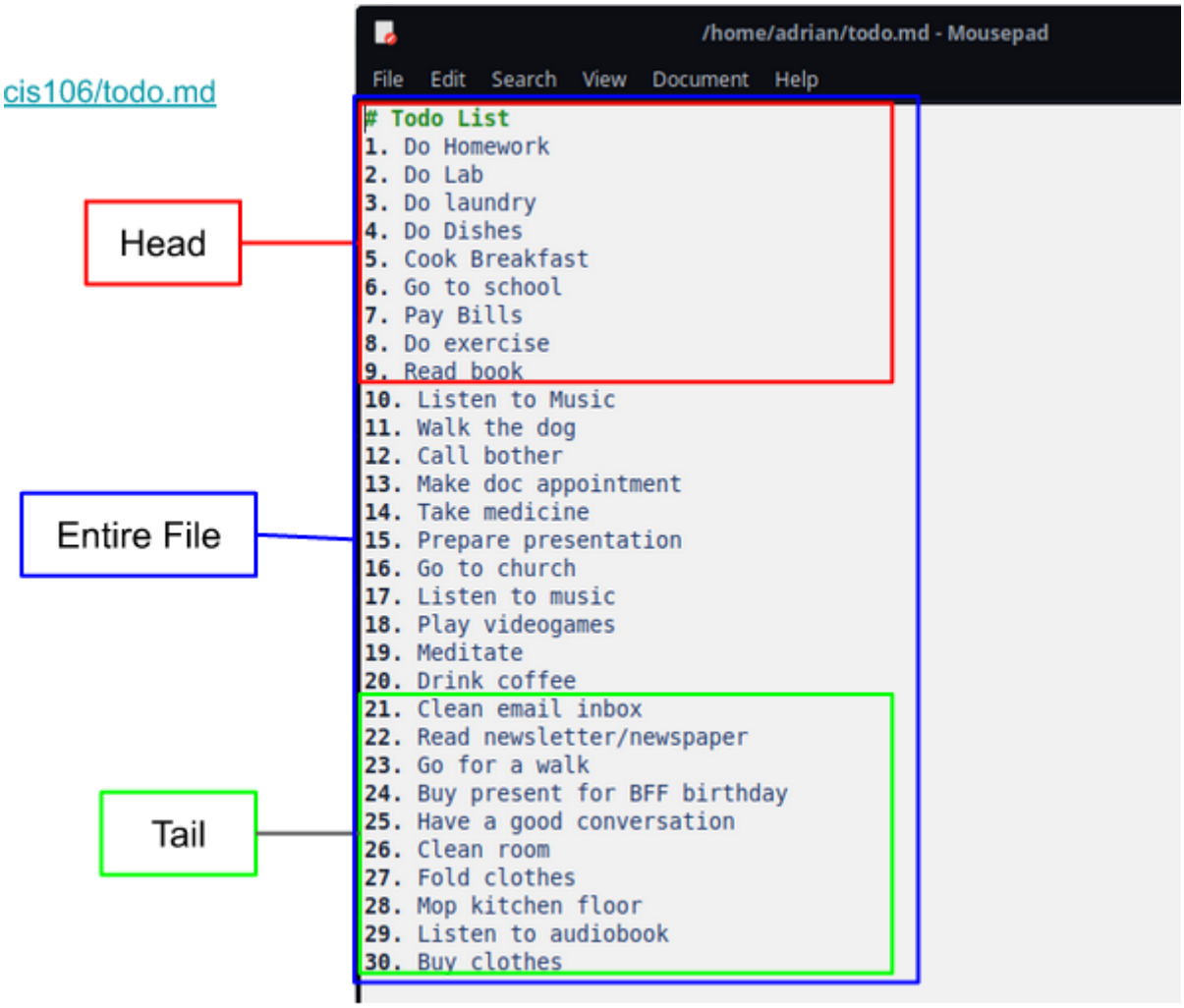
Handling Text Files

- There are a lot of commands for handling text on Linux

Cat
Tac
More
Less
Head
Diff

Tail
Cut
Paste
Sort
Wc
Tr
Grep



- The **Cat** command is used for displaying the content of a file.
- The word **Cat** does not refer to the animal, instead it is short for **concatenate** which means joining two strings together

