Programmation et projet encadré - L7TI005

Shell Unix

Crédits supports : Serge Fleury

Yoann Dupont prenom.nom@sorbonne-nouvelle.fr
Pierre Magistry pierre.magistry@inalco.fr

2024-2025

Université Sorbonne-Nouvelle
INALCO
Université Paris-Nanterre

Introduction

Avant propos

- Unix ?
- Linux ?
- GNU/Linux ?
- Système d'exploitation
  - kernel
  - shell
  - applications
- logiciels et licences

Philosphie derrière Unix

Existence et importance du système de fichiers

Tout est fichier

- dans les années 60, ce n'était pas évident,
- même les programmes, les périphériques, le réseau, ...
- simplicité $\rightarrow$ on ne pense qu'en termes de fichiers et dossiers,
- d'où l'importance de bien les organiser !

Philosphie derrière Unix

Existence et importance du système de fichiers

Tout est fichier

- dans les années 60, ce n'était pas évident,
- même les programmes, les périphériques, le réseau, ...
- simplicité $\rightarrow$ on ne pense qu'en termes de fichiers et dossiers,
- d'où l'importance de bien les organiser !

Un ensemble de commandes pour manipuler les fichiers et leur contenu.

(depuis **la ligne de commandes**)

- Chaque commande doit faire **une** chose et la faire **bien** (K.I.S.S.)
- Chaque commande est pensée pour pouvoir interagir avec les autres.
- On fait des choses complexes en combinant des commandes simples. (*pipelines* ou *scripts*)

Le Système de fichier

Mots clefs de la section

à définir

- fichier
- dossier (ou répertoire)
- dossier «parent»
- arborescence
- racine
- dossier personnel
- dossier courant ou «de travail» (*working directory*)
- chemin absolu
- chemin relatif
- caractères de remplacement (jokers ou *wildcards*),

Le système de fichier – L'arborescence

Points clefs

- les **fichiers** sont dans un **dossier**
- pouvant eux même être dans un **dossier** « parent »
- ce qui forme une **arborescence**

Le système de fichiers – Les chemin

L'ensemble des données sur la machine sont regroupées en une seule **arborescence**, il est intéressant de pouvoir identifier les fichiers et les dossiers par un **chemin** dans l'arbre.

- **/** désigne la **racine** de l'arbre
- **~/** désigne le dossier personnel de l'utilisateur ("*HOME*")
- **./** désigne le dossier courant (*working directory*)
- **../** désigne le dossier parent

un **chemin** est formé par une suite de noms de dossiers séparés par des /, pouvant se finir par le nom d'un fichier.

Le système de fichiers – chemins absolus et relatifs

un chemin absolu indique la position d'un fichier en partant de la **racine**.

ex: `/home/pierre/PPE1`

un chemin relatif indique la position d'un fichier en partant du dossier courant.

ex: `../../dev/input/mouse3`

$\rightarrow$ habituez vous à toujours être capable de donner le chemin relatif et absolu vers les fichiers que vous manipulez.

Le système de fichier – Jokers ! (*wildcards*)

ou « caractères de remplacement »

Dans un chemin, caractères **?** et ***** ont un comportement spécial.

? peut remplacer n'importe quel caractère (unique)

***** peut remplacer n'importe quelle suite de caractères

$\rightarrow$ le chemin peut alors désigner plusieurs fichiers !

À tester avec la commande `ls`.

Mots clefs de la section

à retenir

- fichier
- dossier (ou répertoire)
- dossier «parent»
- arborescence
- racine
- dossier personnel
- dossier courant ou «de travail» (*working directory*)
- chemin absolu
- chemin relatif
- caractères de remplacement (jokers ou *wildcards*),

Confirmez que vous maîtrisez ces notions dans votre journal, ou exprimez y vos doutes. Et posez des questions la semaine prochaine !

Les Commandes

Mots clefs de la section

à découvrir

- commande
- options
- arguments
- page de man(uel)
- et moultes commandes à découvrir

Les commandes

Les commandes sont des fichiers comme les autres,

- qui ont la propriété d'être exécutables
- qui sont placés par convention dans un dossier où le système sait les trouver (typiquement `/usr/bin/`)

La syntaxe d'une commande

```
nom [-options...]  [arguments...]
```

- les **options** peuvent avoir une forme courte (`-o` avec un seul tiret)
- ou une forme longue (`--option` avec deux tirets).
- Les **arguments** sont typiquement des chemins vers des fichiers (mais pas toujours).
- La première action de l'interpréteur de commande est de *tokeniser* la ligne pour découper le nom de la commande, les options et les arguments.

