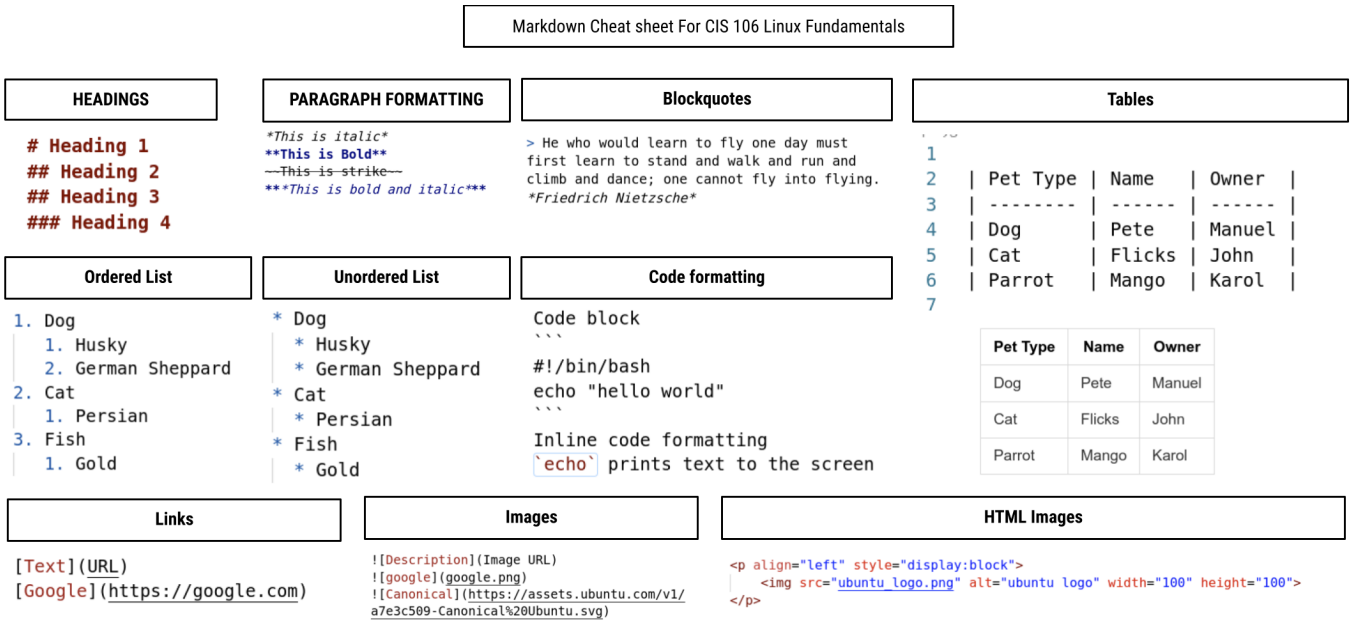
GitHub commands

`git pull` `git add .` `git commit -m ""` `git push`

Markdown file image and Proper heading



- `![Image alt text](/path/to/img.jpg)`
- `![Image alt text](/path/to/img.jpg "title")`
- `![Image alt text][img] [img]: http://foo.com/img.jpg`

To clone git repository

`git clone HTTPs Link`

lab 3 notes

BASH SHELL shortcut

`Ctrl + A`	Go to the start of the command line
`Ctrl + E`	Go to the end of the command line
`Ctrl + K`	Delete from cursor to the end of the command line
`Ctrl + U`	Delete from cursor to the start of the command line
`Ctrl + W`	Delete from cursor to start of word (i.e. delete backwards one word)
`Ctrl + Y`	Paste word or text that was cut using one of the deletion shortcuts after the cursor
`Ctrl + XX`	Move between start of command line and current cursor position (and back again)

`Ctrl + R`	Search the history backwards
`Ctrl + G`	Escape from history searching mode
`Ctrl + P`	Previous command in history (i.e. walk back through the command history)
`Ctrl + N`	Next command in history (i.e. walk forward through the command history)
`Ctrl + F`	Move forward one character
`Ctrl + B`	Move backward one character
`Ctrl + D`	Delete character under the cursor

`Ctrl + H`	Delete character before the cursor
`Ctrl + T`	Swap character under cursor with the previous one
`Ctrl + L`	Clear the screen
`Ctrl + S`	Stops the output to the screen (for long running verbose command)
`Ctrl + Q`	Allow output to the screen (if previously stopped using command above)
`Ctrl + C`	Terminate the command
`Ctrl + Z`	Suspend/stop the command

`Alt + U`	Make uppercase from cursor to end of word
`Alt + l`	Make lowercase from cursor to end of word
`Alt + T`	Swap current word with previous
`Alt + C`	Capitalize to end of word starting at cursor (whole word if cursor is at the beginning of word)
`Alt + B`	Move backward one word (or go to start of word the cursor is currently on)
`Alt + F`	Move forward one word (or go to end of word the cursor is currently on)
`Alt + D`	Delete to end of word starting at cursor (whole word if cursor is at the beginning of word)
`Alt + .`	Use the last word of the previous command

`!!`	Run last command
`!blah`	Run the most recent command that starts with 'blah' (e.g. !ls)
`!blah:p`	Print out the command that !blah would run (also adds it as the latest command in the command history)
`!$`	The last word of the previous command (same as Alt + .)
`!$:p`	Print out the word that !$ would substitute
`!*`	The previous command except for the last word (e.g. if you type '_find somefile.txt /', then !* would give you '_find somefile.txt')
`!*:p`	Print out what !* would substitute

Basic command

- **date** displays the current time and date
- **df** displays disk space usage
- **free** displays the amount of free memory
- **uname** displays information about your system
- **clear** clears the screen
- **history** displays the shell command history
- **lscpu** displays information about your CPU
- **lshw** displays information about your computer hardware
- **figlet** displays a given string as large text

Example

- Options of the date command:
  - `+FORMAT` Customizes the output format of the date.
  - `-d "STRING"` Displays a date specified by the string.
- Display the current date and time
  - `date`
- Display the date in a custom format
  - `date +"%Y-%m-%d %H:%M:%S"`
- Display the date and time from a specific time in the past
  - `date -d "3 years ago"`
- Display the current date with UTC formatted
  - `date -u +"%D-%X-%Z"`

Common formatting characters for the date command:

- %D Display date as mm/dd/yy
- %Y Year (e.g., 2020)
- %m Month (01-12)
- %B Long month name (e.g., November)
- %b Short month name (e.g., Nov)
- %d Day of month (e.g., 01)
- %j Day of year (001-366)
- %u Day of week (1-7)
- %A Full weekday name (e.g., Friday)
- %a Short weekday name (e.g., Fri)
- %H Hour (00-23)
- %I Hour (01-12)
- %M Minute (00-59)
- %S Second (00-60)
- %X time representation (e.g., 23:13:48)
- %Z alphabetic time zone abbreviation

More here: FORMAT control characters supported by the GNU/date command

Uname command

Options of the uname command

Option	Description	Example Output
`-a`	Display **all system information**. Combines all options.	`Linux hostname 5.15.0-50 x86_64...`
`-s`	Display the **kernel name**.	`Linux`
`-n`	Display the **network hostname** of the machine.	`hostname`
`-r`	Display the **kernel release** version.	`5.15.0-50-generic`
`-v`	Display the **kernel version** and build info.	`#56-Ubuntu SMP Tue Sep 6 12:00:00`
`-m`	Display the **machine hardware architecture**.	`x86_64`
`-p`	Display the **processor type** (if available).	`x86_64` or `unknown`
`-i`	Display the **hardware platform** (if available).	`x86_64` or `unknown`
`-o`	Display the **operating system** type.	`GNU/Linux`

Write a shell script

1. Open a text editor
2. Save the file as: file_name.sh
3. The first line in the file must be the shebang or shell interpreter. In the case of bash it would be: **`#!/bin/bash`**
4. The rest of the script includes the commands you want the shell to execute when the file is run.
5. To run the script, use the following command:
   a. **`bash /path/to/script/script_name.sh`**

**Note*: we will discuss other ways of running scripts later in the course. For now let's keep it simple.*

Getting help

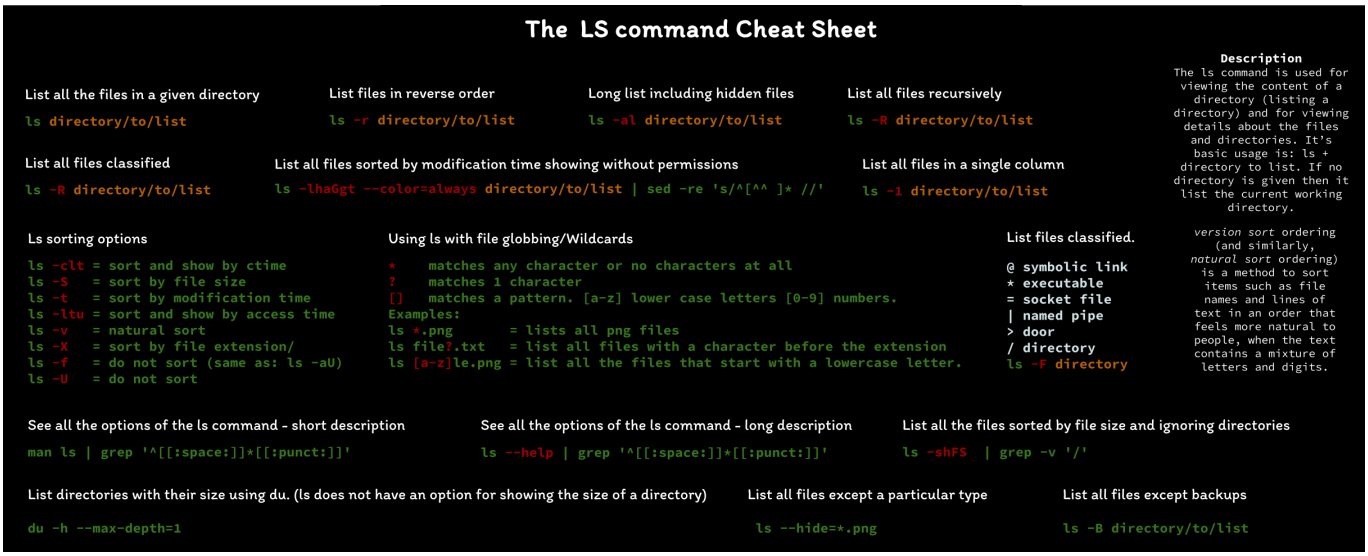
`man echo` exit? `q` ex `read -p "what is name" name`

