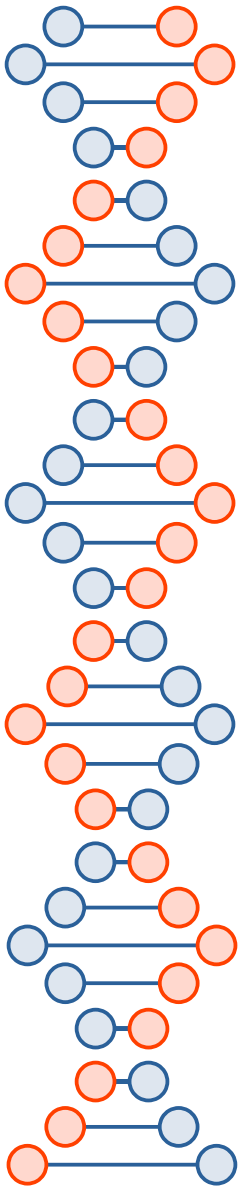
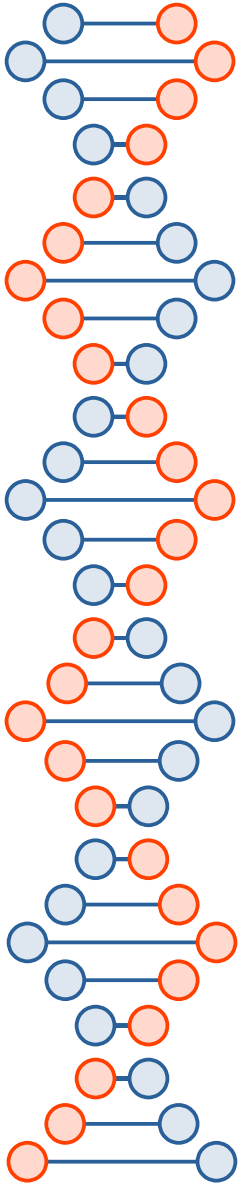
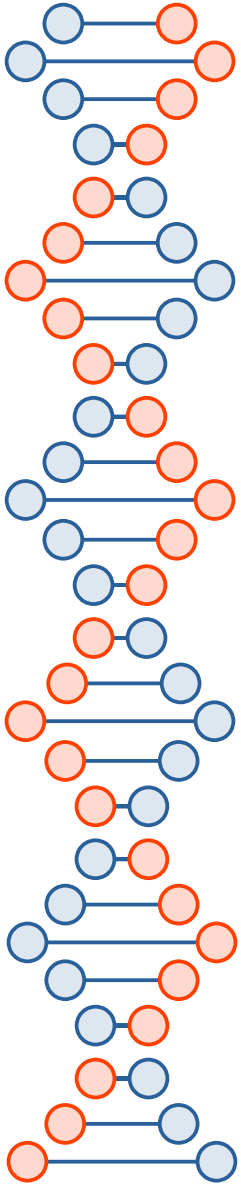


Introduction to Linux Command Line



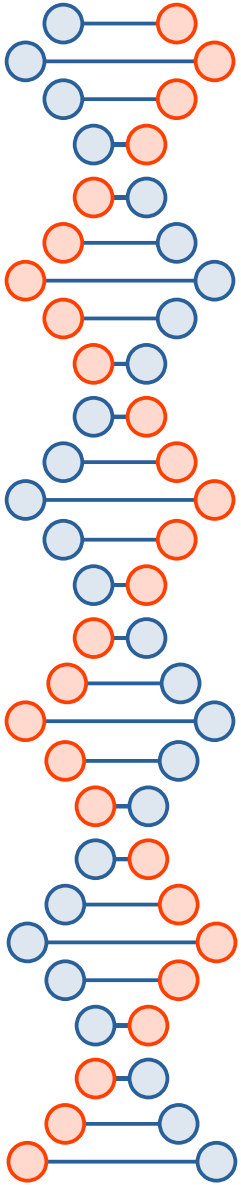
Outline

- What is shell?
- Navigating Directories and Files
- Manipulating Files and Directories
- Working with Commands and Redirection
- Brief Introduction to Regular Expressions
- Installing Miniconda



What is Shell?

- When we think of Command Line...we are really talking about shell
- Shell is a program that take keyboard commands and passes them to the OS to carry out
- Most Linux distributions have a shell program called Bash
- **Activity: Open your terminal.**

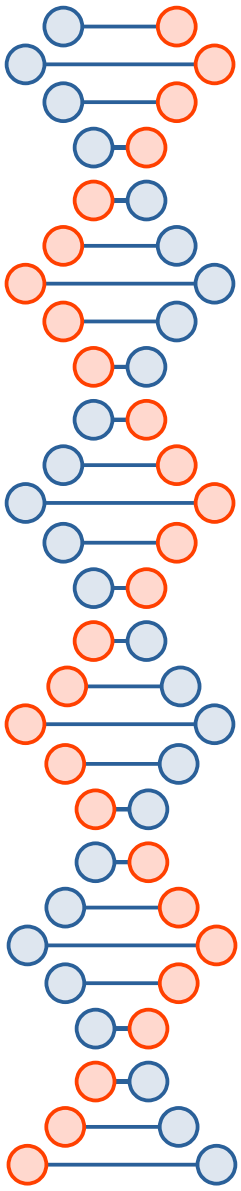


Shell prompt and Command History

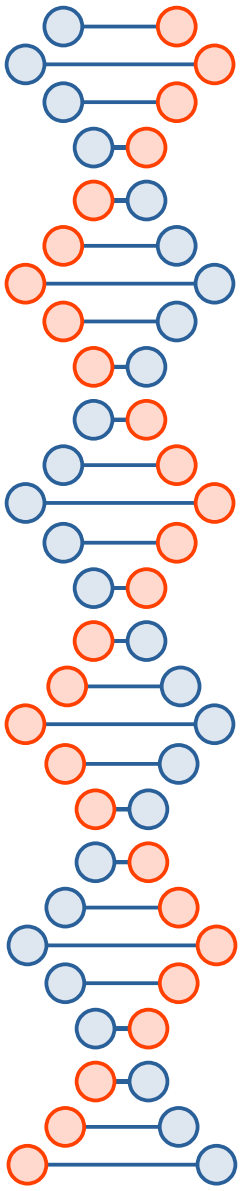
- When you open the terminal, you will see the shell prompt where you write your commands.
- Write jibberish into the terminal
- What does it show?
- Why does it show that?
- Use the up arrow to see the command you typed in.

Try Some Simple Commands

- **Type in: date** ---you see today's date
- **Type in: cal** ---you see this month's calendar
- **Type in: df** ---see the current amount of free space on your disk drives
- **Type in: free** ---display the amount of free memory
- **Type in: exit** ---closes the terminal

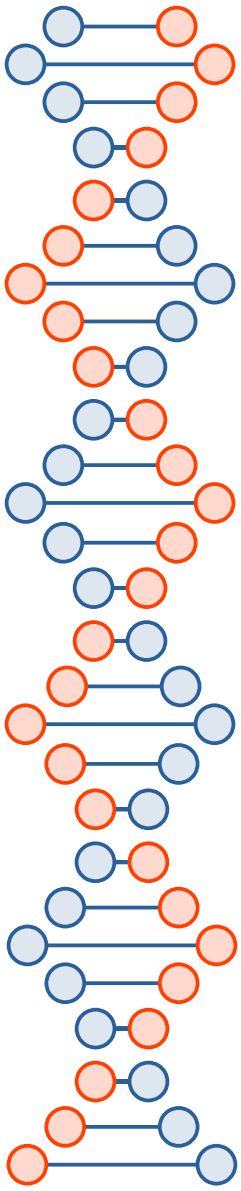


Navigating Directories and Files



Navigation and Exploration

- Linux OS organizes its filkes in a hierarchical directory structure; directories are organized in a tree-like pattern
- First directory in the system is called the root directory
- Storage devices can be mounted at various points in the tree based on what the system administrator wants



Current Working Directory

- Do you want to see where you are in the file system?
  - **Type in: pwd** (print working directory)
    - See the path to your current directory
- Want to see the content of the current working directory?
  - **Type in ls**
    - See a list of everything in the directory

Changing the Current Working Directory

- Use the cd command with either:

  Absolute Pathnames

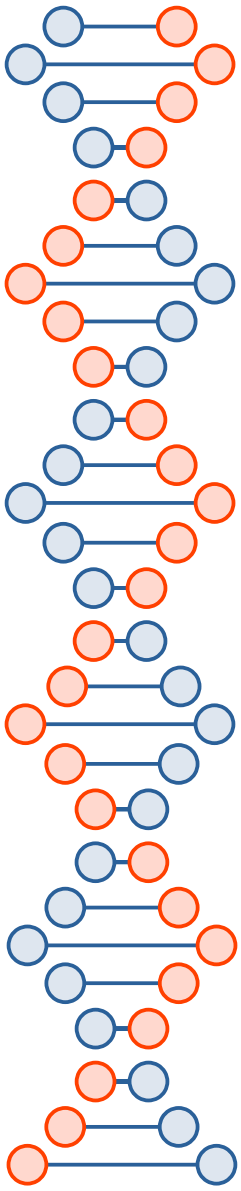
  - Begins with the root directory and follows the tree branch by branch until the path to the desired directory or file is complete
  - For example: /home/haley/Desktop/Comp_Sci_and_Programming_Workshops/Workshop_Directory/dir1

  Relative Pathnames

  - Starts from the current working directory (example: Workshop_Directory)
  - For example: dir1

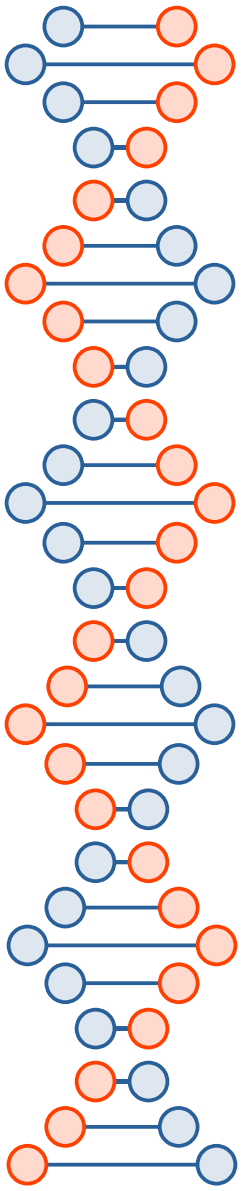
  Going back to a parent directory

- The dot notation can refer to the working directory and the double dot (..) notation refers to its parent
  - Type in: cd ..
  - Returns you to the working directory's parent directory



More about the ls function

- **ls**
  - To get a list of files and directories in the current working directory
- **ls dir1**
  - Can specify the directory
- **ls ~ dir1**
  - Can specify more than one directory, ~ indicates the home directory
- **ls -1**
  - Changes the format of the output to reveal more detail



Options and Arguments

- Commands are often followed by one or more options the modify the behaviour and, by one or more arguments, the items upon which the command acts
- **Type in: ls -1t**
  - 1 option to produce long format output, t option to sort result by the file's modification time
- **Type in: ls -1t --reverse**
  - Reverse reverses the order of the sort

Common ls options

Table 3-1: Common ls Options

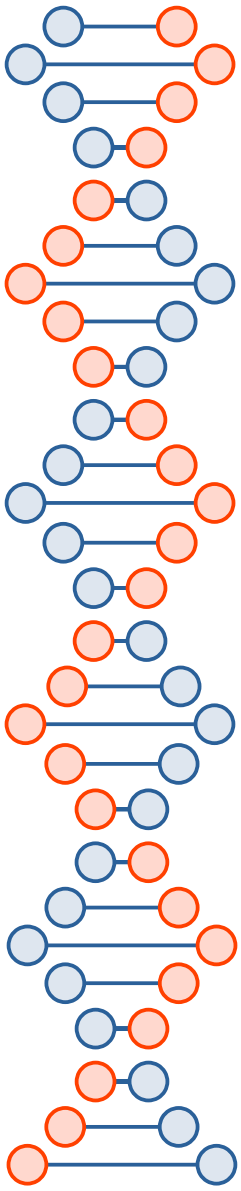
Option	Long option	Description
`-a`	`--all`	List all files, even those with names that begin with a period, which are normally not listed (that is, hidden).
`-A`	`--almost-all`	Like the `-a` option except it does not list `.` (current directory) and `..` (parent directory).
`-d`	`--directory`	Ordinarily, if a directory is specified, `ls` will list the contents of the directory, not the directory itself. Use this option in conjunction with the `-l` option to see details about the directory rather than its contents.
`-F`	`--classify`	This option will append an indicator character to the end of each listed name. For example, it will append a forward slash (`/`) if the name is a directory.
`-h`	`--human-readable`	In long format listings, display file sizes in human-readable format rather than in bytes.
`-l`		Display results in long format.
`-r`	`--reverse`	Display the results in reverse order. Normally, `ls` displays its results in ascending alphabetical order.
`-S`		Sort results by file size.
`-t`		Sort by modification time.

