

UNIX for Non-Programmers

Contents

Introduces the following utilities, listed in alphabetical order:

cancel

cat

chgrp

chmod

chown

clear

cp

date

emacs

file

groups

head

lp

lpr

lprm

lpq

lpstat

ls

mail

man

mkdir

more

mv

newgrp

page

passwd

pwd

rm

rmdir

stty

tail

tset

vi

wc

The Unix System

- ▶ Multiuser
 - Server – connect via terminals
- ▶ Single User
 - Your own PC
- ▶ Interface
 - Text based
 - GUI



What the difference between a...

- ▶ PC and a Workstation?
- ▶ Previously
 - Memory, Hard Drive, Screen
- ▶ Currently – O/S it was designed for
 - Windows = PC
 - Unix = Workstation
- ▶ Connectivity



Basics

- ▶ Remember, Unix is Case Sensitive

CD FILE

cd file

Are not the same!

- ▶ Type mostly in lowercase
- ▶ Backspace
 - Either “Backspace” or “Delete” Key
(Depending on system) or “Ctrl + H”



Logging In

- ▶ Username (Login ID)
 - a unique name that distinguishes you from the other users of the system.
- ▶ Usernames and initial passwords are assigned to you by the system administrator(root – super user).
 - login:
 - Password:
- ▶ User Accounts



User Accounts

- ▶ Typically: first initial lastname

- John Doe = jdoe
- Im Hyunmi = him

- ▶ First Task – Change Passwords

- Passwd

- Passwords must be 6+ characters Must be a good mix letters/digits
 - Can not be a variation of your login
 - Case sensitive

Shells

- ▶ See either a \$, a % or another prompt.
 - default **shell** prompting – a special kind of program
- ▶ **A Shell**
 - a **middleman** between you and the raw UNIX operating system.
 - **run programs**, build **pipelines of processes**, save **output to files**, and run more than one program at the same time.
 - **executes all of the commands**
- ▶ The four most popular shells are:
 - the Bourne shell (sh)
 - the Korn shell (ksh)
 - the C shell (csh)
 - the Bash Shell (bash)



Running another shell

- ▶ If you don't like the shell, just type in the name of the one that you like and press Enter.

```
ka 9: echo $SHELL
/usr/bin/csh
ka 10: sh
$ bash
bash-2.03$ exit ^D
bash
$ ^D
ka 11: _
```

```
--> csh -> sh
--> sh -> bash
--> back to

--> back to sh
--> back to csh
```

Logging Off

- ▶ exit
- ▶ logout
- ▶ ^D (Ctrl + D)
- ▶ *Depends on your version of Unix*

Turning Off a Unix System

(If you are the owner / root)

- ▶ CTRL-ALT-DEL
- ▶ init 5
- ▶ shutdown now
- ▶ halt
- ▶ *Depends on your version of UNIX*

Utility

ls	Display list of file
cat	Display content of file
more	Display file w/pause
cp	Copy file
grep	Find string in file
diff	Compare files
mv	Rename file
man	Get help

Utility

chmod	Change file protection
cd	Create directory
mkdir	Make directory
rmdir	Remove directory
df	Display disk space
vi	Edit a file
lp	print a file

Utility

date	Display Date/Time
who,w	Who else is online
cal	Display a calender
finger	Displays info about a user
talk	Chat with other user
head	First few lines of the file
tail	Last few lines of a file

Utility

echo	Displays text
man	Online manuals
compress	Compress a file
uncompress	Uncompress a file
file	Displays information
zcat	Display a compressed file

Using special characters

stty -a

stty erase ^H

--> obtain terminal line settings

--> setting

Meta-character	Meaning
^h erase	Backspace one character
^u kill	Erase all of the current line.
^w werase	Erase the last word.
^r rprnt	Reprint the line.
^o flush	Ignore any pending input and reprint the line.
^v lnext	Don't treat the next character specially.
^z susp	Suspend the process for a future awakening.
^c intr	Terminate (interrupt) the foreground job with no core dump.
^\ quit	Terminate the foreground job and generate a core dump.
^s stop	Stop/restart terminal output.
^d eof	End of input.