lab 4 notes

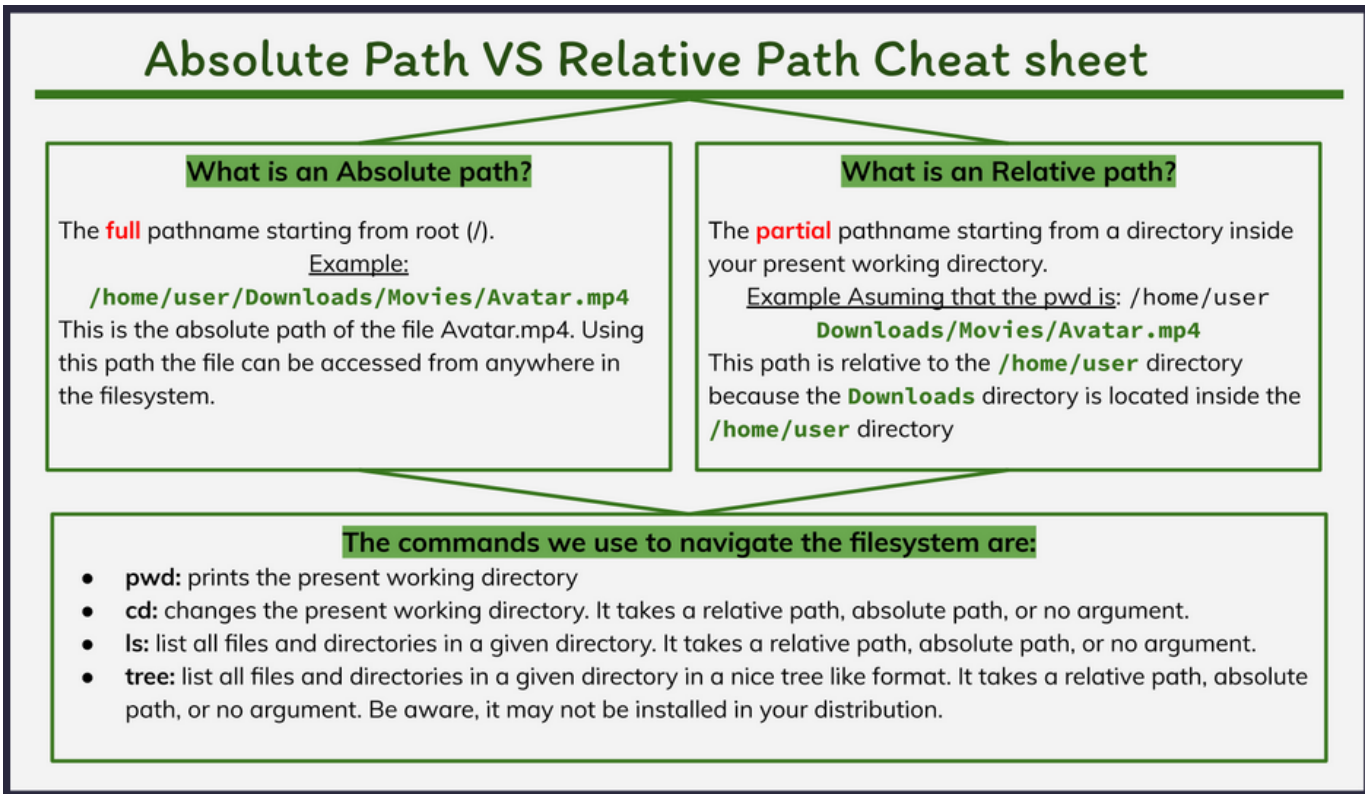
commands

pwd= Print the absolute path of the current working directory. cd = Change the shell current working directory. ls = list files inside a given directory

LS command cheat sheet



A vs R path



SPECIAL CHARACTERS

- . (single period): represents the current directory.
- ~ (tilde character): expands the current users home directory. It is like a variable that the shell uses to store the absolute path of the user's home directory. This ~/Downloads is the same as typing /home/maria53/Downlods
- / (one forward slash): as mentioned earlier, this is the root directory and the shortest path in the system. This is the beginning of the directory tree. There is nothing before it and everything after it.
- (hyphen-minus): is used to move to the previous current working directory.
- # (hash or number sign): This is used for single line comments in shell scripting.
- ! (single exclamation mark): used for repeating a command from the history. For example !5 will repeat the 5th command in the command history. To view the entire command history type history.

- !! (2 consecutive exclamaton marks): are used for repeating the previous command. For example, !! will repeat the previous command while, sudo !! will repeat the previous command but will add sudo at the beginning of the command. This is useful for times when we forget to type sudo when performing administrative tasks.
- `$user = prisha` = variable (bash shell)

CD

- cd (without any arguments, cd will take you home)
- cd ~ (using the ~ special character. as ~ will expand to the absolute path of the user's home directory)
- cd $HOME (using the $HOME environment variable)
- cd /home/$USER/Downloads (using $USER environment variable in the path)
- Go to a specified directory with absolute path:

  ```
  cd /usr/share/themes
  ```

- Go to a specified directory with relative path assuming your current working directory is /home

  ```
  cd maria53/Downloads/
  ```

- Go to the previous working directory. This is useful when you are working with 2 directories located far in the directory tree

  ```
  cd -
  ```

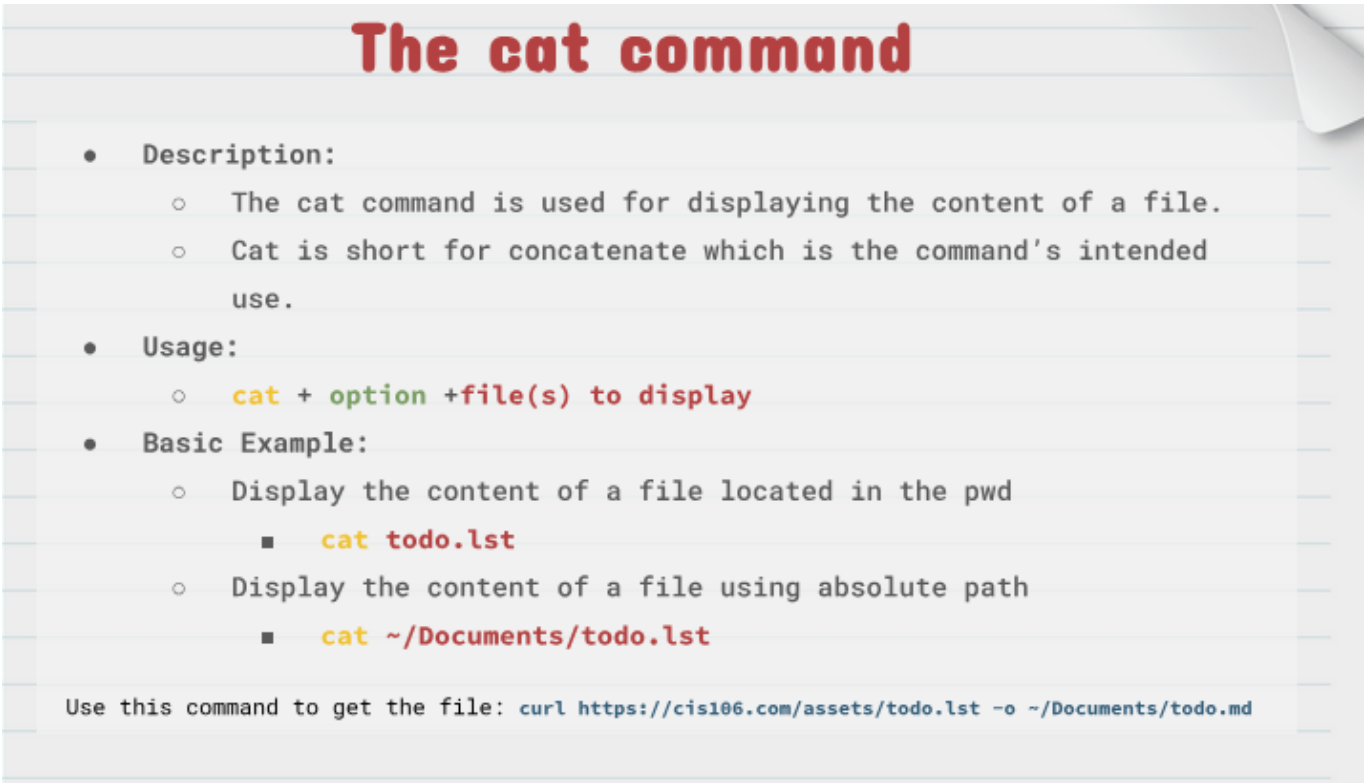
- Go to the previous directory in the directory tree. One directory above.

