves cursor down one line</td>
</tr>
</tbody>
</table>

*(N.B: On some terminals, you can also use the arrow keys to perform these movements.)*
You can also move around from word to word within a file. A word is defined as one or more characters set off by spaces or punctuation.

<table>
<thead>
<tr>
<th>w</th>
<th>Moves to the first character of the next word</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>Moves back to the previous word</td>
</tr>
<tr>
<td>e</td>
<td>Moves to the end of the current word</td>
</tr>
</tbody>
</table>

**Scrolling**

<table>
<thead>
<tr>
<th>&lt;ctrl-d&gt;</th>
<th>Scrolls down one-half screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ctrl-u&gt;</td>
<td>Scrolls up one-half screen</td>
</tr>
<tr>
<td>&lt;ctrl-f&gt;</td>
<td>Scrolls forward one page</td>
</tr>
<tr>
<td>&lt;ctrl-b&gt;</td>
<td>Scrolls backward one page</td>
</tr>
<tr>
<td>&lt;ctrl-g&gt;</td>
<td>Scrolls down to the end of the file</td>
</tr>
</tbody>
</table>

To override the default settings for the scrolling commands, you can type a number before the command to indicate the number of lines you want to scroll. For example, 6<ctrl-d> would scroll down 6 lines.

**Deleting Text**

<table>
<thead>
<tr>
<th>x</th>
<th>Deletes a single character</th>
</tr>
</thead>
<tbody>
<tr>
<td>#x</td>
<td>Deletes a specified number of characters</td>
</tr>
<tr>
<td>dw</td>
<td>Deletes a word</td>
</tr>
<tr>
<td>#dw</td>
<td>Deletes a specified number of words</td>
</tr>
<tr>
<td>dd</td>
<td>Deletes a line</td>
</tr>
<tr>
<td>#dd</td>
<td>Deletes a specified number of lines</td>
</tr>
<tr>
<td>d$ or D</td>
<td>Deletes the remainder of a line</td>
</tr>
<tr>
<td>dG</td>
<td>Deletes through the last line in the file</td>
</tr>
</tbody>
</table>

If you decide after deleting something that you have made a mistake, you can cancel the effect of the delete command using the undo command. This command is discussed later in this section.
Replacing Text

There are numerous commands to replace text in vi. Only the basic ones will be discussed in this section.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>r</code></td>
<td>Replaces a single character</td>
</tr>
<tr>
<td><code>r</code></td>
<td>Replaces multiple characters overtype mode)</td>
</tr>
<tr>
<td><code>s</code></td>
<td>Substitutes multiple characters for a single character</td>
</tr>
</tbody>
</table>

Leaving the Editor

There are many ways to exit from vi depending on whether you want to save changes to a file, rename the file, or just quit without saving.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>:q</code></td>
<td>Quits if the file was previously saved</td>
</tr>
<tr>
<td><code>:q!</code></td>
<td>Quits without saving</td>
</tr>
<tr>
<td><code>:wq</code></td>
<td>Saves changes and quit</td>
</tr>
<tr>
<td><code>ZZ</code></td>
<td>Saves changes and quit (same as <code>:wq</code>)</td>
</tr>
<tr>
<td><code>:wq filename</code></td>
<td>Saves changes to filename and quit</td>
</tr>
</tbody>
</table>

_N.B:_ Use caution when leaving the editor to make sure you are using the proper command. It is always good practice to maintain a backup of important files to minimize the chance of losing data while making changes.
Redirection, Piping, and Filtering

Standard Input and Output

Often, in UNIX, it is advantageous to use the output from one command (or process) as input for another command (or process). There are several commands that are used for the management of what is called standard input and output. Standard input (stdin) is defined as input from a device from which UNIX normally accepts input (usually a terminal). Standard output (stdout) is the same concept applied to the output from UNIX (also, usually a terminal).

Redirection Commands

The concept of redirection is defined as:
- changing the destination of output from a command (or process);
- providing a source of input to a command (or process) from another command (or process).