A Longer Look at Long Format

```
-rw-r--r-- 1 root root 3576296 2017-04-03 11:05 Experience ubuntu.ogg
-rw-r--r-- 1 root root 1186219 2017-04-03 11:05 kubuntu-leaflet.png
-rw-r--r-- 1 root root   47584 2017-04-03 11:05 logo-Edubuntu.png
-rw-r--r-- 1 root root   44355 2017-04-03 11:05 logo-Kubuntu.png
-rw-r--r-- 1 root root   34391 2017-04-03 11:05 logo-Ubuntu.png
-rw-r--r-- 1 root root   32059 2017-04-03 11:05 oo-cd-cover.odf
-rw-r--r-- 1 root root  159744 2017-04-03 11:05 oo-derivatives.doc
```

Table 3-2: ls Long Listing Fields

Field	Meaning
`-rw-r--r--`	Access rights to the file. The first character indicates the type of file. Among the different types, a leading dash means a regular file, while a `d` indicates a directory. The next three characters are the access rights for the file's owner, the next three are for members of the file's group, and the final three are for everyone else. Chapter 9 discusses the full meaning of this in more detail.
`1`	File's number of hard links. See "Symbolic Links" on page 21 and "Hard Links" on page 22.
`root`	The username of the file's owner.
`root`	The name of the group that owns the file.
`32059`	Size of the file in bytes.
`2017-04-03 11:05`	Date and time of the file's last modification.
`oo-cd-cover.odf`	Name of the file.



Determining a File's Type with file

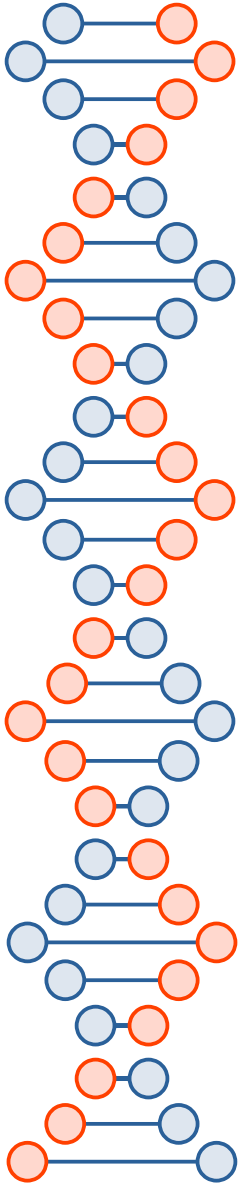
- **Type in: cd dir1**
- **Type in: file test1.txt**
- One common idea in Linux is that "everything is a file"

Viewing File Contents with less

- Less is a program to view text files
- **Type in: less text1.txt**

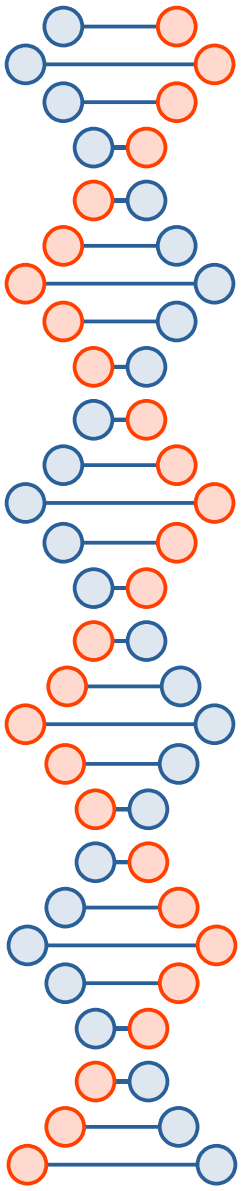
Table 3-3: less Commands

Command	Action
PAGE UP or b	Scroll back one page
PAGE DOWN or space	Scroll forward one page
Up arrow	Scroll up one line
Down arrow	Scroll down one line
G	Move to the end of the text file
1G or g	Move to the beginning of the text file
/characters	Search forward to the next occurrence of characters
n	Search for the next occurrence of the previous search
h	Display help screen
q	Quit less

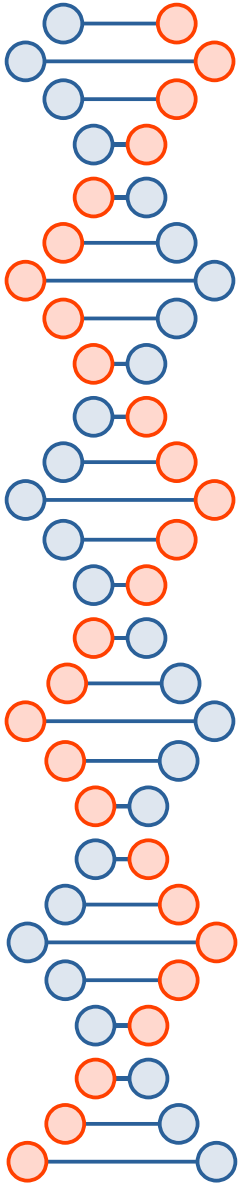


Viewing Text File Content with cat

- **Type in: cat text1.txt**
- Prints text file into terminal



Manipulating Files and Directories



New Commands

- cp: copy files and directories
- mv: move/rename files and directories
- mkdir: create directories
- rm: remove files and directories
- ln: create hard and symbolic links

Wildcards

Table 4-1: Wildcards

Wildcard	Meaning
*	Matches any characters
?	Matches any single character
[*characters*]	Matches any character that is a member of the set *characters*
[!*characters*]	Matches any character that is not a member of the set *characters*
[[:*class*:]]	Matches any character that is a member of the specified *class*

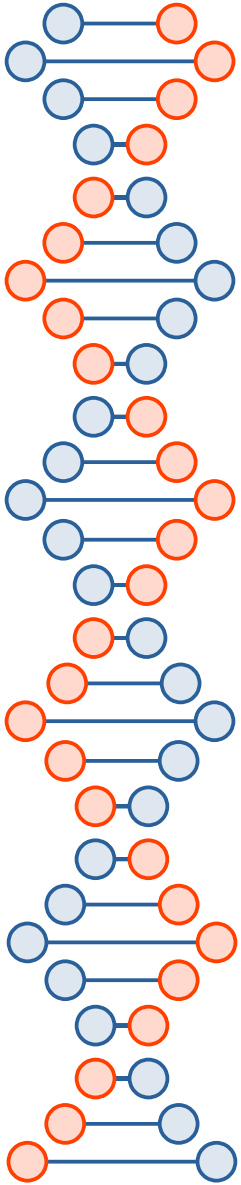
Table 4-2 lists the most commonly used character classes.

Table 4-2: Commonly Used Character Classes

Character class	Meaning
[:alnum:]	Matches any alphanumeric character
[:alpha:]	Matches any alphabetic character
[:digit:]	Matches any numeral
[:lower:]	Matches any lowercase letter
[:upper:]	Matches any uppercase letter

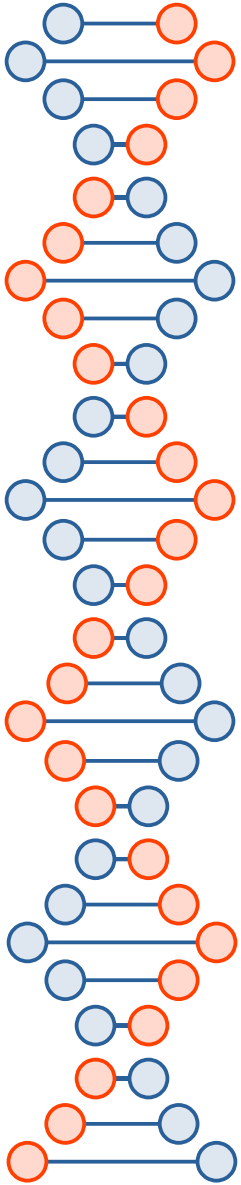
Table 4-3: Wildcard Examples

Pattern	Matches
*	All files
*g**	Any file beginning with *g*
*b**.txt	Any file beginning with *b* followed by any characters and ending with *.txt*
Data???	Any file beginning with *Data* followed by exactly three characters
[*abc*]*	Any file beginning with either an *a*, a *b*, or a *c*
BACKUP.[0-9][0-9][0-9]	Any file beginning with *BACKUP.* followed by exactly three numerals
[[:upper:]]*	Any file beginning with an uppercase letter
[![:digit:]]*	Any file not beginning with a numeral
*[[:lower:]123]	Any file ending with a lowercase letter or the numerals *1*, *2*, or *3*



mkdir –Create Directories

- **Type in: cd ..**
- **Type in: mkdir dir3**
  - You have made a directory called dir3
- **Type in: mkdir chocolate vanilla mint**
  - You have made 3 directories called chocolate, vanilla, and mint



cp – Copy Files and Directories

- **Type in: cd dir1**
- **Type in: cp text1.txt text1_copy.text**
  - Copies text1.txt in the same folder
- **Type in: cd ..**
- **Type in: cp dir1/text1.txt dir2**
  - Copies text1.txt into dir2

cp: Useful Options and Examples

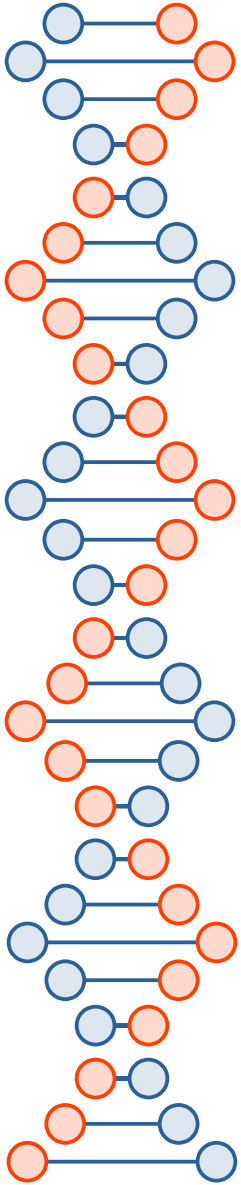
Table 4-4: cp Options

Option	Meaning
`-a, --archive`	Copy the files and directories and all of their attributes, including ownerships and permissions. Normally, copies take on the default attributes of the user performing the copy. We'll take a look at file permissions in Chapter 9.
`-i, --interactive`	Before overwriting an existing file, prompt the user for confirmation. **If this option is not specified, cp will silently (meaning there will be no warning) overwrite files.**
`-r, --recursive`	Recursively copy directories and their contents. This option (or the -a option) is required when copying directories.
`-u, --update`	When copying files from one directory to another, only copy files that either don't exist or are newer than the existing corresponding files in the destination directory. This is useful when copying large numbers of files as it skips files that don't need to be copied.
`-v, --verbose`	Display informative messages as the copy is performed.