$\rightarrow$ habituez vous à anticiper comment les lignes de commandes que vous saisissez vont être découper et à repérer nom, options et arguments, comme l'interpréteur

Quelques commandes à connaître

`cd`, `ls`, `pwd`, `cat`, `less`, `wc`, `echo`, `head`, `tail`, `mkdir`, `cp`, `mv`, `rm`, `rmdir`, `file...`

Et surtout: `man`

La plupart des commandes ont aussi une option `--help` qui permet d'obtenir une description concise du fonctionnement et des option de la commande

Se promener dans l'arbre

pwd ***p**rint **w**orking **d**irectory*

ls ***lis**ter le contenu d'un dossier*

cd ***c**hange **d**irectory*

copier, déplacer, supprimer ou créer des fichiers

cp **cop**ier

mv ***mov***e déplacer

rm ***rem***ove supprimer

mkdir ***mk dir***ectory créer un dossier

touch crée un fichier (effet de bord bien pratique)

zip compresser une archive `zip`

unzip décompresser une archive `zip`

tar manipuler les archives `tar`

voir et manipuler le contenu des fichiers

file donne des informations sur le type de fichier

cat lit le contenu d'un ou plusieurs fichiers

head lit le début d'un fichier

tail lit la fin d'un fichier

cut sélectionne une ou plusieurs colonnes dans un fichier tabulé

less lecteur (interactif)

Mots clefs de la section

à retenir

- commande
- options
- arguments
- page de man(uel)
- et moultes commandes à découvrir

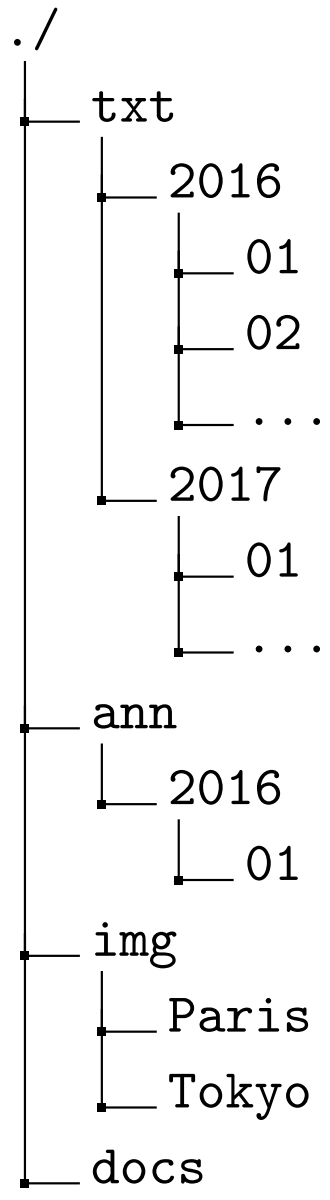
N'hésitez pas à notez les commandes et les options les plus utiles dans votre journal (les pages man sont parfois très longues !)

Exercice

L'adresse d'une archive zip a été ajoutée sur le tableur contenant la liste des inscrits au cours (un fichier par personne, en fin de chaque ligne). copier cette adresse qui sera à saisir sur icampus.
Puis dans un terminal:

- créer un dossier "Exercice1" en sous-dossier de votre répertoire personnel avec la **commande** `mkdir`
- rendez-vous ce dossier (par exemple avec la **commande** `cd ~/Exercice1`)
- Télécharger l'archive `fichiers.zip` avec la **commande** `wget` en lui donnant l'url du zip en **argument**.
- utiliser la commande `unzip` pour décompresser l'archive. (consultez le **man**uel si besoin).
- créer une arborescence pour classer les documents
  - par type de fichier (txt, ann, img, docs)
  - puis par date pour les txt et ann
  - puis par lieux pour les photos.

Exercice



Consignes pour le rendu

- L'arborescence à obtenir peut être schématiser comme ci-contre.
- créer une nouvelle archive contenant les fichiers triés
- obtenez l'historique des commandes avec la commande **history** et copier son contenu (à partir du DEBUT de l'exercice) dans un fichier texte
- le nouveau zip et le fichier texte d'historique sont à déposer sur iCampus avant lundi 30 à 23h59.