<table>
<thead>
<tr>
<th>Redirection Operator</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Redirects stdout to a real file</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Appends stdout to a file</td>
</tr>
<tr>
<td>&lt;</td>
<td>Redirects stdin from a real file</td>
</tr>
<tr>
<td>&gt; &amp;</td>
<td>Redirects both stdout and stderr</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&amp;</td>
</tr>
</tbody>
</table>
Examples

Input:
gigastar % mail karen < schedule
This command mails the contents of the file called schedule to a user named karen. The
schedule file is used as input to the mail command without any intermediate steps.

Output:
gigastar % cal 8 1990 > august
This command takes the output of the calendar command and sends it to a new file called
august. It is important to note that this procedure will overwrite the file august.

gigastar % ls -al > dirlist
This command will create a file called dirlist that will contain the directory listing of the
current working directory.

gigastar % cat file1 > file2
This command will create a copy of file1 and name it file2.

Append Output:
gigastar % ls >> august
This command takes the output of the ls command and appends it to the end of the
august file. The contents of august will not be overwritten.

gigastar % cat file1 > merger
gigastar % cat file2 >> merger
This set of commands will merge file1 and file2 into a third file called merger.

Piping and Filtering Commands

Piping is the process of supplying the output of one command as input to another
command. Piping differs from redirection in that no file is created; rather, the standard
output is sent directly to the standard input. Filtering involves the processing of data into a
specified format. A filter can take raw data and put it in a specific form. An example of this
would be to list all the files in a directory that are older than thirty days. It is easy to see
how this could be useful when working with very large directories.

Examples

Piping:
gigastar % cal 1990 | lpr
This command takes the output of the calendar function and uses it as input to the lpr
command; lpr will, in turn, send the calendar to the local printer.
First, a man page of the command `cp` is requested; then the output of man is piped to the `lpr` command. This will, in effect, print out the man page listing on the local line printer.

**Filtering:**

```
gigastar % ls -al | grep Aug
```

This will list all the files in the current directory that were last modified in the month of August.

```
gigastar % head -50 hosts | tail -20 | lpr
```

This command uses the filters `head` and `tail` to send lines 31 through 50 of the hosts file to the local printer.

```
gigastar % tail -50 /var/adm/messages
```

This command will print out the last 50 system error messages from a file that contains a total listing of all error messages.

```
gigastar %head -5 serial.c | tail -1
```

The output of this command will be line number 5 of a C program called `serial.c`

```
gigastar %find / -name core -print >& /dev/null
```

The `find` command is used to locate any core files on the system starting at the root directory and display any occurrences on the screen. If the user does not own a directory, then an error message will be generated on the screen. To avoid the unnecessary messages, they can be directed to the null device (no designation).

```
gigastar %cat tree | spell
```

All misspelled words in the file tree are printed on the screen.
Printing

Determining Printer Status

On a UNIX system, printers may be directly connected to your local system, or available via other systems connected to your network. To find the names of the printers available and their accessibility, use the lpc status command. The lpc command stands for line printer control. This command is mostly for system administration use by the superuser, but some options are available to all users. Following is an example of how to use this command to list all printers available from your machine and their status.

```
gigastar % lpc status
```

<table>
<thead>
<tr>
<th>Message Displayed</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>lp: (printer name)</td>
<td>(printer name)</td>
</tr>
<tr>
<td>queuing is enabled</td>
<td>print jobs are being queued</td>
</tr>
<tr>
<td>printing is enabled</td>
<td>queued jobs are being printed</td>
</tr>
<tr>
<td>1 entry in spool area</td>
<td>there is 1 job in the queue</td>
</tr>
<tr>
<td>lp is ready and printing</td>
<td>the printer is on-line and printing</td>
</tr>
<tr>
<td>ps: (printer name)</td>
<td>(printer name)</td>
</tr>
<tr>
<td>queuing is enabled</td>
<td>print jobs are being queued</td>
</tr>
<tr>
<td>printing is enabled</td>
<td>queued jobs are being printed</td>
</tr>
<tr>
<td>14 entries in spool area</td>
<td>there are 14 jobs in the queue</td>
</tr>
<tr>
<td>waiting for sun to come up</td>
<td>ps is a remote printer; sun, the machine that ps is connected to, is not on-line</td>
</tr>
</tbody>
</table>

Looking at Print Queues

To view the printer queue for a specific printer:
```
gigastar % lpq -Pprintername
```

To view the queue for the default printer:
```
gigastar % lpq
```

To view the queue of a non-default printer:
```
gigastar % lpq -Pprintername
```
To view the queue of a Postscript printer named ps:

```
gigastar % lpq -Pps
```

Spooling Files to Print Queues

To send a file to the printer queue, the `lpr` command is used. To designate to which printer file will be sent, the `-P` printer name option is used. The only valid options for the `-P` option are those printer names that are displayed with the `lpc` status command.

