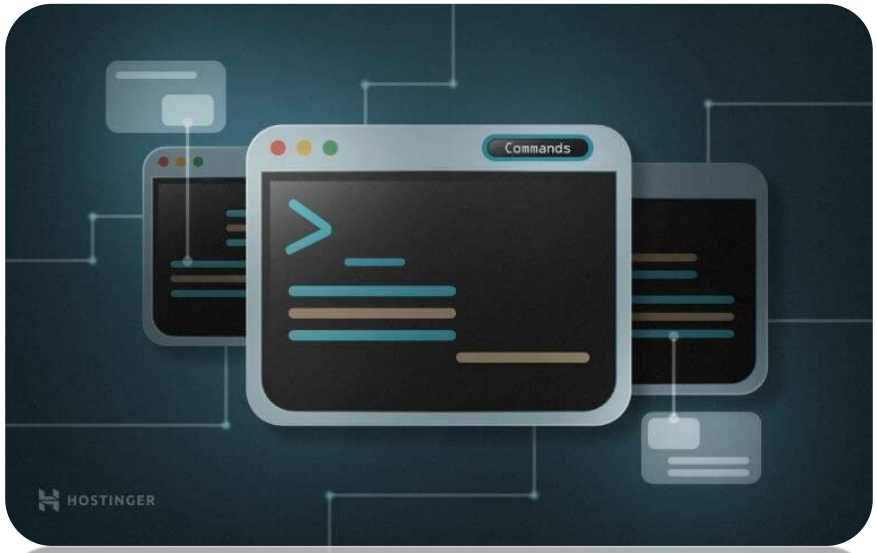
Support de cours

Systèmes d'exploitation



Chapitre IV: Les commandes Linux II

Plan

- Les jokers et les expressions régulières
- La commande de recherche de fichiers
- Les commandes filtres
- La redirection des entrées-sorties
- Les tubes de communication

→ Jokers:

Définition

De nombreuses commandes opèrent sur un ou plusieurs fichiers ou répertoires. Les jokers (**métacaractère)** permettent de définir ces noms de fichiers ou répertoires sans connaitre précisément leur nom. Ce sont des caractères spéciaux qui sont interprétés.

.	désigne le répertoire courant
..	désigne le répertoire père
*****	correspond à 0 ou n caractères
?	correspond à 1 seul caractère
[]	correspond à un des caractères précisé entre les crochets
-	précise un intervalle de caractères

→ **Jokers:**

Exemple

- **Lister les fichiers de 4 caractères dont le 1er et le 3eme est une lettre *t***

  $ **ls t?t?**

- **Lister tous les fichiers qui commence par t et finisse par e**

  $ **ls t*e**

- **lister les fichiers commençant par une lettre entre *a* et *e***

  $ **ls [a-e]*** ou $ **ls [abcde]***

→ Les expressions régulières:

Définition

Une expression régulière est **une série de caractères spéciaux qui nous permettent de décrire un texte que nous voulons trouver**.

→ **Il existe 2 types d'expressions régulières :**

- Les expressions régulières basiques (**ERb**)
- → Les expressions régulières étendues (**ERe**)

→ **Les ERb sont utilisées par les commandes suivantes :**

- grep
- sed

→ **Les ERe sont utilisées par les commandes suivantes :**

- grep avec l'option -E (egrep)
- awk

→ **Les expressions régulières**

- **Liste des caractères spécifiques aux ERb**

Caractère spécial	Signification
\{n\}	n fois le caractère précédent
\{n,\}	Au moins n fois le caractère précédent
\{n,x\}	Entre n et x fois le caractère précédent
\(ERb\)	Mémorisation d'une ERb
\1, \2, ...	Rappel de mémorisation

→ Les expressions régulières

❑ Liste des caractères spéciaux communs aux ERb et aux ERe

Caractère spécial	Signification
^	Début de ligne
$	Fin de ligne
. (point)	Un caractère quelconque
[liste_de_caractères]	Un caractère cité dans la liste
[^liste_de_caractères]	Un caractère qui n'est pas cité dans la liste
*****	0 à n fois le caractère ou regroupement précédent
\<expression	Début d'un mot. Caractères pouvant faire partie d'un mot : [A-Za-z0-9_]
expression\>	Fin d'un mot
\<expression\>	Mot complet

→ **Les expressions régulières**

- **Liste des caractères spécifiques aux ERe**

Caractère spécial	Signification
?	0 ou 1 fois le caractère ou regroupement précédent
+	1 à n fois le caractère ou regroupement précédent
{n}	n fois le caractère ou regroupement précédent
{n,}	Au moins n fois le caractère ou regroupement précédent
{n,x}	Entre n et x fois le caractère ou regroupement précédent
(er1)	Regroupement
er1 | er2 | er3	Alternatives

→ Les expressions régulières

❑ Exemples

- ^$: représente une ligne vide
- ^[A-Z] : une majuscule en début de ligne
- [a-z][a-z]$: 2 caractères minuscules en fin de ligne
- [ABCD]\ {2,10 \}$: entre 2 et 10 caractères. **A**,**B**,**C** ou **D** en fin de ligne
- .* : une séquence vide ou de plusieurs caractères
- [x y z] : **x** ou **y** ou **z**
- [A-G] : intervalle
- [^xyz] : sauf **x**, **y** ou **z**
- \{m,n\} :répétions entre **m** et **n** fois
- [^0-9] : désigne une chaîne ne contenant pas de chiffre.
- ^[^0-9] : désigne une chaîne ne contenant pas de chiffre au début de la ligne

→ Les expressions régulières

❑ Exercice

Créez dans le répertoire *rep1* le fichiers suivants : *fich1*, *fich2*, *fich11*, *fich12*, *fich1a*, *ficha1*, *fich33*, *.fich1*, *.fich2*, *toto*, *afich*.

