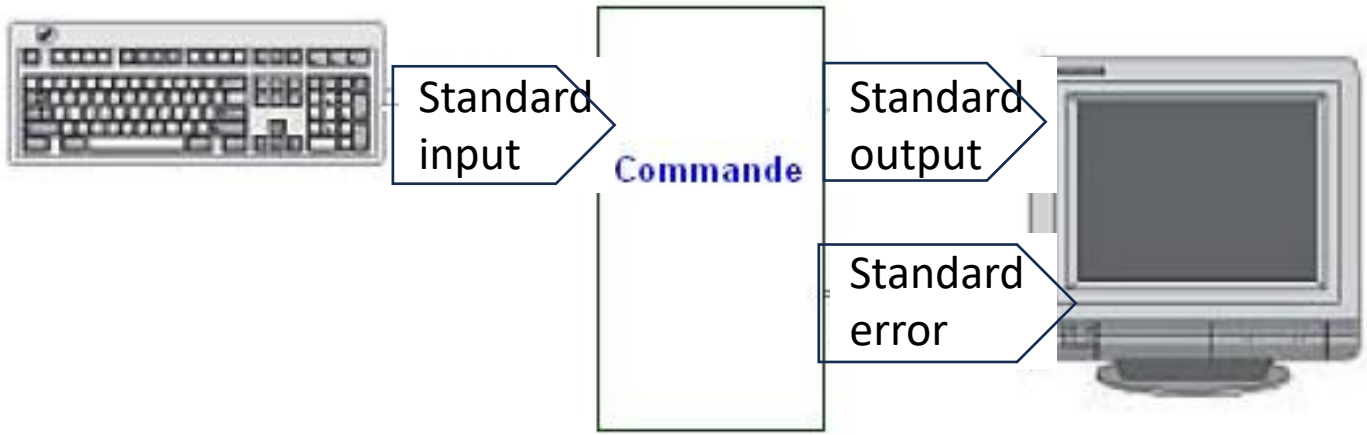
Input/Output Redirections

The inputs/outputs

- All commands use input/output mechanisms:
  - ✓ to read data (input flow) from the keyboard
  - ✓ To transmit their results or execution errors (output flow) on the screen
- There are three kinds of input/output or data flow;