Examples

To spool files to the default printer (lp):

```
gigastar % lpr filenames

gigastar % lpr -Plp filenames
```

To spool files to a different printer:

```
gigastar % lpr -Pps filenames
```

**N.B:** The printer names `ps` or `lw` usually designate Postscript or Laser Writer printers. While `lp`, `lp1`, `lp2`, etc. are used to designate line (text only) printers.
# Basic UNIX Commands

The table below contains often used UNIX commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>display reference manual pages</td>
<td>man <code>cmd_name</code></td>
</tr>
<tr>
<td></td>
<td>Basic Options:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-k locates commands by keyword lookup</td>
<td>man <code>-k keyword</code></td>
</tr>
<tr>
<td></td>
<td>-s specifies a chapter or section to look in</td>
<td>man <code>-s chap# cmd_name</code></td>
</tr>
<tr>
<td></td>
<td>-u basic usage description</td>
<td>man <code>-u</code></td>
</tr>
<tr>
<td>whereis</td>
<td>locate the executable, source, and man page files</td>
<td><code>whereis command</code></td>
</tr>
<tr>
<td></td>
<td>Basic Option:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-h basic usage description</td>
<td><code>whereis -h</code></td>
</tr>
<tr>
<td>finger</td>
<td>lists information about users</td>
<td><code>finger username</code></td>
</tr>
<tr>
<td></td>
<td>Basic Option:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-h basic usage description</td>
<td><code>finger -h</code></td>
</tr>
<tr>
<td>ftp</td>
<td>file transfer protocol</td>
<td><code>ftp computer.domain</code></td>
</tr>
<tr>
<td></td>
<td>Transfers files from one networked computer where you have an account to another where you have an account.</td>
<td></td>
</tr>
<tr>
<td>telnet</td>
<td>login on another computer on the network</td>
<td><code>telnet computer</code></td>
</tr>
<tr>
<td>w</td>
<td>list who is on the system and what they are doing</td>
<td><code>w</code></td>
</tr>
<tr>
<td>who</td>
<td>list who is on the system</td>
<td><code>who</code></td>
</tr>
<tr>
<td>whodo</td>
<td>list who is on the system and what they are doing</td>
<td><code>whodo</code></td>
</tr>
</tbody>
</table>

