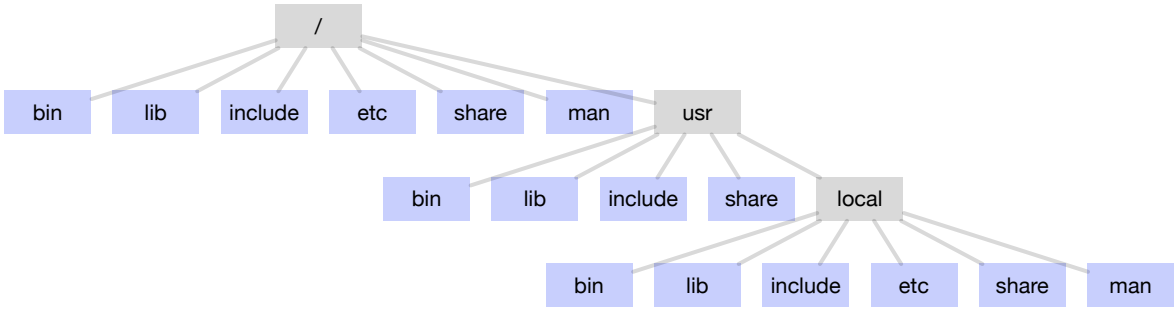
Le système de fichiers
Arborescence Unix

- Les répertoires classiques sous Unix sont :
  - **bin**, **sbin** : binaires et binaires système
  - **lib** : les librairies, partagées ou non
  - **include** : les entêtes décrivant la librairie C
  - **etc** : fichiers système
  - **share** : resources indépendantes de l'architecture
  - **man** (ou **share/man**) : le manuel
  - **var** : les données "variables"
  - **tmp** : les fichiers (en principe) temporaires

Le système de fichiers
Arborescence Unix

- Ces répertoires peuvent se retrouver à plusieurs niveaux du système de fichier :
  - **/** : racine du système, tout ce qui y est directement lié
  - **/usr** : tout ce qui est lié à l'utilisation du système par les utilisateurs
  - **/usr/local** : tout ce qui "local" à un site
  - **/usr/local/samba** : tout ce qui est lié à un produit logiciel donné
  - ...



Le système de fichiers
Nommage de fichiers et répertoires

- Du point de vue du noyau un nom de fichier peut contenir tout caractère sauf deux :
  - **/** qui est réservé pour séparer les répertoires et fichiers dans un nom de chemin.
  - le caractère **nul** (\0) qui est réservé pour terminer les chaînes de caractères en C.
- Cependant pour faciliter le travail avec le shell il est recommandé d'utiliser seulement
  - des lettres minuscules et majuscules
  - des chiffres
  - le point **.**
  - l'underscore **_**
  - le tiret **-**
- Unix n'exige pas de point **.** dans le nom de fichier.
  - On voit par exemple très souvent : **`README`**
  - Un nom peut contenir autant de points qu'on veut : **`archive.tar.gz`**
- Un nom de fichier doit être unique dans son répertoire.

Le système de fichiers
Nommage de fichiers et répertoires

- L'utilisation de l'espace est possible mais il faut faire attention.
  - Le shell interprète l'espace comme séparateur de paramètres.
  - Si un nom de fichier comporte des espaces, le mettre entre guillemets "" : **`"a confusing name"`**
  - ou précéder chaque espace par un backslash \ : **`a\ confusing\ name`**

```
$ ls -l
total 4
-rw-rw-r-- 1 marcel.graf marcel.graf    0 Mar 10 12:36 a confusing name
-rw-rw-r-- 1 marcel.graf marcel.graf 2921 Mar 10 12:37 ch01
$ rm a confusing name
rm: cannot remove `a': No such file or directory
rm: cannot remove `confusing': No such file or directory
rm: cannot remove `name': No such file or directory
$ rm "a confusing name"
$
```

Le système de fichiers

Noms de fichiers cachés

- Un nom de fichier qui commence avec un point **.** est appelé nom de fichier caché *(hidden filename)* parce que normalement la commande **ls** ne le montre pas.
  - Pour afficher tous les fichiers, les fichiers cachés inclus, utiliser l'option **-a**

```
$ ls
play  work
$ ls -a
.   .archive       .bash_logout  .byobu  .emacs.d  play      .screenrc  .tmux.conf
..  .bash_history  .bashrc       .cache  .lesshst  .profile  .ssh       work
$
```

  - Le répertoire personnel contient généralement un grand nombre de fichiers cachés. Ils sont généralement utilisés par divers programmes pour stocker des paramètres de configuration.