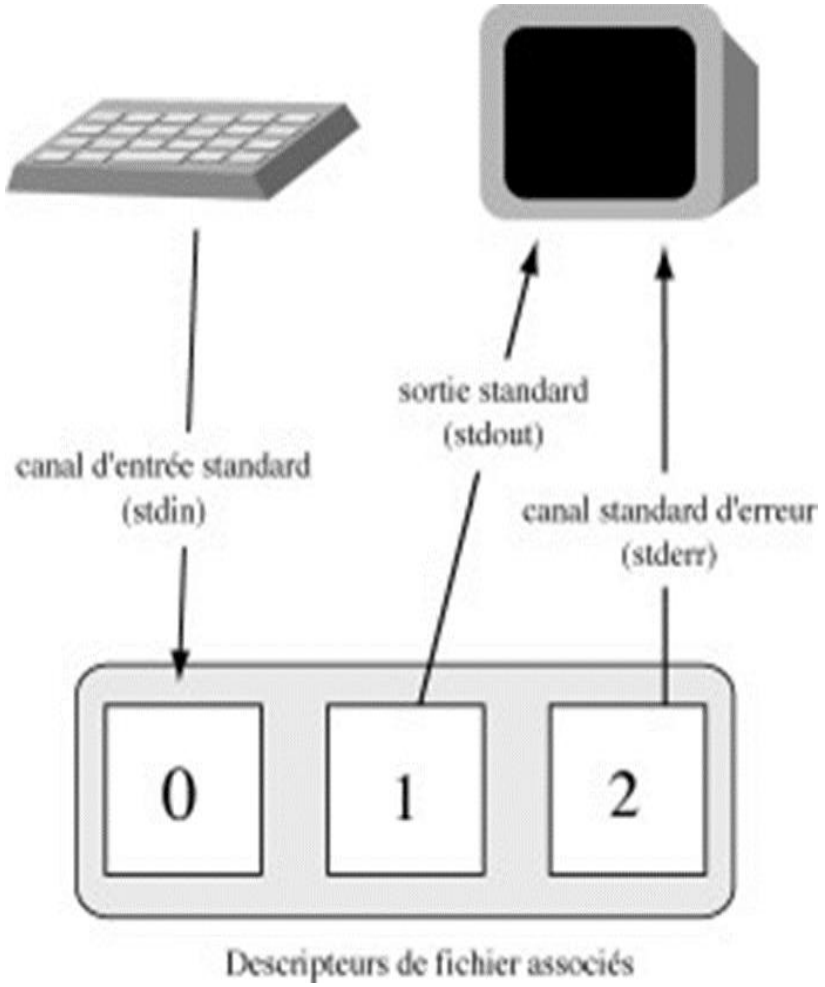
The inputs/outputs

- Each of these data streams is identified by a number descriptor:
  - 0 for standard input
  - 1 for standard output of results,
  - 2 for standard output of error messages
- The shell uses three files located in/dev:

  • stdin (0): Standard input associated with keyboard

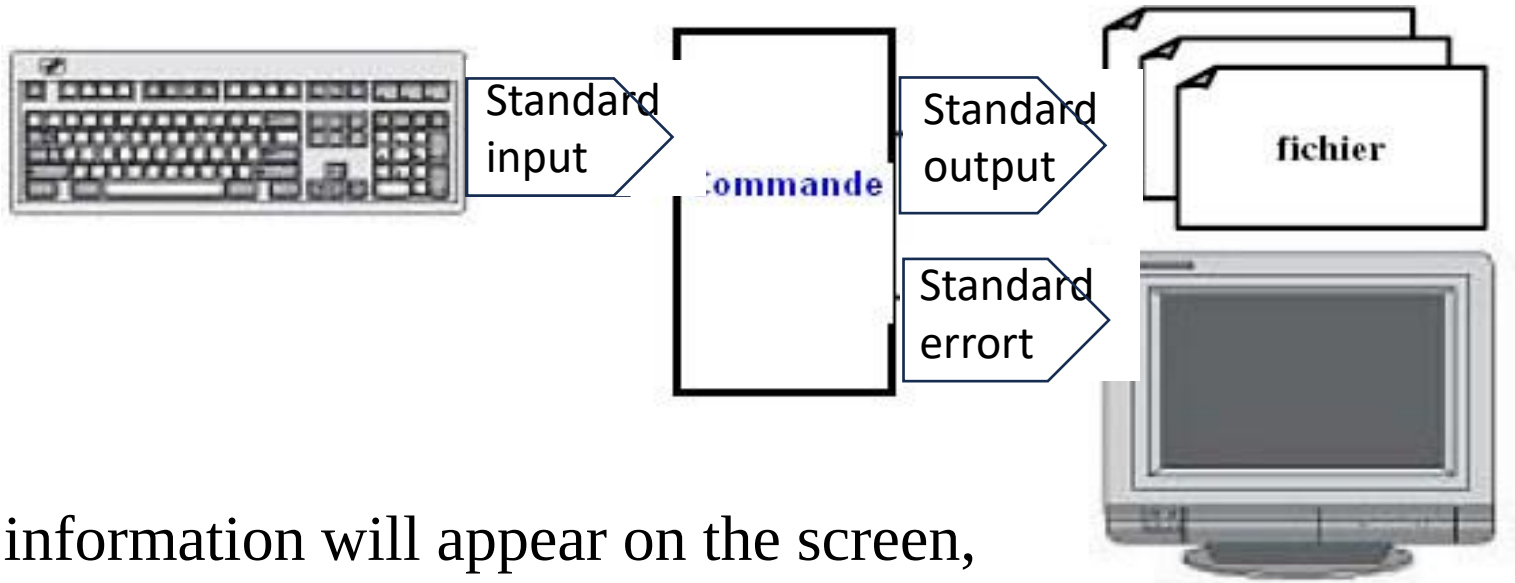
  •stdout (1): Standard output associated with the screen

  • stderr (2): Error output associated with the screen



Principle of Input/Output Redirection

✓ Standard output redirection is to return the text which should appear on the screen to a file.