Table 4-5 provides some examples of these commonly used options.

Table 4-5: cp Examples

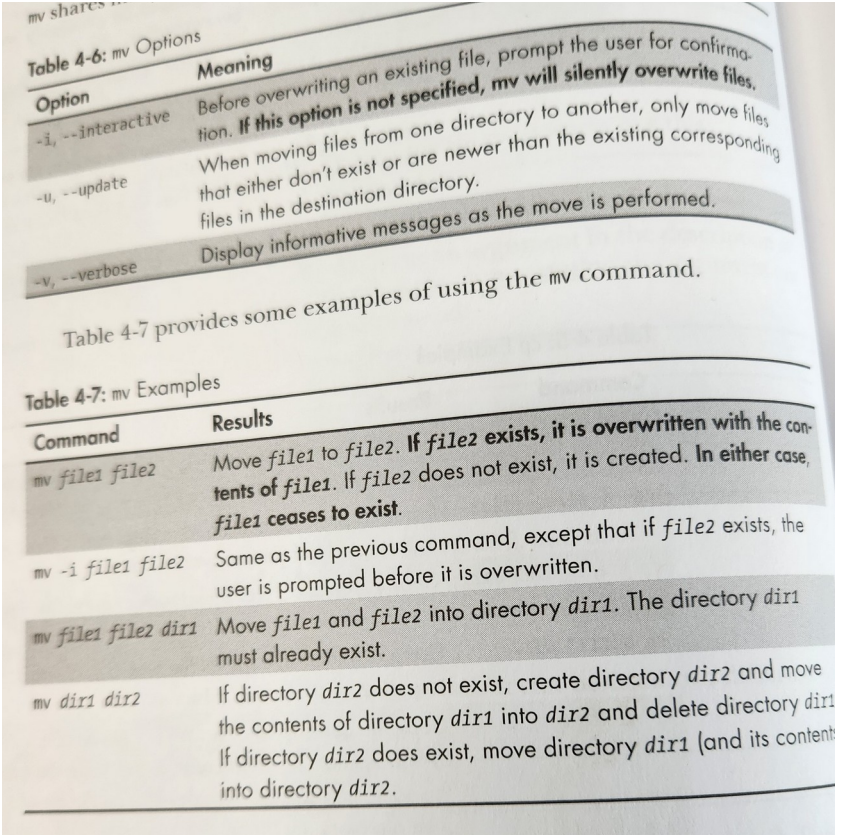
Command	Results
`cp file1 file2`	Copy *file1* to *file2*. If *file2* exists, it is overwritten with the contents of *file1*. If *file2* does not exist, it is created.
`cp -i file1 file2`	Same as previous command, except that if *file2* exists, the user is prompted before it is overwritten.
`cp file1 file2 dir1`	Copy *file1* and *file2* into directory *dir1*. The directory *dir1* must already exist.
`cp dir1/* dir2`	Using a wildcard, copy all the files in *dir1* into *dir2*. The directory *dir2* must already exist.
`cp -r dir1 dir2`	Copy the contents of directory *dir1* to directory *dir2*. If directory *dir2* does not exist, it is created and, after the copy, will contain the same contents as directory *dir1*. If directory *dir2* does exist, then directory *dir1* (and its contents) will be copied into *dir2*.



mv – Move and Rename Files

- **Type in: mv dir1/text1.txt dir1/guinea_pig_fact.txt**
  - Rename text1.txt in dir1
- **Type in mv dir2/text1.txt dir2/text2.txt dir3**
  - Move text1.txt and text2.txt from dir2 to dir3
- **Type in: mv dir2 mint**
  - Move contents of dir2 to mint and delete dir2

mv: Options and Examples



mv shares

Table 4-6: mv Options

Option	Meaning
-i, --interactive	Before overwriting an existing file, prompt the user for confirmation. **If this option is not specified, mv will silently overwrite files.**
-u, --update	When moving files from one directory to another, only move files that either don't exist or are newer than the existing corresponding files in the destination directory.
-v, --verbose	Display informative messages as the move is performed.

Table 4-7 provides some examples of using the mv command.

Table 4-7: mv Examples

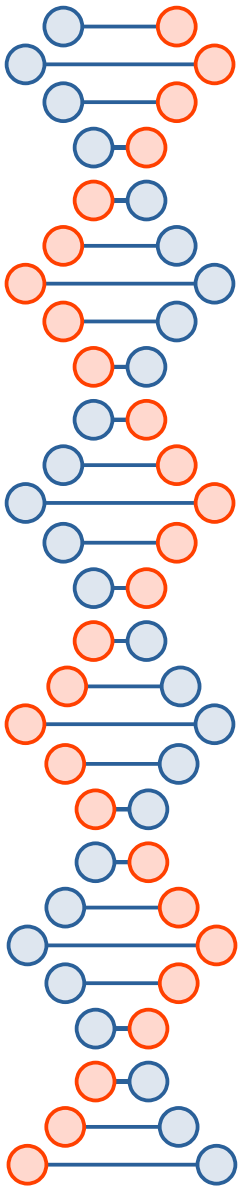
Command	Results
mv *file1 file2*	Move *file1* to *file2*. **If *file2* exists, it is overwritten with the contents of *file1*. If *file2* does not exist, it is created. In either case, *file1* ceases to exist.**
mv -i *file1 file2*	Same as the previous command, except that if *file2* exists, the user is prompted before it is overwritten.
mv *file1 file2 dir1*	Move *file1* and *file2* into directory *dir1*. The directory dir1 must already exist.
mv *dir1 dir2*	If directory *dir2* does not exist, create directory *dir2* and move the contents of directory *dir1* into *dir2* and delete directory dir1. If directory *dir2* does exist, move directory *dir1* (and its contents) into directory *dir2*.

Rm – Removing Files and Directories

- **Type in: rm dir1/guinea_pig_fact.txt**
  - Deletes the file guinea_pig_fact.txt in directory dir1
- **Type in: rm -r dir1**
  - Removes dir1 and all of its content
- Be super careful not to erase everything on your computer using wildcards

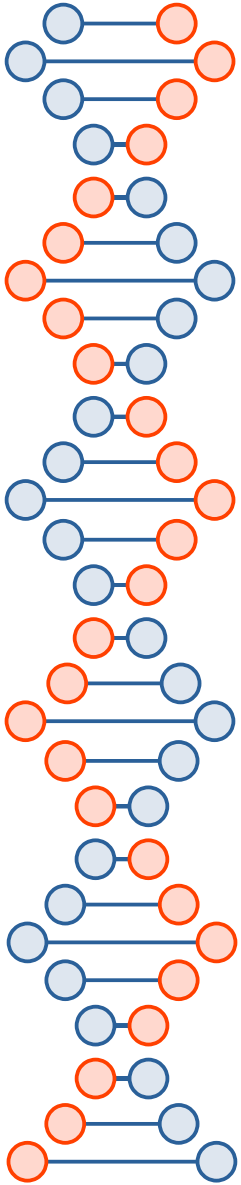
Table 4-8: `rm` Options

Option	Meaning
`-i, --interactive`	Before deleting an existing file, prompt the user for confirmation. **If this option is not specified, `rm` will silently delete files.**
`-r, --recursive`	Recursively delete directories. This means that if a directory being deleted has subdirectories, delete them too. To delete a directory, this option must be specified.
`-f, --force`	Ignore nonexistent files and do not prompt. This overrides the `--interactive` option.
`-v, --verbose`	Display informative messages as the deletion is performed.



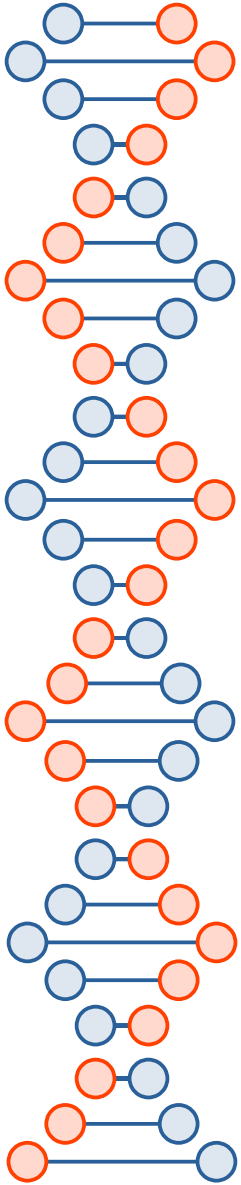
ln –Creating Links

- Hard links
  - Creating an additional directory entry for a file
  - Cannot reference a file outside of its own file system
  - Cannot reference a directory
  - Delete a hard link, keep the file contents until all links are deleted
- Symbolic Links
  - Create a special type of file that contains a text pointer to the referenced file or directory
  - A file pointed to by the symbolic link and the symbolic link itself are largely indistinguishable from one another – edit either, the file is rewritten
  - When deleted only the link is deleted, not the file

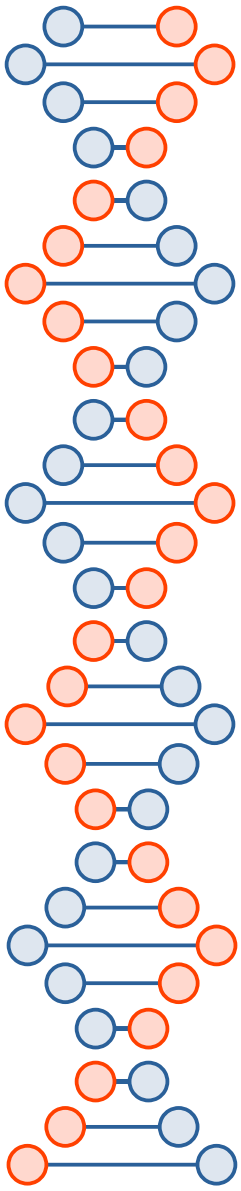


ln – Creating links

- **Type in: ln mint/text1.txt mint/text3.txt**
  - Created a hard link to text1.txt called text3.txt
- **Type in: ln -s mint/text2.txt mint/text2_sym.txt**
  - Created a symbolic link to text2.txt called text2_sym.txt

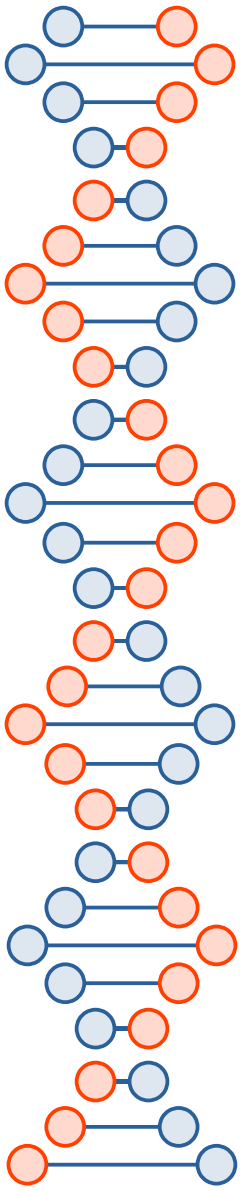


Working with Commands



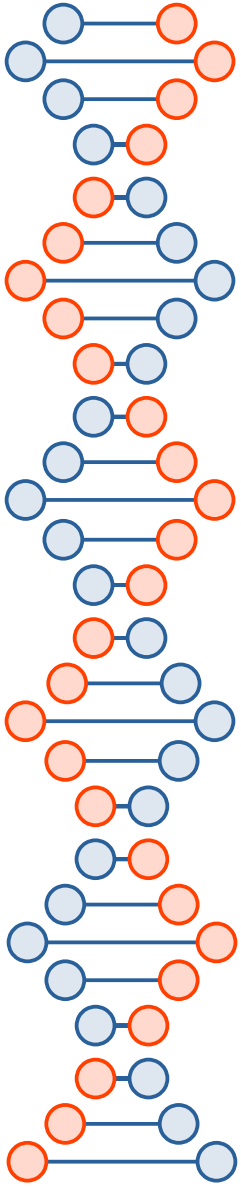
New Commands

- type: indicate how a command name is interpreted
- which: display which executable program with be executed
- help: get help for shell built-ins
- man: display the command’s manual page
- apropos: display a list of appropriate commands
- info: display a command’s info entry
- whatis: display one-line manual page description
- Alias: create an alias for a command



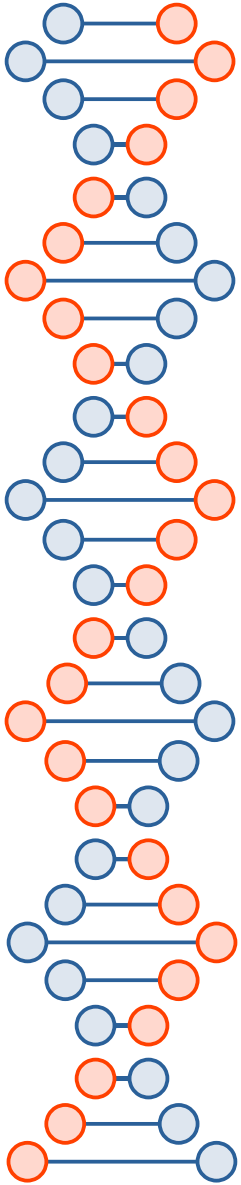
What Exactly are Commands?