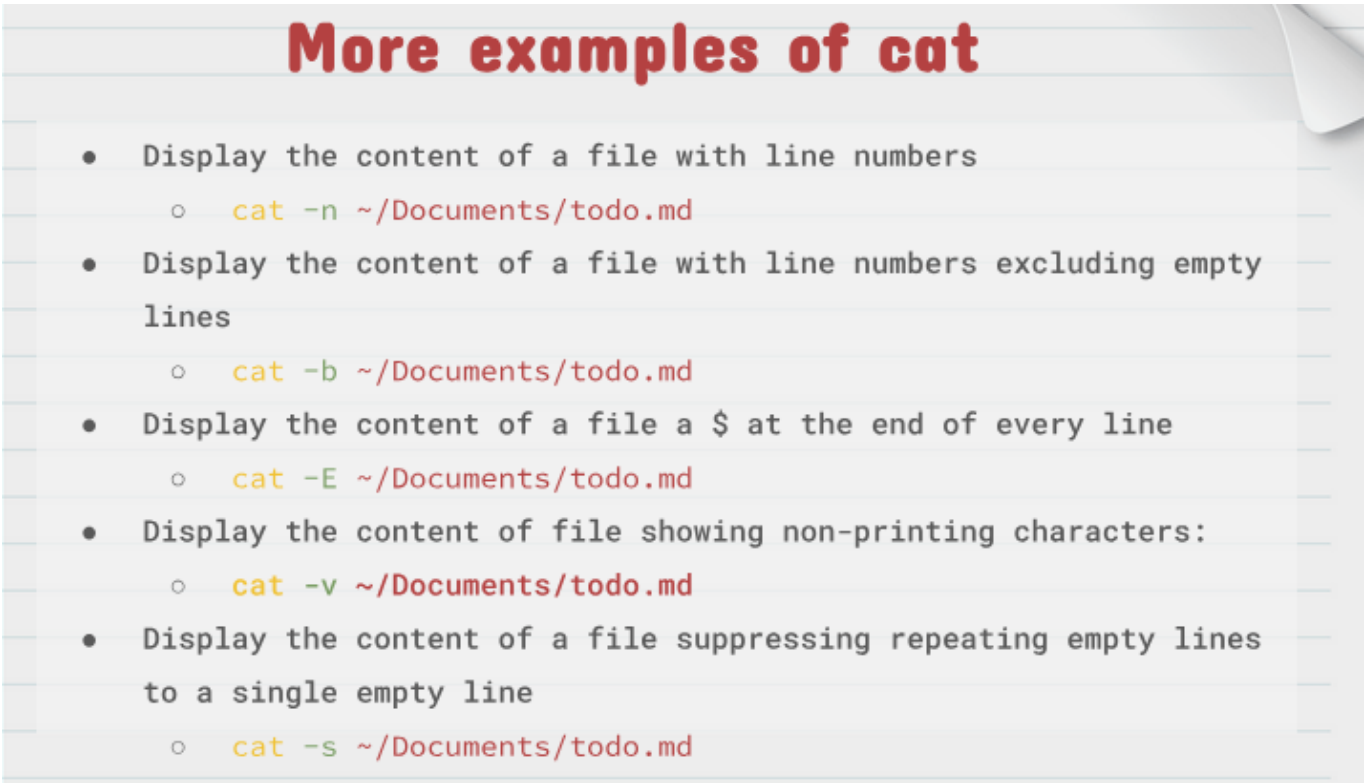
  ```
  cd ../
  ```

- Go to 2 directories above the directory tree

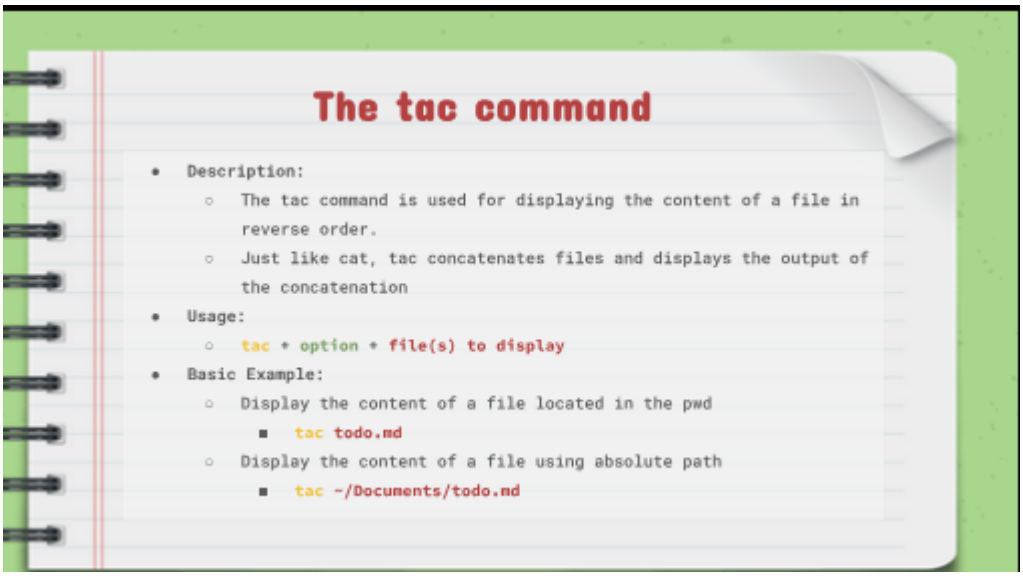
  ```
  cd ../../
  ```

CAT

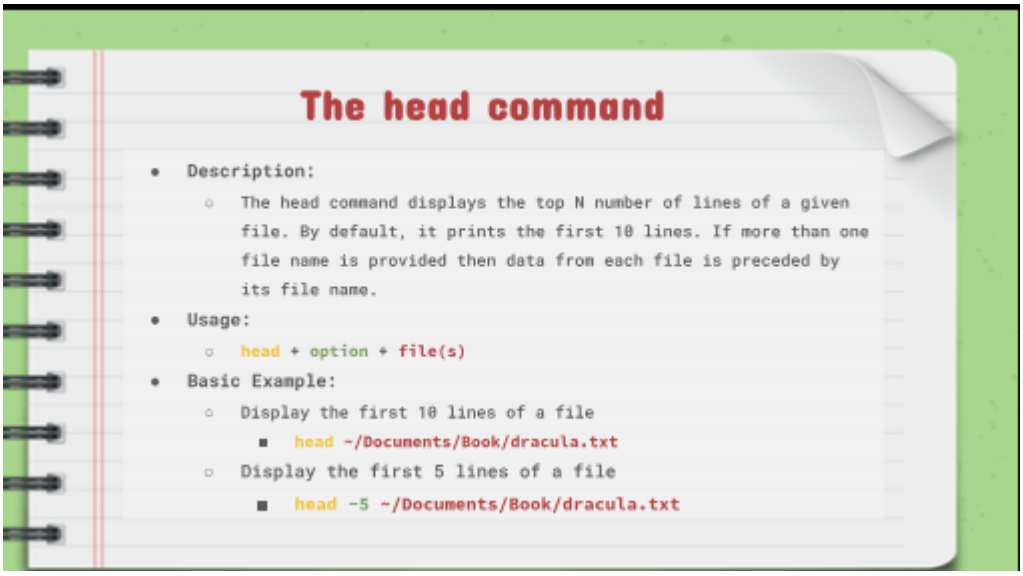

The cat command
Description:
The cat command is used for displaying the content of a file.
Cat is short for concatenate which is the command's intended use.
Usage:
cat + option +file(s) to display
Basic Example:
Display the content of a file located in the pwd
cat todo.lst
Display the content of a file using absolute path
cat ~/Documents/todo.lst
Use this command to get the file: curl https://cis106.com/assets/todo.lst -o ~/Documents/todo.md

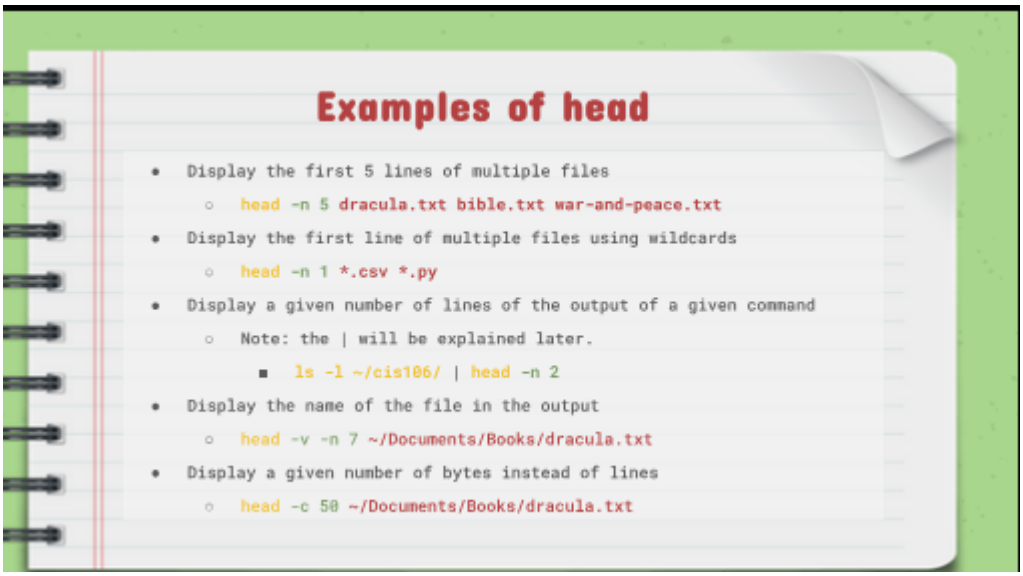

More examples of cat
Display the content of a file with line numbers
cat -n ~/Documents/todo.md
Display the content of a file with line numbers excluding empty lines
cat -b ~/Documents/todo.md
Display the content of a file a $ at the end of every line
cat -E ~/Documents/todo.md
Display the content of file showing non-printing characters:
cat -v ~/Documents/todo.md
Display the content of a file suppressing repeating empty lines to a single empty line
cat -s ~/Documents/todo.md

TAC

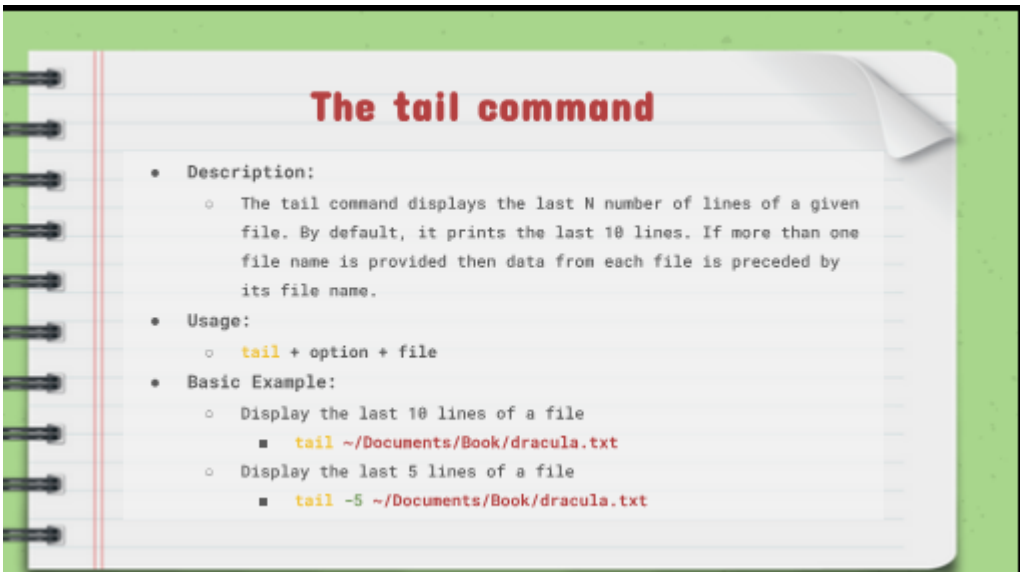

The tac command
Description:
The tac command is used for displaying the content of a file in reverse order.
Just like cat, tac concatenates files and displays the output of the concatenation
Usage:
tac + option + file(s) to display
Basic Example:
Display the content of a file located in the pwd
tac todo.md
Display the content of a file using absolute path
tac ~/Documents/todo.md

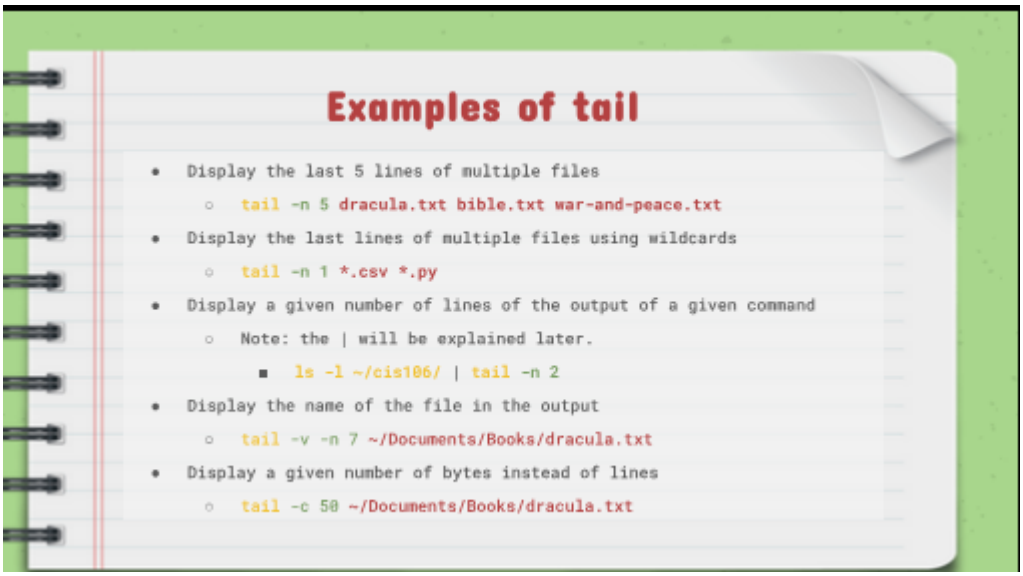
HEAD


The head command
Description:
The head command displays the top N number of lines of a given file. By default, it prints the first 10 lines. If more than one file name is provided then data from each file is preceded by its file name.
Usage:
head + option + file(s)
Basic Example:
Display the first 10 lines of a file
head ~/Documents/Book/dracula.txt
Display the first 5 lines of a file
head -5 ~/Documents/Book/dracula.txt

