Introduction à l'informatique

Les flux

D. Buscaldi, J.-C. Dubacq

IUT de Villetaneuse

S1 2016

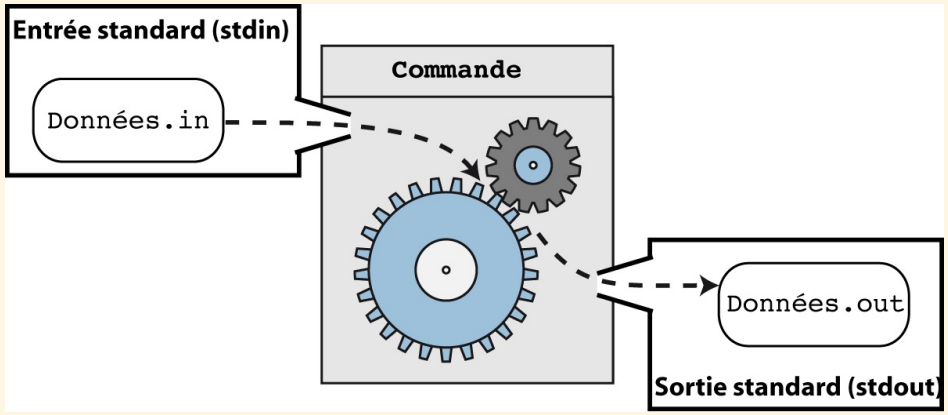
Plan

Plan

Entrée et sortie standard

Rappel : Les programmes informatiques

- Un programme prend des données en entrée. Ces données peuvent être lues dans un fichier ou fournies par un flux du système.
- Le programme manipule ces données.
- Le programme fournit un résultat en sortie (des données). Ces données peuvent être écrites dans un fichier ou exportées comme un flux vers le système.



Les flux de données

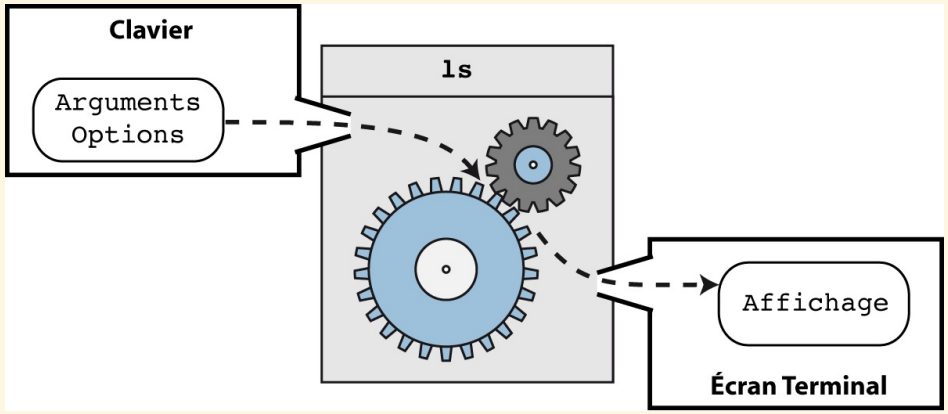
Pour fonctionner, un programme a donc besoin de lire des données (flux d'entrée : input) et d'écrire les résultats de ses évaluations (flux de sortie : output). On distingue 3 types de flux de données :

- **STDIN** : entrée standard (là où sont lues les données),
- **STDOUT** : sortie standard (là où sont écrits les résultats),
- **STDERR** : sortie erreur (là où sont écrit les messages d'erreur).

Entrée et sortie standard

Les commandes qui lisent sur l'entrée standard

- Certaines commandes Linux qui traitent les données d'un fichier (dont le chemin est passé en paramètre) peuvent alternativement, si aucun chemin fichier n'est spécifié, travailler directement avec les données lues sur l'entrée standard.
- Par exemple : `echo`, `cat`, `head`, `tail`, `grep`.
- **Par défaut, l'entrée standard est le clavier**.



Les commandes qui écrivent sur la sortie standard

- Les affichages produits par les commandes Linux sont le résultat de leur évaluation. Ce résultat est écrit sur la sortie standard.
- **Par défaut, la sortie standard est l'écran**.