- `cat + file to display`
- `cat + file 1 + file 2`

```
raalberto@cis106:~$ cat food
pizza
rice
potatoes
raalberto@cis106:~$ cat food drinks
pizza
rice
potatoes
soda
water
juice
raalberto@cis106:~$
```

Display the content of a file with line numbers

```
cat -n /etc/passwd
```

Display the content of a file with line numbers excluding empty lines

```
cat -b /etc/resolv.conf
```

read th
commanc
to reac
passwd, r

Display a $ at the end of every line

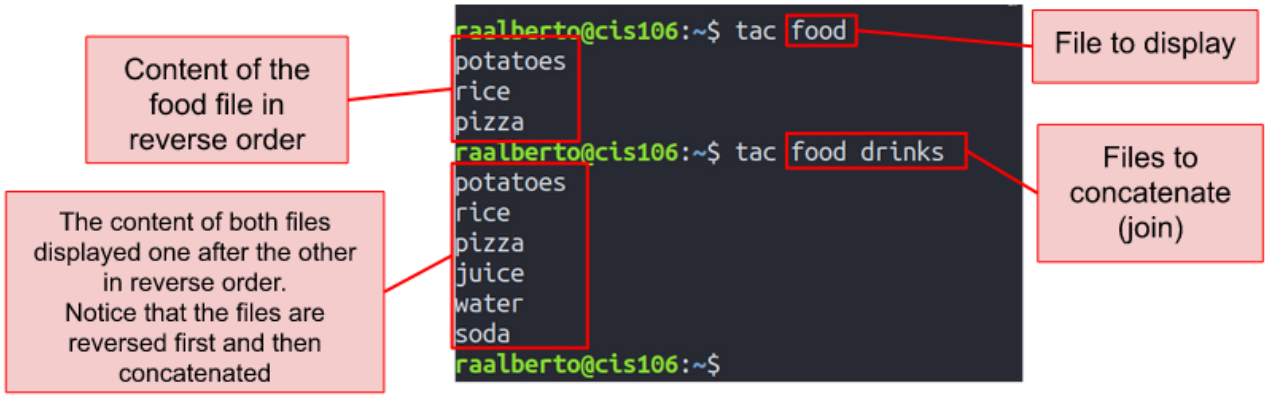
```
cat -E /etc/group
```

Display the content of a file suppressing repeating empty lines to a single empty line

```
cat -s /etc/hosts
```

- The **Tac** command is used for displaying the content of a file in reverse
- The command can also cat commands in reverse

- `tac + file to display`
- `tac + file 1 + file 2`



- The **More** command is used for displaying the content of a text file one page at a time

- Usage:
  - `more + file to view`
- For getting help navigating the menu press h
- Examples of the more command:

Open a file and display guiding information in the bottom

```
more -d /var/log/syslog
```

Open a file 10 lines at a time

```
more -10 /var/log/syslog
```

For more information, read the man page of the more command

- The **Less** command displays the content of a file 1 page at a time, it helps greatly since when dealing with large files using the command loads 1 page at a time

- Usage:
  - less + file to view
- Examples of the less command:

Open a file with line numbers

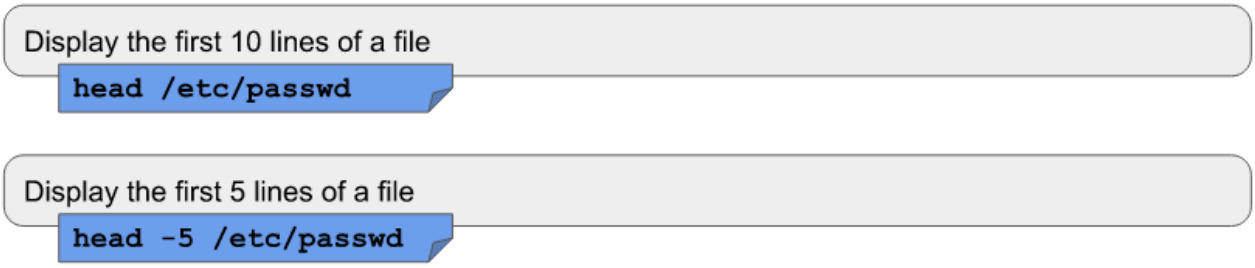
```
less -N /var/log/syslog
```

Open a file at the beginning of the first occurance of a string

```
less -p "nobody" /etc/passwd
```

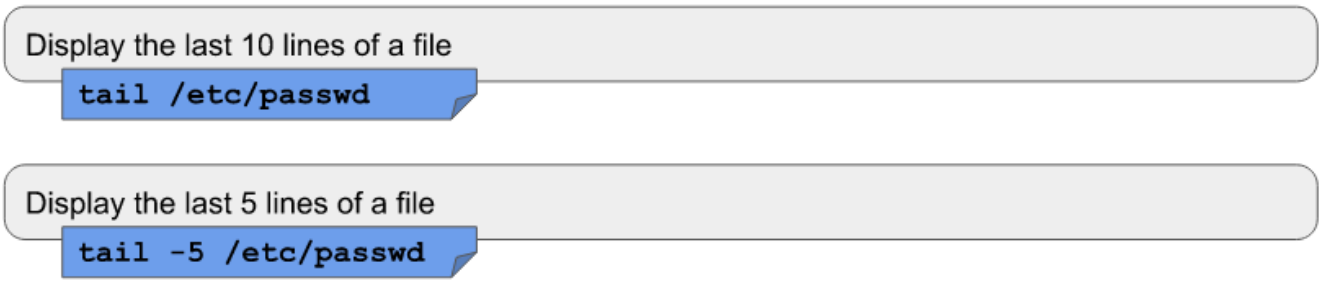
- The **Head** command displays the top number of lines of a given file. It prints the first 10 lines by default

- Usage:
  - **head + option + file**
- Examples of the head command:



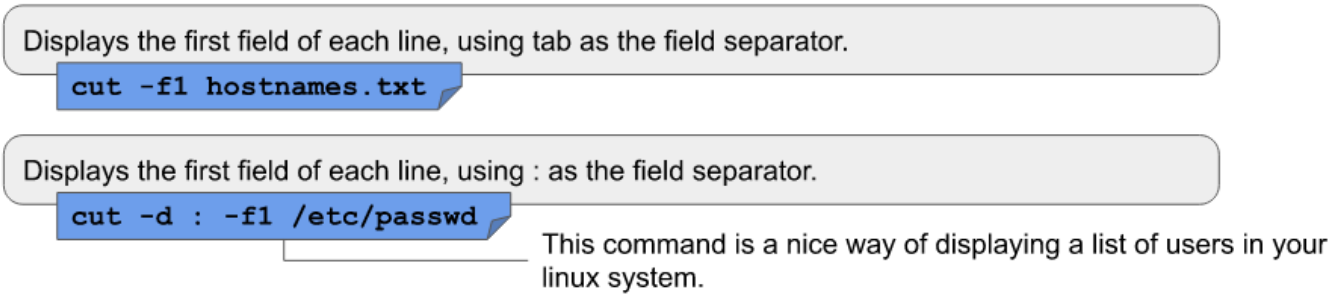
- The **Tail** command does the same as the **Head** command but backwards. It displays the last 10 lines of a given file

- Usage:
  - **head + option + file**
- Examples of the head command:



- The **Cut** command is used to extract a section of a file and display it

- Usage:
  - **cut + option + file**
- Examples of the cut command:



- The **Paste** is used for joining files horizontally in columns

- Usage:
  - `paste + option + files`
- Examples of the paste command:

Merge two files

```
paste users.txt ips.txt
```

Merge two files using a different delimiter

```
paste -d ":" users.txt ips.txt
```

Merge files sequentially instead of horizontally

```
paste -s users.txt ips.txt
```

- The **Sort** is used for sorting files as the name implies
- It sorts information in a particular order
- it sorts the contents of a text file line by line and supports other forms of sorting such as
  - Alphabetically
  - reverser order
  - by number
  - by month
- it can be user for sorting by column number too
- it can be used ignoring case sensitivity
- it can run whatever file is sorted
- it identifies spaced as a default operator

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.

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.

Assuming you have a file with a list of users. Sort the file.

```
sort users.txt
```

```
adrian@G752VL:~$ cat users.txt
users           name            email
aarias          arnold          aarias@email.net
rgerald         ray             rgerald67@parks.com
lemma           liam            lemma_@grmail.com
nolivia         nadine          supernah@citizen.com
wava            william         wava_1988@recreation.com
jisabella       james           jisabella@games720.com
adrian@G752VL:~$ sort users.txt
aarias          arnold          aarias@email.net
jisabella       james           jisabella@games720.com
lemma           liam            lemma_@grmail.com
nolivia         nadine          supernah@citizen.com
rgerald         ray             rgerald67@parks.com
users           name            email
wava            william         wava_1988@recreation.com
adrian@G752VL:~$
```

Sort the file and save the output to a new file

```
sort -o usersSorted.txt users.txt
```

```
adrian@G752VL:~$ sort -o usersSorted.txt users.txt
adrian@G752VL:~$ cat usersSorted.txt
aarias          arnold          aarias@email.net
jisabella       james           jisabella@games720.com
lemma           liam            lemma_@grmail.com
nolivia         nadine          supernah@citizen.com
rgerald         ray             rgerald67@parks.com
users           name            email
wava            william         wava_1988@recreation.com
adrian@G752VL:~$
```

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 phoneNumbers.txt
```

Check if a file is sorted

```
sort -c usersSorted.txt
```

Sort and remove duplicate entries

```
sort -u users.txt
```

```
adrian@G752VL:~$ sort -r users.txt
wava            william         wava_1988@recreation.com
users           name            email
rgerald         ray             rgerald67@parks.com
nolivia         nadine          supernah@citizen.com
lemma           liam            lemma_@grmail.com
jisabella       james           jisabella@games720.com
aarias          arnold          aarias@email.net
adrian@G752VL:~$ sort -k 2 users.txt
aarias          arnold          aarias@email.net
jisabella       james           jisabella@games720.com
lemma           liam            lemma_@grmail.com
nolivia         nadine          supernah@citizen.com
users           name            email
rgerald         ray             rgerald67@parks.com
wava            william         wava_1988@recreation.com
adrian@G752VL:~$ sort -k 3 users.txt
aarias          arnold          aarias@email.net
users           name            email
jisabella       james           jisabella@games720.com
lemma           liam            lemma_@grmail.com
rgerald         ray             rgerald67@parks.com
nolivia         nadine          supernah@citizen.com
wava            william         wava_1988@recreation.com
adrian@G752VL:~$ sort -c usersSorted.txt
adrian@G752VL:~$ sort -u usersSorted.txt
aarias          arnold          aarias@email.net
jisabella       james           jisabella@games720.com
lemma           liam            lemma_@grmail.com
nolivia         nadine          supernah@citizen.com
rgerald         ray             rgerald67@parks.com
users           name            email
wava            william         wava_1988@recreation.com
adrian@G752VL:~$
```

- The **WC** command is used for printing the number of lines, characters and bytes ina a file
  - `wc + option + file`
- Example of the wc command

Display the number of bytes in a file

```
wc -c users.txt
```

Display the number of lines in a file

```
wc -l users.txt
```

Display the number of characters in a file

```
wc -m users.txt
```

Display the number of words in a file

```
wc -w users.txt
```

- The **TR** command is used for translating or deleting characters from standard output

- `Standard output | tr + option + set + set`

- Examples of the tr command:

Translate one character to another. For example a period with a comma.