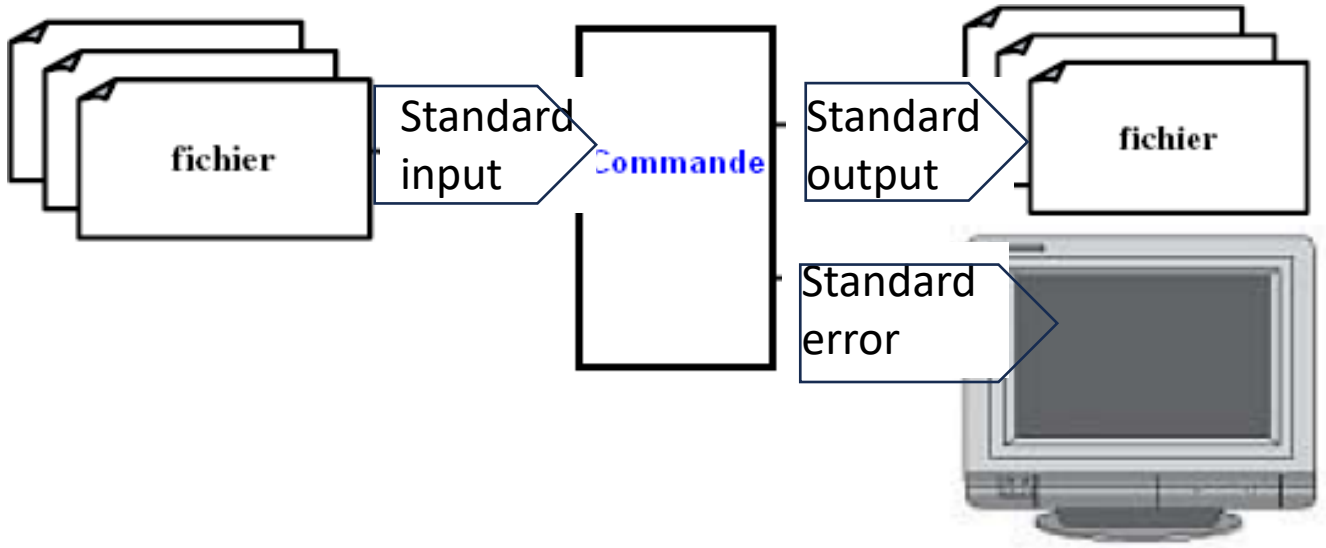
- No result information will appear on the screen,