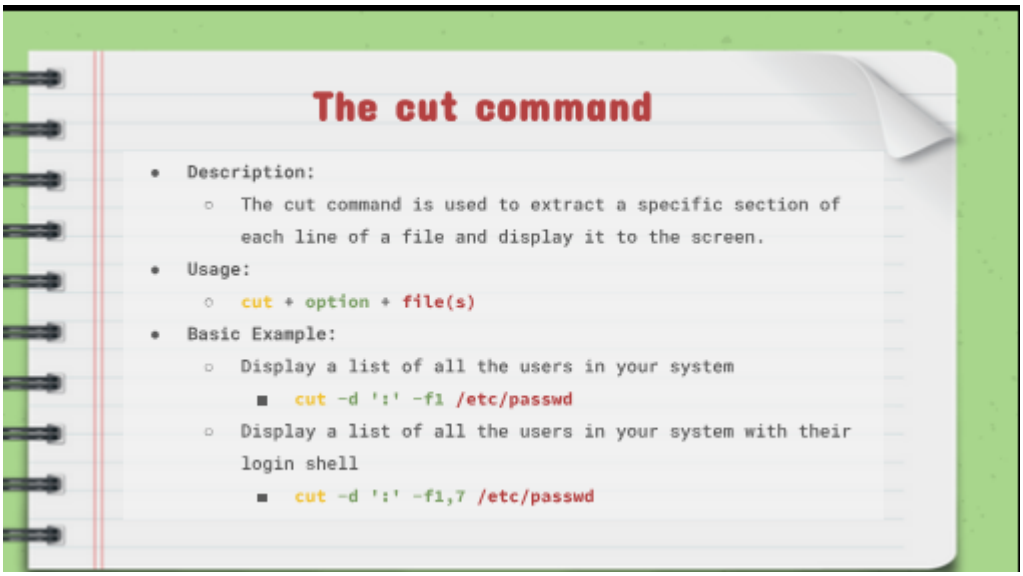

Examples of head
Display the first 5 lines of multiple files
head -n 5 dracula.txt bible.txt war-and-peace.txt
Display the first line of multiple files using wildcards
head -n 1 *.csv *.py
Display a given number of lines of the output of a given command
Note: the | will be explained later.
ls -l ~/cis106/ | head -n 2
Display the name of the file in the output
head -v -n 7 ~/Documents/Books/dracula.txt
Display a given number of bytes instead of lines
head -c 50 ~/Documents/Books/dracula.txt

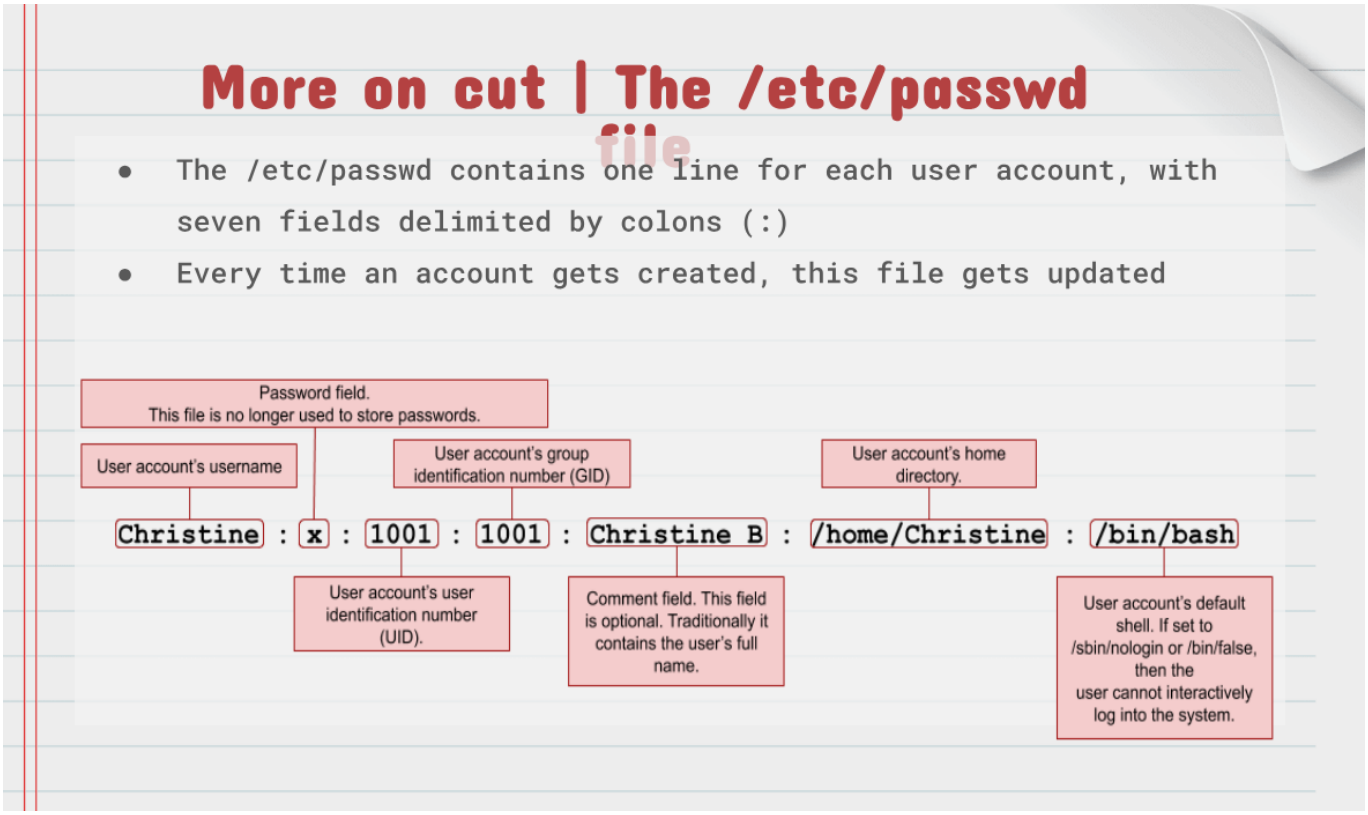
TAIL


The tail command
Description:
The tail command displays the last N number of lines of a given file. By default, it prints the last 10 lines. If more than one file name is provided then data from each file is preceded by its file name.
Usage:
tail + option + file
Basic Example:
Display the last 10 lines of a file
tail ~/Documents/Book/dracula.txt
Display the last 5 lines of a file
tail -5 ~/Documents/Book/dracula.txt

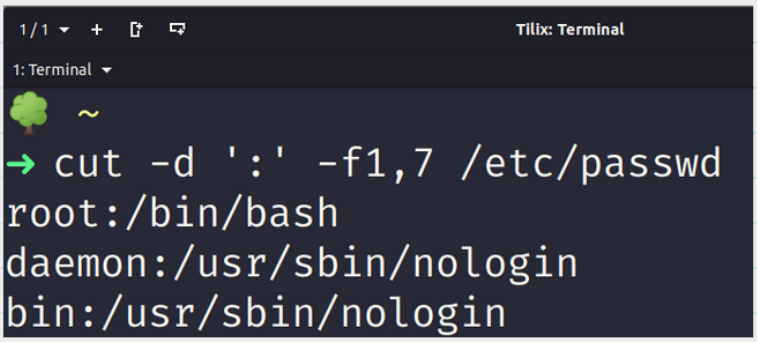

Examples of tail
Display the last 5 lines of multiple files
tail -n 5 dracula.txt bible.txt war-and-peace.txt
Display the last lines of multiple files using wildcards
tail -n 1 *.csv *.py
Display a given number of lines of the output of a given command
Note: the | will be explained later.
ls -l ~/cis106/ | tail -n 2
Display the name of the file in the output
tail -v -n 7 ~/Documents/Books/dracula.txt
Display a given number of bytes instead of lines
tail -c 50 ~/Documents/Books/dracula.txt

CUT


The cut command
Description:
The cut command is used to extract a specific section of each line of a file and display it to the screen.
Usage:
cut + option + file(s)
Basic Example:
Display a list of all the users in your system
cut -d ':' -f1 /etc/passwd
Display a list of all the users in your system with their login shell
cut -d ':' -f1,7 /etc/passwd

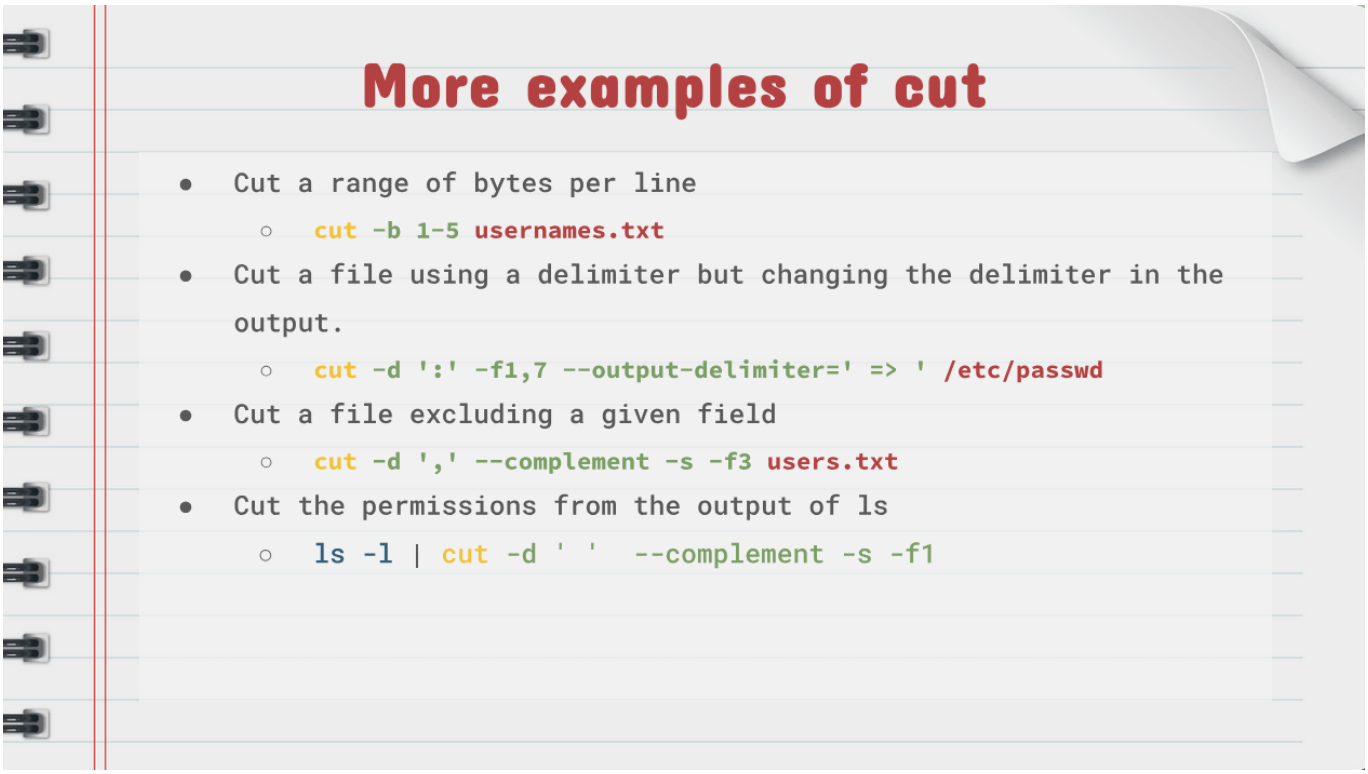

More on cut | The /etc/passwd file
The /etc/passwd contains one line for each user account, with seven fields delimited by colons (:)
Every time an account gets created, this file gets updated
Password field. This file is no longer used to store passwords.
User account's username
User account's group identification number (GID)
User account's home directory.
Christine : x : 1001 : 1001 : Christine B : /home/Christine : /bin/bash
User account's user identification number (UID).
Comment field. This field is optional. Traditionally it contains the user's full name.
User account's default shell. If set to /sbin/nologin or /bin/false, then the user cannot interactively log into the system.