```
cat file.txt | tr '.' ',' file.txt
```

Translate white space into tabs. Useful with python programs.

```
cat program.py | tr "[:space:]" '\t'
```

Translate tabs into space. Again, useful with python ~~headaches~~ programs!

```
cat file.py | tr -s "[:space:]" ' '
```

Check out:

- the **Diff** command compared files and displays their differences between them

- Usage:
  - `diff + option + file1 + file2`
- Examples of the diff command:

Display the difference between two files

```
diff file1 file2
```

Display the difference between two files only if the files are different

```
diff -q file1 file2
```

Display the difference between two files in a column format

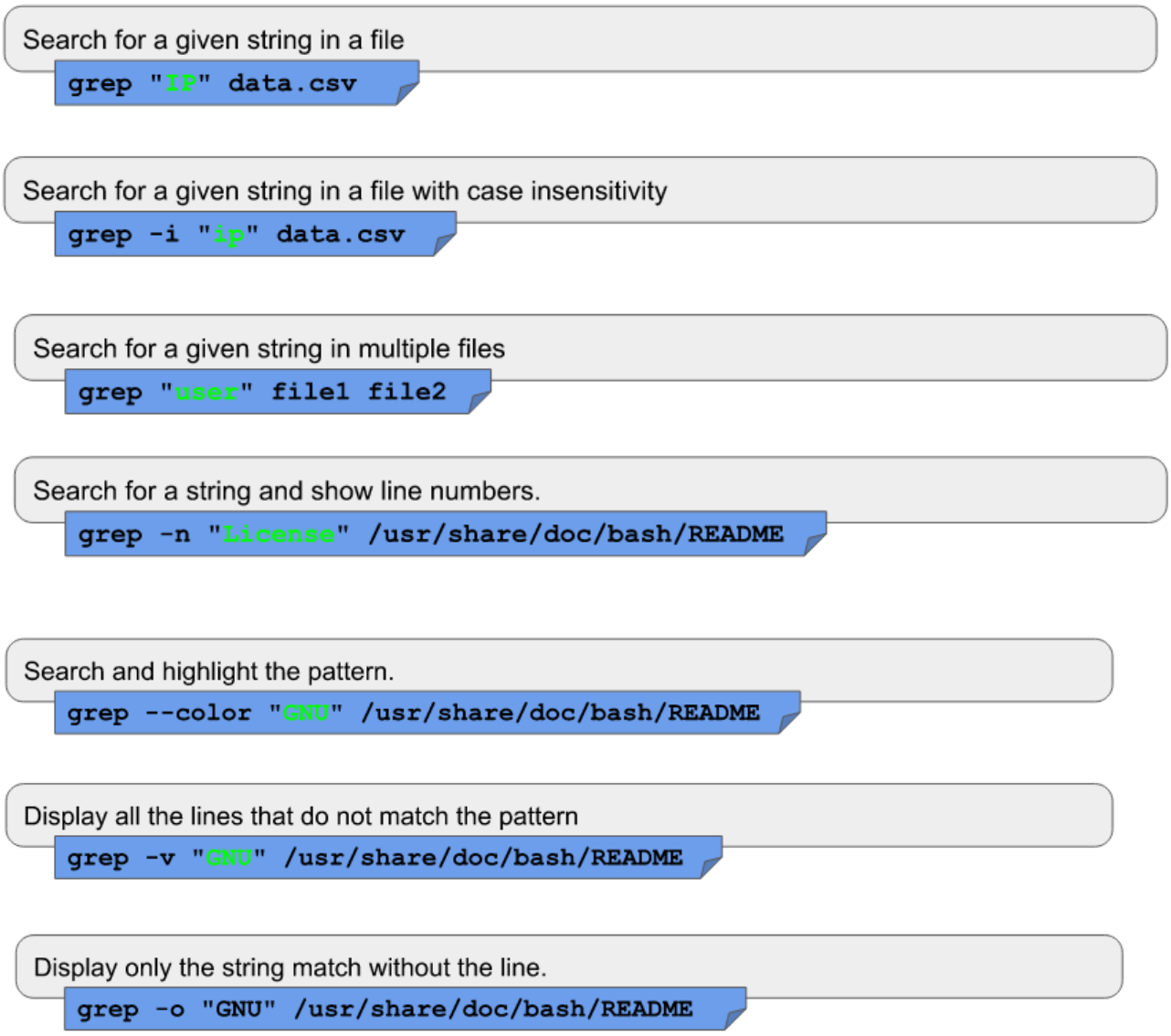
```
diff -y file1 file2
```

- The **GREP** command is used to match a string pattern from a file or standard output using a pipe

- Usage:
  - `grep + option + pattern to match + file`
  - `Standard output + pipe (|) + grep + pattern to match`
- Common options of the grep command

Option	Explanation
-i	Turns case sensitivity off
-n	Displays line number of the each line matched
-E	Treats the pattern as an extended regular expression
-G	Treats the pattern as a basic regular expression
-v	Inverts the search
-o	Only display the string matched

Check: https://robertalberto.com/linuxcommands/commands/grep.html



Check: https://robertalberto.com/linuxcommands/commands/grep.html

- the **REV** command is sued for reversing the characters position in a given text
- Its used byt typing **rev user.txt**

Working with I/O Redirection

- Input and output of commands can be riderected to and from files and multiple commands can be used together using pipelines
- since everything in linux is a file, programs sent their output to a file called **SDOUT** and error messages to **STDERR**
- files are linked to the screen by default meaning that they are not saved into a file and are displayed in the terminal