Les Pipelines

Flux d'entrées-sorties

Généralités

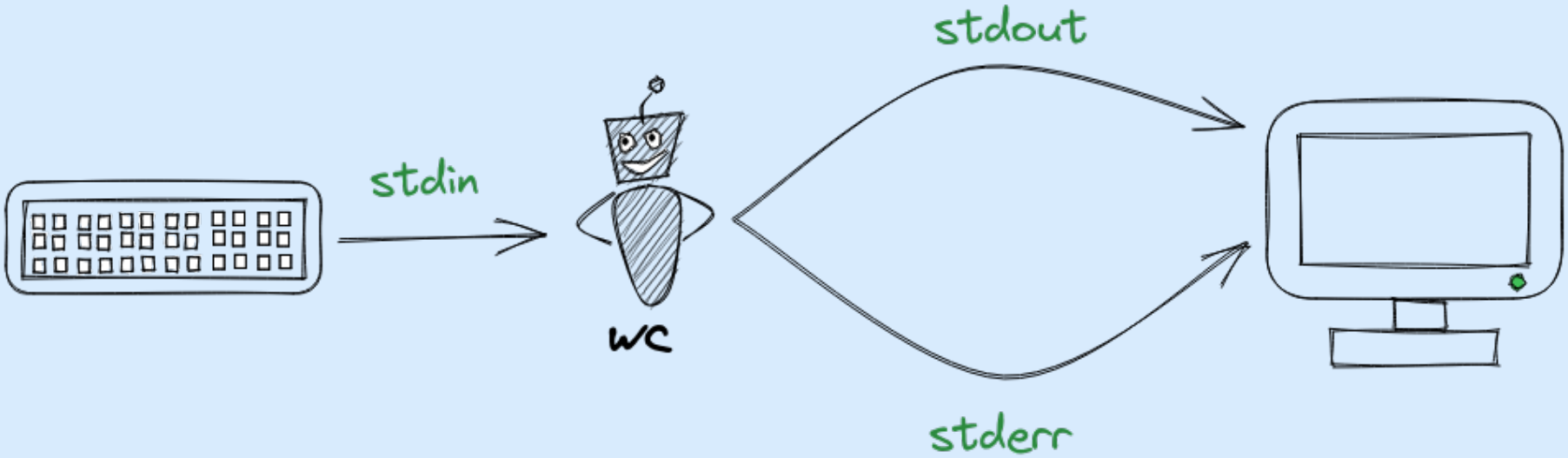
Toutes les commandes communiquent via trois flux de données

0	**stdin**	l'entrée standard (par défaut le clavier)
1	**stdout**	la sortie standard (par défaut l'écran)
2	**stderr**	la sortie d'erreurs standard (par défaut l'écran)

Par défaut, les commandes échangent dans le terminal via le clavier et l'écran, mais on peut **rediriger** ces flux.

wc

Situation par défaut



note: pour indiquer la fin de notre saisie au clavier, on utilise la combinaison de touches **Ctrl-D**

