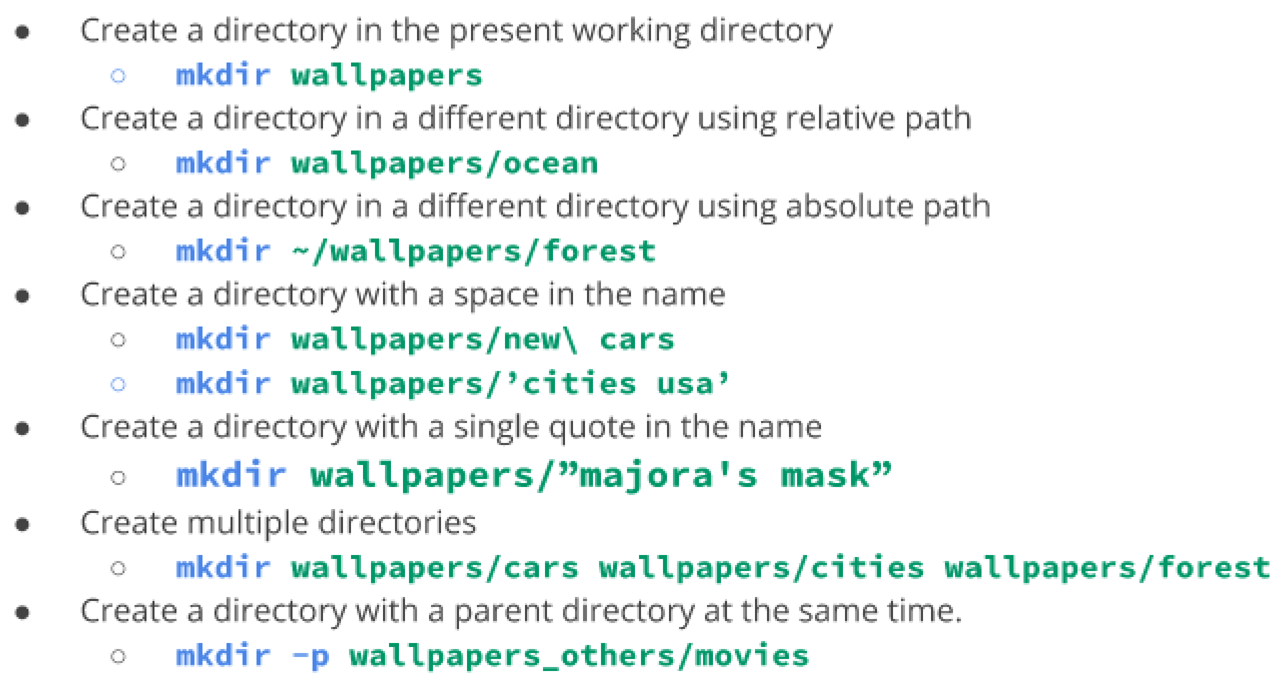
Managing Files and Directories

- Commands are followed by options that modify their behavior
- They are also followed by arguments which are items open which the command acts on

Creating files and directories

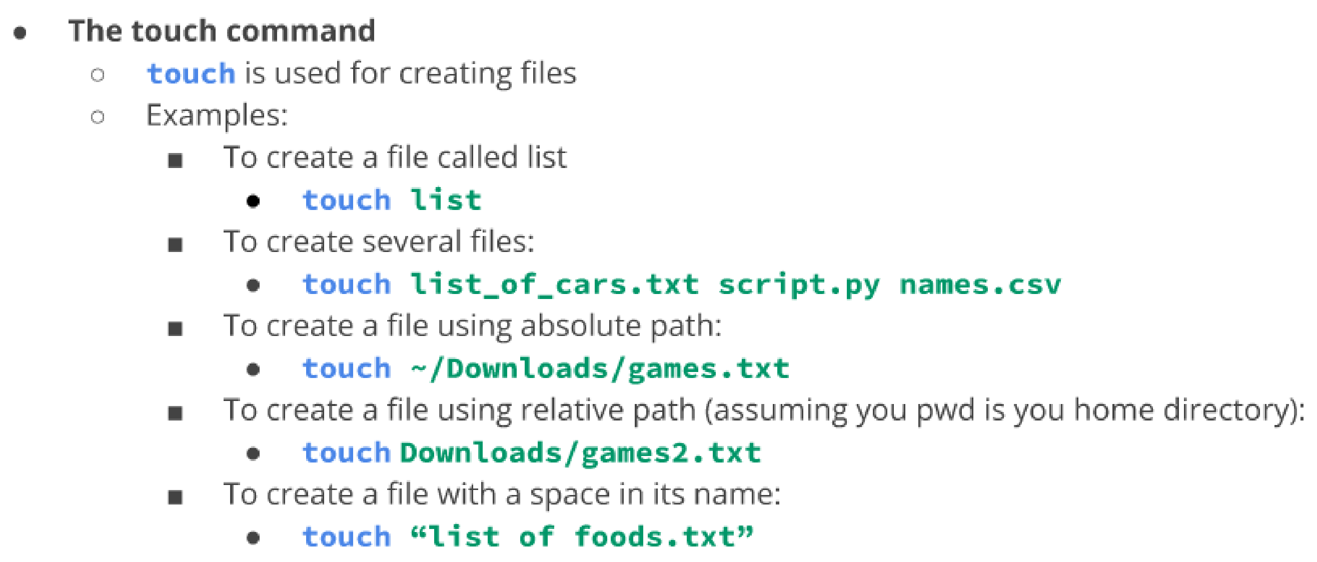
- **mkdir** is used for creating single or multiple directories
- to create one type **mkdir + name of directory**
- separating the name of multiple directories will create them
- it is possible to use absolute path or relative path to create said directories
- it is possible to create a directory with a space in its name by using the () or using quotation marks
- creating a directory that already exist will cause an error