Le système de fichiers

Le *globbing* avec des caractères génériques ou jokers *(wildcards)*

- Le shell connaît un nombre de méta-caractères pour l'interprétation des noms de fichiers
- On peut les utiliser pour spécifier plusieurs fichiers à la fois.
  - ***** représente une chaîne de caractères quelconque (même vide).
  - **?** représente un caractère quelconque.
- **[...]** représente un caractère quelconque dans l'ensemble défini entre [et].
- **[!...]** représente un caractère quelconque hors de l'ensemble.
- Un ensemble est défini par une liste de caractères (**[aeiouy]**) ou un intervalle (**[0-9]**) ou toute combinaison des deux (**[ab0-9AB]**).

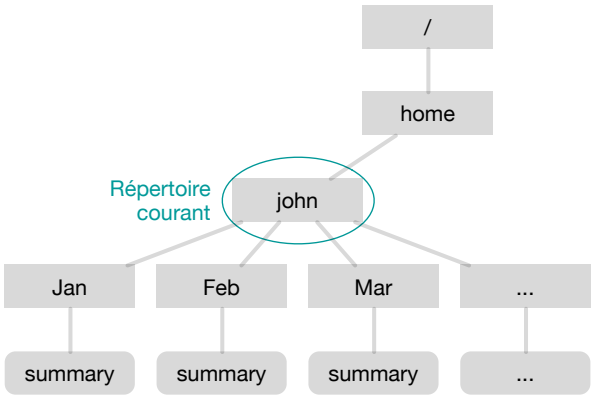
```
$ ls
chap01a.old  chap04  chap10
chap01b      chap05  cold
chap02       chap06  haha
chap03.old   chap07  oldjunk
$ ls chap0?
chap02  chap04  chap05  chap06  chap07
$ ls chap0[5-7]
chap05  chap06  chap07
$
```

```
$ ls chap[!0]*
chap10
$ ls chap??
chap02  chap05  chap07
chap04  chap06  chap10
$ ls *old
chap01a.old  chap03.old  cold
$ ls *a*a*
chap01a.old  haha
$
```

Le système de fichiers

Le *globbing* avec des caractères génériques ou jokers *(wildcards)*

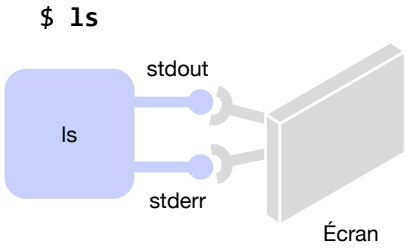
- Les caractères génériques peuvent aussi être utilisés au sein d'un nom de chemin.
- Pour concaténer tous les fichiers nommés **`summary`** dans l'exemple à droite :
  - **`cat */summary`**
  - Presque équivalent à taper **`cat Jan/summary Feb/summary ...`** avec une différence : les noms sont triés par ordre alphabétique, et **`Apr/summary`** sera le premier dans la liste.

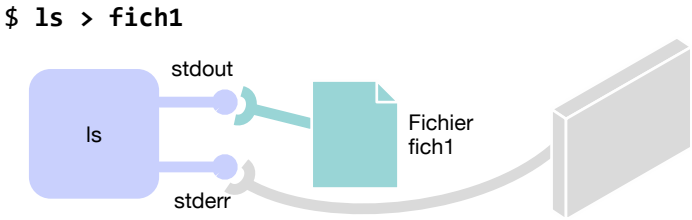


Le shell

Redirection des entrées/sorties

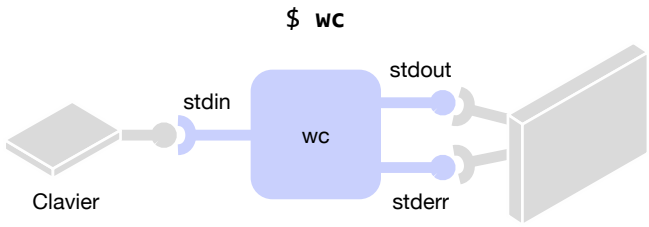
- Quand on tape la commande **`ls`** elle produit un affichage sur le terminal.
  - Cet affichage provient du flux de sortie **stdout** du programme **`ls`**.
  - Par défaut le shell connecte **stdout** à l'écran du terminal.
  - Il y a un deuxième flux de sortie **stderr** pour les messages d'erreur. Par défaut il est aussi connecté à l'écran.
- On peut rediriger la sortie de la commande **`ls`** dans un fichier en ajoutant le symbole **>** et le nom du fichier.
  - Rien ne sera affiché à l'écran (sauf s'il y a une erreur).
  - La sortie se trouvera dans le fichier.