Syntaxe pour cat

```
cat  fichier [fichier_2 ...]
```

Description

- Affiche le contenu des fichiers les uns à la suite des autres.
- Les fichiers sont concaténés dans l'ordre des paramètres.

Exemple d'utilisation:

Cette commande est en générale utilisée pour concaténer des fichiers textes. On l'utilise avec une commande de redirection (cf. Partie Redirections) pour enregistrer le résultat de la concaténation dans un nouveau fichier.

Soient les deux fichiers suivants :

tellur.txt

```
Mercure, Venus
Terre, Mars
```

jov.txt

```
Jupiter, Saturne
Uranus, Neptune
```

La commande :

```
login@host:~$ cat tellur.txt jov.txt
Mercure, Venus
Terre, Mars
Jupiter, Saturne
Uranus, Neptune
login@host:~$
```

Syntaxe pour head

`head < -int > fichier`

Description

- Affiche par défaut les 10 premières lignes d'un fichier.
- Si un entier n précède le nom du fichier, la commande affiche les n premières lignes du fichier.

Exemple d'utilisation:

Soit le fichier `planetes.txt` contenant les lignes suivantes :

planetes.txt

```
# Premier groupe
1  Mercure
Tellurique
2  Venus
Tellurique
3  Terre
Tellurique
4  Mars
Tellurique
# Deuxième groupe
1  Jupiter
Gazeuse
2  Saturne
```

La commande suivante affiche les 5 premières lignes du fichier :

```
login@host:~$ head -5 planetes.txt
# Premier groupe
1  Mercure  Tellurique
2  Venus    Tellurique
3  Terre    Tellurique
4  Mars     Tellurique
login@host:~$
```

Syntaxe pour tail

```
tail < -int > fichier
```

Description

- Affiche par défaut les 10 dernières lignes d'un fichier.
- Si un entier n précède le nom du fichier, la commande affiche les n dernières lignes du fichier.

Exemple d'utilisation:

Soit le fichier `planetes.txt` contenant les lignes suivantes :

planetes.txt

```
# Premier groupe
1  Mercure
Tellurique
2  Venus
Tellurique
3  Terre
Tellurique
4  Mars
Tellurique
# Deuxième groupe
1  Jupiter
Gazeuse
2  Saturne
```

La commande suivante affiche les 4 dernières lignes du fichier :

```
login@host:~$ tail -4 planetes.txt
1  Jupiter  Gazeuse
2  Saturne  Gazeuse
3  Uranus   Gazeuse
4  Neptune  Gazeuse
login@host:~$
```

Syntaxe pour grep

```
grep "motif" fichier
```

Description

- Affiche les lignes du fichier qui comportent le `"motif"`.
- Les lignes sont affichées dans leur ordre d'apparition dans le fichier.

Exemple d'utilisation:

Soit le fichier `planetes.txt` contenant les lignes suivantes :

planetes.txt

```
# Premier groupe
1 Mercure
Tellurique
2 Venus
Tellurique
3 Terre
Tellurique
4 Mars
Tellurique
# Deuxième groupe
1 Jupiter
Gazeuse
```

Commandes :

```
login@host:~$ grep 'Tellurique'
planetes.txt
1 Mercure Tellurique
2 Venus   Tellurique
3 Terre   Tellurique
4 Mars    Tellurique
login@host:~$ grep '1' planetes.txt
1 Mercure Tellurique
1 Jupiter Gazeuse
login@host:~$
```

Exercices

Manipulation du contenu d'un fichier texte

Q1 La commande suivante montre le contenu d'un fichier texte :

```
login@host:~/$ cat /proc/cpuinfo
```

Q2 Quelle sont les informations contenues dans ce fichier ?

Q3 À l'aide des commandes `cat` ou `less` identifiez dans le fichier /proc/cpuinfo le nombre de fois ou le mot 'cpu' apparait

Q4 La commande `grep 'cpu' /proc/cpuinfo` permet d'afficher les lignes du fichier `/proc/cpuinfo` où le mot 'cpu' apparait. Vérifiez qu'il y en le bon nombre ?