→ Listez les fichiers :

1. Dont les noms commencent par ***fich***
2. Dont les noms commencent par ***fich*** suivi d'un seul caractère
3. Dont les noms commencent par ***fich*** suivi d'un chiffre
4. Dont les noms commencent par **.**
5. Dont les noms ne commencent pas par ***f***
6. Dont les noms contiennent ***fich***.

La commande de recherche de fichiers

Syntaxe

find ***[emplacement] [options] [action]***

Description

- Rechercher des fichiers dans une hiérarchie de répertoires
- Cette commande peut également enchaîner des actions sur le résultat des recherches et utiliser des expressions régulières comme motifs de recherche

Options

- **-name** ***[nom]***, **-iname** ***[nom]*** :Recherche par nom de fichier.
- **-type** ***[type]*** :Recherche par type de fichier (bloc **b**, caractère **c**, répertoire **d**, tube nommé **p**, fichier régulier **f**, liens symbolique **l**, socket **s**) .
- **-size** ***[n]*** :Recherche par taille de fichier. Fichier utilisant **n** blocks. *La taille d'un fichier sont affichées avec un nombre suivi d'une unité***:** *c :Octets, k :Kilooctets, M :Mégaoctets, G :Gigaoctets, b :Blocs de 512 octets* .
- **-empty** : fichier vide.

La commande de recherche de fichiers

Options

- **-ctime** ***[d]***, **-mtime** ***[d]***, **-atime** ***[d]*** :Recherche par date de création, date de dernière modification, ou date de dernier accès, respectivement.
- **-user** ***[u]***, **-group** ***[g]*** :Recherche par propriétaire ou par groupe.
- **-perm** ***[mode]*** :Recherche par autorisations d'accès.
- **-links** ***[n]*** : fichiers ayant **n** liens .
- **-inum** ***[n]*** : fichier dont le numéro d'i-node est n.
- **-newer** ***[nom_fich]*** : fichier modifié plus récemment que le fichier indiqué
- **-print** affiche le nom complet du fichier sur la sortie standard.
- **-exec** ***[cmd]*** : exécute la commande jusqu'à ce qu'on rencontre un ' ;'. La chaîne ***\{ \}***ou ***{}*** **** est remplacée par le nom du fichier en cours de traitement.
- **-ok** comme **-exec** mais interroge d'abord l'utilisateur (demande de confirmation).

La commande de recherche de fichiers

Exemples

- Chercher dans l'arborescence **/usr** les répertoires dont le nom commence par "local" :
  $ **find** */usr* **-name** *"local*"* **-type** *d* **–print**
- Chercher dans le répertoire courant les fichiers « .c » de plus de 40Ko
  $ **find** . **-name** *" *.c "* **-type** *f* **-size** *+78* **-print**
- Chercher les liens symboliques qui se nomment local dans /usr
  $ **find** */usr* **-name** *"local"* **-type** *l* **-print**
- Supprimer dans votre répertoire d'accueil les fichiers « .o ».
  $ **find** ~ **-name** *" *.o"* **-exec** ***rm*** *\{\} \;*
- Supprimer tous les répertoires vides dans l'arborescence (/)
  $ **find** */* **-type** *d* **-empty -ok rmdir** *{} \;*
- Rechercher dans le répertoire courant tous les fichiers avec les droits d'accès « 777 » et les changer en « 664 »
  $ **find . -type** *f* **-perm** *777* **-exec chmod** *664 {} \;*

La commande de recherche de fichiers

❑ Exercice

En utilisant la commande *find*, trouvez et listez les noms de :

1. Tous les fichiers sous le répertoire ***/etc*** dont les noms commencent par ***rc***.
2. Tous les sous–répertoires de ***/etc***,
3. Tous les fichiers réguliers se trouvant sous votre répertoire d'accueil et qui n'ont pas été modifiés dans les 10 derniers jours
4. Trouvez à partir de votre répertoire d'accueil, le nombre de fichiers ayant une taille supérieure à 1 Mega-octets
5. Trouver tous les fichiers **'GIARI'** dans l'arborescence et supprimez les après confirmation.
6. Cherchez dans toute l'arborescence les fichiers Dont le nom :
    - Se termine par **.c**
    - Commençant par **X** ou **x**.
    - Dont les noms ne contiennent pas de chiffre.
    - Redirigez les erreurs vers le fichier poubelle **/dev/null**
7. Afficher le nombre des fichiers dans toute l'arborescence vous appartenant et ayant les droits fixés à 666 (-rw-rw-rw-).

Les commandes filtres

Définitions

- Certains processus peuvent utiliser à la fois l'entrée **STDIN** et les sorties **STDOUT** et **STDERR**. Un processus qui lit des données sur l'entrée standard et produit des données sur la sortie standard est appelé **filtre**.
- Un filtre est une commande qui lit les données sur l'entrée standard, effectue des traitements sur les lignes reçues et écrit le résultat sur la sortie standard.
- Les entrées/sorties peuvent être redirigées, et enchainées avec des tubes.
- les principaux filtres utilisés dans GBU/Linux:
    - **cat, more et less, head, tail et wc**
    - **grep**
    - **tr**
    - **sort**
    - **cut**
    - **uniq et past**
    - **join**
    - **sed**
    - **awk**

Les commandes filtres - Simple

Syntaxe

cat *[option] [nom_fich]*

Description

- Défiler à l'écran le contenu d'un ou plusieurs fichiers.