apart from error messages that should appear on the screen

typed on the keyboard

Principle of Input/Output Redirection

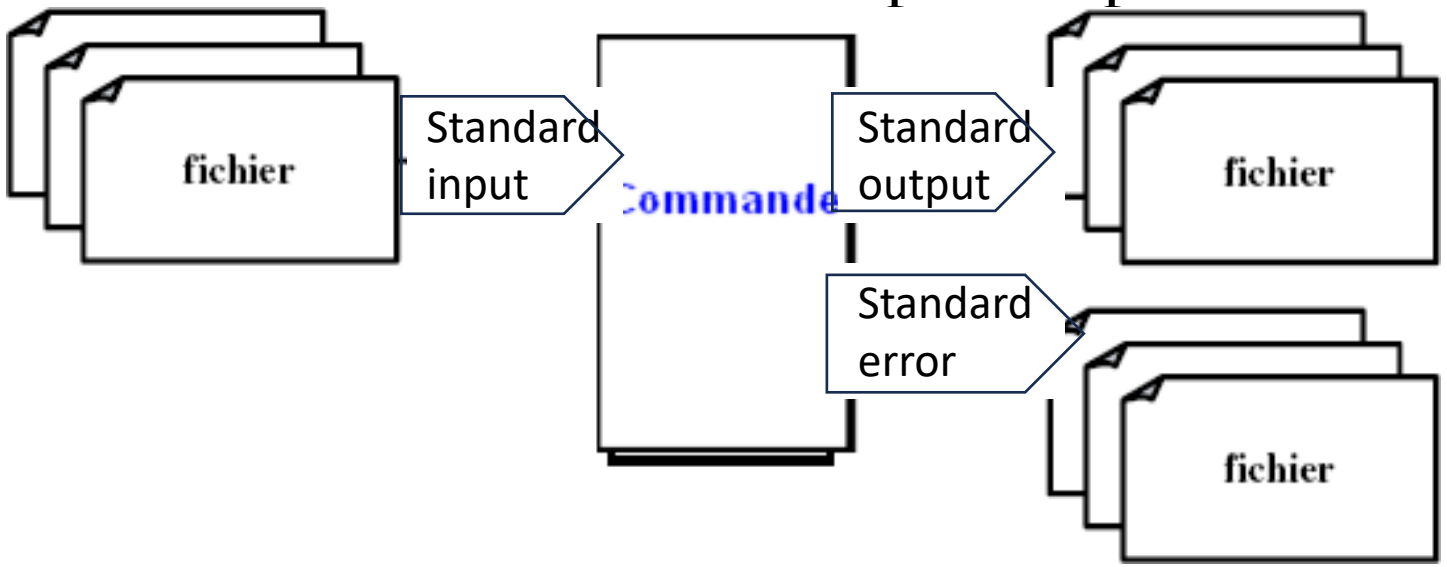
- The redirection of the standard input consists of retrieving the parameters that are usually typed on the keyboard from a file .



- Retrieve information from a file, not from the keyboard
-

Principle of Input/Output Redirection

- In fact, it is possible to redirect all standard input-output of a command



➢ Therefore, the command :

➢Retrieves the information it needs from a file;

➢Writes information in files (results or error messages)

Redirection des E/S standards

- To redirect standard entries/outputs , we use :
- the symbol < to redirect standard input

read from a file instead of the keyboard,

- the symbol > to redirect the standard output :

write to a file instead of on the screen

❑To ADD a data stream to the end of an existing file, use >> instead of the > symbol

equals

Standard Output Redirection

- The syntax is as follows :

  Command > file_out

  - If the file does not exist, it will be created .
  - If the file already exists, it will be overwritten .

- ⇨ command > file equals to command 1> file

- Example
- ls -l /usr/include >list

❑ The result of the ls command **will not be displayed** on the screen,

❑ The list file will **be created**,

❑ It contains the details of the files that exist in /usr/include

Add the /usr/include tree to the list file

Standard Output Redirection

- The double redirection (symbol >>) is used for adding text to a file.
- The syntax is as follows :
  - command >> file_out
- If the file does not exist, it will be created .
- If the file already exists, it will not overwrite; and The new entries will be added at the end of the file.
- Example:
- tree /usr/include >>list

Add the /usr/include tree to the list file

Error redirection

- A command can generate errors during execution;

It sends a message to the standard error output (the screen).

The error output has for descriptor 2:stderr (2).

- The syntax is given as follows
- Command 2> error_file :
- ➔**interest :**
- You can save your error messages to a file for the internal analysis (case of a command or a program that displays a large number of error messages).

Example of output redirection (error cases)

- You are running this command: ls **myfile.txt**
  - It is assumed that the file « myfile.txt" does not exist
  - This error message is displayed at the end :
    - myfile.txt not such file or directory
- be this command:

ls myfile.txt 2> error.file

✓ No error messages will be displayed on the screen;

✓ The error.file **file will be created** ;

✓ the error message " not such file or directory" will be saved in **the error.file file**

Input redirection

- Commands need input parameters to run.
- Commands can read their data from a file instead of typing them by keyboard

->This leads to redirecting the standard input .