Q5 L'option -v permet d'inverser son comportement. Au lieu d'afficher les lignes qui présentent le motif, `grep` affiche alors les lignes qui ne présentent pas le motif. Affichez les lignes du fichier `/proc/cpuinfo` ne présentant pas le mot 'cpu'.

Q6 Proposez une commande permettant d'afficher les premières 5 lignes

Q7 Proposez une commande permettant d'afficher les dernières 5 lignes

Plan

Redirection des Entrée/Sorties

Commandes de Redirection

Il est possible de modifier le comportement par défaut des commandes et de donner une entrée et/ou une sortie standard différente des entrées/sorties standards.

```
command > fichier.out
```

- **Redirige la sortie standard** de la commande `command` vers le fichier `fichier.out`.
- Si le fichier `fichier.out` n'existe pas, il est créé avec comme contenu les affichages produits par la commande `command`.
- **Si le fichier `fichier.out` existe, son contenu est écrasé** et remplacé par les affichages produits par la commande `command`.

```
command » fichier.out
```

- **Redirige la sortie standard** de la commande `command` vers le fichier `fichier.out`.
- Si le fichier `fichier.out` n'existe pas, il est créé avec comme contenu les affichages produits par la commande `command`.
- Si le fichier `fichier.out` existe, les affichages produits par la commande `command` sont **ajoutés à la fin du contenu du fichier**.

```
command 2> fichier.err
```

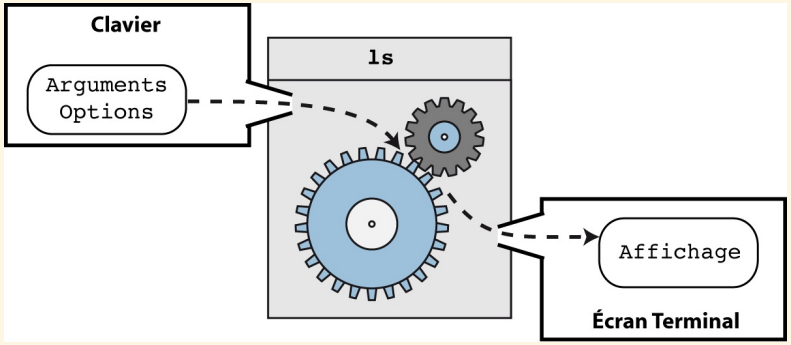
- **Redirige la sortie erreur** de la commande `command` vers le fichier `fichier.err` **avec écrasement du contenu** si le fichier de sortie existe déjà.

```
command 2» fichier.err
```

- **Redirige la sortie erreur** de la commande `command` vers le fichier `fichier.err` **avec préservation du contenu** si le fichier de sortie existe déjà.

Exemple de redirection

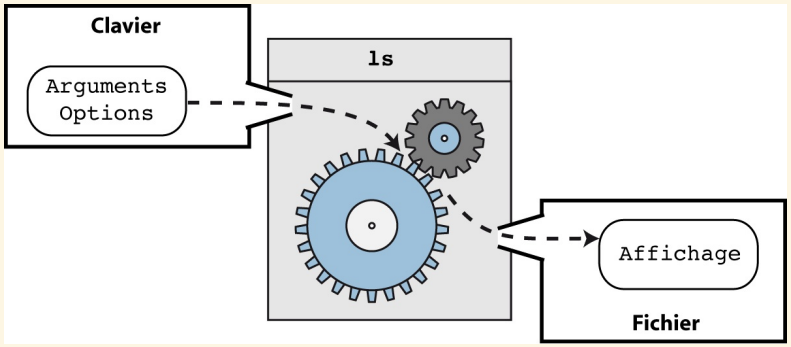
Comportement par défaut de la commande `ls`




```
login@host:~$ ls
aldenaran.jpg alphacentauri.gif
etacentauri.jpg
login@host:~$ ls
aldenaran.jpg alphacentauri.gif
etacentauri.jpg
login@host:~$
```

La sortie standard de la première commande `ls` est l'écran. La liste du contenu du répertoire courant est affichée à l'écran.

Redirection de la sortie de la commande `ls`




```
login@host:~$ ls > 1.out
login@host:~$ ls
1.out aldenaran.jpg alphacentauri.gif
etacentauri.jpg
login@host:~$
```

La sortie standard de la première commande `ls` est redirigée vers le fichier `1.out`. La liste du contenu du répertoire courant est écrite dans le fichier `1.out`.
La deuxième commande `ls`, montre qu'un fichier portant le nom `1.out` a été créé.