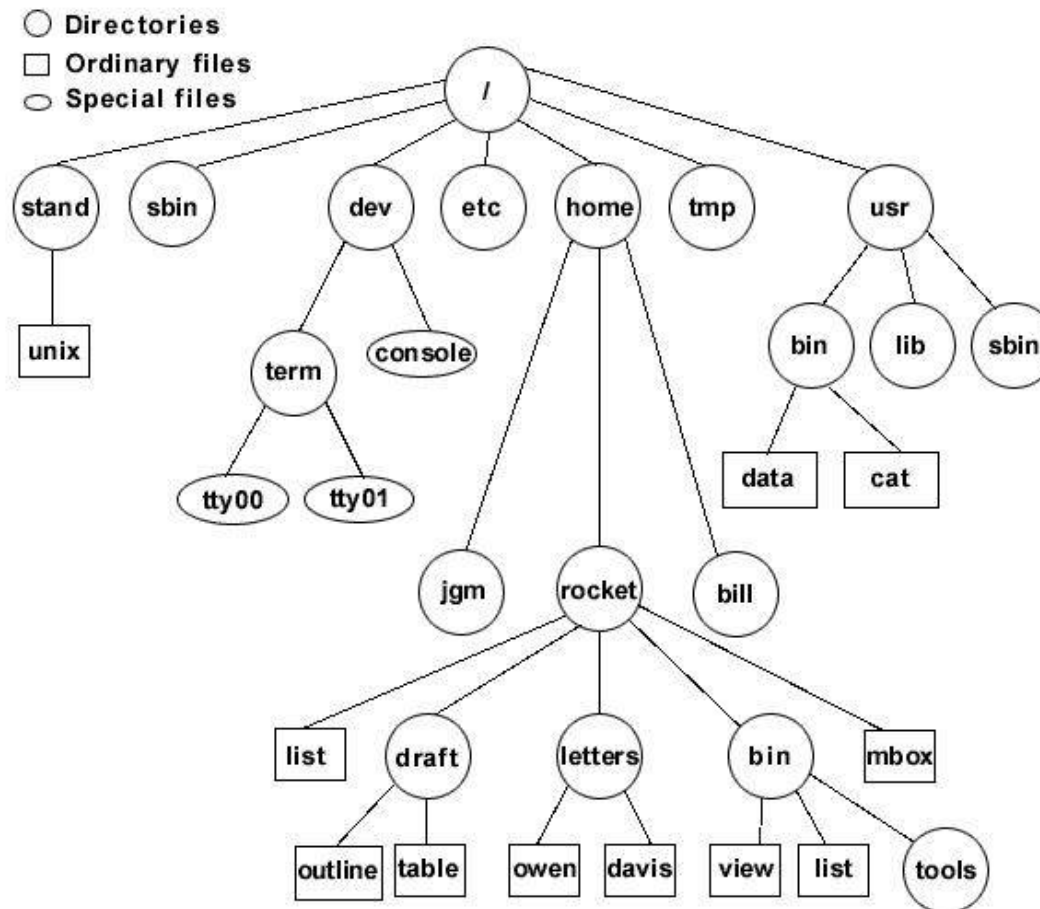
Setting/Changing password

`passwd [user]`

- ▶ When first login to a system, change your initial password.
- ▶ Superuser(root) can access everything.
- ▶ stored in an encrypted password file
“/etc/passwd” or in a “shadow” file (for more security), depending on your version of UNIX.

File Structure

► Hierarchical File Structure



File Structure

- ▶ **Files**
 - Everything is a file
- ▶ **Filename**
 - Characters
 - A–Z, a–z Case Sensitive
 - 0–9 Digits
 - _ Underscore
 - . Period
 - Length: 255 Characters

File Structure

▶ Invisible Filename

- Start with a period
 - .hidden
 - .login
 - .profile
- `ls -a` displays all filenames



File Structure

▶ Command File Extensions

.c	C Program
.cc	C++ Programs
.csh	C shell script
.h	Header
.o	Object file
.sh	Shell program
.tar	Archived using TAR
.uu	UUEncoded File
.Z	compress file

File Structure

▶ Directories

- Adds a level of organization
- Tree based
- Starts at / (root)

▶ Subdirectories

- Very similar to MS/DOS
 - They "borrowed" it from UNIX

Printing Working Directory: pwd

pwd

- ▶ displays your shell's current working directory

Home Directory

- ▶ Where you first log in
- ▶ This is your file area
 - Typically
/home/username
- ▶ Contains hidden start up files
- ▶ The system administrator assigns these home-directory values.

Directory Operations

mkdir	Make Directory
cd	Change Directory
rmdir	Remove (empty) Directory
pwd	Display working Directory

Changing Directories: cd

`cd [directoryName]`

- ▶ changes a shell's **current working directory** to be **directoryName**.
- ▶ If argument is omitted(or “~”), return to its **owner's home directory**.
- ▶ Special Directories
 - . Current Directory
 - .. Parent Directory (up one)
 - ~ home Directory