- The syntax is as follows : command < file_in

Example

- sort command allows to sort data.

ls –l < /etc/passwd

->the , and the result will be displayed on screen.

Combination of multiple redirects

- You can make several redirections at the same time, as follows:

➢ command > file_OUT 2> file_ERROR

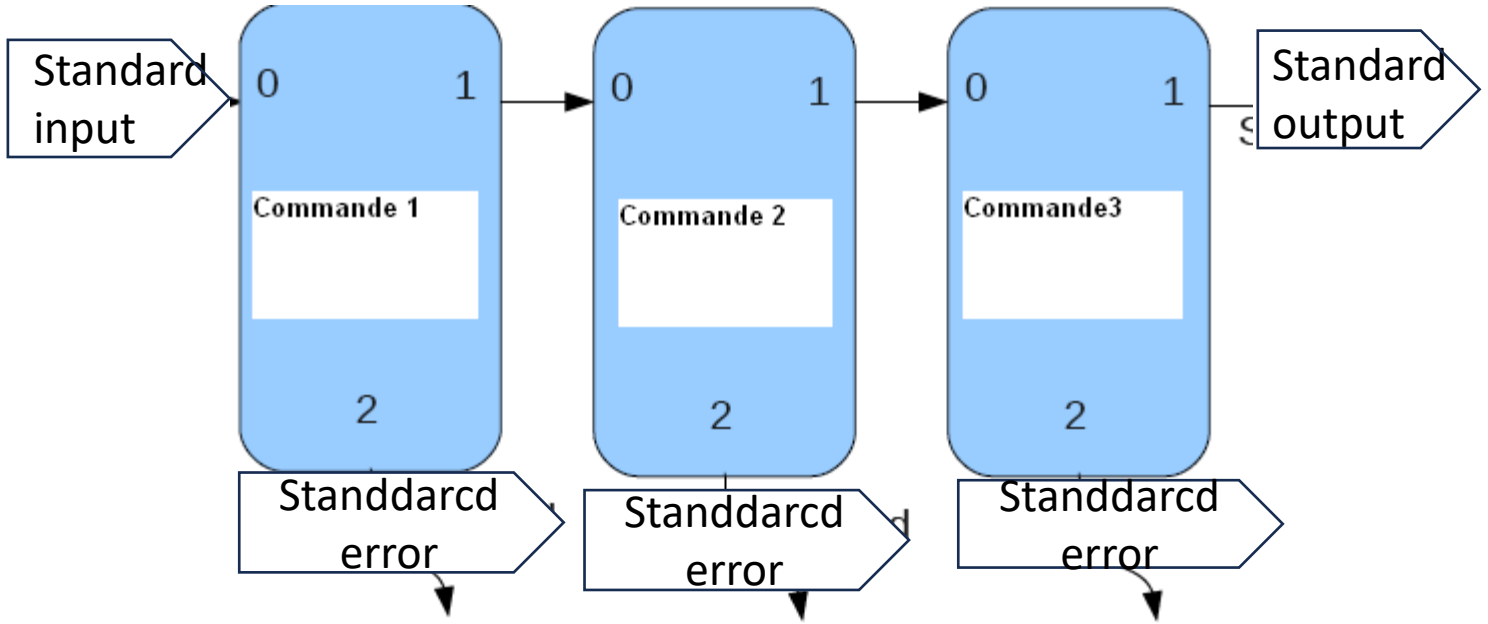
➢ command < file_IN > file_OUT

- Examples:

- sort < /etc/passwd > **sorted**
- To sort the / etc/passwd and get the result in the file named **sorted**
- ls –l > **my_list** 2> **ERR**
- To insert the details of the files in the " **my_list** " file
- and error messages in the " **ERR" file**

The tubes (pipe-line)

- You can use the results of a command (standard output descriptor1) to reinject as inputs of another command (standard input descriptor 0)without going through an intermediate file.