Syntaxe pour echo

```
echo  expression
```

Description

- Affiche sur la sortie standard l'expression après interprétation.

Exemple d'utilisation:

Affiche `'Bonjour'` :

```
login@host:~$ echo Bonjour
Bonjour
login@host:~$
```

Définie une variable puis affiche sa valeur :

```
login@host:~$ Astre=Terre
login@host:~$ echo $Astre
Terre
login@host:~$ echo La planete $Astre
La planete Terre
login@host:~$
```

Exercices

Redirections

Q8 Que font les commandes suivantes ?

```
login@host:~$ echo “Bonjour"
login@host:~$ echo “Bonjour" > bonjour.out
login@host:~$ echo “Salut" > bonjour.out
login@host:~$ echo “Bonjour" » bonjour.out
```

Q9 Entrainez-vous avec les commandes suivantes. Profitez-en pour comprendre les affichages produits par les commandes `ps` et `file` :

```
login@host:~$ ps > essai_ps.out
login@host:~$ file /usr/include/stdio.h > file.out
```

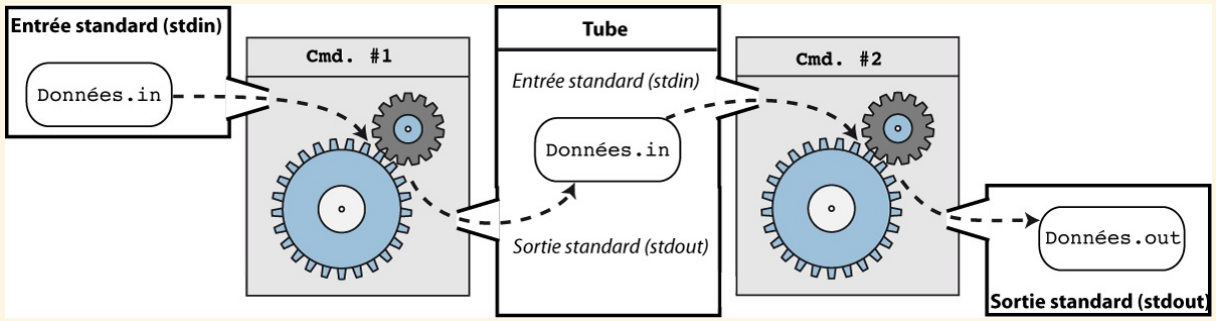
Q10 Proposez une commande pour copier le contenu de /proc/cpuinfo dans un fichier cpuinfo.out sans utiliser la commande `cp`

Plan

Tubes

Principes de fonctionnement des Tubes (Pipe en anglais)

- A la différence des redirections simples qui permettent de rediriger la sortie standard d'une commande vers un fichier,
- Un tube permet de rediriger la sortie standard d'une commande vers l'entrée standard d'une autre commande.



Syntaxe

- Le tube est symbolisé par le caractère |.

```
cmd1 | cmd2
```

- - La sortie standard de la première commande (`cmd1`) est redirigée vers l'entrée standard de la deuxième commande (`cmd2`).
  - L'entrée standard de la commande `cmd1` et la sortie standard de la commande `cmd2` ne sont pas modifiées.

Exemple de Tubes avec les commande `ls` et `more`

Rappel des commandes :

- `ls` affiche à l'écran (stdout) la liste des fichiers contenus dans un répertoire.
- `more` affiche page par page le contenu des données passée sur son entrée standard.