- An executable program
- A command built into the shell itself
- A shell function
- An alias



Identifying Commands: type and which

- Type – Display a Command's Type
  - **Type in: type type**
- Which – Display an Executable's Location
  - Sometimes more than on version, especially on HPC clusters
  - **Type in: which ls**
  - Works only on executable programs, not built-ins or aliases that are substitutes for the actual executable program
  - **Type in: which cd**
    - You get an error message.



Getting a Command's Documentation: help and --help

- Help – Get Help for Shell Built-ins
  - **Type in: help cd**
- --help – Display Usage Information
  - Might not always be supported but usually if it isn't it'll give an error message giving the same information
  - **Type in: cd --help**

Getting a Command's Documentation: man

- Displays a Program's Manual Page
- Most executable programs intended for command line use provide a manual or man page – use the man program to view it
- **Type in: man ls**
- Man uses less to display the manual page
- Can search sections for key words

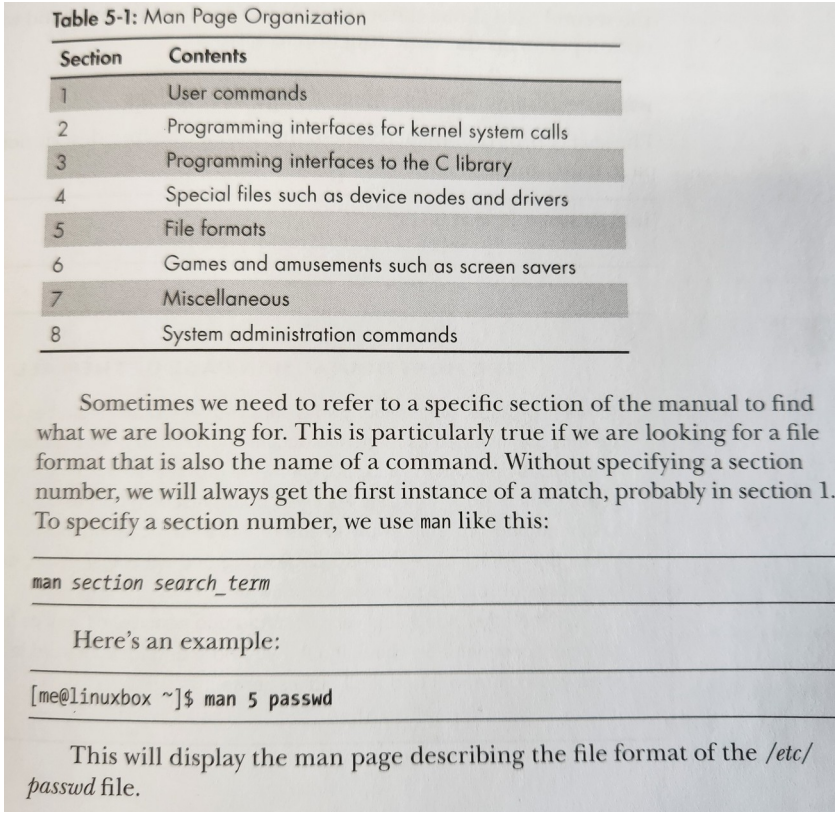


Table 5-1: Man Page Organization

Section	Contents
1	User commands
2	Programming interfaces for kernel system calls
3	Programming interfaces to the C library
4	Special files such as device nodes and drivers
5	File formats
6	Games and amusements such as screen savers
7	Miscellaneous
8	System administration commands

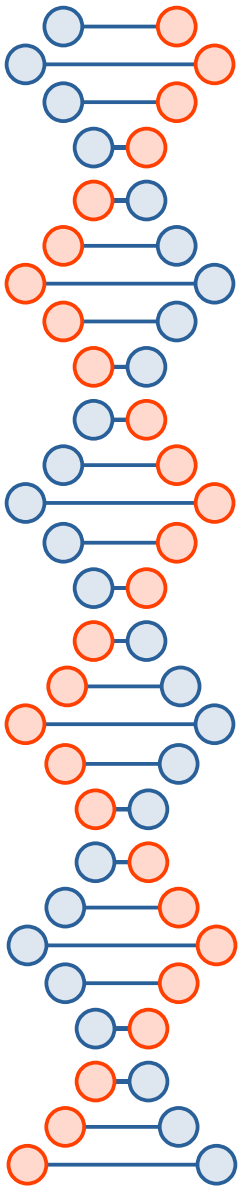
Sometimes we need to refer to a specific section of the manual to find what we are looking for. This is particularly true if we are looking for a file format that is also the name of a command. Without specifying a section number, we will always get the first instance of a match, probably in section 1. To specify a section number, we use man like this:

```
man section search_term
```

Here's an example:

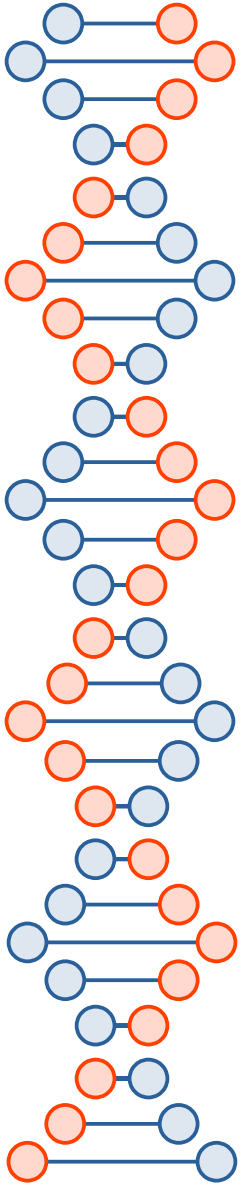
```
[me@linuxbox ~]$ man 5 passwd
```

This will display the man page describing the file format of the */etc/passwd* file.



Getting a Command's Documentation: apropos

- Display appropriate commands
- Search the list of man pages for possible matches based on a search term
- **Type in: apropos partition**
- Each line of output shows man page, section
- man -k does the same thing



Getting a Command's Documentation: whatis

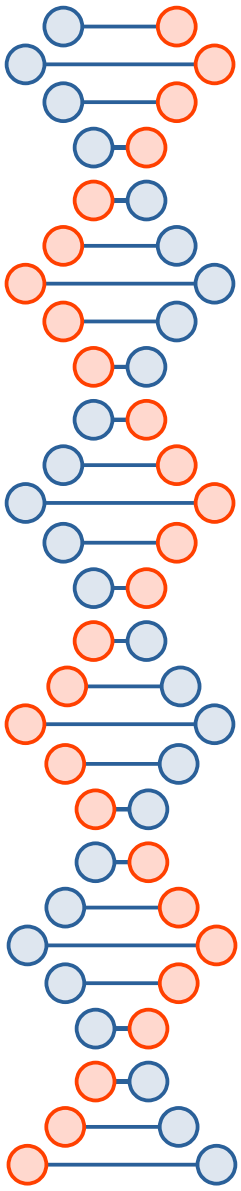
- Displays the name and a one-line description of a man page matching a specified keyword
- **Type in: whatis ls**

Getting a Command's Documentation: info

- Alternative to man pages
- **Type in: info coreutils**
- Info pages have a tree structure with hyperlinks linking them

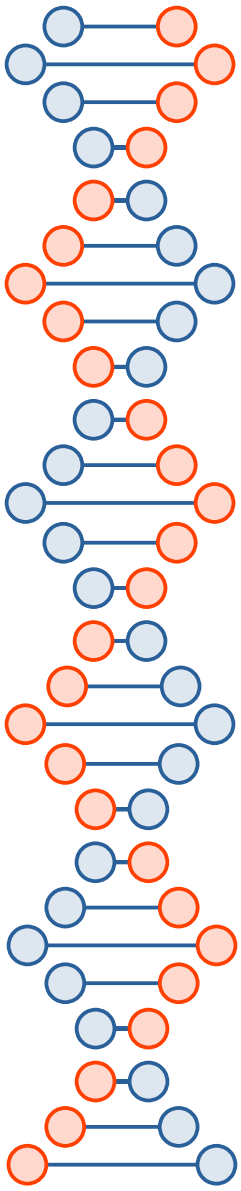
Table 5-2: info Commands

Command	Action
?	Display command help
PAGE UP or BACKSPACE	Display previous page
PAGE DOWN or spacebar	Display next page
n	Next—display the next node
p	Previous—display the previous node
U	Up—display the parent node of the currently displayed node, usually a menu
ENTER	Follow the hyperlink at the cursor location
Q	Quit

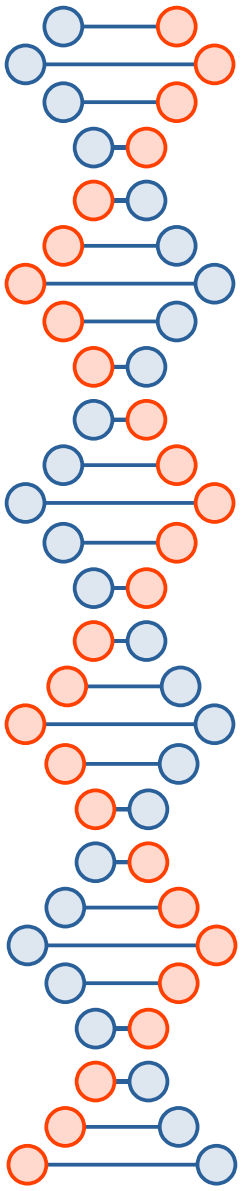


Creating Our Own Commands with alias

- It is possible to put more than one command on a line by separating each command with a semicolon
- **Type in: cd mint;ls;cd -**
  - Change current directory to mint; list everything in it; return to parent directory
- **Type in: alias foo='cd mint;ls;cd -'**
- **Type in: foo**
- **Type in: type foo**
- **Type in: unalias foo**
- An alias only last until your shell session ends