Pathnames

▶ Absolute

- Starts with /

```
/home/shmoon$ cd /tmp  
/tmp$
```

▶ Relative

- Starts in current directory

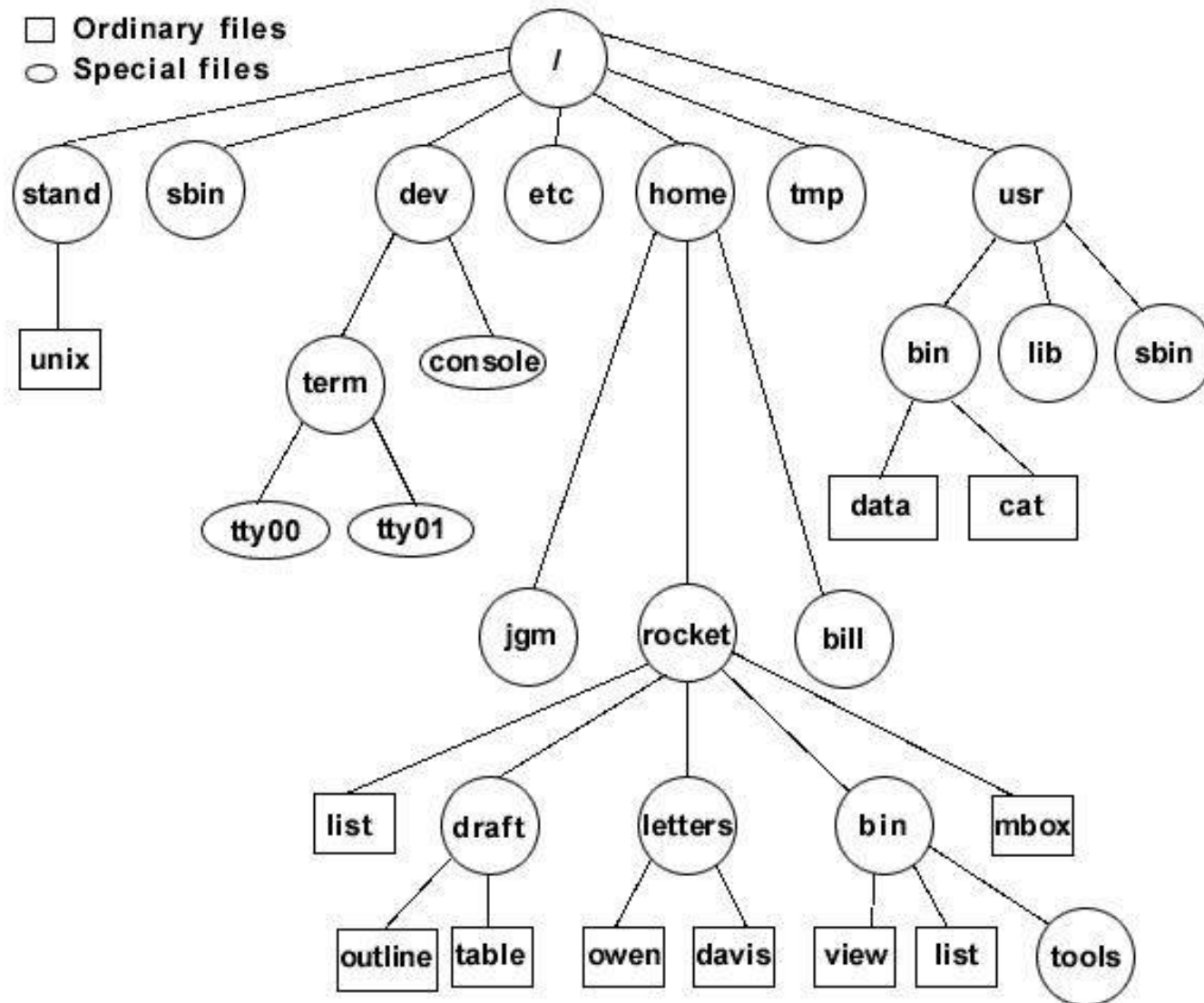
```
/home/shmoon$ cd tmp  
/home/shmoon/tmp$
```

Given this tree

○ Directories

□ Ordinary files

◌ Special files



Inclass

- ▶ Your current directory is
/home/rocket

- ▶ What are the absolute and relative
 - 1. davis
 - 2. Table
 - 3. bill
 - 4. cat
 - 5. sbin
 - 6. list
 - 7. tty01

Making Directory: mkdir

`mkdir newDirectoryName`

- ▶ creates a directory

```
$ pwd
```

```
/home/shmoon
```

```
$ mkdir test
```

or

```
$ mkdir /home/shmoon/test
```

Deleting a Directory: rmdir

`rmdir { directoryName }+`

- ▶ removes all of the directories.
- ▶ A directory must be empty before it can be removed.
 - To remove a directory and all of its contents, use the `rm` with the `-r` option.

Special Directories

Name	Use
/	root
/dev	I/O devices - disks...
/etc	sys admin stuff
/opt	applications
/home	user directories
/tmp	temporary files
/var	system variables
/bin	commands
/usr	commands, libraries

man: online help

man word

man -k keyword

- ▶ Display on-line copies of the original UNIX documentation, which is usually divided into eight sections.



Displaying a File: cat

`cat -n {fileName}*`

- ▶ takes its input from **standard input** or from a list of files and displays them to **standard output**.
- ▶ By default, **the standard input of a process** is from **the keyboard** and **the standard output** is to **the screen**.
- ▶ The **-n** option **adds line numbers to the output**.

Displaying a File: cat

\$ **cat heart** --> list the contents
of the “heart” file.

I hear her breathing.

I’m surrounded by the sound.

Floating in this secret place,

I never shall be found.

\$ _

- ▶ **cat** is good for listing the contents of small files, but it doesn’t pause between full screens of output.

Displaying a File: more

`more -f [+lineNumber] { fileName }*`

- ▶ scrolls a list of files, one page at a time.
- ▶ By default, each file is displayed starting at line 1,
- ▶ `+option` may be used to specify the starting line number.
 - To list the next page, press the space bar.
 - To list the next line, press the Enter key.
 - To quit from more, press the “q” key.
 - `^B` will display the previous page
 - `H` will display help page

```
$ ls -la /usr/bin > myLongFile
```

```
$ more myLongFile
```

Displaying a File: head and tail

`head -n { fileName }*`

- ▶ displays the first 10 lines by default.
- ▶ `-n` option displays `n` lines of a file.