Exemple #1

- Si de très nombreux fichiers sont contenus dans un répertoire, la commande `ls` peut produire un affichage qui ne tient pas dans l'écran, rendant impossible le parcours de la liste des fichiers (seuls les derniers sont visibles).

```
login@host:~$ ls
```

Défilement de tous les fichiers

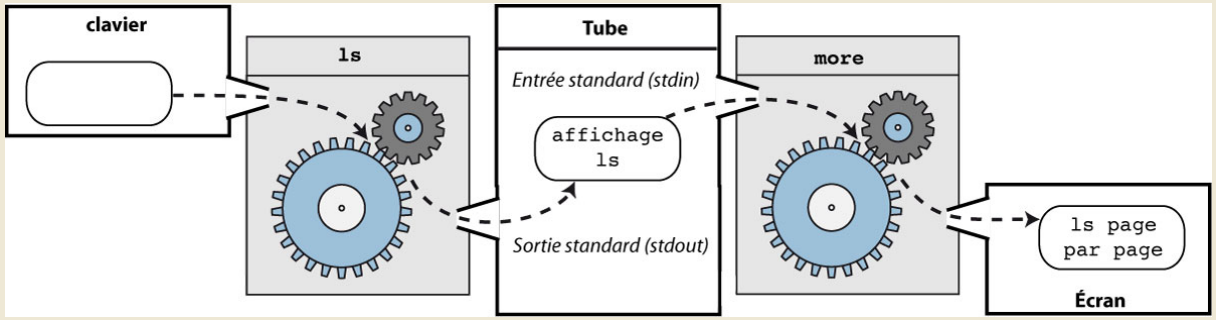
```
betelgeuse.jpg    etacentauri.jpg
soleil.jpg        syrius.gif
vega.png
login@host:~$ ■
```

Images/ . . . . . Répertoire courant
- aldebaran.jpg . . . . . Hors de la fenetre
- alphacentauri.gif . . . . . Hors de la fenetre
- betelgeuse.jpg . . . . . Dans la fenetre
- etacentauri.jpg . . . . . Dans la fenetre
- soleil.jpg . . . . . Dans la fenetre
- syrius.gif . . . . . Dans la fenetre
- vega.png . . . . . Dans la fenetre

Exemple de Tubes avec les commande `ls` et `more`

Exemple #1 (suite) :

- La redirection de la sortie standard de la commande `ls` vers l'entrée standard de la commande `more` permet de passer en revue l'affichage de la commande `ls` page par page.




```
login@host:~$ ls | more
aldebaran.jpg
alphacentauri.gif
betelgeuse.jpg
etacentauri.jpg
soleil.jpg          syrius.gif
```

Affichage d'une première page puis
Presser la touche Enter pour la page suivante

```
soleil.jpg          syrius.gif
vega.png
login@host:~$ ■
```

Images/ . . . . . Répertoire courant
- aldebaran.jpg . . . . . Page 1
- alphacentauri.gif . . . . . Page 1
- betelgeuse.jpg . . . . . Page 1
- etacentauri.jpg . . . . . Page 1
- soleil.jpg . . . . . Page 1&2
- syrius.gif . . . . . Page 1&2
- vega.png . . . . . Page 2

Exemple de Tubes avec les commande `ls` et `grep`

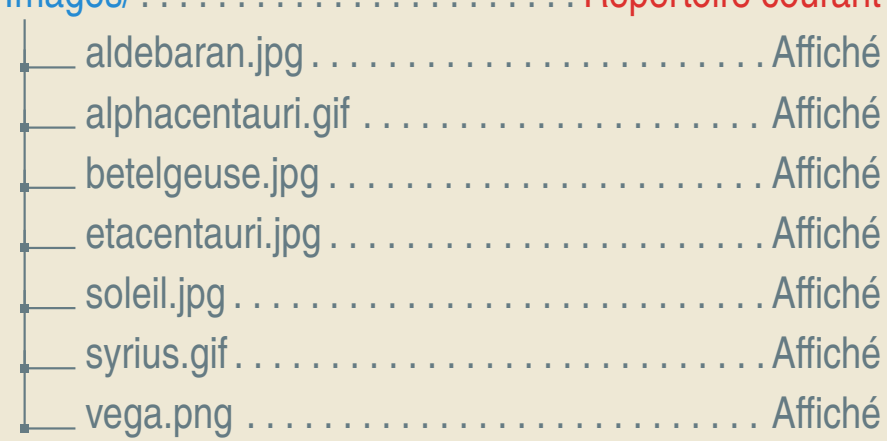
Rappel des commandes :

- `ls` affiche à l'écran (stdout) la liste des fichiers contenus dans un répertoire.
- `grep` affiche les lignes d'un texte qui comportent un certain motif.