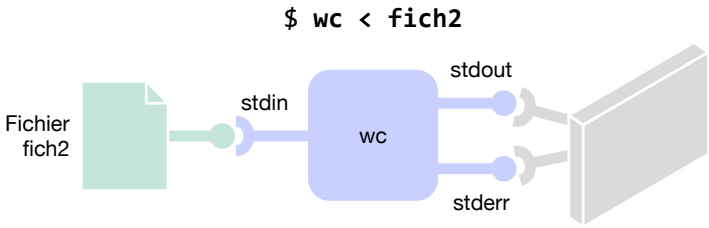


Le shell
Redirection des entrées/sorties

- La commande **`wc`** *(word count)* sert à compter les lignes, les mots et les caractères dans un texte.
  - La commande attend que l'utilisateur saisisse le texte au clavier. Après un **Ctrl-D** pour signaler la fin du texte la commande affiche le résultat.
  - Par défaut le shell connecte le flux d'entrée **stdin** du programme **`wc`** au clavier du terminal.

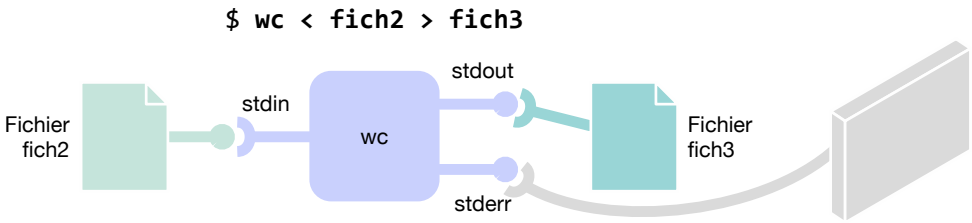


- On peut rediriger l'entrée de la commande **`wc`** afin que la lecture se fasse sur un fichier en ajoutant le symbole **`<`** et le nom du fichier.
  - La commande ne va plus attendre une saisie de l'utilisateur.
  - Le texte sera lu du fichier.



Le shell
Redirection des entrées/sorties

- On peut combiner la redirection de l'entrée et de la sortie sur la même commande.

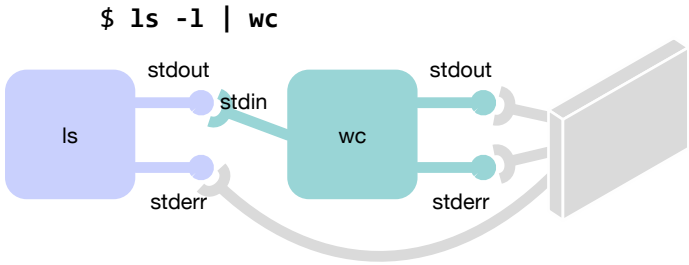


- NB: On peut mettre les redirections n'importe où, commande équivalente **`> fich3 < fich2 wc`**

Le shell

Redirection des entrées/sorties — Les tubes

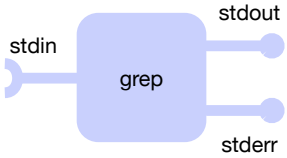
- Un tube *(pipe)* est une zone mémoire permettant à deux programmes de communiquer. L'objectif est de faire agir une commande sur le résultat d'une autre sans fichiers intermédiaires.
- Le symbole **|** placé entre deux commandes redirige la sortie standard **stdout** de la première sur l'entrée standard **stdin** de la seconde grâce à un tube.



Le shell

Redirection des entrées/sorties — Les commandes filtres

- Certaines commandes sont conçues pour être utilisées comme filtre dans une séquence de commandes connectées par des tubes.
  - **`wc`** — Compter les lignes, mots et caractères
  - **`sort`** — Trier des données
  - **`uniq`** — Supprimer des doublons
  - **`grep`** — Extraire des lignes avec des critères de recherche
  - **`sed`** — Substituer des chaînes de caractère
  - **`cut`** — Découper des lignes en champs
  - **`head`** — Extraire les premières lignes
  - **`tail`** — Extraire les dernières lignes
  - ...