Examples of the mkdir command



- Create a directory in the present working directory
  - `mkdir wallpapers`
- Create a directory in a different directory using relative path
  - `mkdir wallpapers/ocean`
- Create a directory in a different directory using absolute path
  - `mkdir ~/wallpapers/forest`
- Create a directory with a space in the name
  - `mkdir wallpapers/new\ cars`
  - `mkdir wallpapers/’cities usa’`
- Create a directory with a single quote in the name
  - `mkdir wallpapers/”majora's mask”`
- Create multiple directories
  - `mkdir wallpapers/cars wallpapers/cities wallpapers/forest`
- Create a directory with a parent directory at the same time.
  - `mkdir -p wallpapers_others/movies`

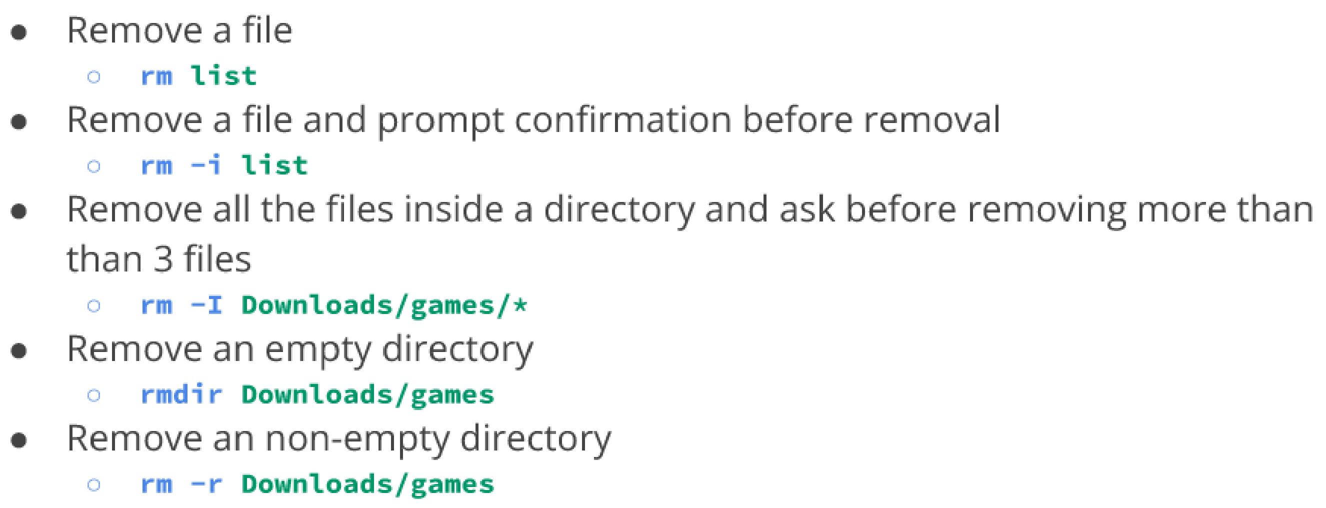
Creating Files



- **The touch command**
  - `touch` is used for creating files
  - Examples:
    - To create a file called list
      - `touch list`
    - To create several files:
      - `touch list_of_cars.txt script.py names.csv`
    - To create a file using absolute path:
      - `touch ~/Downloads/games.txt`
    - To create a file using relative path (assuming you pwd is you home directory):
      - `touch Downloads/games2.txt`
    - To create a file with a space in its name:
      - `touch “list of foods.txt”`

Deleting files and directories

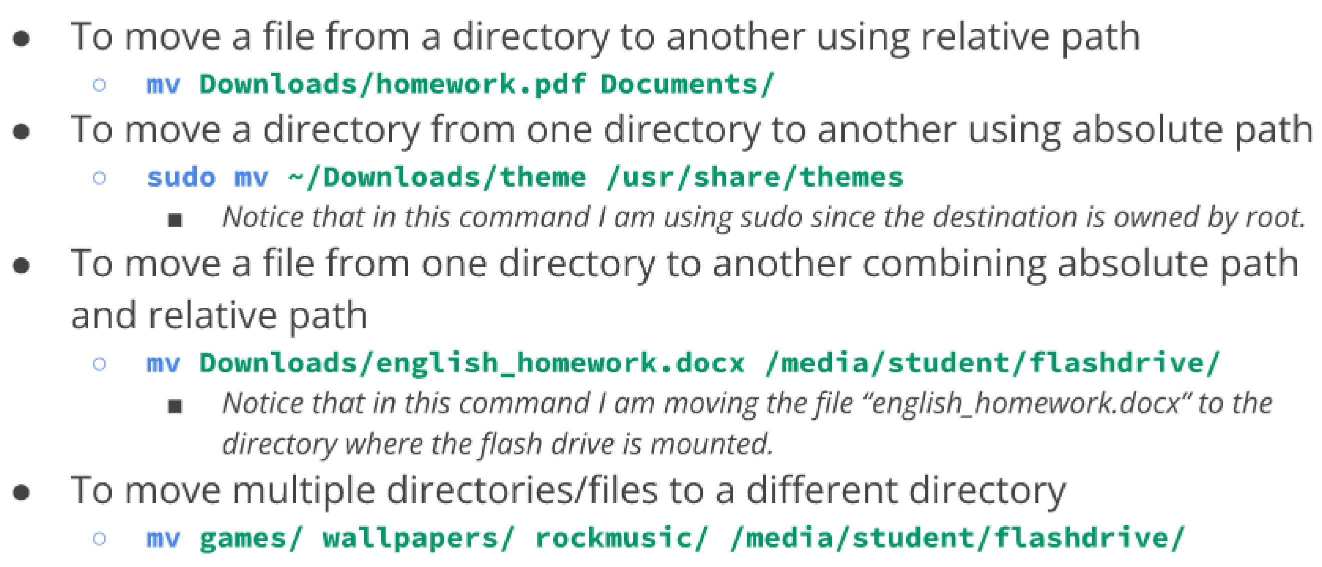
- The rm command
  - removes files
  - does not remove directories by default but using **-r** with it will delete directories
  - use **rmdir** to remove empty directories
  - using the **-r** plus the name of the directories or absolute path