- all input is sent to **STDIN** and is attached to the keyboard in the same way **STDOUT** and **STDERR** are attached to the display by default
- redirection allows users to change where the output goes and where it comes from

Standard file description

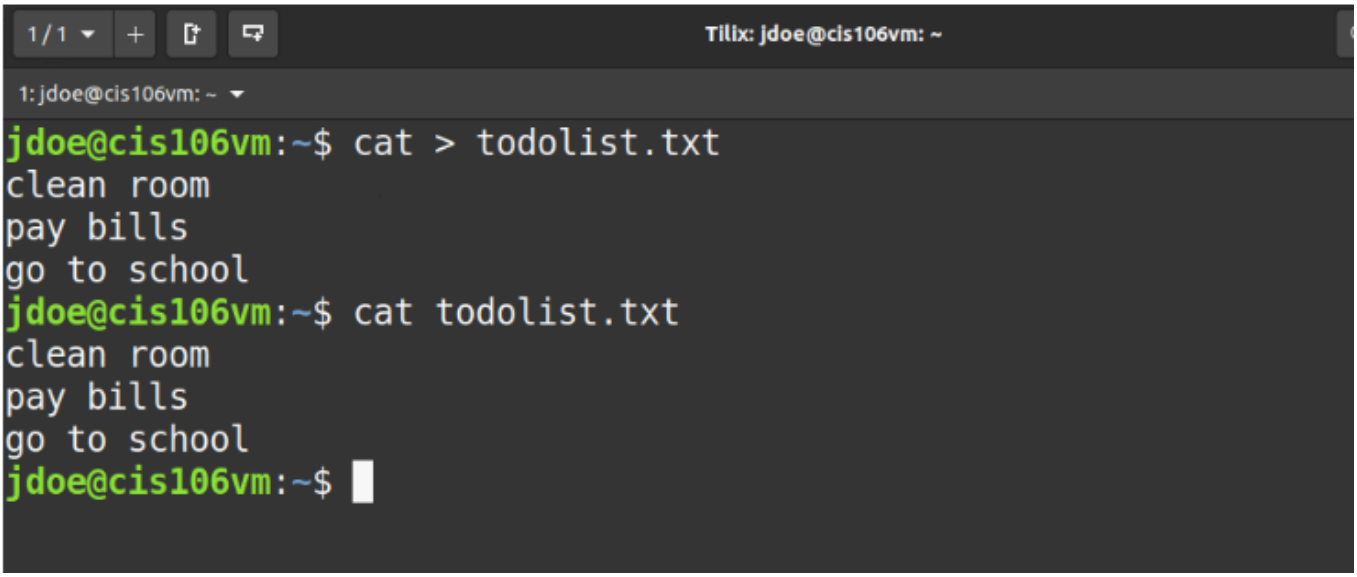
- file descriptors are positive integers used for identifying open files in a given session
- 9 files at a time are allowed for each descriptor
- bash reserves the first 3 file descriptors
- they are used for directing the input and output of commands

- To redirect standard output, we use: >
  - **Example:**
    - `ls -lax > list_of_files.txt`
- To redirect standard error, we use: 2>
  - **Example:**
    - `cat badFile.txt 2> error_cat_command.txt`
- To redirect standard output and append the output to a file, we use: >>
  - **Example:**
    - `ls -1alh >> list_of_files.txt`
- We can use the output redirection to create an empty file:
  - **Example:**
    - `> newfile`
    - `: > newfile` (in older versions of bash)
- We can also get rid of output that we do not want:
  - **Example:**
    - `ls -l ~/Downloads ~/documents 2> /dev/null`

- It is possible to redirect both output and error to the same file. There are two ways:
- The old way
  - `command > file_to_save 2>&1`
  - **Example:**
    - `ls ~/Downloads ~/documents > output.txt 2>&1`
  - Using this method, the redirection of standard error must always occur after redirecting standard output or it doesn't work.
- The new way supported in bash version 4+
  - `command &> file_to_save`
  - **Example:**
    - `ls ls ~/Downloads ~/documents &> output.txt`
  - You can also redirect both and append
    - `ls ls ~/Downloads ~/documents &>> output.txt`

- In the absence of a filename arguments, cat copies input and output and display it in the terminal

How to create a file with cat



Redirect a command error to a file