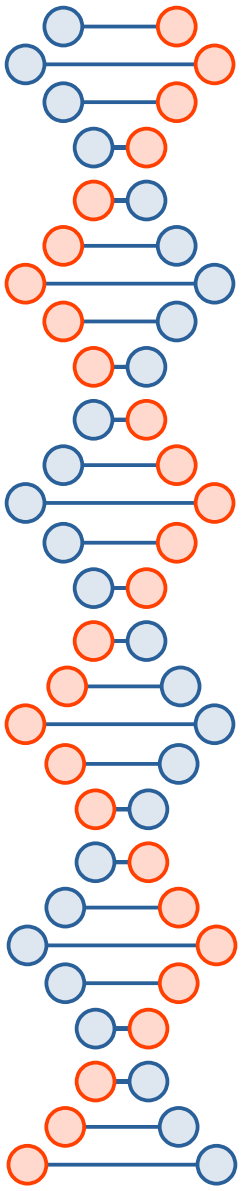


Redirection



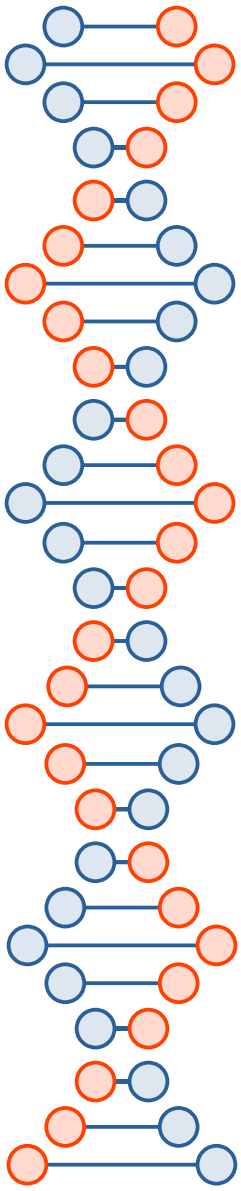
New Commands

- cat: concatenate files
- sort: sort lines of text
- uniq: report or omit repeated lines
- grep: print lines matching a pattern
- wc: print newline, word, and byte counts for each file
- head: output the first part of a file
- tail: output the last part of a file
- tee: read from standard input and write to standard output and files



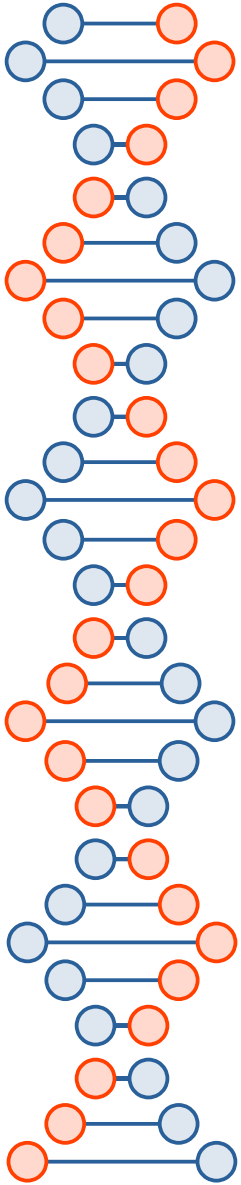
Input and Output

- Two types:
  - The program's results; the data the program is designed to produce
  - Status and error messages that tell us how the program is doing
- Usually displays both of screen; Linux send to stdout and stderr which are linked to the screen and not saved to disk
- Input is usually from stdin which is attached to the keyboard
- I/O redirection allows us to change where output goes and where input comes from



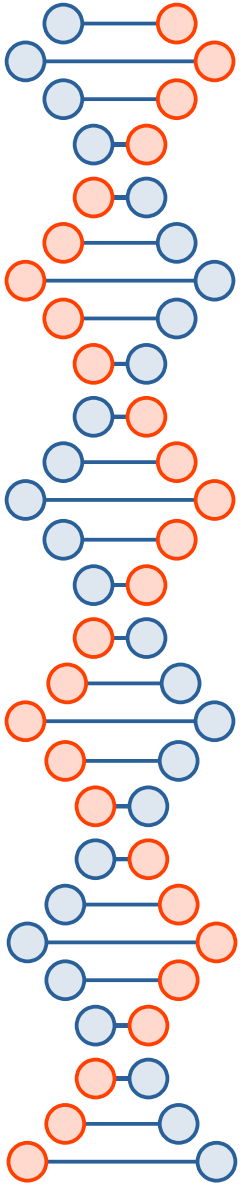
Redirecting Standard Output

- To redefine where standard output goes
- Use the > redirection operator followed by the name of the file
- **Type in: ls -1 mint > mintyfresh.txt**
- **Type in: less mintyfresh.txt**
- **Type in: ls -1 mitn > mintyfresh.txt**
- **Type in: ls mintyfresh.txt**
- **Type in: > mintyfresh.txt**
- **Type in: less mintyfresh.txt**
- **Type in: ls -1 mint >> mintyfresh.txt** (x3) –append rather than overwrite
- **Type in: ls -1 mintyfresh.txt**



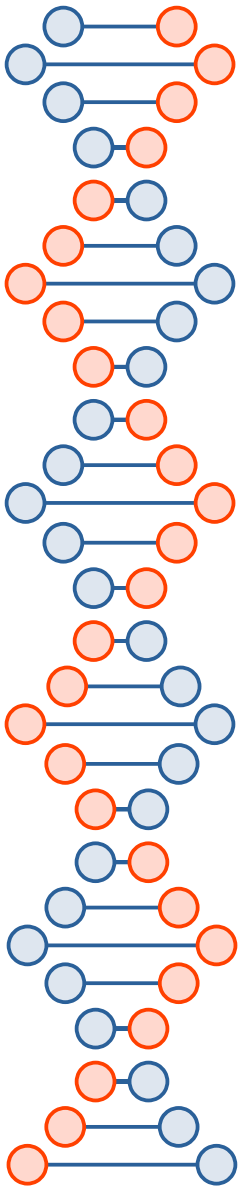
Redirecting Standard Error

- Standard input, output and error is references internally as file descriptors 0, 1, 2, respectively
- **Type in: ls -1 mint 2> mintyerror.txt**



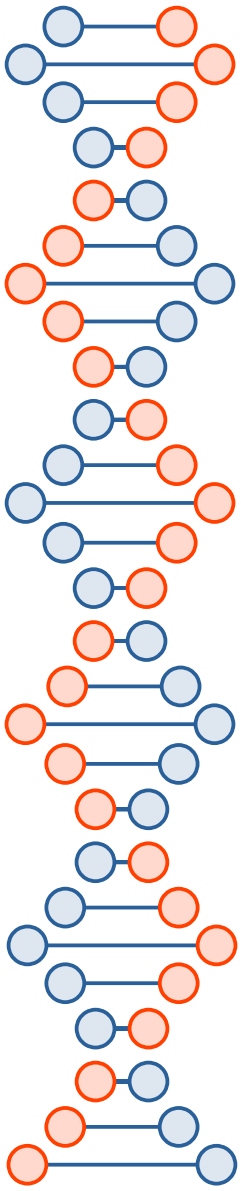
Redirecting Standard Output and Standard Error to One file

- Redirect output to the file then redirect file descriptor 2 to file descriptor 1
  - **Type in: ls -1 mint > mintyfresh.txt 2>&1**
  - **Type in: ls -1 mint &> mintyfresh2.txt**
- Append standard output and error to a single file
  - **Type in: ls -1 mint &>> mintyfresh3.txt**



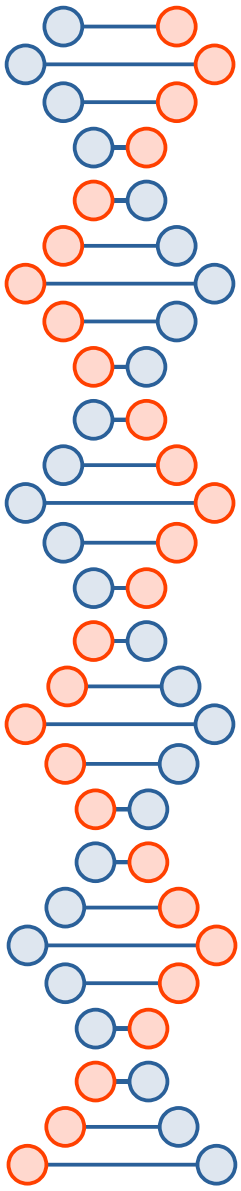
Disposing of Unwanted Output

- **Type in: ls -1 mint 2> /dev/null**
- Redirect special file that is a bit bucket, which accepts input and does nothing with it



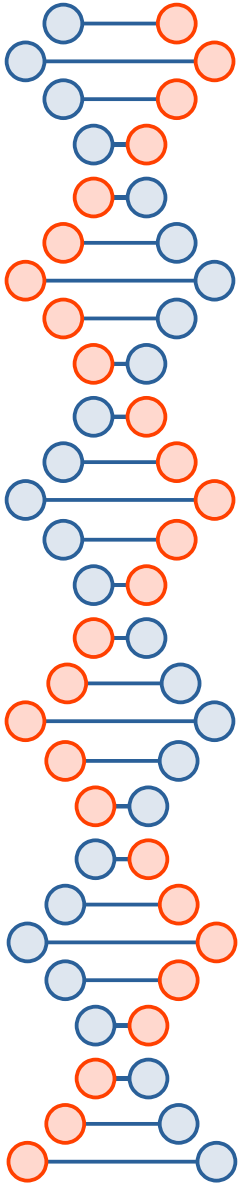
Redirecting Standard Input

- cat: Concatenate Files
- Reads one or more files and copies them to standard output
- **Type in: cat mintyfresh.txt**
- Often used to display short text files
- If we want to join files together:
  - **Type in: cat mintyfresh.txt mintyfresh2.txt mintyfresh3.txt > totalmintyfresh.txt**
  - **Type in: cat mintyfresh* >totalmintyfresh.txt**



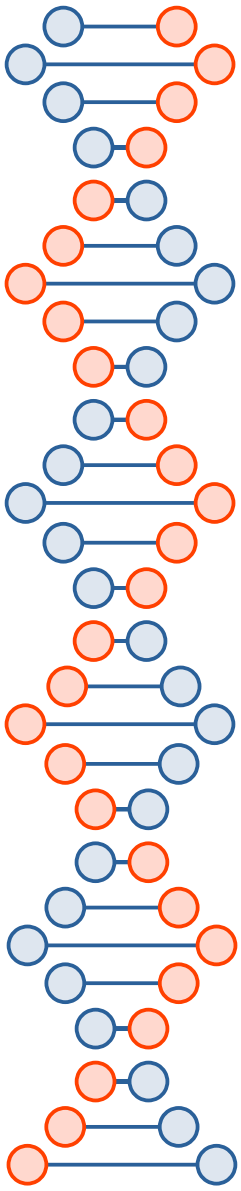
More Standard Input Redirection

- You can write input with cat and your keyboard
  - **Type in: cat**
  - **Type in: Who rule the world? Guinea pigs.**
  - **Type in: CTRL-D –to tell that you've reached the end of the file**
  - **Type in: cat > guinea_pigs_rule.txt**
  - **Type in: Who rule the world? Guinea pigs. (press CTRL-D)**
  - **Type in: cat guinea_pigs_rule.txt**
- The < redirection operator changes the source of input to the file from the keyboard
  - **Type in: cat < guinea_pigs_rule.txt**



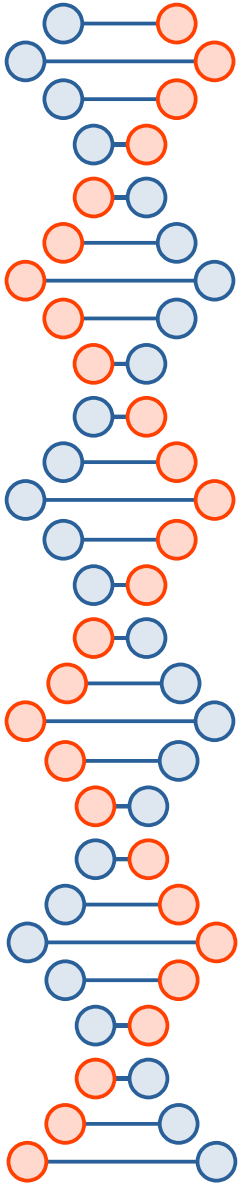
Pipelines

- Utilizes capability of commands to read data from standard input and send to standard output
- Used pipe operator (|), standard output of a command can be piped into the standard input of another
- **Type in: ls -1 mint | less**