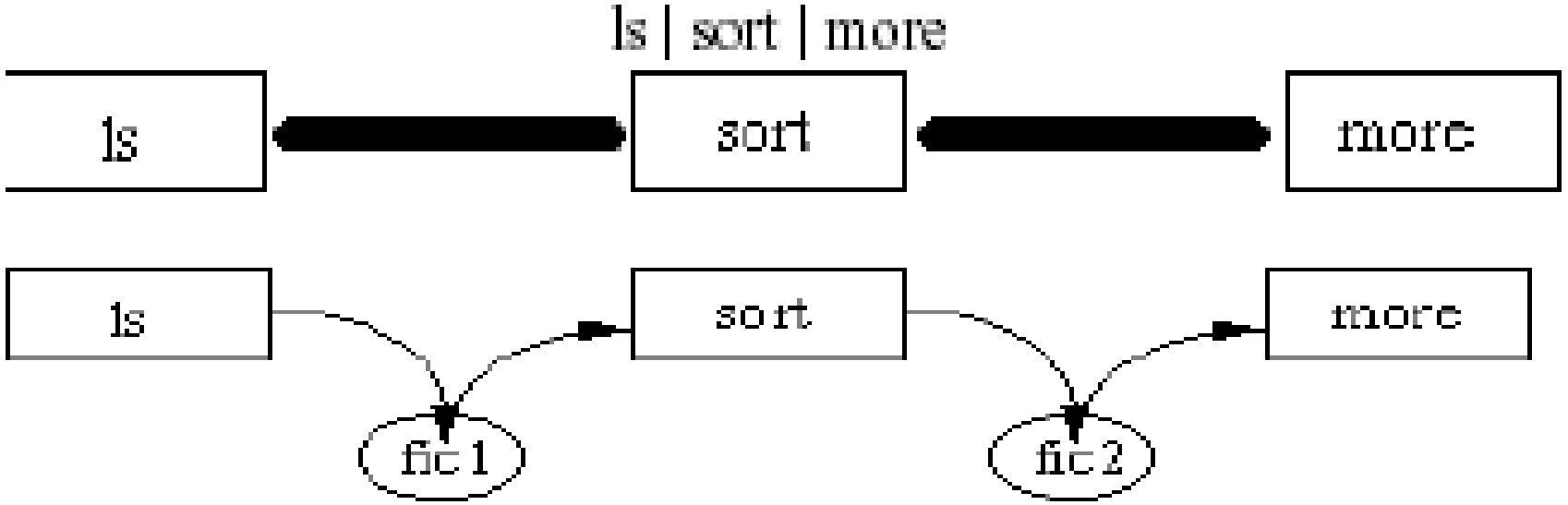
The tubes (pipe-line)

- For the redirection of the output of a command to another command, we use the the symbol '|'
- you can get this symbol by pressing the "Altgr + 7" keys
- The combination is as follows: command 1 | command2
- For example :
- The more **command** displays a data stream page by page
- **tree /usr/include | more**
- Page view of the / **usr /include tree**
- You can combine several pipes as much as you want
- **command1 | command2 | command3 ... | commandN**

The tubes (pipe-line)

- Example:
- We want to display the list of files in the current directory sorted by name and get a page-by-page view.
- The combination of commands is as follows:

ls | sort | more



command sequencing

•One can chain commands that will execute independently.

- We use the character ‘;’ which is a command separator.
- Example:

The **echo** command displays text on the screen

The whoami **command** displays the name of the currently logged-in user.

date; echo "Hello, I am:"; whoami

The result will be displayed as follows:

Sat Nov 11 14:56:12 +01 2023

Hello, I am

meskaldji

command chain

DO NOT MIX UP!!!!!!

➔ **(echo start; date ; echo end) > FicP**

which does not display anything on the screen and will create the FicP having this content :

start

Sat Nov 11 15:01:06 +01 2023

end

With!!!!!!!!!!!!!!!

➔**echo debut ; date; echo fin > FicP**

Which displays

debut

Sat Nov 11 15:04:56 +01 2023

- and then it will create the FicP file with content just the word «end»