Options

- **-n** ou **-b** : Numérote les lignes

Remarques

- **<ctrl-c>** : Mettre fin au défilement
- **<ctrl-s>** : Interrompre le défilement avec possibilité de reprise
- **<ctrl q>** : Reprendre le défilement

Exemple

```
$ cat etc/passwd
$ cat file1 file2 > file3
```

Les commandes filtres - Simple

Syntaxe

more ***[nom_fich]***

Description

- Défiler à l'écran le contenu d'un fichier page par page.
- **less** ***[nom_fich]*** :comme la commande **more**, mais on peut utiliser la touche Page Précédente. <Pas disponible sur tous les distributions>.

Remarques

more reconnaît un jeu de commandes interactives basé sur celui de **vi**. Certaines commandes peuvent être précédées d'un nombre décimal noté k ci-dessous. ^X signifie control-X.

- ***ESPACE***: page suivante
- ***RETURN ou ENTREE*** : ligne suivante
- ***=*** :Affiche le numéro de ligne courant.
- ***/chaîne*** :cherche la chaîne de caractère spécifiée
- ***:n*** :passe au fichier suivant
- ***:p*** :revient au fichier précédent
- ***! <cmd>*** :exécute *commande* Unix
- ***Q ou q*** :termine affichage (sortie de more)
- ***.*** :Répète la commande précédente
- ***:f*** :Affiche le nom et le numéro de ligne du fichier courant
- ***^L*** :Réaffiche l'écran.
- ***v*** :Entre dans l'éditeur vi et se positionne sur la ligne courante
- ***?*** ou ***h :***aide en ligne sur l'utilitaire more

Les commandes filtres - simple

Syntaxe

head ***[- int] [nom_fich]***

Description

- Affiche par défaut les 10 premières lignes d'un fichier.

Options

- **-[n]** : Si un entier **n** précède le nom du fichier, la commande affiche les **n** premières lignes du fichier.

Syntaxe

tail ***[- int] [nom_fich]***

Description

- Affiche par défaut les 10 dernières lignes d'un fichier.

Options

- **-[n]** : Si un entier **n** précède le nom du fichier, la commande affiche les **n** dernières lignes du fichier.

Les commandes filtres- simple

Syntaxe

wc *[option] [nom_fich]*

Description

- Compte le nombre de lignes, de mots et de caractères du *fichier* spécifié (ou de l'entrée standard) et l'écrit sur la sortie standard [*word count*]

Options

- ***-l*** :Imprime le nombre de lignes
- ***-w*** :Imprime le nombre de mots
- **-m** :Imprime le nombre de caractères
- **-c** :Imprime le nombre d'octets
- **-L** :Imprime la longueur de la ligne la plus longue

Exemple

```
$ wc -l /etc/passwd
33  /etc/passwd
```

Les commandes filtres

Syntaxe

grep *[options] {expreg} {nom_fich}*

Description

- Recherche dans les fichiers ou sur son entrée standard des lignes de texte qui satisfont l'expression régulière ***{expreg}*** indiquée.
- Sa sortie peut être redirigée dans un fichier.
- Les lignes sont affichées dans leur ordre d'apparition dans le fichier

Options

- **-c** : Donne seulement le nombre de lignes trouvées obéissant au critère
- **-l** : Donne seulement le nom des fichiers où le critère a été trouvé
- **-v** : Donne les lignes où le critère **n'a pas** été trouvé
- **-i** : Ne pas tenir compte de la casse (ne pas différencier majuscules minuscules)
- **-n** : Pour n'afficher que les numéros des lignes trouvées
- **-w** : Pour imposer que le motif corresponde à un mot entier d'une ligne
- ***-G*** : Interprète le motif comme une expression régulière simple (par défaut)
- ***-E*** : Interprète le *motif* comme une expression régulière étendue
- **-F** : Recherche des chaînes littérales

Les commandes

egrep ***[options] {expreg} {nom_fich}***

fgrep ***[options] {expreg} {nom_fich}***

Description

- **egrep** équivaut à la commande **grep -E** et permet d'utiliser des expressions régulières étendue. Par défaut, la commande Grep fonctionne avec des expressions régulières de base. Si vous souhaitez une recherche plus complexe, vous devez utiliser les regex étendus.
- **fgrep** est équivalent à la commande **grep -F** et utilise une chaîne fixe pour la recherche et il effectue donc une recherche plus rapide.

egrep et **fgrep** sont des commandes obsolètes et doivent être évitées

Exemples

```
$ more file
Génie informatique
Administration des réseaux informatiques
Management logistique et transport
$ grep "*info*" file
Génie informatique
Administration réseau informatique
$ cat file | grep -v "*info*"
Management logistique et transport
```

- Affichage avec la numérotation de toutes les lignes du fichier *prog.f* contenant *read*.
  ```
  $ grep -n "read" prog.f
  ```
- Affichage de la ligne (ou des lignes) contenant un caractère numérique dans tous les fichiers du répertoire courant.
  ```
  $ grep " *[0-9] *" .
  ```
- Recherche de tous les fichiers contenant la chaîne *toto* et affichage de leur nom.
  ```
  $ grep -l 'toto' /
  ```

Les commandes filtres

❑ Exercice

1. Chercher toutes les lignes commençant par «***a***» ou «***A***».
2. Chercher toutes les lignes finissant par «***rs***».
3. Chercher toutes les lignes commençant par une majuscule.
4. Chercher toutes les lignes commençant par «***B***», «***E***» ou «***Q***».
5. Chercher tous les mots contenant un «***r***» précédé de n'importe quelle lettre majuscule ou minuscule.