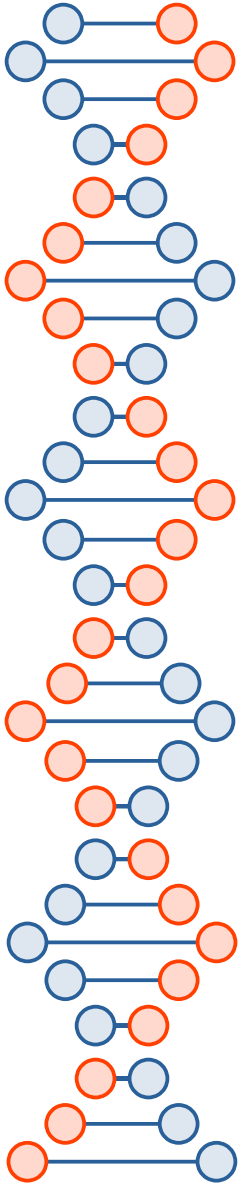
Filtering

- sort: sort one or more lists
  - **Type in: ls -1 chocolate mint | sort | less**
- uniq: report or omit repeated lines, accepts sorted lists
  - **Type in ls -1 chocolate mint | sort | uniq | less**
  - **Type in: ls -1 chocolate mint| sort | uniq -d | less** – to see duplicates



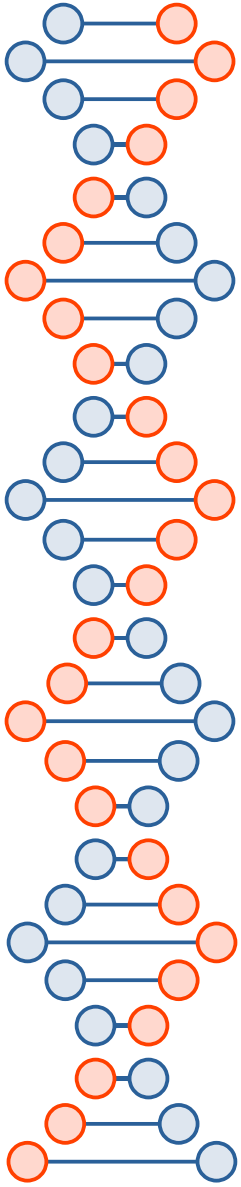
Filtering: wc – Print Line, Word, and Byte Count

- wc (word count): used to display the number of lines, words, and bytes contained in files
- **Type in: wc mintyfresh.txt**
- You can limit the output by selecting the -1 option to report only lines
- **Type in: ls chocolate mint | sort | uniq | wc -1**



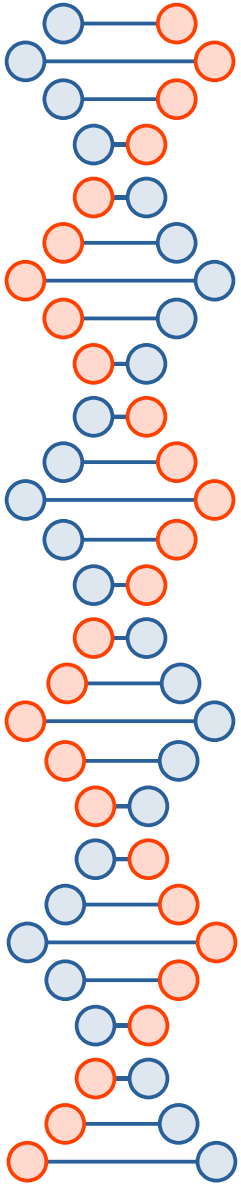
Filtering: grep – Print Lines Matching a Pattern

- Used to find text patterns within files; when it encounters the pattern it prints out the line containing it
- **Type in: ls chocolate mint| sort | uniq | grep text**
- -i gets grep to ignore all cases when performing the search
- -v tells grep to print only thopse lines that do not match the pattern



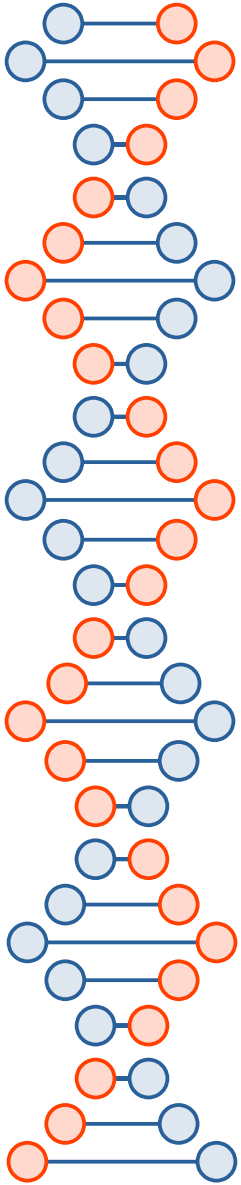
Filtering: head/tail – Print First/Last Part of Files

- **Type in: head -n 5 mintyfresh.txt**
- **Type in: ls mint| tail -n 5**

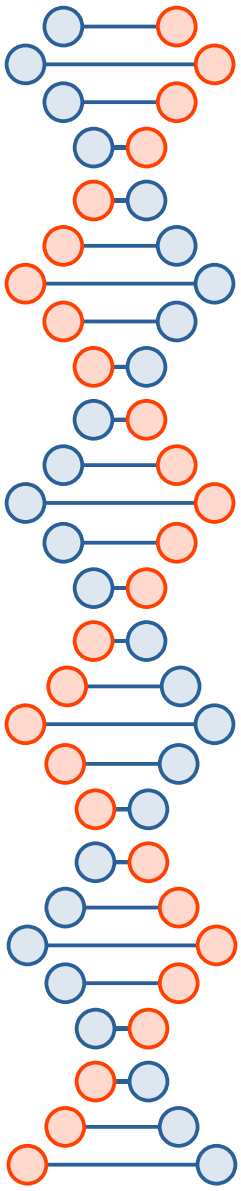


Filtering: tee -- Read from Stdin and Output to Stdout and Files

- **Type in ls mint| tee ls.txt| grep text**



A (Brief) Introduction to Regular Expressions



What are Regular Expressions?

- Symbolic notations used to identify patterns in text
- Regular expressions can change based on the language you are using so we are focusing on the POSIX standard

grep (global regular expression print)

- Searches text files for text matching a specified regular expression and outputs any line containing a match to standard output

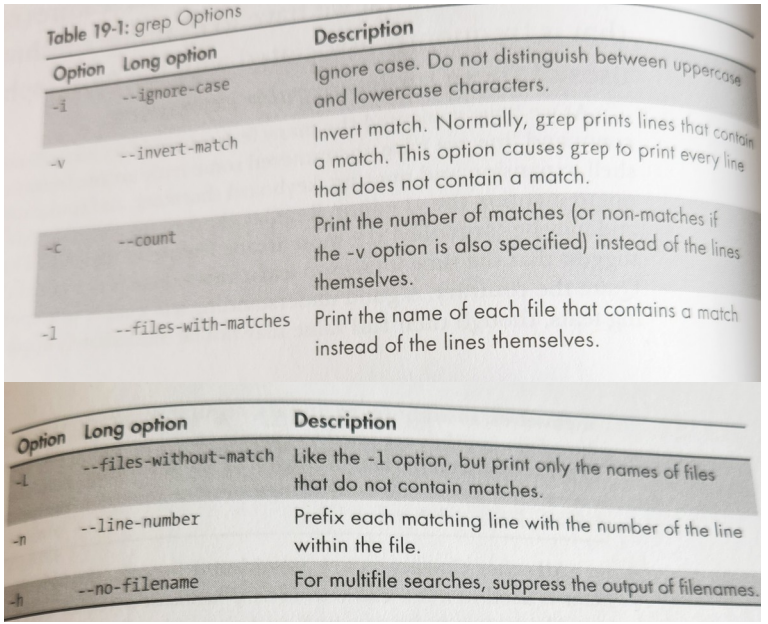
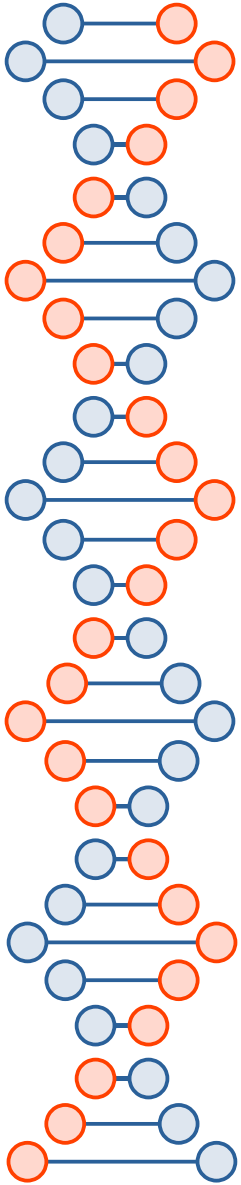


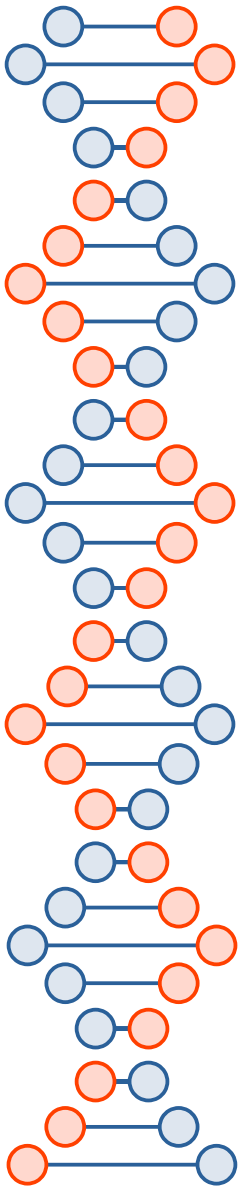
Table 19-1: grep Options

Option	Long option	Description
-i	--ignore-case	Ignore case. Do not distinguish between uppercase and lowercase characters.
-v	--invert-match	Invert match. Normally, grep prints lines that contain a match. This option causes grep to print every line that does not contain a match.
-c	--count	Print the number of matches (or non-matches if the -v option is also specified) instead of the lines themselves.
-l	--files-with-matches	Print the name of each file that contains a match instead of the lines themselves.
-L	--files-without-match	Like the -l option, but print only the names of files that do not contain matches.
-n	--line-number	Prefix each matching line with the number of the line within the file.
-h	--no-filename	For multifile searches, suppress the output of filenames.