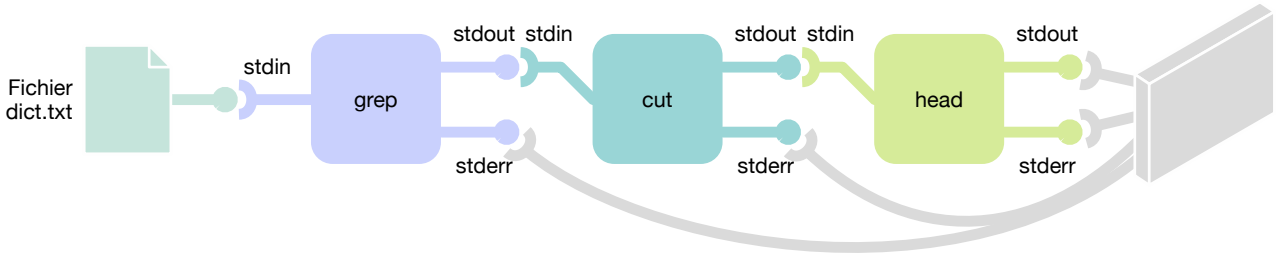
Le shell

Redirection des entrées/sorties — Les pipelines

- À partir de commandes simples on peut construire des pipelines de traitement complexes.
- Pour alimenter le pipeline on peut utiliser une redirection de l'entrée depuis un fichier ...
- ... et le résultat final peut être redirigé vers un autre fichier.

- Exemple : Chercher dans un dictionnaire anglais tous les mots d'origine française et afficher les dix premiers.

```
$ grep french < dict.txt | cut -d ' ' -f 1 | head
```



Le shell

Redirection des entrées/sorties — Précisions

- La redirection de la sortie avec **>** va écraser le fichier s'il existe déjà !
  - Pour éviter cela utiliser la variante avec double **>>**. La sortie sera **ajoutée** à la fin du fichier.
  - Si le fichier n'existe pas encore il sera créé.

```
$ date > who.log
$ who >> who.log
```

- Pour rediriger la sortie d'erreurs standard **stderr** utiliser la notation **2>**
  - Il ne doit pas y avoir d'espace entre le **2** et le **>**

```
$ ls 2014-03-?? > march2014.txt 2> errors.txt
```

  - Parfois on souhaite supprimer complètement les messages d'erreur. Dans ce cas rediriger la sortie d'erreurs standard dans **`/dev/null`**. **`/dev/null`** est un pseudo-fichier qui absorbe et vaporise toutes les données qu'on lui donne.

```
$ ls 2014-03-?? > march2014.txt 2> /dev/null
```

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Recherche de lignes — **grep**

- La commande **`grep`** parcourt les fichiers pour chercher un motif. Elle émet seulement les lignes contenant le motif et supprime le reste.
- On donne les fichiers à chercher en derniers paramètres. Si on les omet, **`grep`** lit l'entrée standard.
- Le motif peut contenir des métacaractères
  - **`.`** représente un caractère quelconque
  - **`[...]`** représente un caractère quelconque dans l'ensemble défini entre **`[`** et **`]`**.
  - **`[^...]`** représente un caractère quelconque hors de l'ensemble.
    - Un ensemble est défini par une liste de caractères (**`[aeiouy]`**) ou un intervalle (**`[0-9])`** ou toute combinaison des deux (**`[ab0-9AB]`**).

```
$ cat to_be.txt
To be, or not to be, that is the question—
Whether 'tis Nobler in the mind to suffer
The Slings and Arrows of outrageous Fortune,
Or to take Arms against a Sea of troubles,
$ cat to_be.txt | grep or
To be, or not to be, that is the question—
The Slings and Arrows of outrageous Fortune,
$
```

  - **`*`** Répétition du motif précédant l'astérisque de 0 à n fois, par exemple **`a*`** remplace **`a`**, **`aa`**, **`aaa`**, ...
  - **`^`** La chaîne recherchée devra être en début de ligne
  - **`$`** La chaîne recherchée devra être en fin de ligne
- Pour éviter que le shell interprète les métacaractères il faut mettre le motif entre apostrophes : **`grep 'Chapter [0-9]'`**

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Recherche de lignes — **grep**

- Quelques options utiles
  - **`-i`** : ne différencie pas les minuscules et les majuscules
  - **`-v`** : effectue la recherche inverse : toutes les lignes ne correspondant pas aux critères sont affichées
  - **`-c`** : ne retourne que le nombre de lignes trouvées sans les afficher
  - **`-n`** : affiche le numéro de ligne pour chaque ligne trouvée

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Sélection de colonnes et champs — **cut**