Syntaxe

tr *[option] [chaine1] [chaine2]*

Description

- Transposer ou éliminer des caractères substitution ou suppression de caractères sélectionnés.
- *tr* remplace chaque caractère du flot d'entrée appartenant à la chaîne ***[chaine1]***par le caractère de même rang dans la chaîne ***[chaine2]***
- Un filtre ne reconnaissant pas les expr. régulières.
- Dans toutes les utilisations de ***tr***, la notation *a* à *z* est autorisée pour représenter la suite des caractères de *a* à *z* (inclus) et les caractères non imprimables sont représentés par leur code ASCII en octal.
- Cette commande est le plus souvent associée à des redirections.

Options

- **-d** : Suppression de certains caractères
- **-s** : Suppression de répétitions
- **-c : tr -c *[ch] [caract]*** | remplace tout caractère NON INCLUS dans la chaine *[ch]* par le caractère *[caract]*

Exemples

- Supprime toutes les minuscules non accentuées
  $ echo « électronique" | tr -d a-z
  lectronique

- Supprime les espaces multiples entre les mots
  echo "gi   ari" | tr -s ‘ ’
  gi ari

- Pour convertir et afficher la ligne saisie au clavier en majuscules
  $ echo   "giari" | tr [a-z] [A-Z]
  GIARI

- Convertir les séquences de sauts de lignes en un seul saut de ligne (ceci supprime les lignes blanches)
  $ cat file | tr –s ‘\n’

- remplace tous les caractères différents de A à Z par un espace
  $ echo « giAri" | tr -c a-z ‘ ‘
  gi ri

Les commandes filtres

Syntaxe

sort *[option] [nom_fich]*

Description

- Permet de lire et de trier le contenu des fichiers par ordre croissant.

Options

- **-c** :Test si l'entrée est déjà triée.
- **-m** :Fusion des fichiers d'entrée qui doivent être triés.
- **-d :**Tri seulement sur l'ordre alphabétique.
- **-n :**Tri numérique.
- **-r :**Ordre du tri inversé.
- **-t** *[c]* **:**Choix du séparateur de champs *[c]* au lieu de la valeur par défaut (espace ou tabulation).
- **-k** *[n] :* le champs considéré pour le tri
- **-o** *[file]* : le résultat est mis dans fichier 'file'.

Exemple

```
$ cat  fichier
0 :Casablanca:maroc:22
1 :Rabat :maroc :37
2 :Fes :maroc :35
3 :Marrakech :maroc :24
```

```
$ cat  fichier | sort -t : -k4 -n
0 :Casablanca:maroc:22
3 :Marrakech :maroc :24
2 :Fes :maroc :35
1 :Rabat :maroc :37
```

```
$ ls | sort -d
#Lister les fichiers de répertoire courant triés par ordre alphabétique
```

Les commandes filtres - Simple

Syntaxe

cut *[option] [nom_fich]*

Description

- Permet de sélectionner (d'extraire) des caractères (colonnes) ou des champs de chaque ligne et les afficher à la sortie standard (écran).

Options

- **-d** *[séparateur]* : le séparateur des champs
- **-f** *{listes de champs}* : les champs séparés
- **-c** : Sélection sur le rang du caractère
- **-b** : Sélection sur le no d'octet
- **-s (**avec **-f)** : supprime les lignes vides

Les commandes filtres - Simple

Exemple

```
$ cat fichier
0 :Casablanca:maroc:022
1 :Rabat :maroc :037
2 :Fes :maroc :035
3 :Marrakech :maroc :024
```

```
$ cat fichier | cut -d : -f1,2
0 :Casablanca
1 :Rabat
2 :Fes
3 :Marrakech
```

Les commandes filtres

Syntaxe

uniq ***[option] [nom_fich]***

Description

- Élimine les lignes dupliquées dans un fichier.

Options

- **-d** : affiche seulement les lignes dupliquées
- **-u** : affiche seulement les lignes non dupliquées (par défaut)
- **-c** : donne le nombre d'exemplaires de chaque ligne

Syntaxe

paste ***[nom_fich1] [nom_fich2]***

Description

- Regroupe les lignes du même n° de différents fichiers

Les commandes filtres

Exemples

```
$ cat fichier
un
deux
deux
trois
quatre
quatre
cinq
```

```
$ cat fichier | uniq
un
deux
trois
quatre
cinq
```

```
$ cat fichier | uniq -d
deux
quatre
```

```
$ cat  fichier1
0 :Casablanca
1 :Rabat
2 :Fes :maroc
```

```
$ cat  fichier2
0 :maroc:22
1 :maroc :37
2 :Fes :maroc :35
```

```
$ paste fichier1 fichier2
0 :Casablanca:maroc:22
1 :Rabat :maroc :37
2 :Fes :maroc :Fes :maroc :35
```