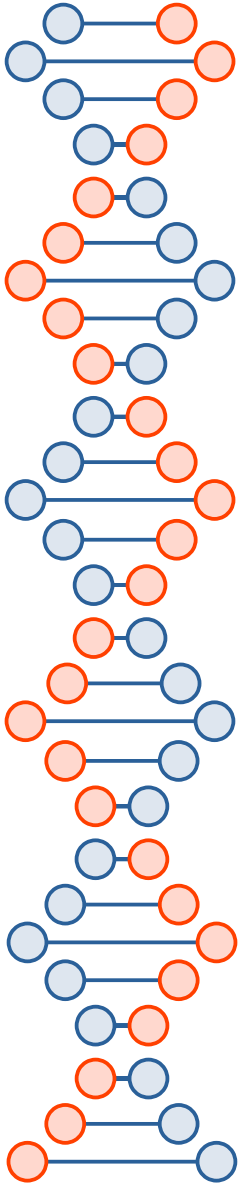
Practice with grep

- **Type in: grep zip random.txt**
- **Type in: grep -1 zip random.txt** – lists files with the string zip
- **Type in: grep -L cat random.txt** –lists files without the string match



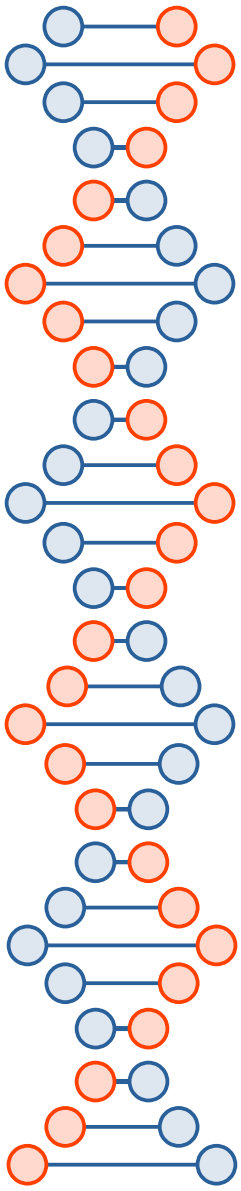
Metacharacters and Literals

- Literals
  - Regular expression zip is taken to mean that a match will occur only if the line in the file contains z,i, and p found in that order, with no characters in between
  - The characters in the string zip are all literal characters, in that they match themselves
- Metacharacters
  - Used to specify more complex matches
  - Including: ^ $. [ ] { } - ? * + ( ) | \



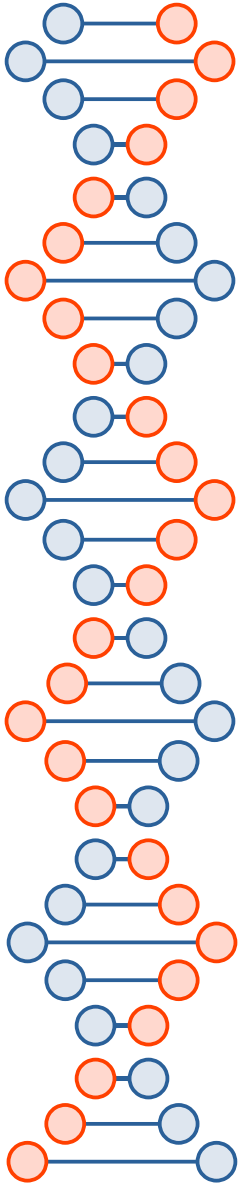
Metacharacters and Literals

- All other characters are literal except metacharacters
- Backslash character is used in a few cases to create metasequences, as well as allowing the metacharacters to be escaped and treated as literals



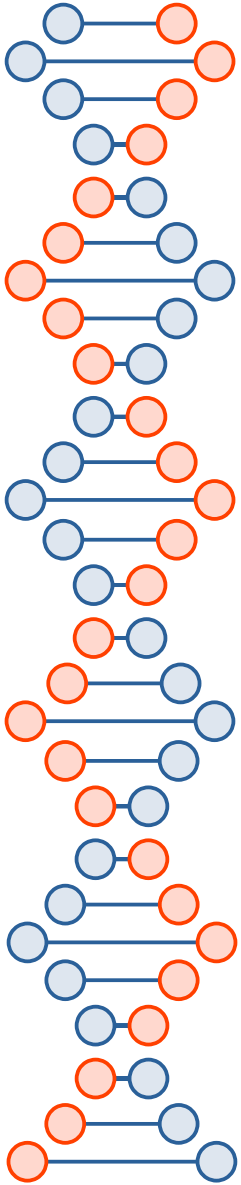
The Any Character

- Metacharacter: the dot (.) which is used to match any character
- **Type in: grep -h '.zip' random.txt**
- This will match a 4 character string with any character before the characters z,i, and p



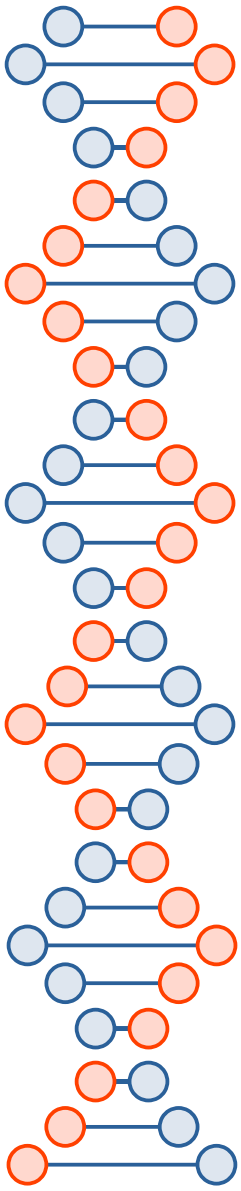
Anchors

- Caret (^) and dollar sign ($) cause the match to occur at the start of the line (^) or the end of the line ($)
- **Type in: grep -h ‘^zip’ random.txt**
- **Type in: grep -h ‘zip$’ random.txt**
- **Type in: grep -h ‘^zip$’ random.txt**
- **Type in: grep -h ‘^$’ random.txt** (matches to blank lines)



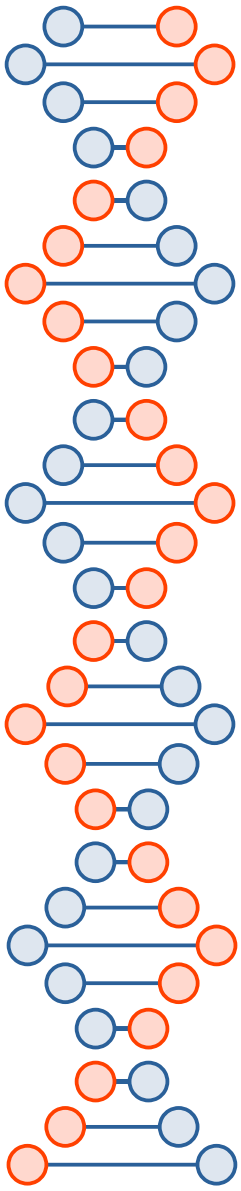
Bracket Expression and Character Classes

- Match a single character from a specified set of characters
- Specify a set of characters to be matched
- **Type in: grep -h ‘[zl]ip’ random.txt**



Negation

- If the first character in a bracket expression is a caret (^), the remaining characters are taken to be a set of characters that must not be present at a given character position.
- **Type in: grep -h ‘[^zl]ip’ random.txt**



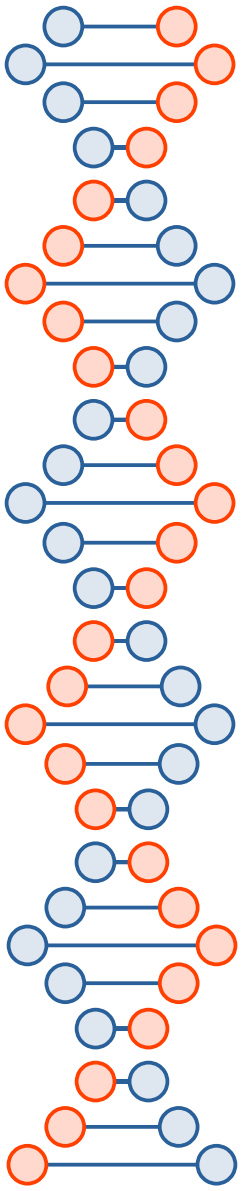
Traditional Character Ranges

- Want to find every line in our list beginning with an uppercase latter?
  - **Type in: grep -h '^[ABCDEFGHIJKLMNOPQRSTUVWXYZ] random.txt**
- You could also use this and get the same thing:
  - **Type in: grep -h '^[A-Z]' random.txt**
- You can include multiple ranges:
  - **Type in: grep -h '^[A-Za-z0-9]' random.txt**
- The location of the dash matters
  - **Type in: grep -h '[-AZ]' random.txt**

POSIX Character Classes

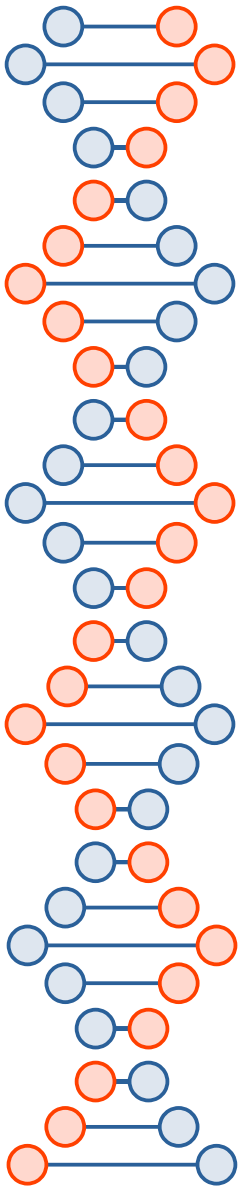
Table 19-2: POSIX Character Classes

Character class	Description
`[:alnum:]`	The alphanumeric characters. In ASCII, equivalent to: `[A-Za-z0-9]`
`[:word:]`	The same as `[:alnum:]`, with the addition of the underscore (_) character.
`[:alpha:]`	The alphabetic characters. In ASCII, equivalent to: `[A-Za-z]`
`[:blank:]`	Includes the space and tab characters.
`[:cntrl:]`	The ASCII control codes. Includes the ASCII characters 0 through 31 and 127.
`[:digit:]`	The numerals 0 through 9.
`[:graph:]`	The visible characters. In ASCII, it includes characters 33 through 126.
`[:lower:]`	The lowercase letters.
`[:punct:]`	The punctuation characters. In ASCII, equivalent to: `[-!"#$%&'()*+,./:;<=>?@[\\\]_`{|}~]`
`[:print:]`	The printable characters. All the characters in `[:graph:]` plus the space character.
`[:space:]`	The whitespace characters including space, tab, carriage return, newline, vertical tab, and form feed. In ASCII, equivalent to: `[ \t\r\n\v\f]`
`[:upper:]`	The uppercase characters.
`[:xdigit:]`	Characters used to express hexadecimal numbers. In ASCII, equivalent to: `[0-9A-Fa-f]`



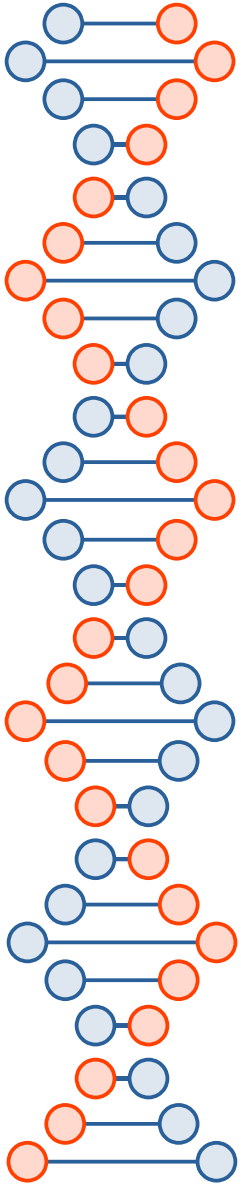
POSIX Basic vs Extended Regular Expression

- The difference? Metacharacters
- BRE has the following metacharacters:
  - ^ $. [ ] *
- ERE adds the following metacharacters:
  - ( ) { } ? + |
- The (,), { and } characters are treated as metacharacters in BRE if they are escaped with a backslash
- In ERE, preceding any metacharacter with a backslash causes it to be treated as a literal
  - ERE can be used with egrep or grep -E



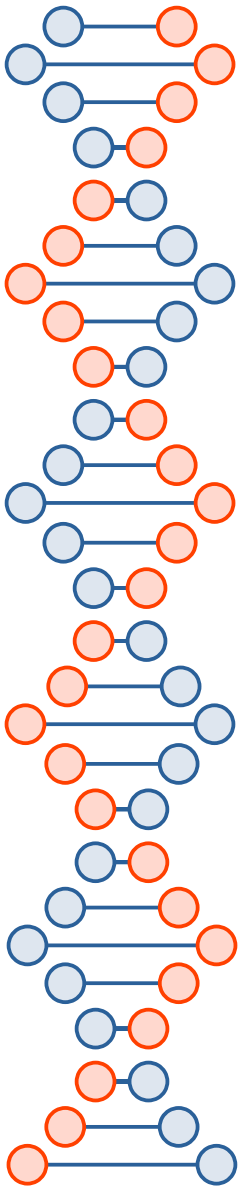
Alternation

- Allows a match to occur from among a set of expressions
- Allows matches from a set of strings or other regular expressions
- **Type in: echo “AAA”| grep AAA**
- **Type in: echo “BBB”| grep AAA**
- **Type in: echo “AAA” | grep -E ‘AAA|BBB’** – means match either string
- **Type in: echo “BBB” | grep -E ‘AAA|BBB’**
- **Type in: echo “CCC” | grep -E ‘AAA|BBB’**