`tail -n { fileName }*`

- ▶ displays the last 10 lines of a file.
- ▶ `-n` option displays last `n` lines of a file.

- ▶ The first two lines and last two lines of my “heart” file.

`$ head -2 myLongFile`

--> list the first two lines.

`$ tail -2 myLongFile`

--> list the last two lines.

`$ head -15 myLongFile`

--> see what happens

Creating a file with cat

\$ **cat > heart** --> store keyboard input into a file called "heart".

I hear her breathing,
I'm surrounded by the sound.
Floating in this secret place,
I never shall be found.

^D --> tell cat that the end of input has been reached.

\$ _

Creating a file

- ▶ Three way to create file

1. `cat > test.txt`

2. `touch test.txt`

3. `vi test.txt`

Listing Contents of a Directory: ls

`ls -adglFR { fileName }* {directoryName}*`

- ▶ lists information about a file or a directory in alphabetical order, excluding files whose names start with a period.
 - The `-a` option display all files
 - The `-l` option generates a long listing, including permission flags, the file's owner, and the last modification time.

Directory Listing

- ▶ `$ ls -->` list all files in current directory.

heart

`$ ls -l heart`

`-->` long listing of "heart."

```
-rw-r--r--  1  glass  106  Jan 30 19:46  heart
```

the hard-link count

permission mode of the file

the username of the owner of the file

the size of the file, in bytes

the time that the file was last modified

the name of the file

Renaming/Moving a File: mv

```
mv -i oldFileName newFileName
```

```
mv -i {fileName}* directoryName
```

```
mv -i oldDirectoryName newDirectoryName
```

1. renames oldFileName as newFileName.
2. moves a collection of files to a directory.
3. moves an entire directory.

▶ The `-i` option prompts you for confirmation(interactive)



Copying Files: cp

```
cp -i oldFileName newFileName
```

```
cp -ir { fileName }* directoryName
```

- ▶ copies the contents of oldFileName to newFileName.
- ▶ The -r option causes any source files that are directories to be recursively copied,

Deleting files and directories

`rm -fir {fileName}*`

- ▶ removes a files from the directory hierarchy.
- ▶ the `-r` option causes all of directory's contents, including subdirectories, to be recursively deleted.
- ▶ The `-f` option inhibits all error messages and prompts. It overrides the `-i` option (also one coming from an alias). **This is dangerous!**

Filtering Files

`grep 'reg' fileName`

- ▶ filters out, all lines that do not contain a specified pattern.

`$ cat grepfile` ----> list the file to be filtered

Well you know it's your bedtime,
So turn off the light,
Say all your prayers and then,
Oh you sleepy young heads dream of wonderful things,

`$ grep the grepfile` ----> search for the word "the"

So turn off the light,
Say all your prayers and then,

Comparing Files: diff

`diff fileName1 fileName2`

- ▶ compares two files and displays a list of editing changes that would convert the first file into the second file.

\$ `cat lady1` --> look at the first test file.

Lady of the night,
I hold you close to me,
And everything you say to me is right.

\$ `cat lady2` --> look at the second test file.

Lady of the night,
I hold you close to me,

\$ `diff lady1 lady2` --> compare lady1 and lady2.

3d2

< And all those loving words you say are right.

Counting Lines, Words and Characters in Files: wc

wc filename

- ▶ obtains a count of the number of lines, words, and characters.

Eg)

```
9    43   213  heart.final
```

- ▶ The **-l option** requests a line count,
- ▶ the **-w option** requests a word count,
- ▶ the **-c option** requests a character count.

clear

- clears your screen.

date [mmddhhmm[[CC]YY][.ss]]

- Without any arguments, displays the current date and times.
- with arguments , sets the date
- `rdate -s time.bora.net`



Archives: tar, gzip, bzip2

`tar {ctxfvuz} {archiveName} {fileOrDirectoryName}*`

- ▶ **saves directory structures** onto a single volume or disk archives.
- ▶ designed specifically for maintaining an archive of files on **a magnetic tape**. However, is also used commonly when archiving into an archive file rather than a tape.

Tape Archiving: tar

```
tar {ctxfvuz} {archiveName}  
    {fileOrDirectoryName}*
```

- **c**: creates a tar-format file
- **f**: use the next file name as the output file
- **v**: encourages output
- **t**: generates a table of contents
- **x**: extract
- **z**: compress with gzip – in most implementations
- **C**: specifies target directory

Archiving: gzip, bzip2

gzip {fileName}*

gunzip {compressedFileName}*

bzip2 {fileName}*

bunzip2 {compressedFileName}*

► Used with tar

- -z, -j

File Attributes

Field #	Field value	Meaning
1	-rw-r--r--	the type and permission mode of the file, which indicates who can read, write, and execute the file
2	1	the hard-link count
3	glass	the username of the owner of the file
4	cs	the group name of the file
5	213	the size of the file, in bytes
6	Jan 31 00:12	the time that the file was last modified
7	heart.final	the name of the file

File Attributes

▶ Filenames

- up to 255 characters in length.
- cannot use “.” and “..”,
 - > current working directory and its parent directory

▶ Time of Last File Modification

- the time that the file was last modified and is used by several utilities.