Les commandes filtres

Syntaxe

join ***[option] [nom_fich1] [nom_fich2]***

Description

- Fusionner les lignes de deux fichiers ayant un champ commun

Options

- **-j1** *[n]* **:** jointure sur n-ième champs du premier fichier
- **-j2** *[n]* **:** jointure sur n-ième champs du deuxième fichier
- **-t** *[caract]* **:** le caractère séparateur de champs

Exemple

```
$ cat file1
0 :Casablanca:maroc:022
1 :Rabat :maroc :037
2 :Fes :maroc :035
3 :Marrakech :maroc :024
```

```
$ cat file2
Touristique : 3
Industrielle : 0
Administratif : 1
```

```
$ join  -t : -j1 1 -j2 2 file1 file2
0 :Casablanca:maroc:022: Industrielle
1 :Rabat :maroc :037: Administratif
3 :Marrakech :maroc :024: Touristique
```

→ L'utilitaire 'sed'

sed éditeur en ligne pour filtrer et transformer du texte

Description

- Permet d'appliquer automatiquement des commandes d'édition sur un fichier
- **sed** est un éditeur d'un texte arrivant sous forme d'un flux (stream) = tube venant d'une précédente commande ou redirection d'entrée.

```
... | sed -re 'commandes d'édition' | ...
```

Exemple :

```
$ ls -l | sed -re '/^-/ifichier :'
```

→ L'utilitaire 'awk'

awk langage de manipulation de motifs (patterns)

Description

- AWK permet d'écrire des traitements numériques sur des fichiers.
- Le nom de l'utilitaire awk vient de ses 3 auteurs Alfred Aho , Peter Weinberger et Brian Kernighan . Il peut être classé dans la catégorie des *filtres*. Mais c'est beaucoup plus qu'un utilitaire de gestion des fichiers textes, il intègre un langage interprété très voisin du C.

Principe général de *awk*

3 éléments :
– Un texte en entrée (redirection ou tube précédent)
– Un texte en sortie (redirection ou tube suivant)
– Un programme appelé script fourni en paramètre

La redirection des entrées-sorties

→ **Lors de l'exécution d'une commande, un processus est créé. Par défaut le processus:**

- lit ses données d'entrée à partir de l'entrée standard (par défaut le clavier).
- écrit sa sortie (les résultats et les erreurs) dans la sortie standard (par défaut l'écran).

→ **Ces canaux de communications sont désignés par les fichiers :**

- « **stdin** » : pour l'entrée standard, identifiée par l'entier **0**.
- « **stdout** »: pour la sortie standard, identifiée par l'entier **1**.
- « **stderr** »: pour la sortie d'erreur standard, identifiée par l'entier **2**.

Remarque

Il est possible de rediriger les entrées/soties standards des commandes afin que la lecture se fasse à partir d'un fichier et que l'écriture se fasse dans autre fichier.

La redirection des entrées-sorties

→ Redirection de l'entrée standard

→ On peut rediriger **l'entrée standard** d'une commande afin que la lecture se fasse à partir d'un fichier grâce au symbole « < ».

Syntaxe

commande < fichier_entree

- La commande effectue ses entrées à partir du fichier «fichier_entree »

Exemple

```
$ cat < donnees_entree
#La commande lit le fichier «donnees_entree » et l'affiche sur la sortie standard, au lieu de lire les données au clavier.
```

La redirection des entrées-sorties

→ Redirection de la sortie standard

→ On peut rediriger **la sortie standard** d'une commande afin que l'écriture se fasse dans un fichier grâce aux symboles « > » ou « >> ».

Syntaxe

commande > fichier_sortie

commande >> fichier_sortie

- Si le fichier « *fichier_sortie* » n'existe pas, alors il sera créé.
- Si le fichier « *fichier_sortie* » existe, alors:
  - Si le symbole « > » est utilisé dans ce cas le fichier « *fichier_sortie* » sera écrasé et remplacé par la nouvelle sortie standard de la commande.
  - Si le symbole « >> » est utilisé, dans ce cas la nouvelle sortie standard sera rajoutée à la fin du fichier « *fichier_sortie* ».

La redirection des entrées-sorties

→ Redirection de la sortie standard

Exemple

- On trie le fichier « fiche1 » et on écrit le résultat dans le fichier « fiche_trie » au lieu d'afficher le résultat sur l'écran.
  $ sort fiche1 > fiche_trie
- On trie le fichier « fiche2 » et on écrit le résultat à la fin du fichier « fiche_trie ».
  $ sort fiche2 >> fiche_trie

Remarque:

- On peut rediriger l'entrée et la sortie en même temps.

Exemple:

- Trier le fichier « fiche » et écrire le résultat dans le fichier « fichier_trie »
  $ sort < fiche > fichier_trie

→ Redirection de la sortie standard

Remarque

La commande « cat » permet de faire la concaténation des fichiers en redirigeant la sortie standard à l'aide des opérateurs de redirections « > » ou « >> ».

Exemple:

concaténer les fichiers « fiche1 » et « fiche2 » dans « fiche 3 »

```
$ cat fiche1 fiche2 > fiche3
```

La redirection des entrées-sorties

→ Redirection de la sortie d'erreurs standard (stderr)

→ Lors de l'exécution d'une commande, des messages d'erreurs peuvent être affichés à l'écran si un problème survient en cours d'exécution.