Flux d'entrées-sorties

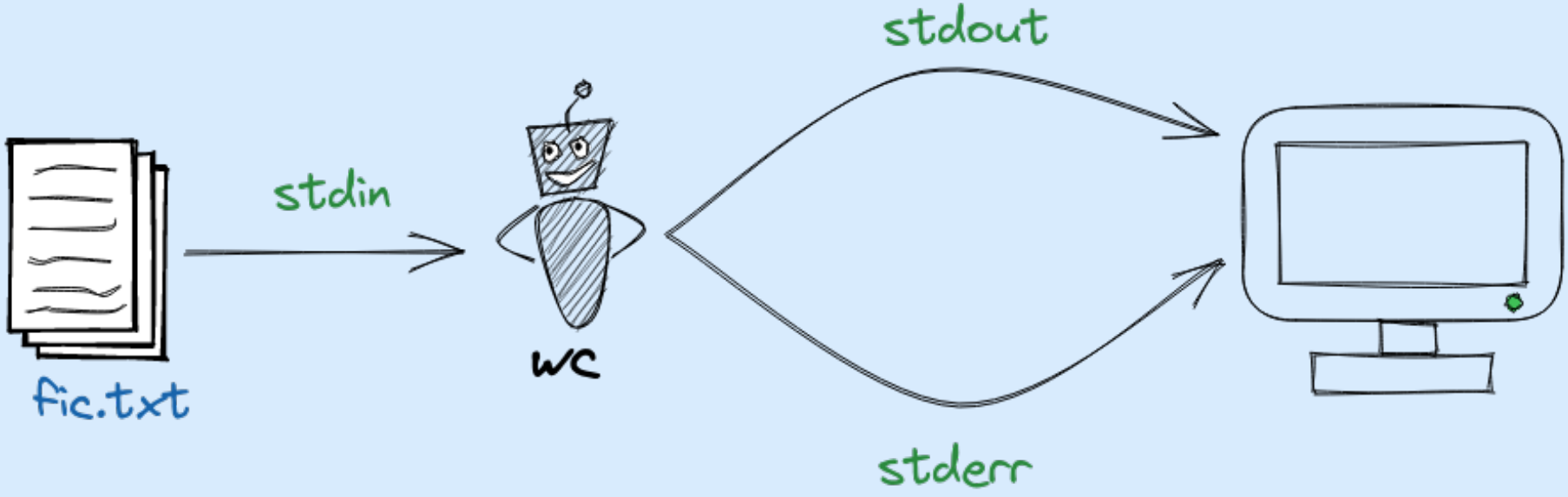
Redirections vers et depuis des fichiers

<	remplace le clavier par le contenu d'un fichier
1> ou >	écrit stdout dans un fichier
2>	écrit stderr dans un fichie
&>	écrit stdout et stderr dans un fichierr

En écriture, si on double le chevron (>>, >>&, 2>>), on écrit en ajoutant la sortie à la fin d'un fichier.
ATTENTION: les chevronts simples (>, >&, 2>) écrasent le fichier si il existe déjà.

`wc < fic.txt`

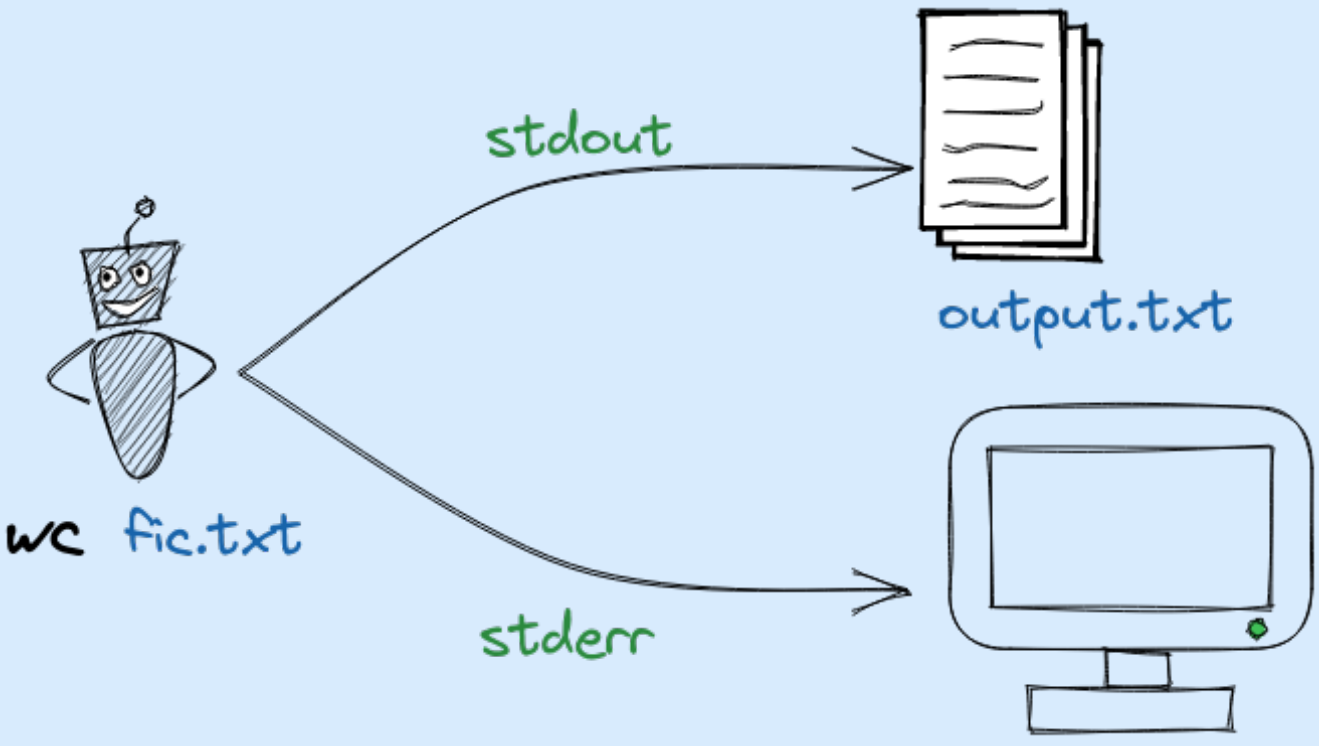
Redirection du contenu d'un fichier dans `stdin`