File Attributes

► File Owner

- the owner of the file. which is typically the same as the username of the person who started it.
- the username(text) is typically refer to a user, representing user ID which is typically integer.
 - The username is easier for humans to understand than a numeric ID.

► File Group

- a member of a group. It is assigned by the system administrator and is used as part of the UNIX security mechanism.

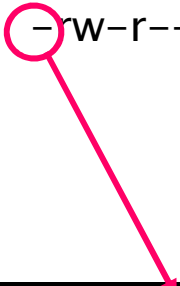
File Attributes

► File Types

- describes the file's type and permission settings.

In `ls -l` example:

`-rw-r--r-- 1 glass cs 213 Jan 31 00:12 heart.final`



character	File Type
-	regular file
d	directory file
b	buffered special file(such as a disk drive)
c	unbuffered special file(such as a terminal)
l	symbolic link
p	pipe
s	socket

File Permissions (Security)

- File permissions are the basis for file security. They are given in three clusters.

-rw-r--r-- 1 glass cs 213 Jan 31 00:12 heart.final

User (owner)	Group	Others
rw-	r--	r--

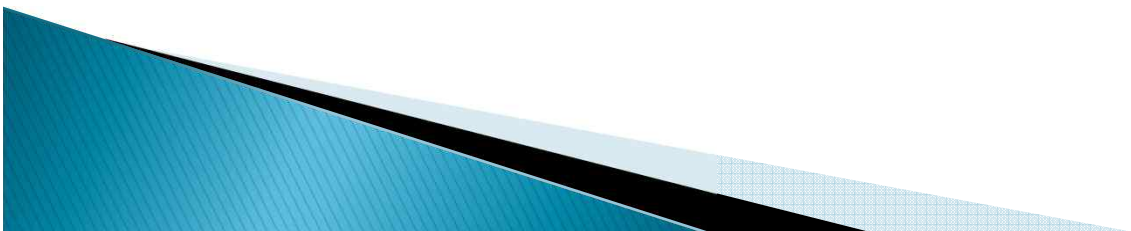
← clusters

Each cluster of three letters has the same format:

Read permission	Write permission	Execute permission
r	w	x

File Permission

- ▶ Who is allowed to see what
- ▶ Three types of users
 - User (Owner), Group, Other
- ▶ Three types of permissions
 - Read, Write, Execute



File Permission

- ▶ The meaning of the read, write, and execute permissions depends on the type of file:

	Regular file	Directory file	Special file
Read	read the contents	read the directory (list the names of files that it contains)	read from the file using the read() system call.
Write	change the contents	Add or remove files to/from the directory	write to the file using the write() system calls.
Execute	execute the file if the file is a program	access files in the directory	No meaning.

File Security

- ▶ When a process creates a file, the **default permissions** given to that file are modified by a special value called the **umask**.
- ▶ It's perfectly possible, although unusual, for the owner of a file to have fewer permissions than the group or anyone else.
- ▶ **Hard-Link Count**
 - indicates **how many labels in the hierarchy are pointing to the same physical file**.



File Security

- ▶ four values related to file permissions:

1. A real user ID
2. An effective user ID
3. A real group ID
4. An effective group ID

- ▶ When a process runs

1. If the process' effective user ID is the same as the owner of the file, the User permission apply.
2. If the process' effective user ID is different, but its effective group ID matches the file's group ID, then the Group permissions apply.
3. If neither the process's effective user ID nor the effective group ID matches the owner of the file and the group ID, respectively, the Others permission apply.

Listing File Group: groups

groups [userId]

- ▶ Lists all of the groups
- ▶ If the name of a user, lists of the groups to which that user belongs



Changing File Group: chgrp

`chgrp -R groupname { fileName }*`

- ▶ changes the group of files.
- ▶ A super-user(root) can change the group of any file.
- ▶ **The -R option** recursively changes the group of the files in a directory.

Changing File Owner: chown

```
chown -R newUserId {fileName}+
```

- ▶ allows a super-user to change the ownership of files
- ▶ The **-R option** recursively changes the owner of the files in directories.

Change File Permissions: chmod

`chmod -R change{, change}* {fileName}+`

- ▶ changes the **modes (permissions)** of the specified
- ▶ **symbolic**
 - Use shorthand name
- ▶ **absolute**
 - Use an octal value to set

Chmod Symbolic

`clusterSelection+newPermissions` (add permissions)

`clusterSelection-newPermissions` (subtract permissions)

`clusterSelection=newPermissions` (assign permissions absolutely)

▶ where `clusterSelection` is any combination of:

`u` (user/owner)

`g` (group)

`o` (others)

`a` (all)

and `newPermissions` is any combination of

`r` (read)

`w` (write)

`x` (execute)

`s` (set user ID/set group ID)

Changing File Permissions

- ▶ The **-R option** recursively changes the modes of the files in directories.
- ▶ **changing a directory's permission settings** doesn't change the settings of the files in the directory.
- ▶ combination

Requirement	Change parameters
Add group write permission	g+w
Remove user read and write permission	u-rw
Add execute permission for user, group, and others.	a+x
Give the group read permission only.	g=r
Add write permission for user, and remove group read permission.	u+w,g-r

Chmod absolute

- ▶ use an octal number to set permissions.
- ▶ Each octal digit represents a permission triplet.