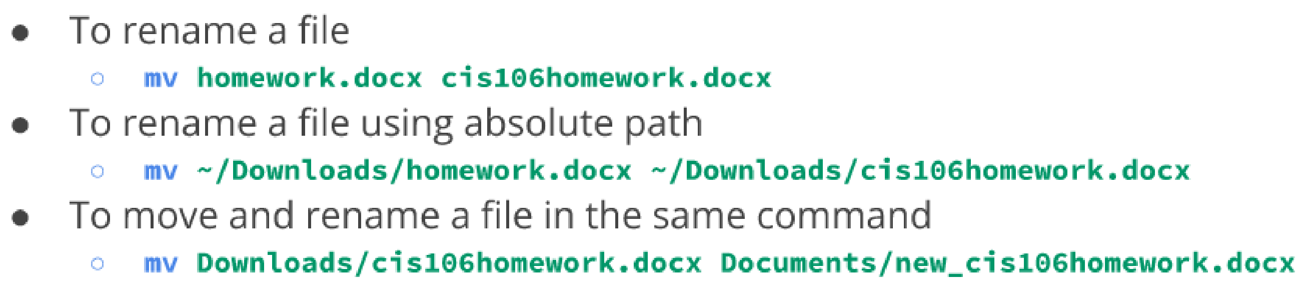
- Remove a file
  - `rm list`
- Remove a file and prompt confirmation before removal
  - `rm -i list`
- Remove all the files inside a directory and ask before removing more than than 3 files
  - `rm -I Downloads/games/*`
- Remove an empty directory
  - `rmdir Downloads/games`
- Remove an non-empty directory
  - `rm -r Downloads/games`

Moving and copying files and directories

- The **mv** command moves and removes directories
- The command is used by using **mv + source + destination**
- To rename a directory the formula is similar **mv + file/directory to rename + new name**
- absolute path and relative path can both be used



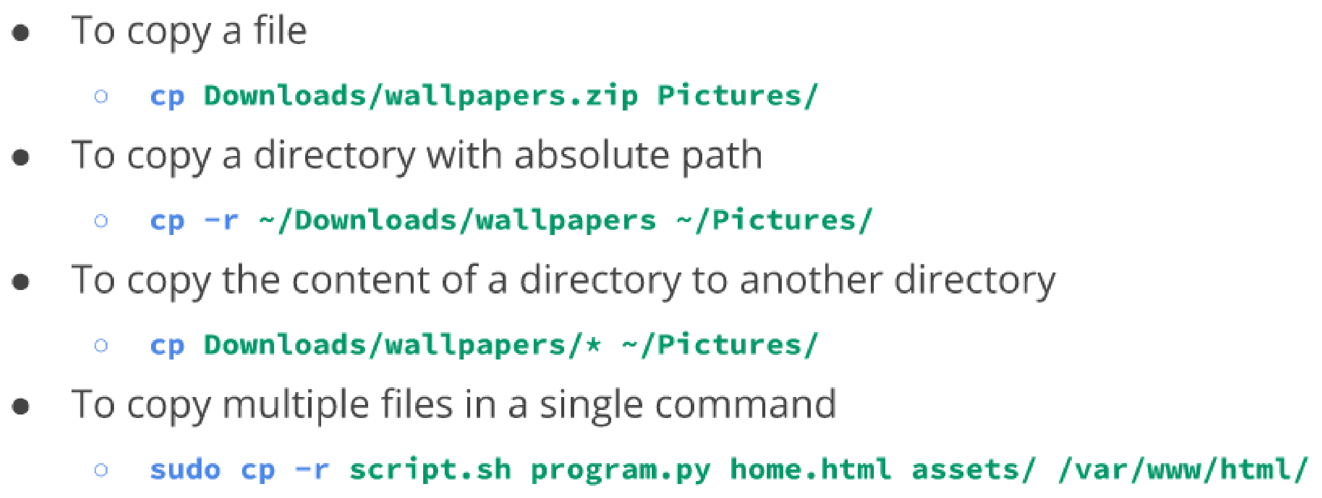
- To move a file from a directory to another using relative path
  - `mv Downloads/homework.pdf Documents/`
- To move a directory from one directory to another using absolute path
  - `sudo mv ~/Downloads/theme /usr/share/themes`
    - *Notice that in this command I am using sudo since the destination is owned by root.*
- To move a file from one directory to another combining absolute path and relative path
  - `mv Downloads/english_homework.docx /media/student/flashdrive/`
    - *Notice that in this command I am moving the file "english_homework.docx" to the directory where the flash drive is mounted.*
- To move multiple directories/files to a different directory
  - `mv games/ wallpapers/ rockmusic/ /media/student/flashdrive/`