More on cut | explained



More examples of cut

- Cut a range of bytes per line
  - `cut -b 1-5 usernames.txt`
- Cut a file using a delimiter but changing the delimiter in the output.
  - `cut -d ':' -f1,7 --output-delimiter=' => ' /etc/passwd`
- Cut a file excluding a given field
  - `cut -d ',' --complement -s -f3 users.txt`
- Cut the permissions from the output of ls
  - `ls -l | cut -d ' '  --complement -s -f1`


More examples of cut
Cut a range of bytes per line
cut -b 1-5 usernames.txt
Cut a file using a delimiter but changing the delimiter in the output.
cut -d ':' -f1,7 --output-delimiter=' => ' /etc/passwd
Cut a file excluding a given field
cut -d ',' --complement -s -f3 users.txt
Cut the permissions from the output of ls
ls -l | cut -d ' ' --complement -s -f1

PASTE


The paste command
Description:
The paste command is used for joining files horizontally in columns
Usage:
paste + option + files
Basic Example:
Merge two files
paste users.lst ip_address.lst
Merge two files using a different delimiter
paste -d ":" users1.lst ip_addresses.lst

SORT

The sort command

- **Description:**
  - The sort command is used for sorting files. The sort command supports sorting: alphabetically, in reverse order, by number, and by month.
  - The sort command follows this order unless specified otherwise:
    - Lines starting with a number will appear before lines starting with a letter.
    - Lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet.
    - Lines starting with a lowercase letter will appear before lines starting with the same letter in uppercase.
- **Usage:**
  - `sort + option + file`
- **Basic Example:**
  - Sort a file
    - `sort users.lst`

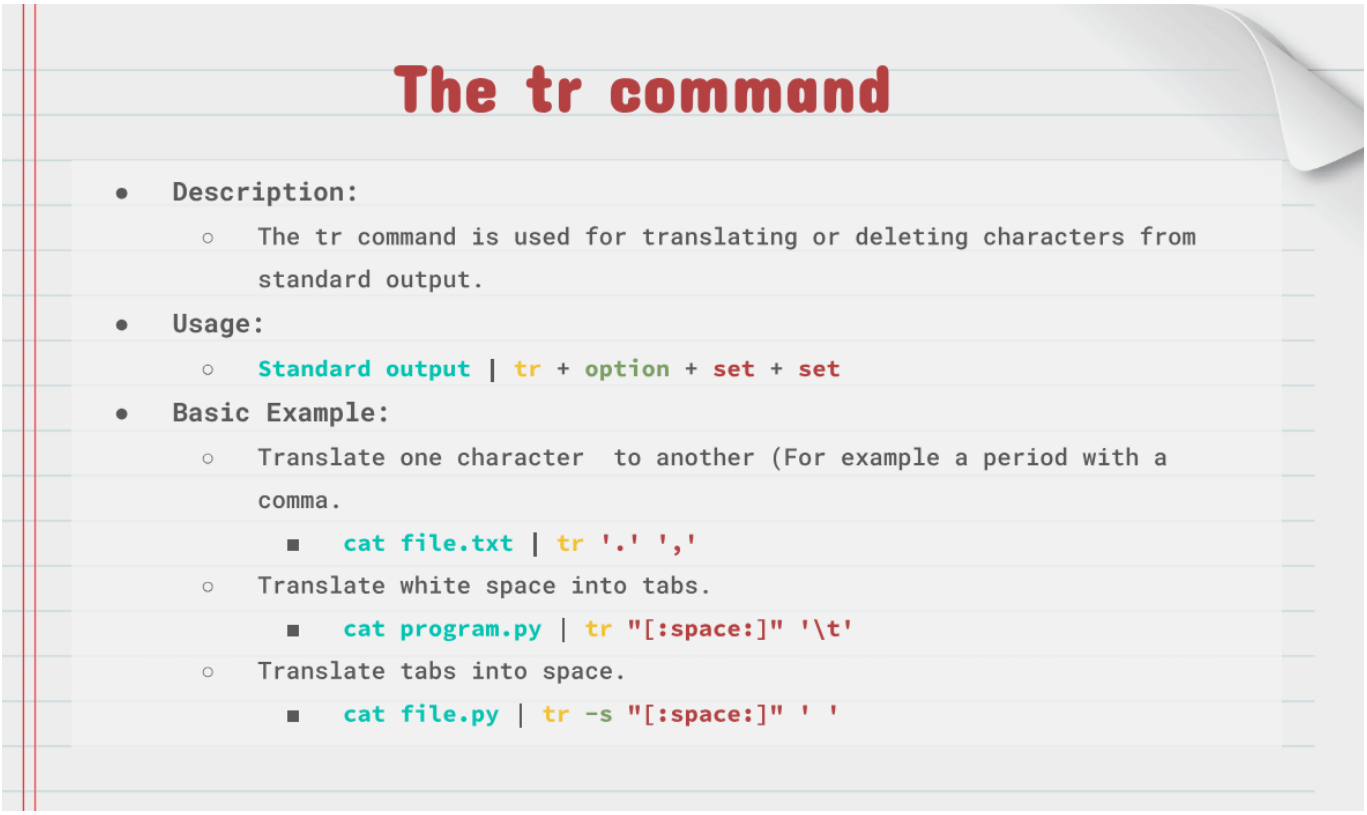
More examples of sort

- Sort a file and save the output to a new file
  - `sort -o sorted.lst users.lst`
- Sort a file in reverse order
  - `sort -r users.txt`
- Sort by column number
  - `sort -k 2 users.txt`
- Sort a file with numeric data
  - `sort -n phones.txt`
- Check if a file is sorted
  - `sort -c sorted.lst`
- Sort and remove duplicate entries
  - `sort -u users.lst`

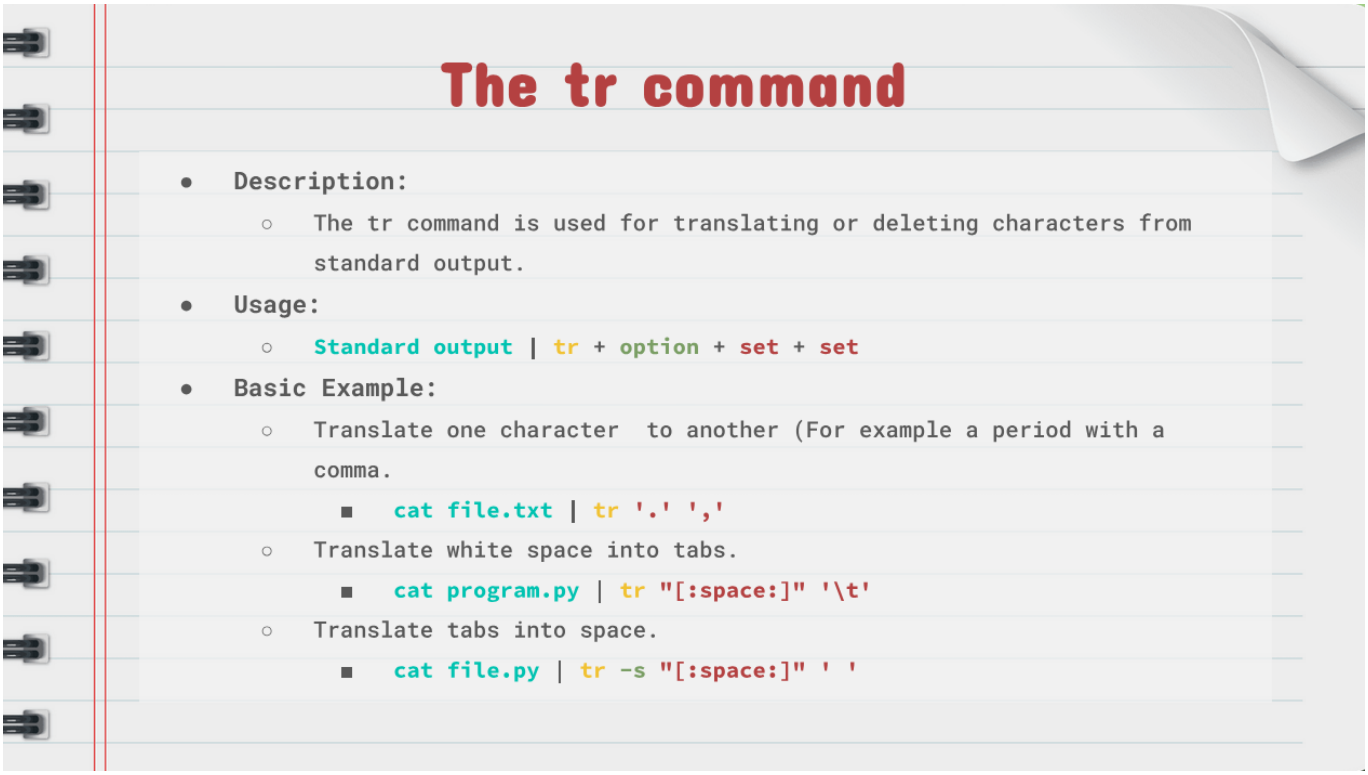
Note:
Use the -t option to specify a delimiter.
For example

`sort -t";" -k3 cereal.csv`

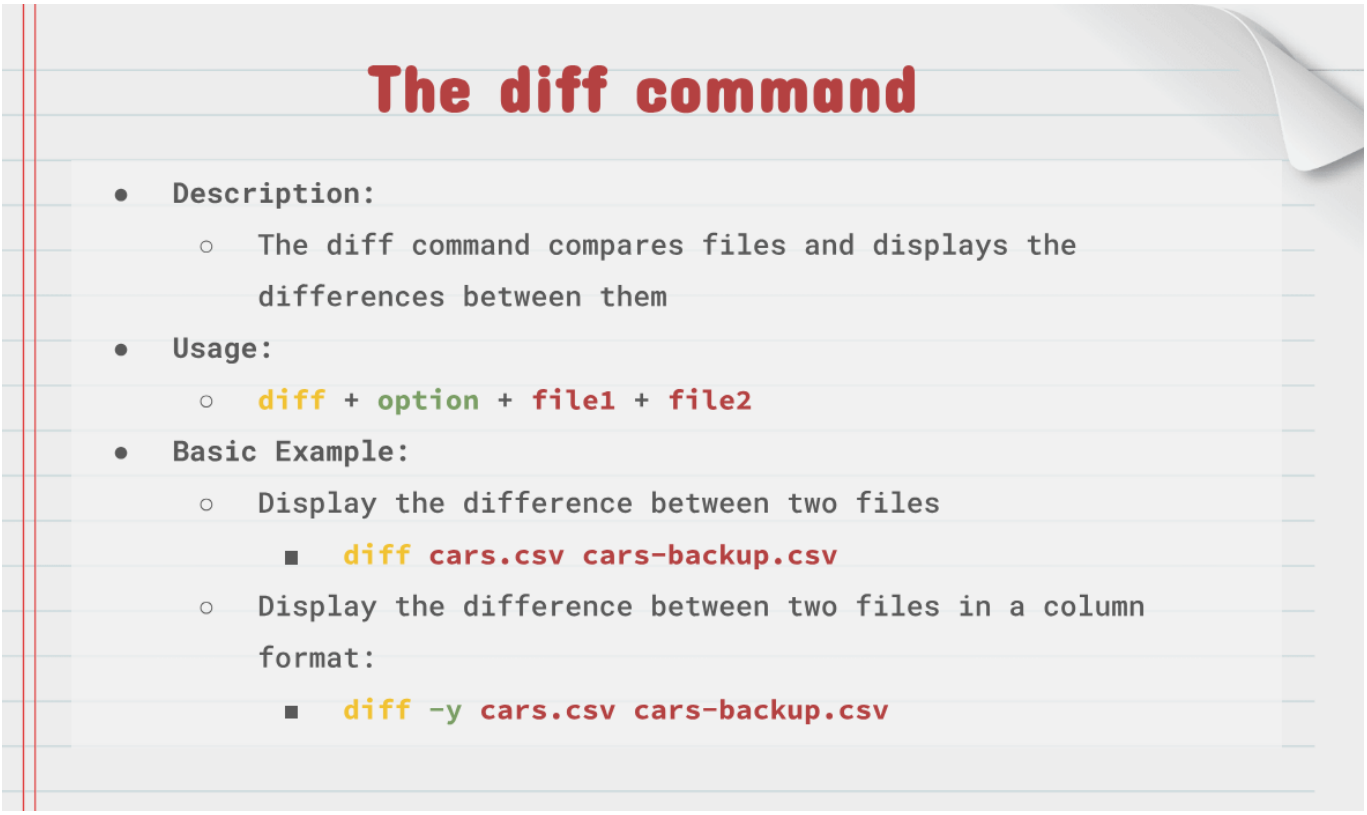
WC


The tr command
Description:
The tr command is used for translating or deleting characters from standard output.
Usage:
Standard output | tr + option + set + set
Basic Example:
Translate one character to another (For example a period with a comma.
cat file.txt | tr '.' ','
Translate white space into tabs.
cat program.py | tr "[:space:]" '\t'
Translate tabs into space.
cat file.py | tr -s "[:space:]" ' '