For example, settings of `rwxr-x--`
the octal permission `s: 750`, calculated as follows:

	User	Group	Others
setting	rwX	r-X	--
binary	111	101	000
octal	7	5	0

```
$ chmod 755 myprog
```

```
rwXr-xr-x
```


Chmod asolute

Permission	On	Off
Read	04	0
Write	02	0
Execute	01	0

- Read + Execute = 5

Number value	Octal representation	rwX permissions
7	111	rwx
6	110	rwx
5	101	r-x
4	100	r--
3	011	-wx
2	010	-w-
1	001	--x
0	000	---

umask

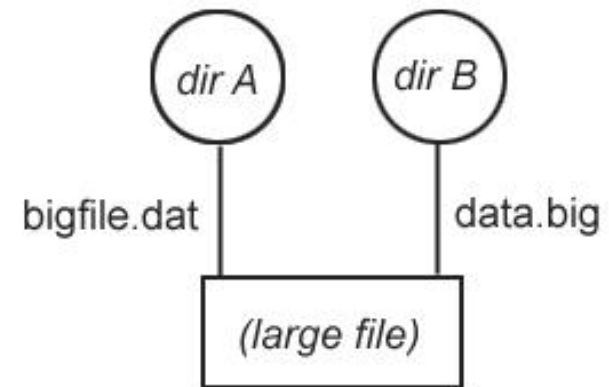
- ▶ Set the default file attributes
 - ▶ It is the 1's complement
 - ▶ Default is 022
 - ▶ Based on 666 for files
 - ▶ Based on 777 for directories
- 
- ▶ Subtract umask to get permissions

umask

- ▶ File: 666 -rw-rw-rw
 umask 022
 result 644 -rw-r--r--
- ▶ Directory: 777 -rwxrwxrwx
 umask 022
 result 755 -rwxr-xr-x

Links

- ▶ Like shortcuts in Windows
- ▶ Links files together
- ▶ Also works for directories
- ▶ **Hard Links**
 - Not copies – the same file
 - Different names
- ▶ **Symbolic Links**
 - Only symbolic – two files
 - One is a pointer to the other



Hard Links: In

- ▶ create both **hard links** and **symbolic (soft) links** between files.
- ▶ add a new label “hold” to the file referenced by the existing label “hold.3”.

\$ **ls -l**
directory.

--> look at the current contents of the

total 2

-rw-r--r-- 1 glass 123 Jan 12 17:32 hold.1

-rw-r--r-- 1 glass 91 Jan 12 17:34 hold.3

Hard Links: In

```
$ ln hold.3 hold
```

--> create a new hard link.

```
$ ls -l
```

--> look at the new contents of the directory.

```
total 3
```

```
-rw-r--r--  2  glass   91 Jan 12 17:34 hold
-rw-r--r--  1  glass 124 Jan 12 17:32 hold.1
-rw-r--r--  2  glass   91 Jan 12 17:34 hold.3
```

```
$ rm hold
```

--> remove one of the links.

```
$ ls -l
```

```
Total 2
```

```
-rw-r--r--  1  glass 123 Jan 12 17:32 hold.1
-rw-r--r--  1  glass  91 Jan 12 17:34 hold.3
```

Soft Links: ln

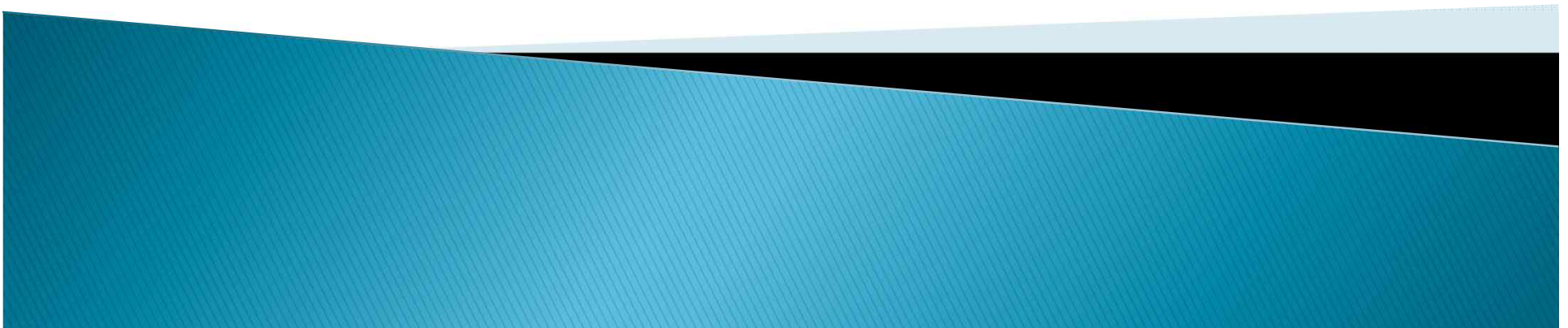
To create a soft link, use `ln -s`

```
$ ls -l
total 48
-rw-r--r--          1 user  users  84 26 Sep 17:08 tmp
-rw-r--r--          1 user  users  24 26 Sep 19:41 tmp1
$ ln -s tmp tmp3
$ ls -l
total 64
-rw-r--r--          2 user  users 84 26 Sep 17:08 tmp
-rw-r--r--          1 user  users 24 26 Sep 19:41 tmp1
lrwxr-xr-x          1 user  users  3 27 Sep 09:08 tmp3 -> tmp
$ _
```