- To rename a file
  - `mv homework.docx cis106homework.docx`
- To rename a file using absolute path
  - `mv ~/Downloads/homework.docx ~/Downloads/cis106homework.docx`
- To move and rename a file in the same command
  - `mv Downloads/cis106homework.docx Documents/new_cis106homework.docx`

copying files and directories

- **cp** command copies files/directories from a source ot a destination
- the structure of the command is similar to the **mv** command **cp + files to copy + destination**
- to copy directories the **-r** option must be used



- To copy a file
  - `cp Downloads/wallpapers.zip Pictures/`
- To copy a directory with absolute path
  - `cp -r ~/Downloads/wallpapers ~/Pictures/`
- To copy the content of a directory to another directory
  - `cp Downloads/wallpapers/* ~/Pictures/`
- To copy multiple files in a single command
  - `sudo cp -r script.sh program.py home.html assets/ /var/www/html/`

Working with links

Inodes

- an inode is a data structure that contains all the info about a file except its name and content
- every file has an inode
- every inode is identified by a index number
- the inode table is a database of the location of the data on a partition on linux
- use the **-i** command to view the inodes number
- use the stat command to see the inode data **stat script.sh**

hard links

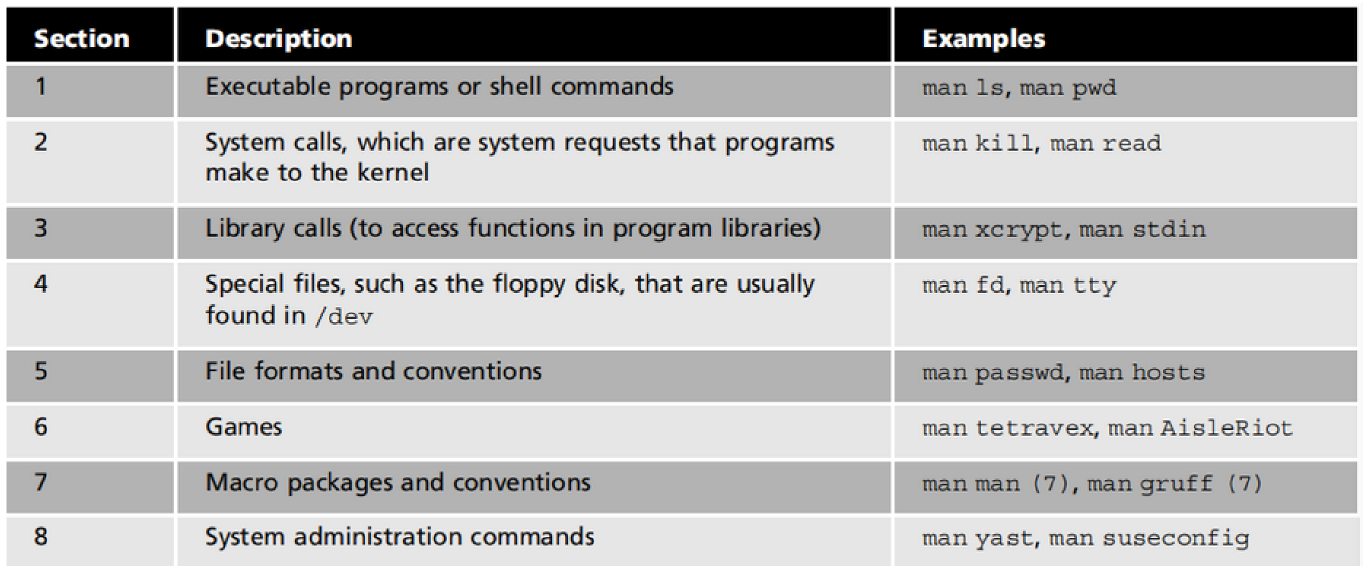
- they are files that point to data on the hard drive
- when a file is created it automatically links to the data in the hard drive
- hard links must be created in the same partition
- data on a hard drive is not eliminated until ever link is deleted
- all hard links are changed once the data on the hard drive is changed
- to create a hard link use **ln file ~/Downloads/fileHL**

soft links

- **symbolic links (soft links)** are files tha point to other files instead of data
- soft links do not share the same inode number as hard links
- when the soft link is modified the target file is also modified
- advantages of soft links is that they can point to files in other partitions
- to create a soft link use **ln -s file fileSL**

getting help

- the **man** command describe commands, executables, system calls, special files and so forth
- to exit the **man** page press **q**