```
ls -l wrongfile 2> error.txt
```

Sometimes you want to store the data and the errors in case any is given. Specially working with scripts.

```
ls -l goodfile badfile 1> longlist.txt 2> errors.txt
```

You can also discard all standard output data by sending it to the null file, which is a file that contains nothing. The null file is also called "the black hole".

```
ls -l badfile > /dev/null
```

Also if you only want to discard error messages

```
ls -l badfile goodfile 2> /dev/null
```

- The pipe **|** allows you to redirect the output of a command to the input of another

- Usage:
  - `command 1 + | (pipe) + command 2 + | (pipe) +command #`
- Examples of the pipe:

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:]]"
```

Display all 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}'
```

Alias

- alias in linux are shorthands of more complicated commands

How to create an alias?

- `alias name_of_alias="command here"`

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 obtain a computers public ip:
  - `alias publicip="curl ifconfig.me && echo ''"`
- Some useful git aliases:
  - `alias add="git add ."`
  - `alias push="git push"`
  - `alias merge="git merge"`
  - `alias pull="git pull"`
  - `alias checkout="git checkout -b"`
  - `alias commit="git commit -m"`
  - `alias qpush="git add .; git commit -m 'quick push'; git push"`

- before creating an alias, it is best to check to see if the words you are choosing is already reserved
- you can use the **type** command to find out if its reserved or not
- if you make an alias to a reserved word, the system may break since the original command will not be used when using the alias
- to make aliases permanent you can place them on either at the end of the **.bashrc** file or in Ubuntu, place it in the **.bash_aliases** located in the home directory
- they are good for remembering difficult commands
- to remove an alias simply type **unalias *name***