Removing links

- ▶ Use `rm` utility
- ▶ When you delete a file, the symbolic link can still exist.
- ▶ Be sure to clean up your mess!

Editing Files with vi



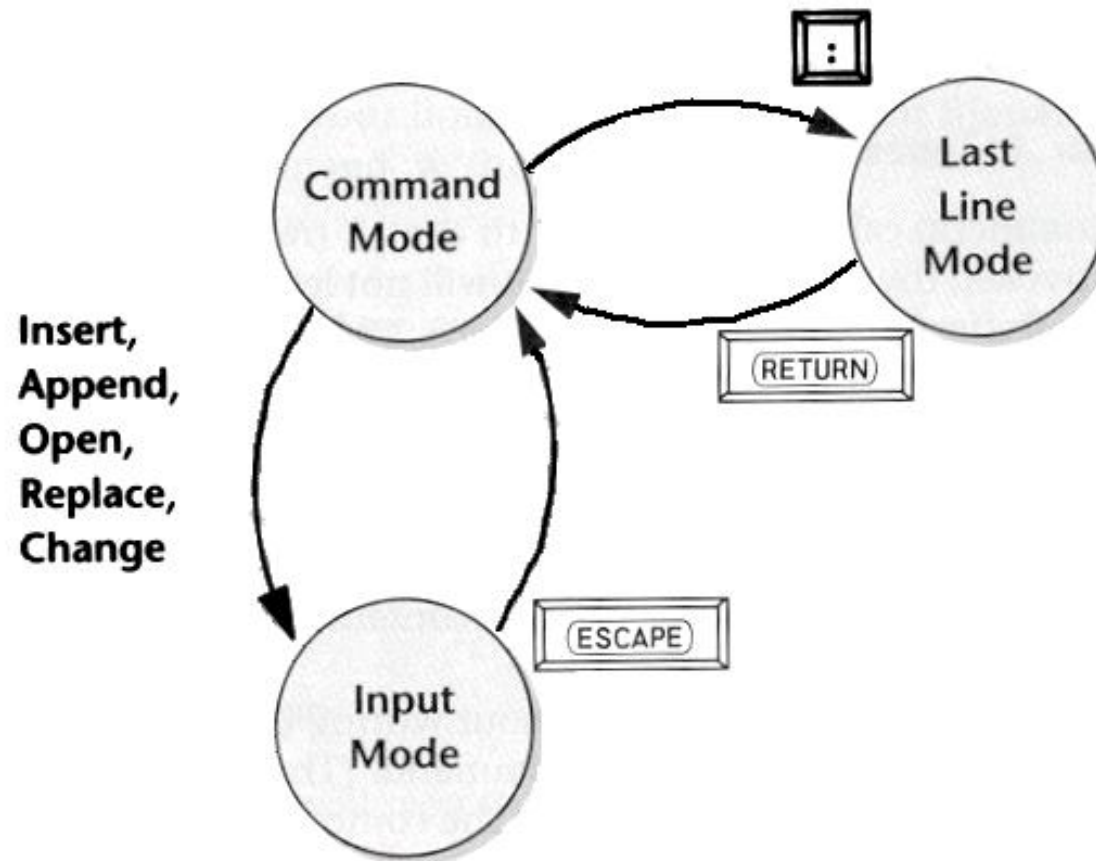
Editing Files with vi

- ▶ The two most popular UNIX text editors are called **vi** and **emacs**.
- ▶ The **vi editor** was originally developed for **BSD UNIX** by Bill Joy of the University of California at Berkeley.
- ▶ This is a standard utility for **System V** and most other versions of UNIX.
 - “vi” stands for “visual editor”.



```
$ vi poem.txt
```

Editing Files with vi



Text Entry Mode in vi

- ▶ To enter **text-entry mode** from command mode, press **one of the keys** in the table below.

Key	Action
i	Text is inserted in front of the cursor.
I	Text is inserted at the beginning of the current line.
a	Text is added after the cursor.
A	Text is added to the end of the current line.
o	Text is added after the current line.
O	Text is inserted before the current line.
R	Text is replaced (overwritten) .
:	Enter an extended command .

vi: Command Mode

- ▶ To transfer from **text-entry mode** to command mode, **press the Esc key**.
- ▶ A lot of editing features **require parameters and are accessed** by pressing **the colon (:) key**, followed by the command sequence, followed by **the Enter key**.
- ▶ For example, **to delete lines one through three,**

:1,3d<Enter>



vi: Command Mode

vi allows you to use the “\$” to denote the line number of the last line in the file and the “.” to denote the line number of the line currently containing the cursor.