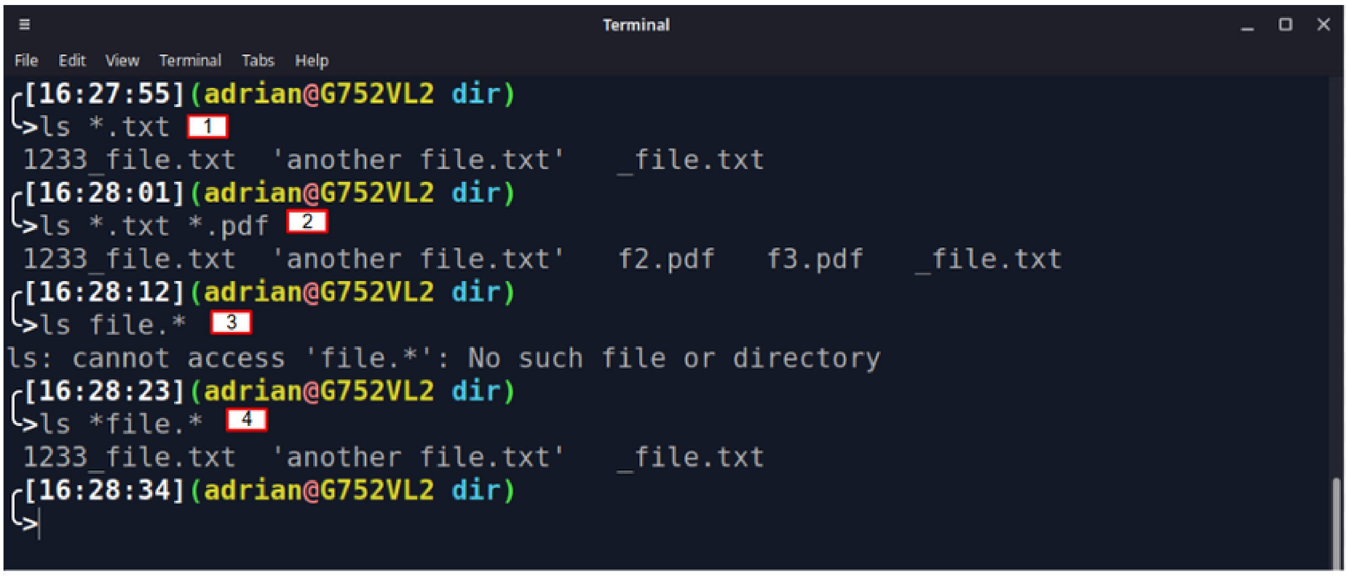


Section	Description	Examples
1	Executable programs or shell commands	`man ls`, `man pwd`
2	System calls, which are system requests that programs make to the kernel	`man kill`, `man read`
3	Library calls (to access functions in program libraries)	`man xcrypt`, `man stdin`
4	Special files, such as the floppy disk, that are usually found in `/dev`	`man fd`, `man tty`
5	File formats and conventions	`man passwd`, `man hosts`
6	Games	`man tetravex`, `man AisleRiot`
7	Macro packages and conventions	`man man (7)`, `man gruff (7)`
8	System administration commands	`man yast`, `man suseconfig`

- Open the man page of the passwd command
  - `man passwd`
- Open a specific man page for the passwd command
  - `man 5 passwd`
- Show the man page section of the passwd command
  - `man -f passwd`
- Show all the available pages of a command
  - `man -a passwd`
- Searches for a man page for a given word or regular expression or phrase.
  - `man -k file`

Using wildcards

- it represents letters and characters used ot specify a filename for searches
- wildcards are officially called metacharacter wildcards
- the main wildcard is a star, or asterisk
- a star alone matches anything and nothing and matches any number of characters
- an example is *_ls .txt_ will match all files that end in .txt regardless of size