Exemple #2 :

- Si de très nombreux fichiers sont contenus dans un répertoire, la commande `ls` peut produire un affichage qui ne tient pas dans l'écran, rendant compliqué l'identification de certain type de fichier (fichiers au format `gif` par exemple).

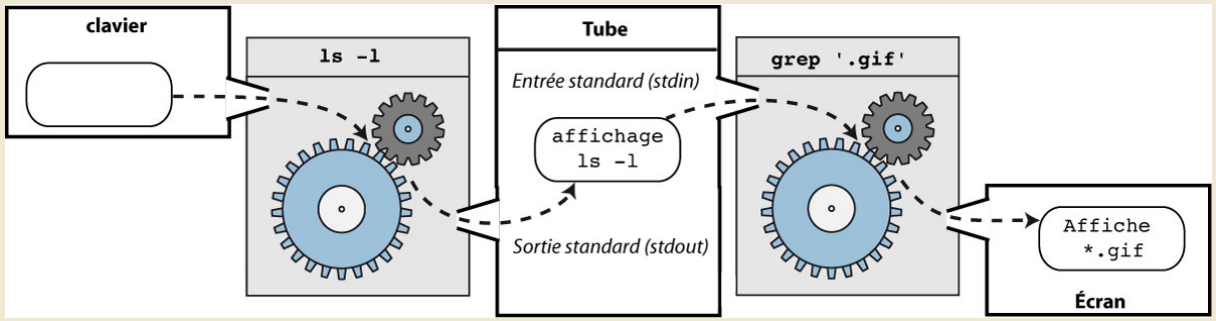
```
login@host:~$ ls
aldebaran.jpg
alphacentauri.gif
betelgeuse.jpg
etacentauri.jpg
soleil.jpg
syrius.gif
vega.png
login@host:~$ ■
```



Exemple de Tubes avec les commande `ls` et `more`

Exemple #2 (suite) :

- La redirection de la sortie standard de la commande `ls` vers l'entrée standard de la commande `grep` permet d'effectuer un filtrage des fichiers présents dans le répertoire sur la base d'un motif présent dans leur nom (par exemple l'extension `.gif`).




```
login@host:~$ ls | grep '.gif'
alphacentauri.gif
syrius.gif
login@host:~$ ■
```

Images/ . . . . . Répertoire courant
- aldebaran.jpg . . . . . Retenu par le filtre
- alphacentauri.gif . . . . . Affiché
- betelgeuse.jpg . . . . . Retenu par le filtre
- etacentauri.jpg . . . . . Retenu par le filtre
- soleil.jpg . . . . . Retenu par le filtre
- syrius.gif . . . . . Affiché
- vega.png . . . . . Retenu par le filtre

Syntaxe pour wc

```
wc  fichier <fichier_2 ...>
```

Description

- Affiche des statistiques sur le nombre de lignes, de mots et de caractères (comptés en nombre d'octets) contenus dans le fichier dont le chemin est donné en paramètre.

Exemple d'utilisation:

Soit le fichier suivant :

tellur.tsv

```
1 Mercure Venus
2 Terre Mars
```

Commande #1 :

```
login@host:~$ wc tellur.tsv
2     6     29  tellur.tsv
login@host:~$
```

L'affichage produit indique que le fichier `tellur.tsv` comporte :

- 2 lignes,
- 6 mots et
- 29 caractères. La taille du fichier texte est donc de 29 octets …

Exercices

Tubes

Q11 Étudiez et comparez les commandes suivantes. Pour vous aider vous pouvez évaluer les commandes pas à pas en vous arrêtant avant chaque tube.

```
login@host:~$ cat /proc/cpuinfo | wc -l
login@host:~$ head /proc/cpuinfo | wc -l
login@host:~$ cat /proc/cpuinfo | grep 'cpu' | wc
-l
login@host:~$ head /proc/cpuinfo | grep 'cpu' | wc
-l
```

Q12 Proposez une commande pour afficher le nombre de fichiers dans votre répertoire home

Q13 Proposez une commande pour afficher le nombre des processus

Q14 Proposez une commande pour afficher les premières 5 lignes des dernières 10 lignes du fichier /proc/cpuinfo