TR


The tr command
Description:
The tr command is used for translating or deleting characters from standard output.
Usage:
Standard output | tr + option + set + set
Basic Example:
Translate one character to another (For example a period with a comma.
cat file.txt | tr '.' ','
Translate white space into tabs.
cat program.py | tr "[:space:]" '\t'
Translate tabs into space.
cat file.py | tr -s "[:space:]" ' '

DIFF


The diff command
Description:
The diff command compares files and displays the differences between them
Usage:
diff + option + file1 + file2
Basic Example:
Display the difference between two files
diff cars.csv cars-backup.csv
Display the difference between two files in a column format:
diff -y cars.csv cars-backup.csv

GREP

The grep command

- **Description:**
  - Grep is used to search text in given file. Grep works line by line basis (it matches the search criteria in a line by line basis).
- **Usage:**
  - `grep + option + search criteria + file(s)`
- **Basic Example:**
  - Search any line that contains the word "dracula" in the given file:
    - `grep 'dracula' ~/Documents/dracula.txt`

Common options of grep

Option	Explanation
-i	Enables case insensitivity. (it will match regardless of case)
-n	Displays line number for every line matched
-E	Treats the pattern (search criteria) as an extended regular expression
-G	Treats the pattern (search criteria) as a basic regular expression
-v	Inverts the search (looks for the opposite of the given criteria)
-o	Only displays the matched string
-c	Search and display the total number of times a pattern is matched
-w	Matches only the given word (pattern) by itself.
-r, -R	Matches recursively. Useful for searching files in a given directory

More examples of grep

- Search any line that contains the word 'dracula' regardless of the case
  - `grep -i 'dracula' ~/Documents/Books/dracula.txt`
- Search any line that contains the word dracula regardless of case and with number line
  - `grep -in 'dracula' ~/Documents/Books/dracula.txt`
- Search for all the lines that do not contain the word 'war'
  - `grep -v 'war' ~/Documents/Books/war-and-peace.txt`
- Search and display only the matched string (pattern)
  - `grep -o 'pride' ~/Documents/Books/war-and-peace.txt`
- Display how many lines contain the matched string
  - `grep -c 'dracula' ~/Documents/Books/dracula.txt`

More examples of grep

- Search for a given strings inside files in a given directory.
  - `grep -iR 'conf' /etc/`
- Search and display the total number of times a given word appears in a file
  - `grep -wc '/bin/bash' /etc/passwd`
- The ^ (caret) symbol matches the empty string at the beginning of a line. Search for all the lines that start with a given word.
  - `grep -ni '^dracula' ~/Documents/Books/dracula.txt`
- Search for all the lines that ends with the string "nologin"
  - `grep -n 'nologin$' /etc/passwd`

More examples of grep

- Search for all the lines that contain a single word in all the files in your system
  - `grep -niR '^linux$' /`
- The . (period) symbol is a meta-character that matches any single character. Search for all the lines that contain a word that starts with letter d has 4 characters after.
  - `grep -n '^d....' ~/Documents/Books/dracula.txt`
- Bracket expressions allows match a group of characters by enclosing them in brackets []. Match all the lines that contain a the words list, last, lost.
  - `grep -n 'l[aio]st' ~/Documents/Books/dracula.txt`

More advance examples of grep

- Search for all the lines that start with a capital letter
  - `grep -n '^[A-Z]' ~/Documents/Books/war-and-peace.txt`
- Search for more than one word per line
  - `grep -Ewn 'horror|love|scare' ~/Documents/Books/dracula.txt`
- Match only lines containing IPv4 addresses
  - `grep -E '[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}' Documentation.txt`
- Search all lines that contain a character repeated 3 times
  - `grep -E "A{3}" file.txt`
- Search all lines that contain a phone number of the format 973-111-2222
  - `grep "[[:digit:]]\{3\}[-][[:digit:]]\{3\}[-][[:digit:]]\{4\}" contacts.csv`
- Display only the options of the of any command from its man page
  - `man ls | grep "^[[:space:]]*[[:punct:]]"`

AWK

The awk command

- Description:
  - Awk is a scripting language used for processing and displaying text. Awk can work with a text file or from standard output. Awk was created in Bell Labs during the 70s by Alfred Aho, Peter Weinberger, and Brian Kernighan and its name comes from its authors' initials. There are several implementations of Awk: nawk, mawk, gawk, and busybox.
  - Awk performs operations line by line
- Usage:
  - awk + options + {awk command} + file + file to save (optional)
- Basic Example:
  - Print the first column of every line of a file
    - awk '{print $1}' ~/Documents/Csv/cars.csv

Awk Variables

$0	Whole line
$1,$2..$NF	First, second… last field
NR	Total Number of Records
NF	N number of Fields
OFS	Output Field Separator (default " ")
FS	input Field Separator (default " ")
ORS	Output Record Separator (default "\n")
RS	input Record Separator (default "\n")
FILENAME	Name of the file
ARGC	Number or arguments

ARGV	Array of arguments
FNR	File Number of Records
OFMT	Format for numbers (default "%.6g")
RSTART	Location in the string
RLENGTH	Length of match
SUBSEP	Multi-dimensional array separator (default "\034")
ARGIND	Argument Index
ENVIRON	Environment variables
IGNORECASE	Ignore case
CONVFMT	Conversion format
ERRNO	System errors
FIELDWIDTHS	Fixed width fields

More examples of AWK

- Print first field of /etc/passwd file
  - `awk -F: '{print $1}' /etc/passwd`
- Print the last field of the /etc/passwd file
  - `awk -F: '{print $NF}' /etc/passwd`
- Print the first and last field of the /etc/passwd
  - `awk -F: '{print $1," = ",$NF}' /etc/passwd`
- Print the first and 3 field with line numbers
  - `awk -F: '{print NR,$1,$3}' /etc/passwd`
- Print the first and 4th field with a different field separator
  - `awk -F: '{OFS="="}{print $1,$4}' /etc/passwd`
- Start printing a file from a given line(exclude the first 2 lines)
  - `awk 'NR > 3 { print }' /etc/passwd`

More examples of awk

- Convert the first field to upper/lower case
  - `awk -F: '{print toupper($1)}' /etc/passwd`
- Prints the length of a line(record)
  - `awk '{print length($0)}' /etc/passwd`
- Print specific fields based on a command output. For example, the size and file name

```
ls -lhF Documents/ | awk 'BEGIN { printf "%s\t%s\n", "Size","Name"} {print $5,"\t",$9}'
```

  - BEGIN block is executed once at the start
- Print specific fields with a head of the /etc/passwd file

```
awk -F: 'BEGIN { printf "%s\t\t%s\n","User", "Shell" } {print $1,"\t",$7}' /etc/passwd
```

A more interesting example of awk

```
→ ls -lhd --time-style=+%D ./* | awk -v OFS='\t' 'BEGIN { printf "%s\t%s\t%s\t%s\t%s\t%s\t%s\n", "Permissions", "Links","User","Group","Size","Date Modified","Name"} {print $1,$2,$3,$4,$5,$6,$7}'
Permissions     Links   User    Group   Size    Date Modified   Name
drwxrwxr-x      3       adrian  adrian  4.0K    03/02/22        ./Applications
drwxrwxr-x      3       adrian  adrian  4.0K    04/03/22        ./Calibre
drwxrwxr-x      2       adrian  adrian  4.0K    04/06/22        ./challenge-Lab6
drwxr-xr-x      2       adrian  adrian  4.0K    03/01/22        ./Desktop
drwxr-xr-x      5       adrian  adrian  4.0K    04/12/22        ./Documents
lrwxrwxrwx      1       adrian  adrian  28      03/03/22        ./dotfiles
drwxr-xr-x      10      adrian  adrian  24K     04/12/22        ./Downloads
drwxrwxr-x      2       adrian  adrian  4.0K    04/03/22        ./filesAwkExamples
drwxrwxr-x      2       adrian  adrian  4.0K    04/02/22        ./games
drwxrwxr-x      27      adrian  adrian  60K     04/10/22        ./gdrive
drwxrwxr-x      9       adrian  adrian  4.0K    04/02/22        ./gems
drwxrwxr-x      2       adrian  adrian  4.0K    04/12/22        ./json-files
drwxrwxr-x      7       adrian  adrian  4.0K    04/02/22        ./lab5
-rw-rw-r--      1       adrian  adrian  4.1M    04/02/22        ./lab5.zip
drwxrwxr-x      2       adrian  adrian  4.0K    04/11/22        ./lab6
drwxrwxr-x      2       adrian  adrian  4.0K    04/02/22        ./movies
drwxr-xr-x      3       adrian  adrian  4.0K    04/04/22        ./Music
drwxr-xr-x      3       adrian  adrian  4.0K    03/05/22        ./Pictures
drwxr-xr-x      2       adrian  adrian  4.0K    03/01/22        ./Public
-rw-rw-r--      1       adrian  adrian  0       04/04/22        ./song.mp3
drwxrwxr-x      3       adrian  adrian  4.0K    03/02/22        ./Steam
drwxr-xr-x      2       adrian  adrian  4.0K    03/01/22        ./Templates
-rw-rw-r--      1       adrian  adrian  108     04/09/22        ./todo.md
-rwxrw-r--      1       adrian  adrian  511     04/09/22        ./urls.sh
drwxr-xr-x      2       adrian  adrian  4.0K    04/07/22        ./Videos
drwxrwxr-x      7       adrian  adrian  4.0K    03/21/22        ./VirtualBox
```