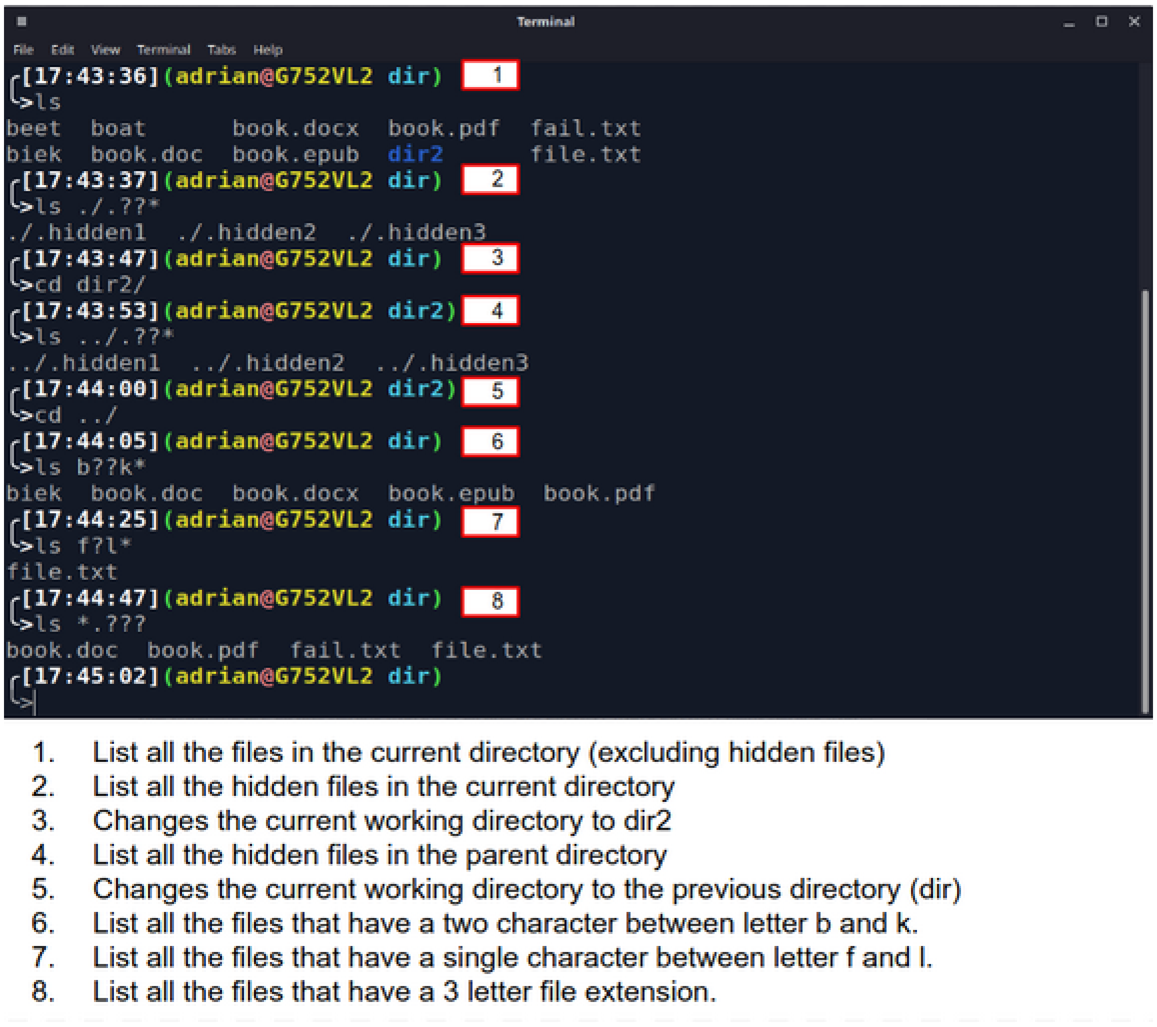
```
┌[16:27:55](adrian@G752VL2 dir)
└>ls *.txt
 1233_file.txt  'another file.txt'   _file.txt
┌[16:28:01](adrian@G752VL2 dir)
└>ls *.txt *.pdf
 1233_file.txt  'another file.txt'   f2.pdf   f3.pdf   _file.txt
┌[16:28:12](adrian@G752VL2 dir)
└>ls file.*
ls: cannot access 'file.*': No such file or directory
┌[16:28:23](adrian@G752VL2 dir)
└>ls *file.*
 1233_file.txt  'another file.txt'   _file.txt
┌[16:28:34](adrian@G752VL2 dir)
└>
```

1. ls *.txt lists all the files that end in .txt
2. ls *.txt *.pdf list all the files that end in .txt and .pdf
3. ls file.* lists all the files that start with the string “file.” regardless of their file extension. In this example, there were no files in the directory that matched this criteria.
4. ls *file.* list all the files that have any letter before the string “file.” and after as well.

the ? wildcard

- is is a metacharacter that matches exactly one character
- proves very useful when working with hidden fiels
- if you want to list all hidden files use **ls .??*.** and it will match all files that start with . or .. and have a character after

- Isn't this the same as using the * character on its own? **NO!**
    - The problem with dot files and wildcard expressions is that the **current directory** and the **parent directory** have a name.
    - The current directory is called/represented with a single dot (.) and the parent directory is called/represented with two dots (..)
        - *Remember cd ../../../ that's what I am talking about!*
    - If you use a wildcard expression such as `.*` to list all files that start with a dot. The shell will also match . and ..
    - To go around this problem you can use `./.*` to match all the files in the current directory with a file name starting with a dot.
    - You can also match all the files that start with a . in the parent directory using `../.*`