ATTENTION: ici `fic.txt` n'est PAS un argument.

`wc fic.txt > output.txt`

Redirection de `stderr` dans un fichier.

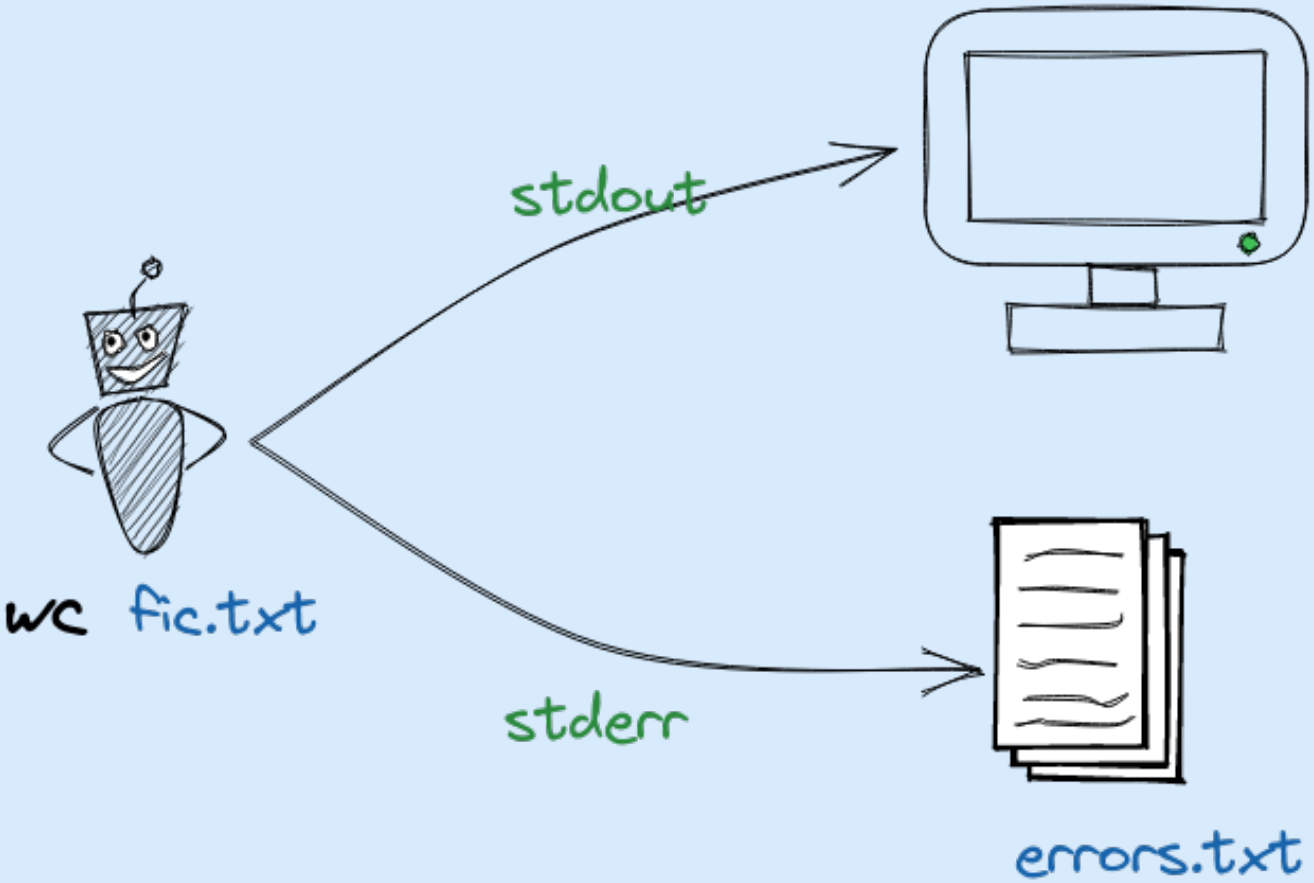


ATTENTION: ici `fic.txt` est donné en argument (et `wc` ignore l'entrée standard).