- La commande **`cut`** permet de sélectionner des colonnes et des champs dans un fichier. Elle émet seulement les éléments sélectionnés et supprime le reste.
  - Colonnes : On considère que le premier caractère dans une ligne est dans la colonne 1, le deuxième dans la colonne 2, et ainsi de suite. On peut sélectionner
    - une colonne seule : **`-c2`**
    - une plage : **`-c2-4`** pour les colonnes 2, 3 et 4.
      - les premières colonnes : **`-c-3`** pour les trois premières colonnes,
      - les dernières colonnes : **`-c4-`** pour les dernières colonnes à partir de la colonne 4.
    - une liste de colonnes : **`-c1,3,6`**
    - les trois notations combinées : **`-c1-3,5,6,12-`**
  - Champs : On considère que le texte contient un caractère qui sépare des champs (séparateur).
    - Par défaut, le séparateur est la tabulation. Pour le changer utiliser l'option **`-d`** : Par exemple **`-d ' '`** pour mettre l'espace comme séparateur.
    - La sélection des champs se fait avec l'option **`-f`** en utilisant la même syntaxe comme pour les colonnes : **`-f2`** pour un champ seul, **`-f2-4`** pour les champs 2, 3 et 4, et ainsi de suite.
- S'il n'y a pas de caractère séparateur dans une ligne, **`cut`** émet toute la ligne.

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Sélection de colonnes et champs — **cut**

- Exemple : On suppose un fichier contenant des commandes de clients. Chaque ligne contient la date de la commande, le nom du client, l'article commandé et le prix.

```
$ cat orders.txt
2014-03-11 Jones Apples 23.45
2014-03-11 Smith Apples 12.33
2014-03-12 Taylor Oranges 18.20
2014-03-12 Jones Bananas 4.88
2014-03-13 Taylor Apples 33.30
$
```

- Les champs sont séparés par des espaces. Sélectionner les articles (3ème champ)

```
$ cat orders.txt | cut -d ' ' -f3
Apples
Apples
Oranges
Bananas
Apples
$
```

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Décompte de lignes — **wc**

- La commande **`wc`** *(word count)* permet de compter les lignes, les mots et les caractères d'un fichier.
- Appelée sans options la commande affiche trois nombres :
  - le nombre de lignes
  - le nombre de mots
  - le nombre de caractères
- Dans les scripts on veut souvent afficher seulement un nombre. Alors utiliser les options
  - **`-l`** : affiche le nombre de lignes
  - **`-w`** : affiche le nombre de mots
  - **`-c`** : affiche le nombre d'octets
  - **`-m`** : affiche le nombre de caractères

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Tri de lignes — **sort**

- La commande **`sort`** permet de trier les lignes d'un fichier. On peut spécifier quels champs doivent être utilisés pour le tri (critère de tri).
  - Appelée sans options la commande par défaut
    - applique un critère de tri qui est le contenu entier de la ligne
    - suppose que le critère de tri est un texte et non pas une valeur numérique
    - trie en ordre croissant.
  - Critère de tri : On spécifie un ou plusieurs champs qui déterminent l'ordre du tri. Utiliser l'option **`-k`** pour donner le critère de tri :
    - **`-k2`** pour le champ 2
    - **`-k2,4`** pour le champ 2, 3 et 4
    - Si les champs sont numériques, ajouter un **`n`** : **`-k5n`** pour le champ 5 interprété comme valeur numérique
    - Pour trier en ordre inverse ajouter un **`r`** : **`-k5nr`** pour le champ 5 numérique en ordre décroissant
    - On appelle l'option **`-k`** la *clé primaire*. S'il y a égalité dans les valeurs de la clé, on peut spécifier une *clé secondaire* (en répétant l'option **`-k`** avec d'autres champs) qui détermine l'ordre pour ces lignes. On peut aussi spécifier une *clé tertiaire*, et ainsi de suite.
- Par défaut la commande considère un champ tout ce qui est séparé par un ou plusieurs whitespace.
  - Pour utiliser un autre caractère utiliser l'option **`-t`**, par exemple **`-t '<tab>'`** (pour insérer le caractère tab dans la commande il faut taper **Ctrl-V** puis la touche **Tabulation**).