→ On peut rediriger **la sortie des messages d'erreurs** dans un fichier. Ceci permet d'éviter d'avoir un mélange entre les résultats normaux et les messages d'erreurs dans une même sortie (standard ou fichier). Ceci peut se faire grâce aux chaines « **2>** » ou **« 2>> ».**

Syntaxe

commande 2> fichier_sortie
commande 2>> fichier_sortie

- Comme pour la redirection de la sortie standard,
  - Si le *fichier_sortie* n'existe pas alors il sera créé.
  - Si le *fichier_sortie* existe alors:
    - Si on utilise la chaine **« 2> »** alors le fichier sera écrasé et remplacé par la nouvelle sortie standard.
    - Si on utilise la chaine **« 2>> »** , alors la sortie standard se rajoute à la fin du fichier sortie.

La redirection des entrées-sorties

→ Redirection de la sortie d'erreurs standard (stderr)

Exemple

On exécute la commande « find » sans arguments. Les résultats de la recherche sont redirigés vers le fichier «res_find» alors que les messages d'erreurs (par exemple on ne peut pas explorer un répertoire protégé en lecture) seront redirigés vers le fichier « erreur_find ».

```
$ find . > res_find 2>erreur_find
$ find . >>res_find 2>>erreur_find
```

Remarques

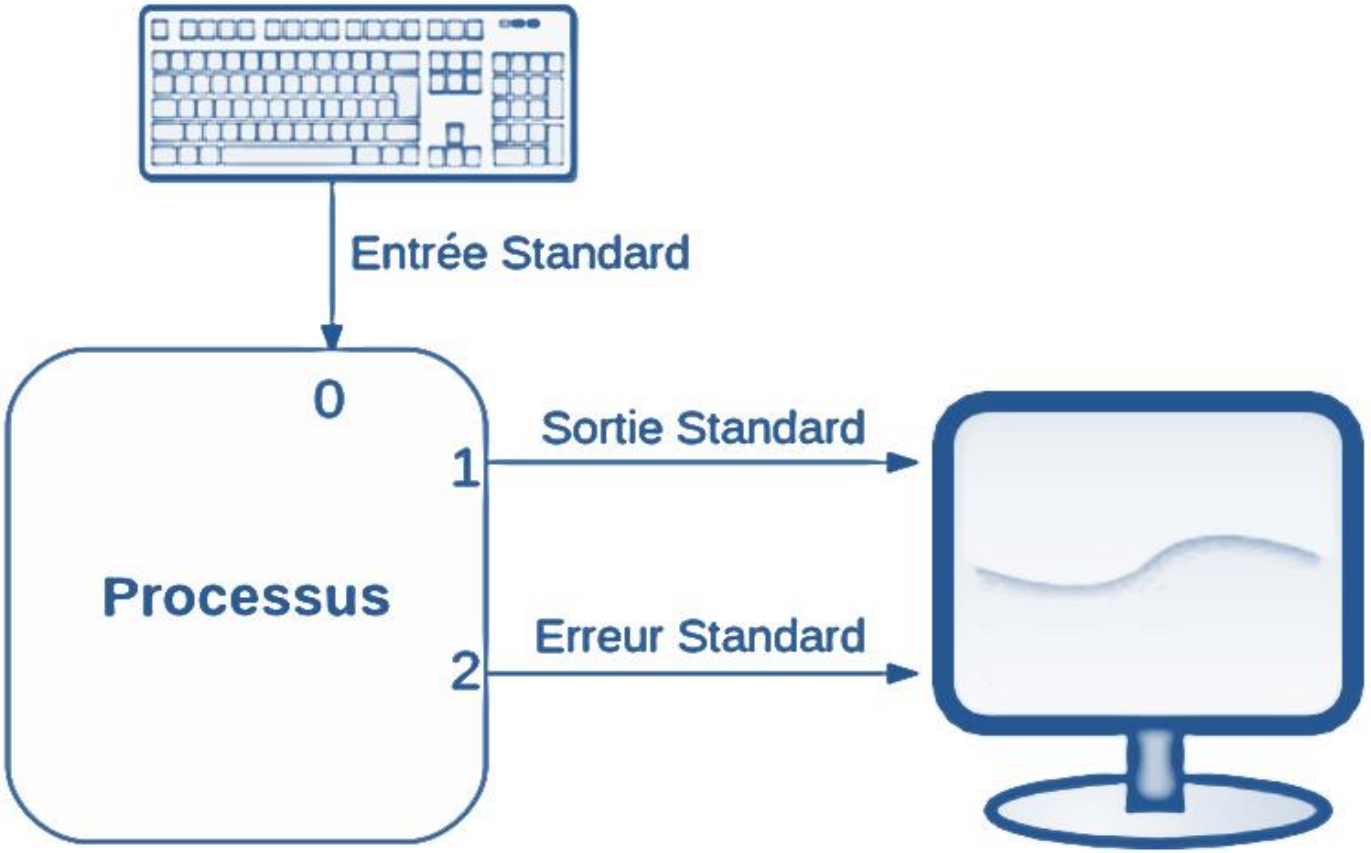
- Pas d'espace entre le "**2**" et le « > »
- Les résultats et les erreurs peuvent être redirigés vers un même fichier grâce aux chaines «&>» ou « &>> ».