La sortie standard reste l'écran par défaut.

`wc fic.txt 2> errors.txt`

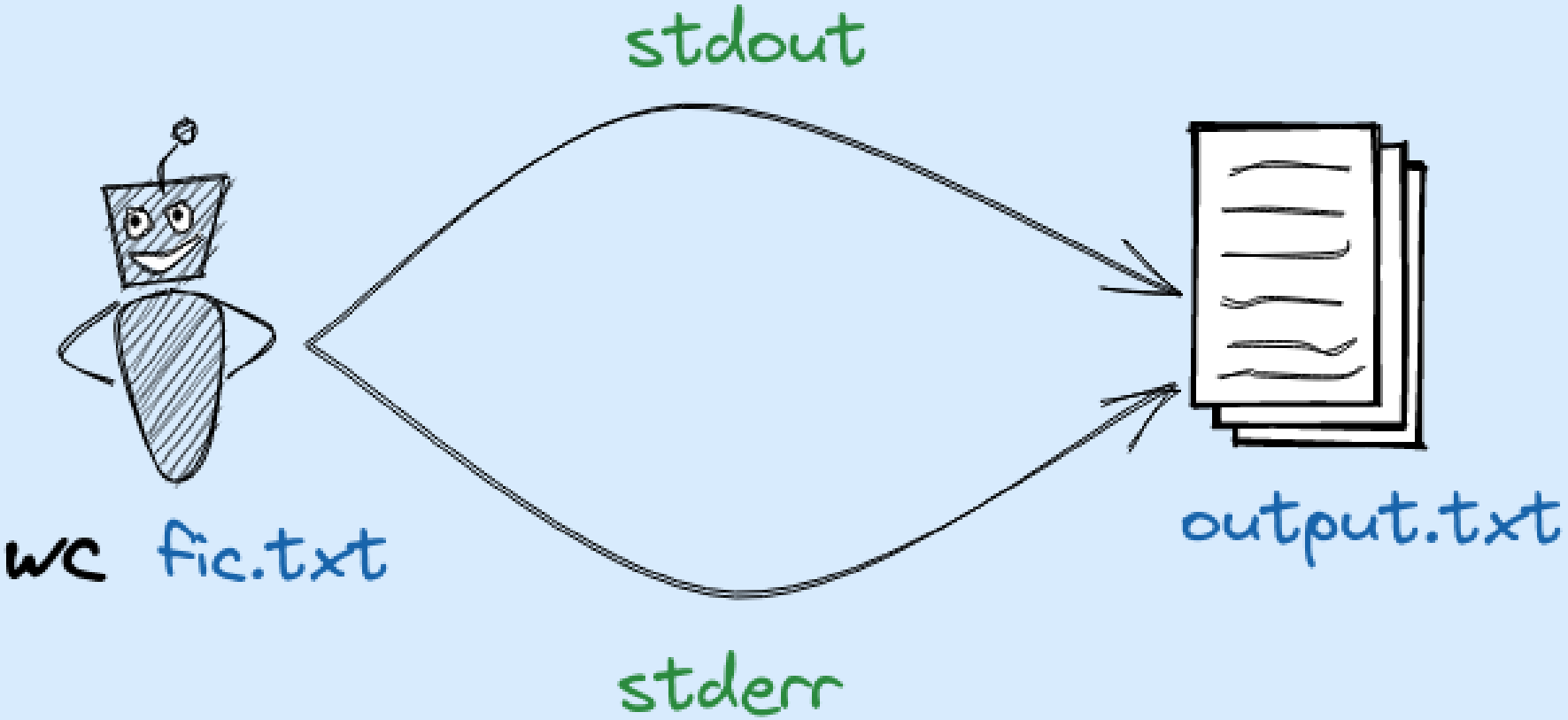
Redirection de `stderr` dans un fichier.