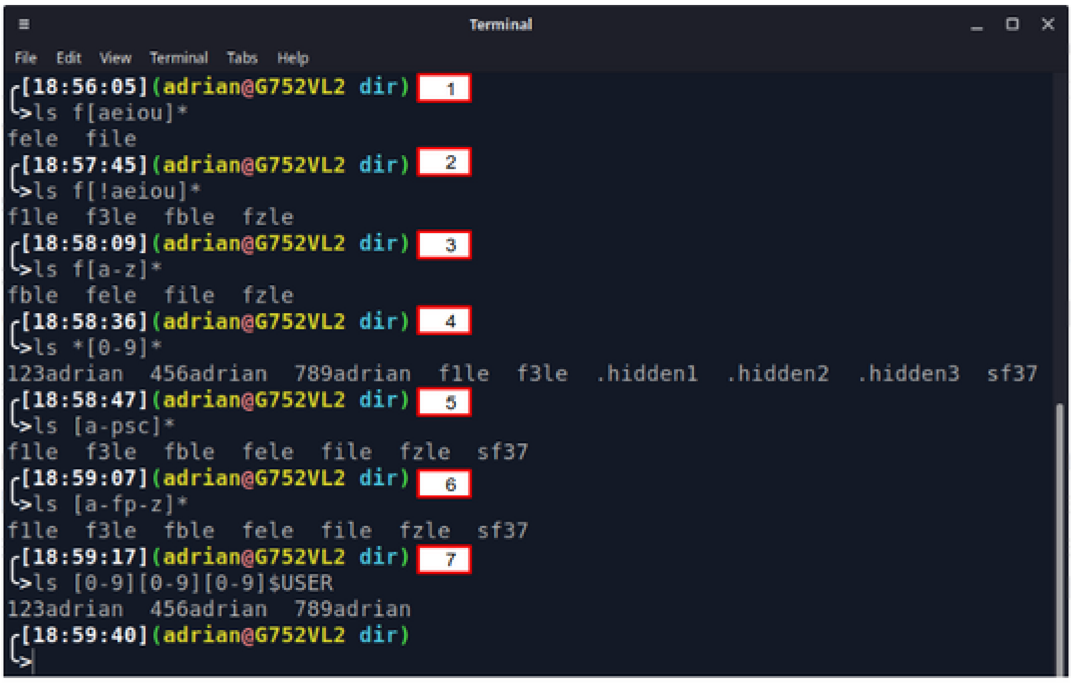


```
┌[17:43:36](adrian@G752VL2 dir)   1
└>ls
beet  boat      book.docx  book.pdf  fail.txt
biek  book.doc  book.epub  dir2      file.txt
┌[17:43:37](adrian@G752VL2 dir)   2
└>ls ./.??*
./.hidden1  ./.hidden2  ./.hidden3
┌[17:43:47](adrian@G752VL2 dir)   3
└>cd dir2/
┌[17:43:53](adrian@G752VL2 dir2)  4
└>ls ../.??*
../.hidden1  ../.hidden2  ../.hidden3
┌[17:44:00](adrian@G752VL2 dir2)  5
└>cd ../
┌[17:44:05](adrian@G752VL2 dir)   6
└>ls b??k*
biek  book.doc  book.docx  book.epub  book.pdf
┌[17:44:25](adrian@G752VL2 dir)   7
└>ls f?l*
file.txt
┌[17:44:47](adrian@G752VL2 dir)   8
└>ls *.???
book.doc  book.pdf  fail.txt  file.txt
┌[17:45:02](adrian@G752VL2 dir)
└>
```

1. List all the files in the current directory (excluding hidden files)
2. List all the hidden files in the current directory
3. Changes the current working directory to dir2
4. List all the hidden files in the parent directory
5. Changes the current working directory to the previous directory (dir)
6. List all the files that have a two character between letter b and k.
7. List all the files that have a single character between letter f and l.
8. List all the files that have a 3 letter file extension.

the [] wildcard

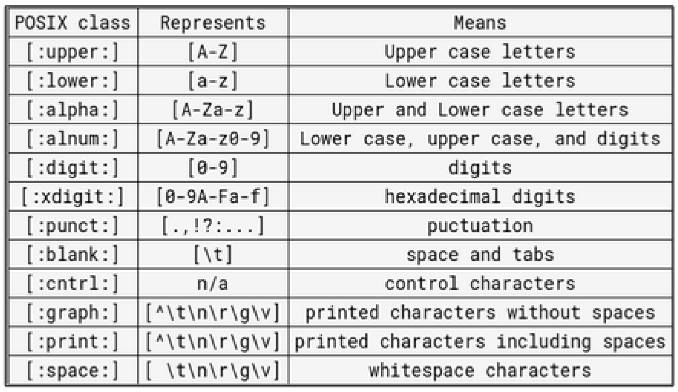
- The brackets wildcard match a single character in a range.
- The brackets wildcard use the exclamation mark to reverse the match. For example, match everything except vowels [!aeiou] or any character except numbers [!0-9]
- **Examples:**
  - To match all files that have a vowel after letter f:
    - `ls f[aeiou]*`
  - To match all files that do not have a vowel after letter f:
    - `ls f[!aeiou]*`
  - To match all files that have a range of letters after f:
    - `ls f[a-z]*`
  - To match all files whose name has at least one number:
    - `ls *[0-9]*`
  - To match all the files whose name does not have a number in their file name:
    - `ls *[!0-9].*`
  - To match all files whose name begins with a letter from a-p or start with letters s or c:
    - `ls [a-psc]*`
  - To match all files whose name begins with any of these two sets of characters: letters from a-f or p-z:
    - `ls [a-fp-z]*`
  - To match all files whose name begins with any 3 combination of numbers and the current user's username:
    - `ls [0-9][0-9][0-9]$USER`



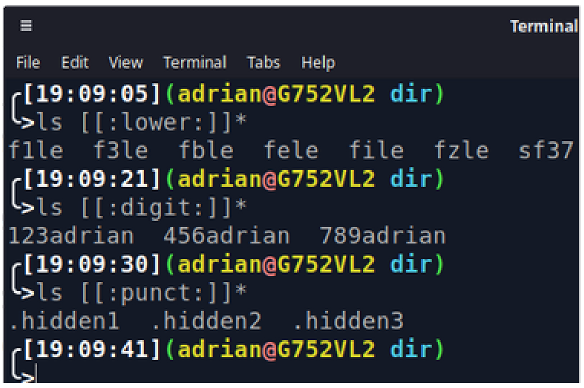
```
┌[18:56:05](adrian@G752VL2 dir) 1
└>ls f[aeiou]*
fele  file
┌[18:57:45](adrian@G752VL2 dir) 2
└>ls f[!aeiou]*
f1le  f3le  fble  fzle
┌[18:58:09](adrian@G752VL2 dir) 3
└>ls f[a-z]*
fble  fele  file  fzle
┌[18:58:36](adrian@G752VL2 dir) 4
└>ls *[0-9]*
123adrian  456adrian  789adrian  f1le  f3le  .hidden1  .hidden2  .hidden3  sf37
┌[18:58:47](adrian@G752VL2 dir) 5
└>ls [a-psc]*
f1le  f3le  fble  fele  file  fzle  sf37
┌[18:59:07](adrian@G752VL2 dir) 6
└>ls [a-fp-z]*
f1le  f3le  fble  fele  file  fzle  sf37
┌[18:59:17](adrian@G752VL2 dir) 7
└>ls [0-9][0-9][0-9]$USER
123adrian  456adrian  789adrian
┌[18:59:40](adrian@G752VL2 dir)
└>
```