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Tri de lignes — **sort**

- Exemple : Trier les commandes par article

```
$ cat orders.txt | sort -k3
2014-03-11 Smith Apples 12.33
2014-03-11 Jones Apples 23.45
2014-03-13 Taylor Apples 33.30
2014-03-12 Jones Bananas 4.88
2014-03-12 Taylor Oranges 18.20
$
```

- Trier les commandes par prix (champ numérique) en ordre décroissant

```
$ cat orders.txt | sort -k4nr
2014-03-13 Taylor Apples 33.30
2014-03-11 Jones Apples 23.45
2014-03-12 Taylor Oranges 18.20
2014-03-11 Smith Apples 12.33
2014-03-12 Jones Bananas 4.88
$
```


La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Suppression des doublons — **uniq**

- La commande **`uniq`** est surtout utile pour agir sur le résultat de la commande sort. Elle permet de supprimer les doublons et compter les occurrences des valeurs.
  - Appelée sans options la commande remplace toute ligne répétée par une seule.
    - C'est utile pour produire une liste de valeurs uniques
  - Pour compter les occurrences des valeurs, utiliser l'option **`-c`**

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Suppression des doublons — **uniq**

- Exemple : Afficher une liste unique des articles qui surviennent dans les commandes

```
$ cat orders.txt | cut -d ' ' -f3 | sort | uniq
Apples
Bananas
Oranges
$
```

- Exemple : Compter le nombre de commandes effectuées par chaque client

```
$ cat orders.txt | cut -d ' ' -f2 | sort | uniq -c
      2 Jones
      1 Smith
      2 Taylor
$
```


La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Remplacement de caractères — **tr**

- La commande **tr** permet de substituer des caractères à d'autres et n'accepte que des données provenant du canal d'entrée standard, pas les fichiers.
  - **tr** *[options] original destination*
  - L'original et la destination représentent un ou plusieurs caractères. Les caractères originaux sont remplacés par les caractères de destination dans l'ordre indiqué. Les crochets permettent de définir des plages.
- Exemple : Remplacer les espaces par des double points

```
$ cat orders.txt | tr ' ' ':'
2014-03-11:Jones:Apples:23.45
2014-03-11:Smith:Apples:12.33
2014-03-12:Taylor:Oranges:18.20
2014-03-12:Jones:Bananas:4.88
2014-03-13:Taylor:Apples:33.30
$
```

- Quelques options utiles :
  - **-s** *(squeeze)* : Si le caractère est répété les répétitions sont supprimées. Utile pour supprimer plusieurs espaces.

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Téléchargement du web — **curl**

- La commande **`curl`** permet de télécharger des pages HTML et des fichiers du web.
  - **`curl`** *[options] [URL...]*
- Appelée sans options avec un URL seulement la commande télécharge la page ou le fichier demandé et l'envoie sur la sortie standard.
  - **`curl 'http://www.heig-vd.ch/'`** : Télécharge la home page de l'HEIG-VD et l'envoie sur la sortie standard
  - Souvent les URL contiennent des caractères que le shell pourrait interpréter comme méta-caractères. Prendre l'habitude d'envelopper les URL par des apostrophes ' '.
- Quelques options utiles :
  - **`-O`** : Stocker le résultat dans un fichier nommé comme le fichier demandé (dernière partie de l'URL).
  - **`-o`** *fichier* : Stocker le résultat dans le fichier spécifié.

La ligne de commande Unix

Les outils pour la manipulation de fichiers de texte — Substitution de texte — **sed**

- La commande **`sed`** *(**s**tream **ed**itor)* permet de substituer un mot (ou une séquences de caractères) dans un fichier texte par un autre mot.
  - En réalité la commande **`sed`** permet des manipulations de texte beaucoup plus générales, mais dans ce cours nous nous limitons au cas d'utilisation le plus courant.
  - **`sed`** lit le flux d'entrée ligne par ligne et exécute la substitution demandée
  - **`sed -e 's/`**<motif>**`/`**<remplacement>**`/`**<drapeaux>**`'`**
    - <motif> : Le motif recherché
    - <remplacement> : Le premier motif trouvé sur une ligne sera remplacé par ce remplacement
    - <drapeaux> : Le drapeau **`g`** indique que toutes les occurrences du motif, pas seulement la première, sont à remplacer.
- Exemple : remplacer toutes les occurrences du motif **`OA`** avec **`Oceanic Airlines`**

```
$ cat announcement.txt
OA welcomes you on our OA flight to Oakland.
Please remain seated with your seatbelts
fastened until the seatbelt sign is switched
off.
```

```
$ sed -e 's/OA/Oceanic Airlines/g'
< announcement.txt
Oceanic Airlines welcomes you on our Oceanic
Airlines flight to Oakland.
Please remain seated with your seatbelts
fastened until the seatbelt sign is switched
off.
```