Exemples

```
$ find . &>res
$ find . &>>res
```

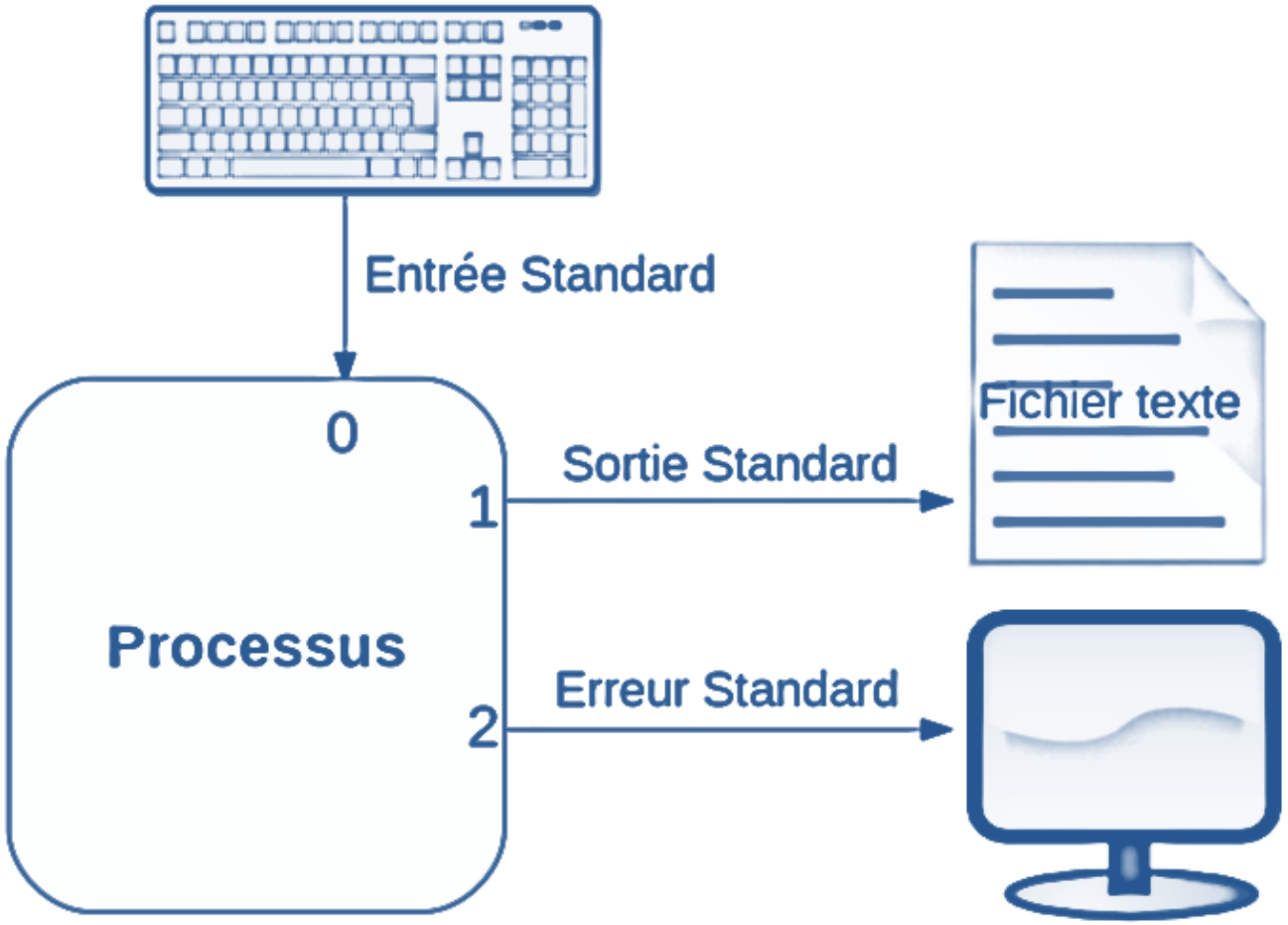
La redirection des entrées-sorties

→ **Entrée/sorties standard d'un processus**



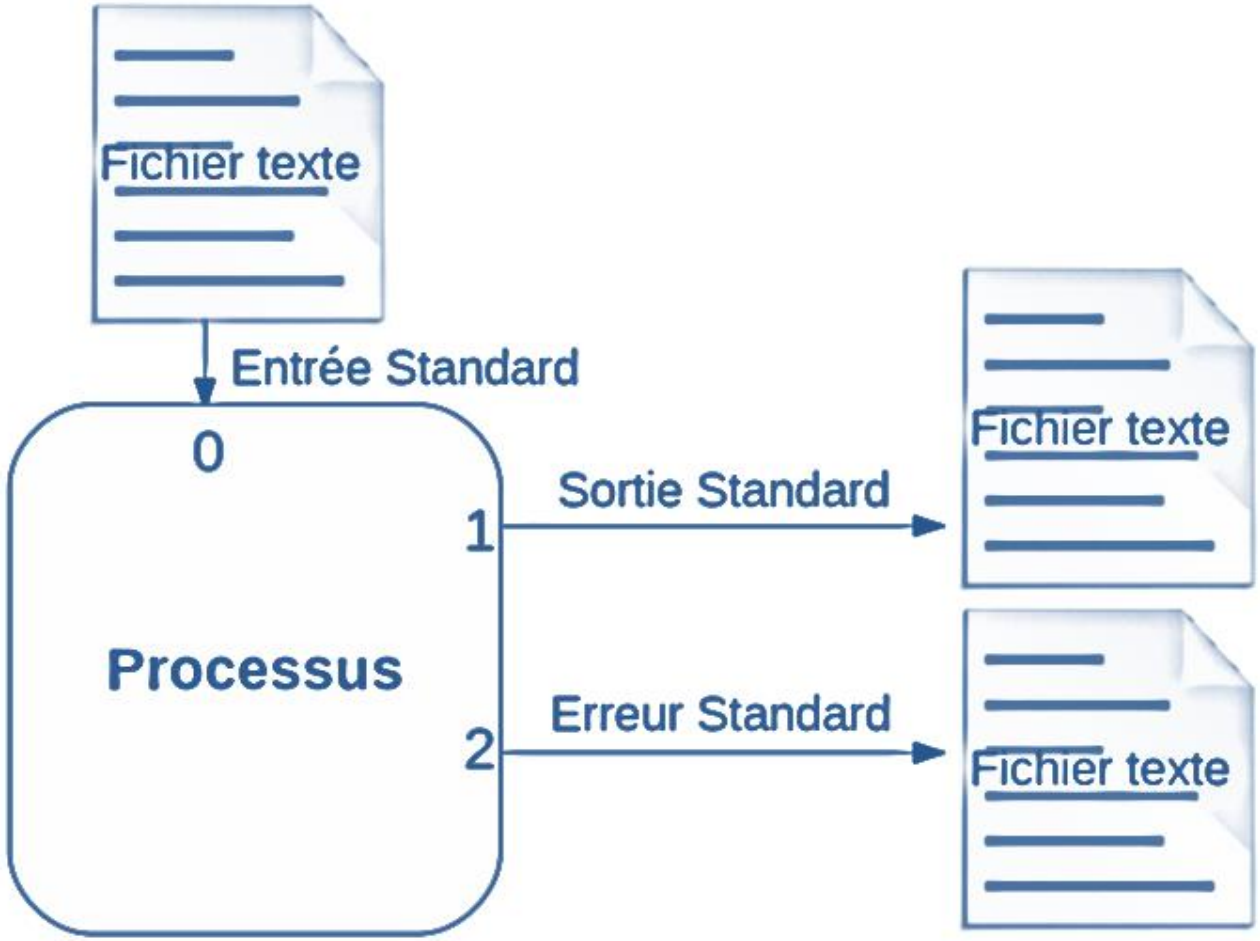
La redirection des entrées-sorties

→ **Redirection de la sortie standard vers un fichier**



La redirection des entrées-sorties

→ **La redirection vers des fichiers**



❑ Métacaractères de redirection

	C shell	Bourne shell
entrée standard	<	<
entrée standard direct	<<	<<
sortie standard	>	>
sortie standard concaténée	>>	>>
sortie erreur		2> ou 2>>
sortie standard + erreur	>& ou >>&	
pipe	|	|

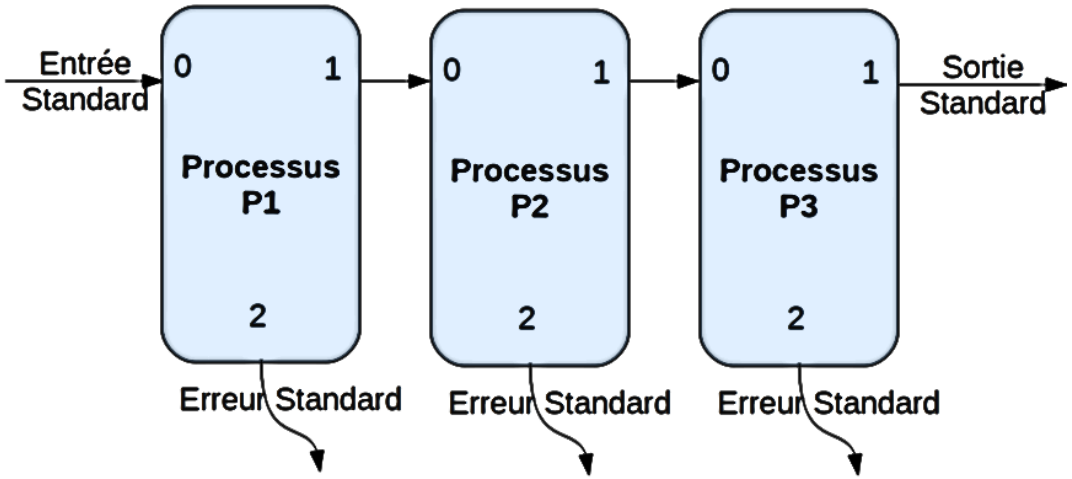
Les tubes de communication- Pipes

→ Les **pipes** (**tubes**) permettent de rediriger la sortie d'un processus vers l'entrée d'un autre. Les processus s'exécutent en parallèles.

Syntaxe

CmdA | cmdB

- « **cmdA** » et « **cmdB** » sont deux commandes.
- Le symbole « | » est appelé « pipe ». Il permet de relier deux commandes entre elles.
- La sortie standard de la commande « **cmdA** » est utilisée comme entrée standard de la commande « **cmdB** ».



P1 | P2 | P3

Les tubes de communication- Pipes

Exemples

- Exécuter la commande « ls -l » et rediriger la sortie de la commande vers le fichiers « resultat.txt »
  $ **ls -l > resultat.txt**

- Compter le nombre de lignes, le nombre de mots et le nombres de caractères du fichier « resultat.tx » et rediriger le résultat vers le fichier « cp.txt »
  $ **wc < resultat.txt > cp.txt**
  ou
  $ **wc resultat.txt > cp.txt**

- Les deux commandes peuvent être combinées pour élaborer une seule:
  $ **ls -l | wc > cp.txt**