ATTENTION: ici `fic.txt` est donné en argument (et `wc` ignore l'entrée standard).

`wc fic.txt &> output.txt`

Redirection des sorties `stdin` et `stdout` dans un fichier.



ATTENTION: ici `fic.txt` est donné en argument
(et `wc` ignore l'entrée standard).

Flux d'entrées-sorties

Redirections entre les commandes

Plusieurs commandes peuvent communiquer en connectant les sorties aux entrées aux moyen du caractère «pipe» |

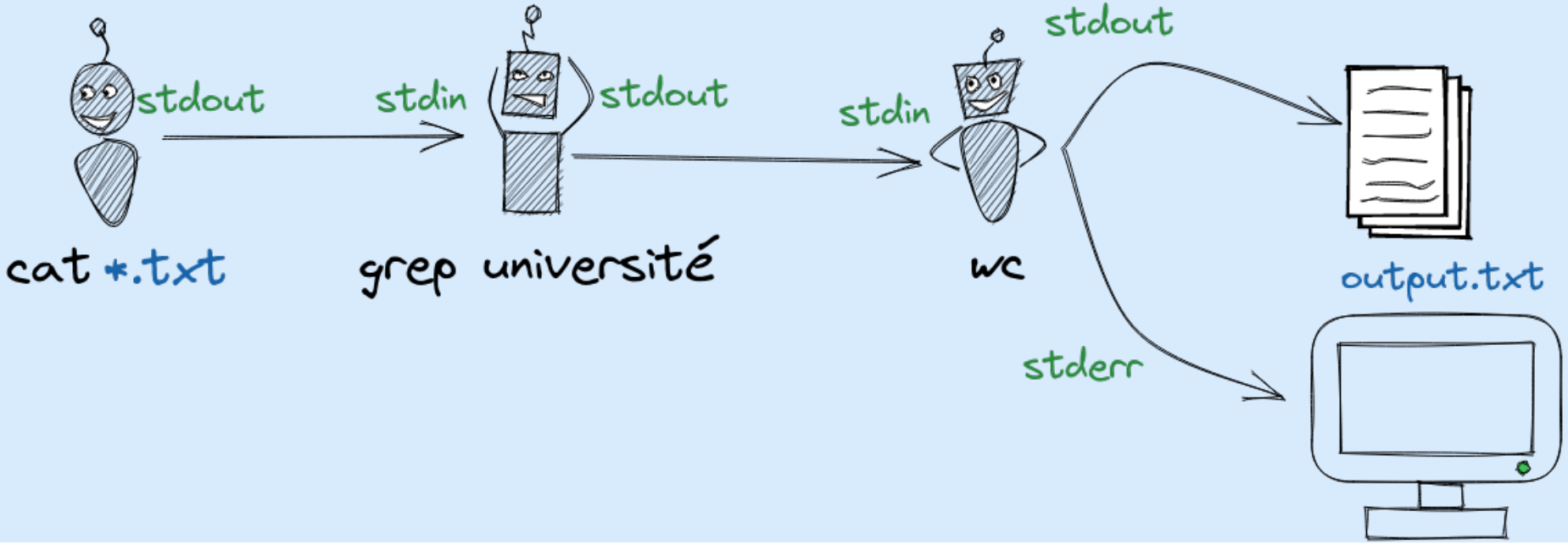
`cmd1 | cmd2` la sortie standard (stdout) de cmd1 est envoyée en stdin de cmd2.

`cmd1 |& cmd2` les sorties stdout et stderr de cmd1 sont toutes deux redirigées vers stdin de cmd2.

flux d'entrées-sorties

```
cat *.txt | grep université | wc > output.txt
```

Redirection de `stderr` dans un fichier.



Exercices

Quelques commandes de plus

grep recherche de motifs dans l'entrée (ou dans des fichiers)

sort trier des lignes

uniq supprimer les lignes qui se répètent

echo affiche un texte (pour formater vos résultats)

avec les fichiers `*.ann`, Sauriez vous... ?

- Compter le nombre d'annotations par année (2016, 2017 et 2018),
- limiter ce comptage aux lieux (Location),
- sauvegarder ces résultats dans un (seul) fichier,
- établir le classement des lieux les plus cités.
- trouver les annotations les plus fréquentes pour votre mois de naissance, toutes années confondues.