## Dealing with the File System

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>concatenate and display arguments to standard output</td>
<td><code>cat file</code></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Usage</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| cd      | change directory  
Note: given a full or relative path, set the working directory as specified; without arguments set the working directory to the login directory | cd path |
|         | Basic Option:  
- h basic usage description | cd -h |
| cp      | copy files | cp fromfile tofile |
|         | Basic Option:  
--help basic usage description | cp --help |
| du      | display disk usage  
NOTE: On rci 'du' and 'du -k' both yield results in 1 kilobyte blocks, while on eden 'du' yields results in \( \frac{1}{2} \) kilobyte blocks and 'du -k' yields results in 1 kilobyte blocks. | du |
|         | Basic Options:  
-k show usage in 1 kilobyte blocks  
--help basic usage description | du -k  
du ---help |
| file    | determine the type of a file | file filename |
|         | Basic Option:  
- h basic usage description | file -h |
| ftp     | file transfer protocol  
Transfers files from one networked computer where you have an account to another where you have an account. | ftp computer.domain |
| grep    | search for a character string in a file | grep string file |
|         | Basic Options:  
-v show lines that do not contain the string  
- h basic usage description | grep -v string file  
grep -h |
| head    | Show the first 10 lines of a file | head filename |
|         | Basic Options:  
-# show first # lines of the specified file  
--help basic usage description | head -20 filename  
head --help |
| ispell  | check the spelling of the contents of a file | ispell file |
|         | Basic Option:  
- h basic usage description | ispell -h |
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpq</td>
<td>check the status of a print queue</td>
<td><code>lpq</code></td>
</tr>
<tr>
<td></td>
<td>Basic Option:</td>
<td><code>-P printer</code> specifies the printer to be checked</td>
</tr>
<tr>
<td>lpr</td>
<td>send a job to a print queue</td>
<td><code>lpr options file(s)</code></td>
</tr>
<tr>
<td></td>
<td>Basic Options:</td>
<td><code>-P printer</code> specifies the printer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>-B bin</code> specifies the bin</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>-N note</code> print note on burster page</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>-m</code> send mail upon completion</td>
</tr>
<tr>
<td>lprm</td>
<td>remove a print job from a print queue</td>
<td><code>lprm print_job_#</code></td>
</tr>
<tr>
<td>ls</td>
<td>list the contents of a directory</td>
<td><code>ls options files(s)</code></td>
</tr>
<tr>
<td></td>
<td>Basic Options:</td>
<td><code>-a</code> all files</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>-d</code> list directories not their contents</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>-F</code> mark directories with <code>/</code>, executable files with <code>*</code>, symbolic links with <code>@</code>, and sockets with <code>=</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>-l</code> long listing showing protections, number of links, owner, size, and time of last modification</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>-s</code> size in kilobytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>--help</code> basic usage description</td>
</tr>
<tr>
<td>mkdir</td>
<td>create a new subdirectory in the current directory</td>
<td><code>mkdir subdir</code></td>
</tr>
<tr>
<td></td>
<td>create a new subdirectory, in the indicated target directory</td>
<td><code>mkdir subdir targetdir</code></td>
</tr>
<tr>
<td></td>
<td>Basic Option:</td>
<td><code>--help</code> basic usage description</td>
</tr>
<tr>
<td>mv</td>
<td>move or rename files</td>
<td><code>mv fromfile tofile</code></td>
</tr>
<tr>
<td></td>
<td>Basic Option:</td>
<td><code>--help</code> basic usage description</td>
</tr>
<tr>
<td>pr</td>
<td>prepare text for printing with headers and page breaks</td>
<td><code>pr file</code></td>
</tr>
<tr>
<td></td>
<td>Basic Options:</td>
<td><code>-h &quot;header text&quot;</code> set the page header</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>--help</code> basic usage description</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>pr --help</code></td>
</tr>
<tr>
<td>pwd</td>
<td>display the current directory's full pathname</td>
<td><code>pwd</code></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Usage</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>quota</strong></td>
<td>check to see if over allowed usage, no response if not</td>
<td><strong>quota</strong></td>
</tr>
<tr>
<td><strong>quota</strong></td>
<td>Basic Option: -v verbose, show status even if not over quota With this option eden will display more information than rci.</td>
<td></td>
</tr>
<tr>
<td><strong>rm</strong></td>
<td>remove (delete) files</td>
<td><strong>rm</strong> <strong>file</strong></td>
</tr>
</tbody>
</table>
| **rm** | Basic Options: -i interactive, ask before removal -r recursive, descend into subdirectories removing files and then containing directories. --help basic usage description | | **rm** **-i** **file**  
| **rm** | Note: Once a file is deleted, you can't undelete it. Use the -i option to have UNIX ask if you are sure about removing the file. | | **rm** **-r** directory  
| **rm** | **rm** **--help** |
| **rmdir** | remove empty directories | **rmdir** **dirname** |
| **rmdir** | Basic Option: --help basic usage description | | **rmdir** **--help** |
| **wc** | display the number of lines, words and characters in a file | **wc** **filename** |
| **wc** | Basic Option: --help basic usage description | | **wc** **--help** |
| **exit** | log off the system | **exit** |
| **id** | display your system ID numbers if given a username, display their ID numbers | **id** **username** |
| **logout** | log off the system | **logout** |
| **printenv** | show your current environment variable settings if given an environment variable name display that setting | **printenv**  
| **printenv** | **printenv** **varname** |
| **setenv** | set environment variables | **setenv** **varname** **value** |