For example, the sequence

`:.+2d<Enter>`

would delete the current line and the two lines that follow it.

vi: Line Ranges

- ▶ Here are some other examples of commands for line ranges:

Range	Selects
1,\$	all of the lines in the file
1,.	all of the lines from the start of the file to the current line, inclusive
.,\$	all of the lines from the current line to the end of the file, inclusive
.-2	the single line that's two lines before the current line

vi: Common Editing Features

- ▶ The most common **vi editing features** can be grouped into the following categories:
- ▶ cursor movement
- ▶ deleting text
- ▶ replacing text
- ▶ pasting text
- ▶ searching text
- ▶ search/replacing text
- ▶ saving/loading files
- ▶ miscellaneous (including how to quit vi)

vi: Cursor Movements

- ▶ Here is a table of the common **cursor-movement commands**:

Movement	Key sequence
Up one line	Arrow Up or the “k” key
Down one line	Arrow Down or the “j” key
Right one character	Arrow Right or the “l” key (will not wrap around)
Left one character	Arrow Left or the “h” key (will not wrap around)
To start of line	^
To end of line	\$
Back one word	the “b” key
Forward one word	the “w” key
Down a half screen	Control-d
Forward one screen	Control-f
Up a half screen	Control-u
Back one screen	Control-b
To line nn	:nn<Enter> (nn G also works)

vi: Deleting Text

Here is a table of the common **text-deletion** commands:

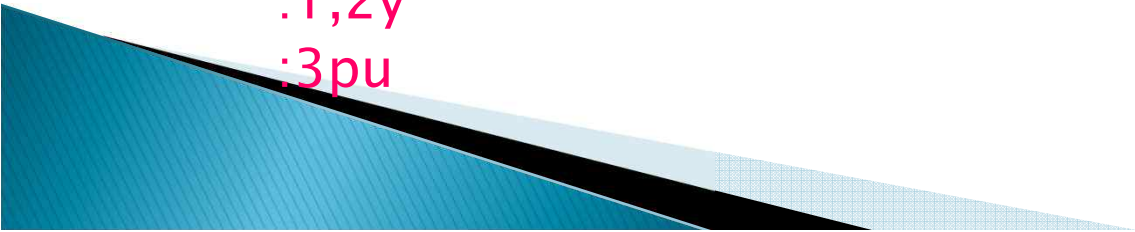
Item to delete	Key sequence
Character	Position the cursor over the character and the press “x” key.
Word	Position the cursor at the start of word and then type the two character “dw” .
Line	Position the cursor anywhere on the line and then type the two characters “dd” (Typing a number ahead of “dd” will cause vi to delete the specified number of lines beginning with the current line.)
Current position to end of current line	Press the “D” key.
Block of lines	:<range>d<Enter>

vi: Pasting Text

Here is a table of the most **common pasting operations**:

Action	Key sequence
Copy(yank) lines into paste buffer.	:yy or :<range>y<Enter>
Insert (put) paste buffer after current line.	p or :pu<Enter> (contents of paste buffer unchanged)
Insert paste buffer after line nn	:nnpu<Enter> (contents of paste buffer unchanged)

- ▶ For example, to copy the first two lines of the poem into the paste buffer and then paste them after the third line, I entered the following two commands:



```
:1,2y  
:3pu
```

vi: Searching

- ▶ Here is a table of the most common [search operations](#):

Action	Key Sequence
Search forward from current position for string <code>sss</code> .	<code>/sss <Enter></code>
Search backward from current position for string <code>sss</code> .	<code>?sss <Enter></code>
Repeat last search.	<code>n</code>
Repeat last search in the opposite direction	<code>N</code>

- ▶ For example, I searched for the substring “ark” in line one of the poem by entering the following commands:

```
:1 <Enter>  
/ark <Enter>
```

vi: Searching/Replacing

- ▶ You may perform **global search-and-replace operations** by using the following commands:

Action	Key Sequence
Replace the first occurrence of sss on each line with ttt .	:<range>s/sss/ttt/<Enter>
Replace every occurrence of sss on each line with ttt (global replace).	:<range>s/sss/ttt/g<Enter>

- ▶ % – entire file

vi: Saving/Loading Files/exit

- ▶ Here is a table of the most common [save/load file](#) commands:

Action	Key Sequence
Save file as <name>.	<code>:w<name><Enter></code>
Save file with current name.	<code>:w<Enter></code>
Forced save into a file that exists	<code>:w!<Enter></code>
Save only certain lines to another file.	
<code>:<range>w<name><Enter></code>	
Read in contents of another file at current position	<code>:r<name><Enter></code>
Quit vi if work is saved.	<code>:q<Enter></code>
Quit vi and discard unsaved work .	<code>:q!<Enter></code>
Save and Quit vi .	<code>:wq<Enter></code>