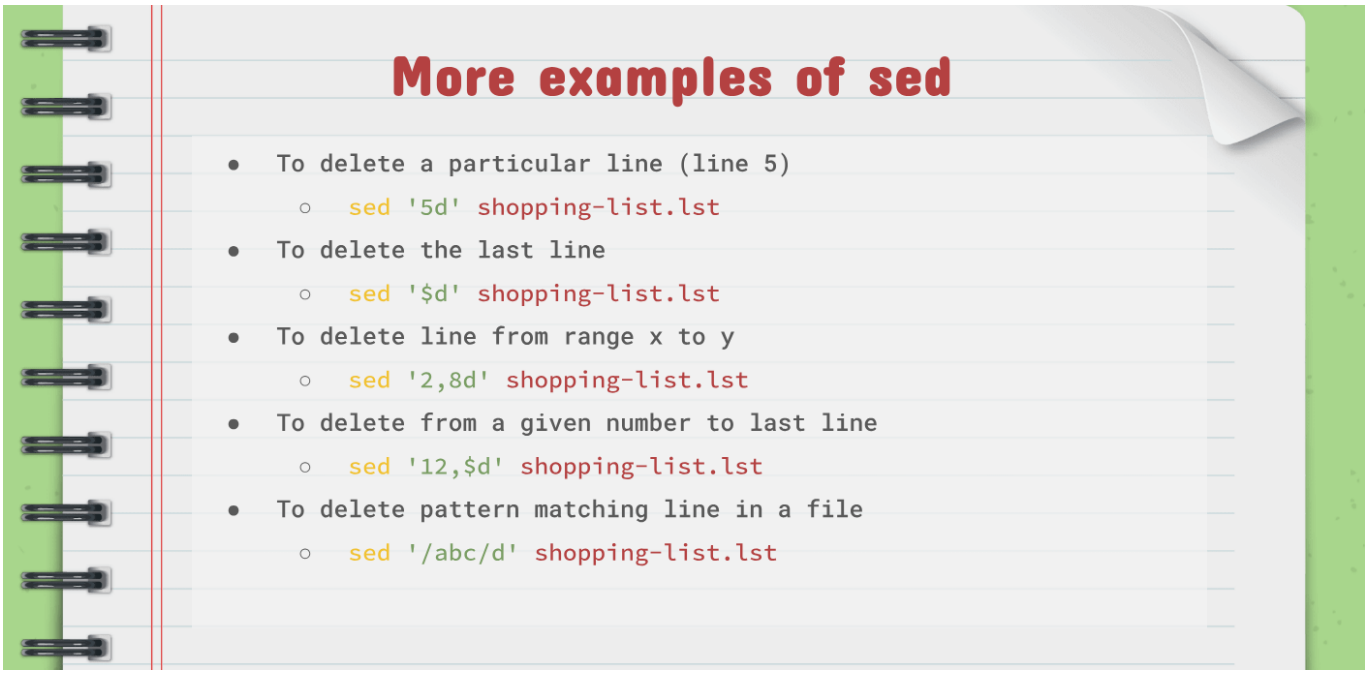
SED

More examples of sed

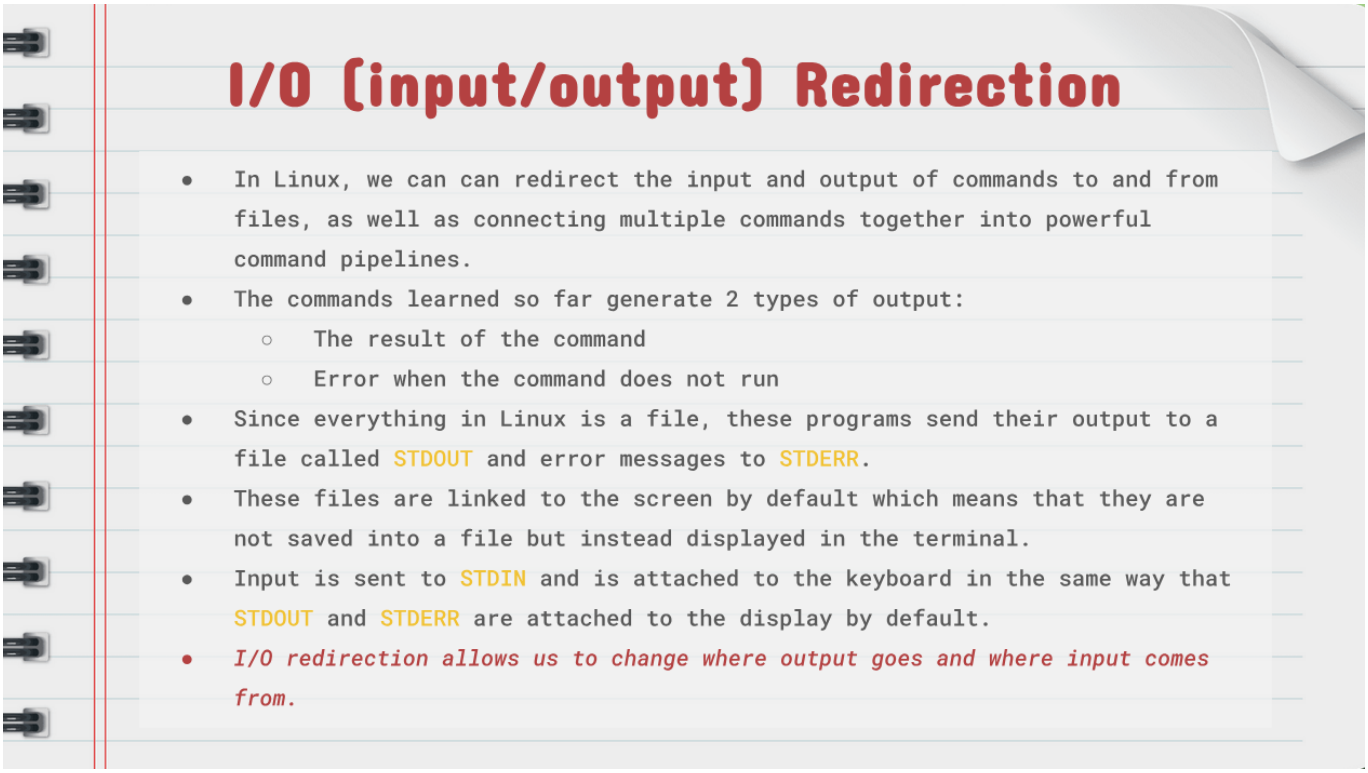
- To delete a particular line (line 5)
  - `sed '5d' shopping-list.lst`
- To delete the last line
  - `sed '$d' shopping-list.lst`
- To delete line from range x to y
  - `sed '2,8d' shopping-list.lst`
- To delete from a given number to last line
  - `sed '12,$d' shopping-list.lst`
- To delete pattern matching line in a file
  - `sed '/abc/d' shopping-list.lst`

More examples of sed

- Replacing the number of occurrences of a pattern in a file
  - `sed 's/pizza/rice/4' shopping-list.lst`
- Replacing all the occurrence of the pattern in a file
  - `sed 's/pizza/rice/g' shopping-list.lst`
- Replacing from the given number occurrence to the rest occurrences in a file. Start at the second time the word appears and continue to till the end of the file
  - `sed 's/pizza/rice/3g' shopping-list.lst`
- Replacing string on a specific line number
  - `sed '3 s/pizza/rice/' shopping-list.lst`
- Replacing string on a range of lines
  - `sed '1,3 s/pizza/rice/' shopping-list.lst`



working with 1/0



1/0

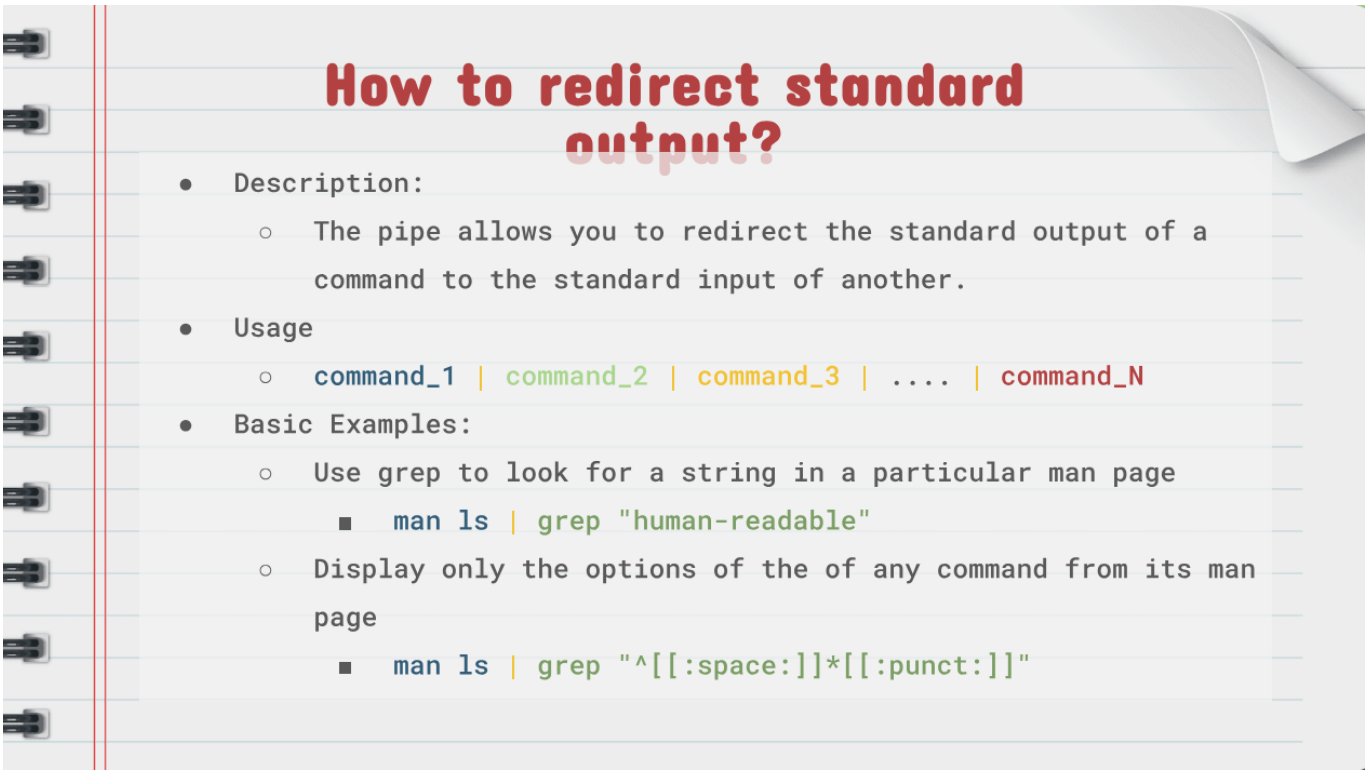
How to save standard output

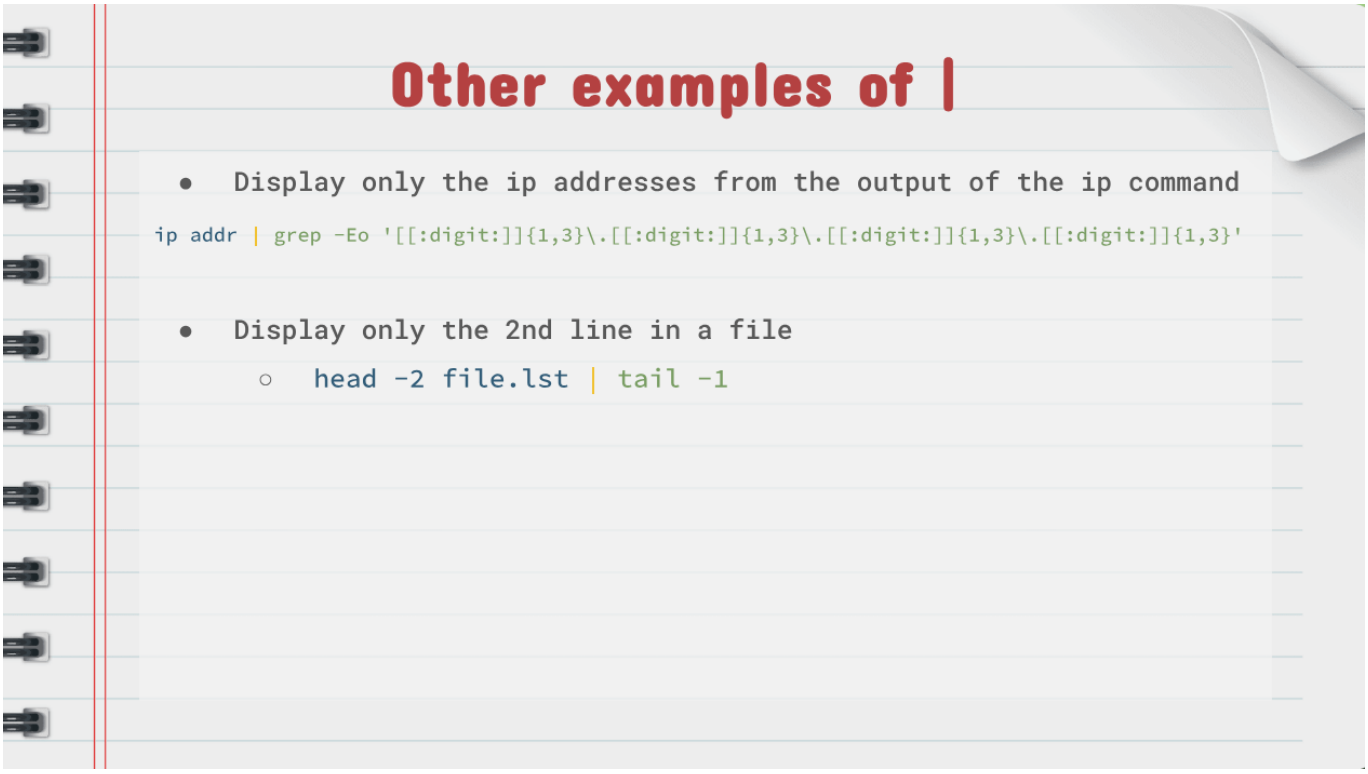
How to save standard output?