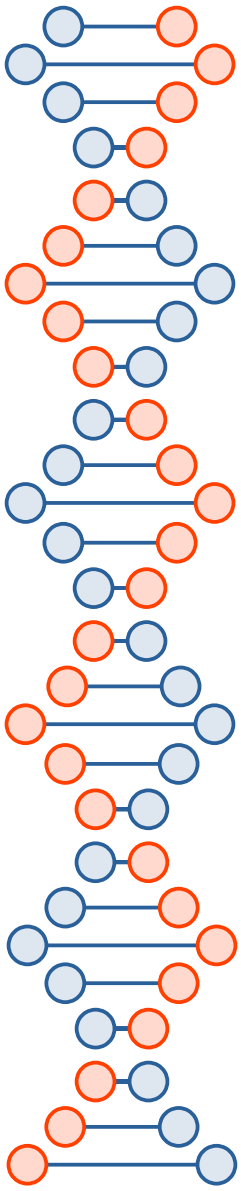
Alternation can be done with other regular expressions

- **Type in: grep -Eh '^(b|l|zip)' random.txt**
  - Match lines in our list that start with b, l, or zip
- **Type in: grep -Eh '^b|l|zip' random.txt**
  - Without the parentheses , the meaning changes to match any line that begins with b or contains l or zip



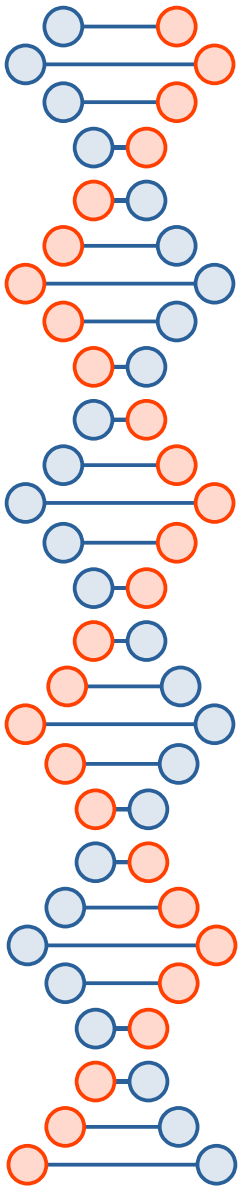
Quantifiers: ? --Match an Element Zero or One Time

- Make the preceding element optional
- Example if we want to accept phone numbers in two forms
  - (nnn) nnn-nnnn
  - nnn nnn-nnnn
  - ^\(?[0-9][0-9][0-9]\)? [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
  - **Type in: echo “(555) 123-4567” | grep -E ^\(?[0-9][0-9][0-9]\)? [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$**
  - **Type in: echo “AAA 123-4567” | grep -E ^\(?[0-9][0-9][0-9]\)? [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$**



Quantifiers: * --Match an Element Zero or More Times

- Used to denote an optional item but the item may occur any number of times, not just once
- See if a string is a sentence: ^[[:upper]][[:upper:] [:lower]]*\.
- **Type in: echo “This works” | grep -E ‘^[[:upper]] [[:upper:][:lower]]*\.’**
- **Type in: echo “this does not” | grep -E ‘^[[:upper]] [[:upper:][:lower]]*\.’**



Quantifiers: + -- Match an Element One or More Times

- Requires at least one instance of the preceding element to cause a match
- This will match only the lines consisting of groups of one or more alphabetic characters separated by single spaces: ^([[:alpha:]]+ ?)+$
- **Type in: echo “a b c” | grep -E ‘^([[:alpha:]]+ ?)+$’**
- **Type in: echo “abc d” | grep -E ‘^([[:alpha:]]+ ?)+$’**
- **Type in: echo “a b 9” | grep -E ‘^([[:alpha:]]+ ?)+$’**

Quantifiers: {} -- Match an Element a Specific Number of Times

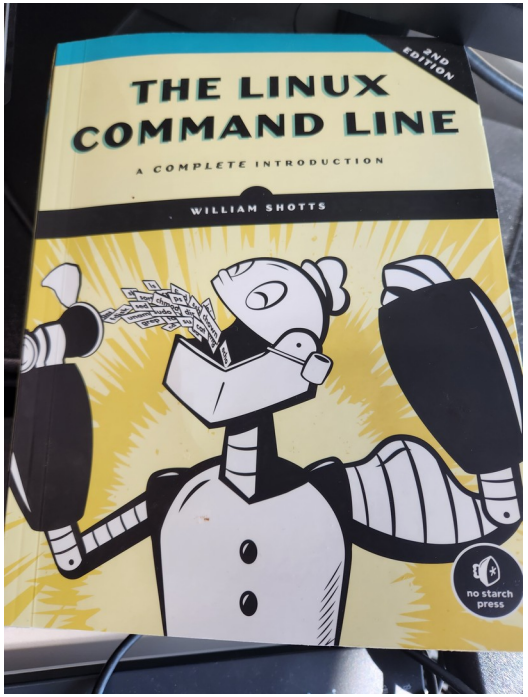
as outlined in Table

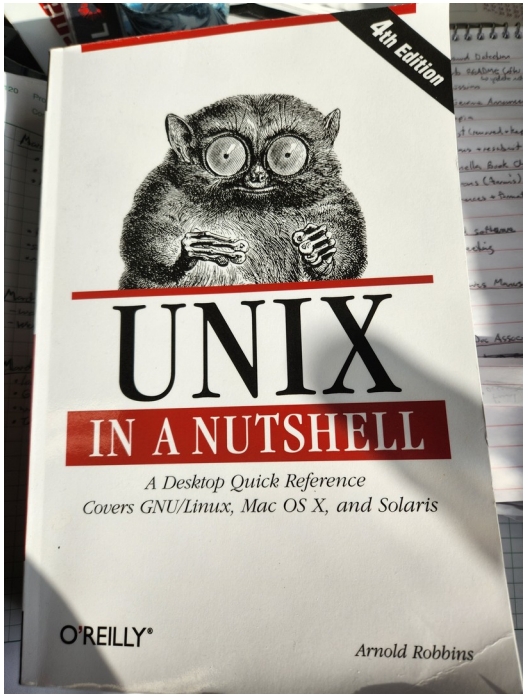
Table 19-3: Specifying the Number of Matches

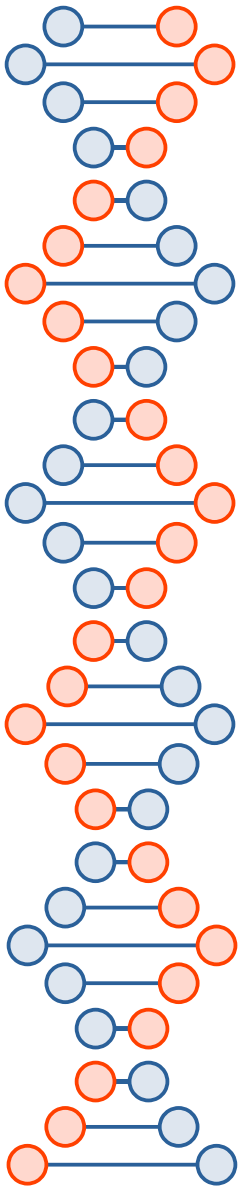
Specifier	Meaning
{n}	Match the preceding element if it occurs exactly *n* times.
{n,m}	Match the preceding element if it occurs at least *n* times but no more than *m* times.
{n,}	Match the preceding element if it occurs *n* or more times.
{,m}	Match the preceding element if it occurs no more than *m* times.

- ^\(?[0-9][0-9][0-9]\)? [0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
- Turns into: ^\(?[0-9]{3}\)? [0-9]{3}-[0-9]{4}$
- **Type in: echo “(555) 123-4567” | grep -E ^\(?[0-9]{3}\)? [0-9]{3}-[0-9]{4}$**
- **Type in: echo “5555 123-4567” | grep -E ^\(?[0-9]{3}\)? [0-9]{3}-[0-9]{4}$**

References


2ND EDITION
THE LINUX COMMAND LINE
A COMPLETE INTRODUCTION
WILLIAM SHOTTS
no starch press


4th Edition
UNIX
IN A NUTSHELL
A Desktop Quick Reference
Covers GNU/Linux, Mac OS X, and Solaris
O'REILLY®
Arnold Robbins

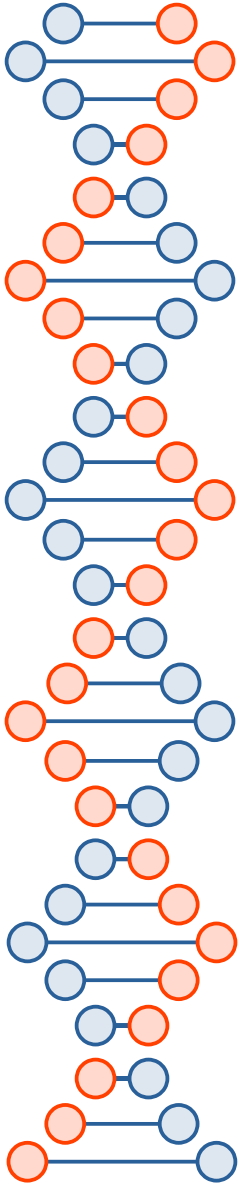


Installing Miniconda

Download the Installer

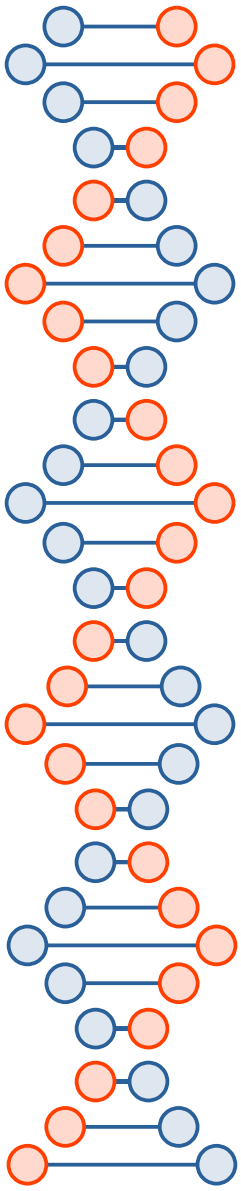
- https://docs.conda.io/en/latest/miniconda.html#linux-installers

Python version	Name	Size	SHA256 hash
Python 3.10	Miniconda3 Linux 64-bit	71.0 MiB	32d73e1bc33fda089d7cd9ef4c1be542616bd8e437d1f77afeeaf7afdb01978
	Miniconda3 Linux-aarch64 64-bit	51.6 MiB	80d6c306b015e1e3b01ea59dc66c676a81fa30279bc2da1f180a7ef7b2191d6
	Miniconda3 Linux-ppc64le 64-bit	52.4 MiB	9ca8077a0af8845fc574a120ef8d68690d7a9862d354a2a4468de5d2196f406
	Miniconda3 Linux-s390x 64-bit	67.6 MiB	0d00a9d34c5fd17d116bf4e7c893b7441a67c7a25416ede90289d87216104a9



Open your terminal

- Open terminal in folder where file downloaded or navigate there
- **Type in: bash Miniconda3-latest-Linux-x86_64.sh**
- Follow the prompts



Next Workshop

- We start Python workshops.
- Expect to be here the whole time.
- Prepare for installing software from miniconda