1. To match all files that have a vowel after letter f
2. To match all files that do not have a vowel after letter f
3. To match all files that have a range of letters after f
4. To match all files whose name has at least one number
5. To match all files whose name begins with a letter from a-p or start with letters s or c
6. To match all files whose name begins with any of these two sets of characters: letters from a-f or p-z
7. To match all files whose name begins with any 3 combination of numbers and the current user's username

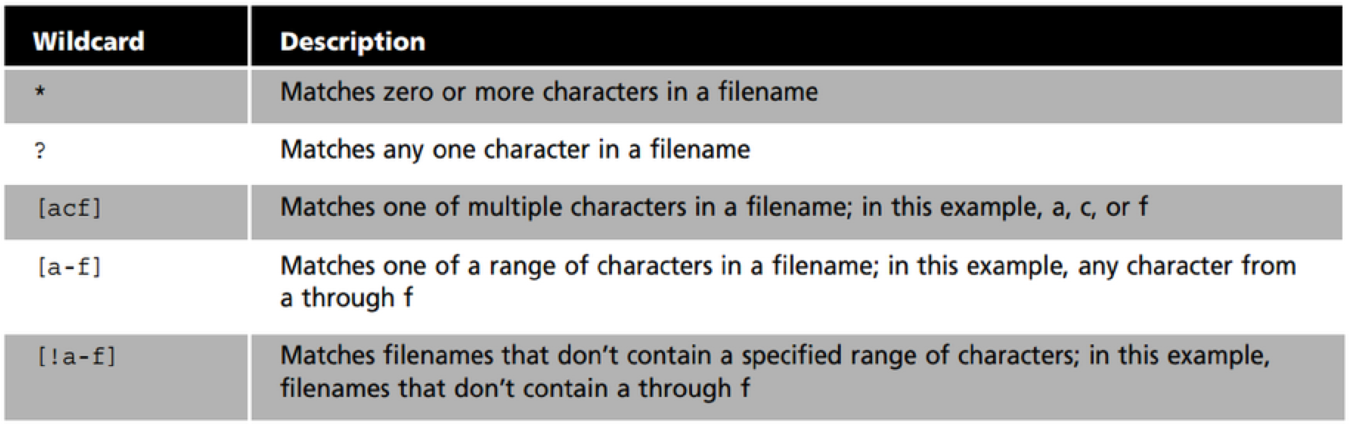
You can use POSIX or Character Classes with the [] wildcard



POSIX class	Represents	Means
[:upper:]	[A-Z]	Upper case letters
[:lower:]	[a-z]	Lower case letters
[:alpha:]	[A-Za-z]	Upper and Lower case letters
[:alnum:]	[A-Za-z0-9]	Lower case, upper case, and digits
[:digit:]	[0-9]	digits
[:xdigit:]	[0-9A-Fa-f]	hexadecimal digits
[:punct:]	[.,!?:...]	puctuation
[:blank:]	[\t]	space and tabs
[:cntrl:]	n/a	control characters
[:graph:]	[^\t\n\r\g\v]	printed characters without spaces
[:print:]	[^\t\n\r\g\v]	printed characters including spaces
[:space:]	[\t\n\r\g\v]	whitespace characters



```
┌[19:09:05](adrian@G752VL2 dir)
└>ls [[:lower:]]*
f1le  f3le  fble  fele  file  fzle  sf37
┌[19:09:21](adrian@G752VL2 dir)
└>ls [[:digit:]]*
123adrian  456adrian  789adrian
┌[19:09:30](adrian@G752VL2 dir)
└>ls [[:punct:]]*
.hidden1  .hidden2  .hidden3
┌[19:09:41](adrian@G752VL2 dir)
└>
```



Wildcard	Description
*	Matches zero or more characters in a filename
?	Matches any one character in a filename
[acf]	Matches one of multiple characters in a filename; in this example, a, c, or f
[a-f]	Matches one of a range of characters in a filename; in this example, any character from a through f
[!a-f]	Matches filenames that don't contain a specified range of characters; in this example, filenames that don't contain a through f

- Brace expansion {} is not a wildcard but another feature of bash that allows you to generate arbitrary strings to use with commands.
- For example,
  - To create a whole directory structure in a single command:
    - `mkdir -p music/{jazz,rock}/{mp3files,vidoes,oggfiles}/new{1..3}`
  - To create a N number of files use:
    - `touch website{1..5}.html`
    - `touch file{A..Z}.txt`
    - `touch file{001..10}.py`
    - `touch file{{a..z},{0..10}}.js`
  - Remove multiple files in a single directory
    - `rm -r {dir1,dir2,dir3,file.txt,file.py}`