- Usage
  - `Command output + > + file`
- Basic Example:
  - Save the output of a command to a file
    - `ls -lA ~ > all-files-in-home.txt`
  - Save the error generated by a command to a file
    - `ls -lA downloads/ 2> error-of-ls`
  - Save the error to a file and the success to another
    - `ls -lA downloads/ Pictures > success.txt 2> error.txt`
  - Save the error and success to the same file
    - `ls -lA downloads/ Pictures &> allouput.txt`
  - Do not display errors. Send errors to the black hole
    - `ls -lA downloads/ 2> /dev/null`

Appending output to a file

- Append means to add more to a file instead of overwriting its content. When we use > on a file that already exist and contains data, we overwrite whatever is already inside the file. For instance take this example:
  - `ls -la > allmyfiles.lst`
- In this example, if the file allmyfiles.lst had any data prior executing the command, that data will be overwritten by the output of `ls -la`.
- What happens if we want to keep the old data? Then we use `>>` for example
  - `ls -la >> allmyfiles.lst`
- Will add the output of `ls -la` to the end of the file `allmyfiles.lst`


How to redirect standard output?
Description:
The pipe allows you to redirect the standard output of a command to the standard input of another.
Usage
command_1 | command_2 | command_3 | .... | command_N
Basic Examples:
Use grep to look for a string in a particular man page
man ls | grep "human-readable"
Display only the options of the of any command from its man page
man ls | grep "^[[:space:]]*[[:punct:]]"


Other examples of |
Display only the ip addresses from the output of the ip command
ip addr | grep -Eo '[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}'
Display only the 2nd line in a file
head -2 file.lst | tail -1

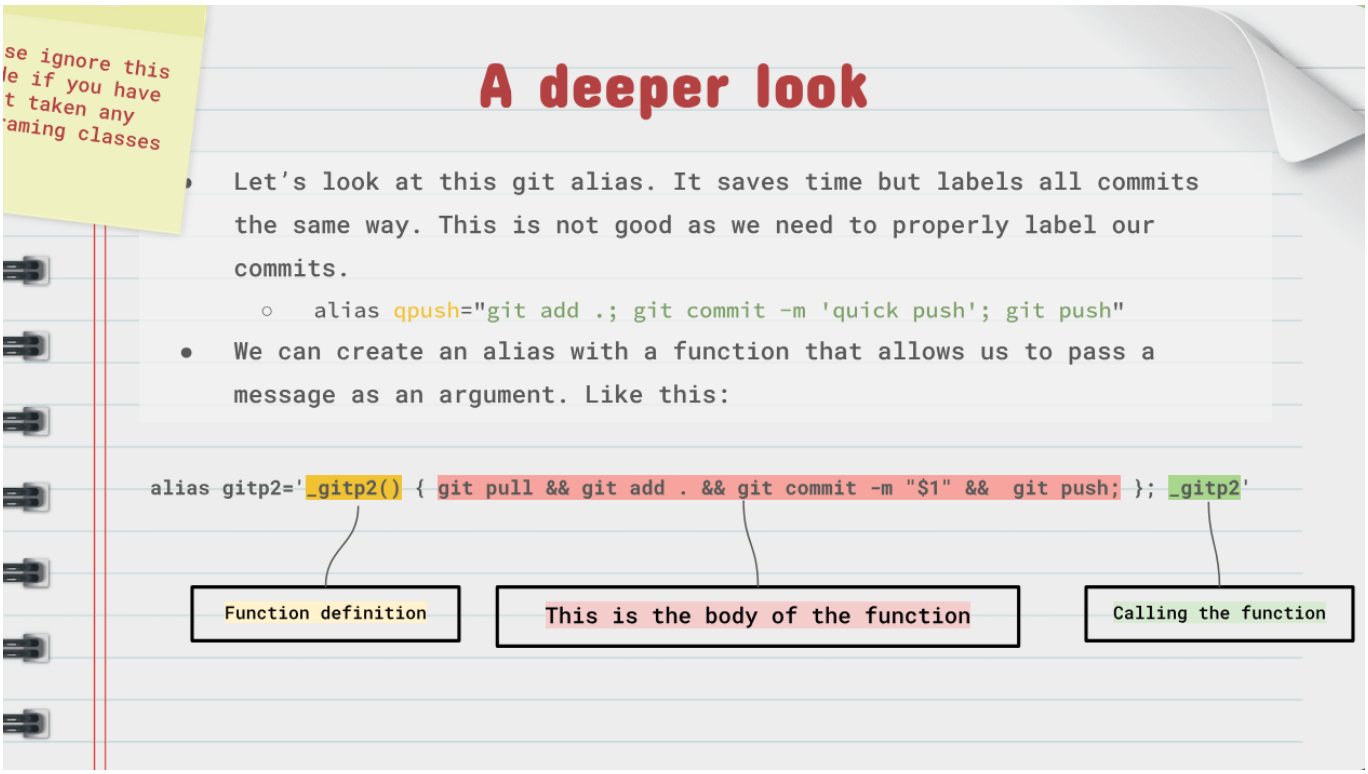
Aliases

Creating you own commands with alias

- What is an alias?
  - A shorthand for a more complicated command.
  - Alias do not persist unless you save them in your .bashrc or .bash_aliases file
- How to create an alias?
  - `alias name_of_alias="command here"`
  - To see all aliases in the system type: `alias`
- Basic Examples:
  - An alias to upgrade a linux (debian system):
    - `alias update="sudo apt update; sudo apt upgrade -y; sudo apt full-upgrade -y"`
  - An alias to clean your system from unneeded packages
    - `alias clean="sudo apt autoremove -y; sudo apt autoclean; sudo apt purge;"`

Some useful aliases

- Some useful aliases:
  - `alias add="git add ."`
  - `alias push="git push"`
  - `alias merge="git merge"`
  - `alias pull="git pull"`
  - `alias commit="git commit -m"`
  - `alias qpush="git add .; git commit -m 'quick push'; git push"`
  - `alias publicip="echo $(curl -s ifconfig.me)"`
  - `alias lA="ls -A"`
  - `alias la="ls -a"`
  - `alias ll="ls -alF"`
  - `alias lr="ls -R"`
  - `alias lsnoperm="ls -lGAgtht1 --color=always | cut -d ' ' --complement -s -f1"`
  - `alias lsops='man ls | grep '\''^[[:space:]]*[[:punct:]]'\'''`



mkdir

Command	Definition	Formula	Example
mkdir	makes directories	`mkdir` + option + `new directory(ies) name`	`mkdir ~/Downloads/games`
touch	makes files	`touch` + option + `new file(s) name`	`touch ~/Downloads/games/list.txt`
rm	removes files	`rm` + option + `files to delete`	`rm ~/Downloads/games/list.txt`
mv	moves files & directories	`mv` + option + `source path` + `destination path`	`mv ~/Downloads/games ~/Documents/`
mv	rename files & directories	`mv` + option + `old name` + `new name`	`mv ~/Documents/games/old.txt ~/Documents/games/mylist.txt`
cp	copies files & directories	`cp` + option + `sources path` + `destination`	`cp -r ~/Downloads/oldGames/ ~/Games/`

WILDCARD

POSIX class	Represents	Means	Example using `ls`
`[:upper:]`	`[A-Z]`	Uppercase letters	`ls *[[:upper:]]*` - List files with uppercase letters
`[:lower:]`	`[a-z]`	Lowercase letters	`ls *[[:lower:]]*` - List files with lowercase letters
`[:digit:]`	`[0-9]`	Digits	`ls *[[:digit:]]*` - List files containing digits
`[:alpha:]`	`[A-Za-z]`	Alphabetic characters	`ls *[[:alpha:]]*` - List files with alphabetic characters
`[:alnum:]`	`[A-Za-z0-9]`	Alphanumeric characters	`ls *[[:alnum:]]*` - List files with alphanumeric characters
`[:space:]`	`[ \t\n\r\f\v]`	Whitespace characters	`ls *[[:space:]]*` - List files with spaces in names
`[:punct:]`	Punctuation	Punctuation characters	`ls *[[:punct:]]*` - List files with punctuation characters
`[:blank:]`	`[ \t]`	Space and tab	`ls *[[:blank:]]*` - List files with spaces or tabs

`[:blank:]`	`[ \t]`	Space and tab	`ls *[[:blank:]]*` - List files with spaces or tabs
`[:xdigit:]`	`[0-9A-Fa-f]`	Hexadecimal digits	`ls *[[:xdigit:]]*` - List files with hexadecimal characters
`[:cntrl:]`	Control characters	Control characters	`ls *[[:cntrl:]]*` - List files with control characters
`[:print:]`	Printable	Printable characters	`ls *[[:print:]]*` - List files with printable characters
`[:graph:]`	Graphical	Visible characters (not spaces)	`ls *[[:graph:]]*` - List files with graphical characters
`[:word:]`	`[A-Za-z0-9_]`	Word characters (alphanumeric + underscore)	`ls *[[:word:]]*` - List files containing word characters
`[:ascii:]`	ASCII characters	All ASCII characters (0-127)	`ls *[[:ascii:]]*` - List files containing ASCII characters

There are only 3 wildcards. Here is how they work:

Wildcard	definition	example
`*`	matches 0 to any number of characters	`ls ~/Downloads/*.png`
`?`	matches 1 character	`ls ~/Downloads/f?ll.sh`
`[]`	matches 1 character from a set	`ls ~/Downloads/f[0-9]ll.sh`

Examples:

- List all the files that contain a 4 letter file extension.
  - `ls -1X wildcard_extra_practice/*.????`
- List all the files that contain a 4 letter file extension and start with letter i:
  - `ls -1X wildcard_extra_practice/i*.????`
- List all th files Microsoft Office 365 files.
  - `ls -1X wildcard_extra_practice/*.???x`

LS

ls command main options:

option	description
`ls -a`	list all files including hidden file starting with '.'
`ls --color`	colored list [=always/never/auto]
`ls -d`	list directories - with ' */'
`ls -F`	add one char of */=>@| to enteries
`ls -i`	list file's inode index number
`ls -l`	list with long format - show permissions
`ls -la`	list long format including hidden files
`ls -lh`	list long format with readable file size
`ls -ls`	list with long format with file size
`ls -r`	list in reverse order
`ls -R`	list recursively directory tree
`ls -s`	list file size
`ls -S`	sort by file size
`ls -t`	sort by time & date
`ls -X`	sort by extension name