



# Outils indispensables

CSC3102 – Introduction aux systèmes d'exploitation  
Elisabeth Brunet





# Plan

## ■ Outils incontournables

- Nature d'une entrée
- Pour les fichiers texte : affichage, tri, recherche de motif
- Occupation disque
- Archivage de fichiers
- Recherche de fichiers

# Nature d'une entrée du système de fichiers

- Traitement applicable à un fichier dépend de sa nature
  - Est-ce un fichier texte ? Une image ? Une archive ? Un pdf ?
- Commande `file` : affiche la nature d'une entrée
  - Si texte, précise le type de codage
    - ASCII s'il n'y a que des caractères, UTF-8 si caractères accentués, etc.

```
$ file *
TP3                directory
TP3.html           exported SGML document, UTF-8 Unicode text
Ci3.pdf            PDF document, version 1.5
Ci3.pptx           Microsoft Powerpoint 2010
Notes.txt          ASCII text
Pedagogie.txt      UTF-8 Unicode text
```

# Nature d'une entrée du système de fichiers

- La commande `test` teste aussi la nature d'un fichier  
(*rappel `test cond` peut s'écrire `[ cond ]` avec `bash`*)
  - `[ -e fichier ]` : vrai si fichier existe
  - `[ -f fichier ]` : vrai si fichier existe et est normal
  - `[ -d fichier ]` : vrai si fichier existe et est répertoire
  - `[ -L fichier ]` : vrai si fichier existe et est un lien symbolique  
(remarque : les autres tests suivent les liens symboliques)

# Taille de l'occupation disque

- `df` : connaître l'état d'occupation des partitions
- `ls -lh chem ...` : taille des chemins cibles
  - Si répertoire, donne la taille nécessaire au stockage de sa table d'entrées mais n'inclut pas celle de ses sous-entrées
  - Si lien symbolique, donne sa taille, i.e. l'espace nécessaire au stockage du chemin vers sa cible, ce qui correspond au nombre de caractères de ce chemin
- `du` : totalise l'occupation disque d'une entrée
  - Si répertoire, parcours récursif de son arborescence
  - Par défaut, donne le nombre de blocs occupés
    - Option `-h`, pour afficher l'équivalent de ce nombre de blocs de manière « lisible pour un humain » en o/K/M/G
    - Option `-d0` pour éviter l'affichage des tailles des sous-répertoires

# Archivage

- Commande `tar` (pour **t**ape **a**rchive) ⇒ manipuler des archives  
Archive = rassemblement d'une arborescence de fichiers en un seul fichier
  - `tar -czf fic.tgz rep` : crée l'archive `fic.tgz` à partir de `rep`
  - `tar -xf fic.tgz` : extrait l'archive `fic.tgz`
  - Option `-c chem`, pour créer l'archive à partir du chemin `chem`
  - Option `-v`, pour un affichage en mode verbeux
  - Option `-z`, pour une compression des données au format gzip
  - Option `-f nom.tgz`, pour préciser le nom de l'archive voulue
  - Par convention, extension `.tgz` ou `.tar.gz`
  - Option `-x`, pour extraire (`-z`, pour la décompression via gzip)  
⇒ décompression dans le répertoire courant

# Affichage d'un fichier en mode texte

- Consultation du contenu d'un fichier ordinaire

- `more fichier`
- `less fichier`

} affichage simple page par page

- `head -n k <fichier>` : affichage des `k` premières lignes

- `tail -n k <fichier>` : affichage des `k` dernières lignes

- `cat fic1 fic2...` : affiche la concaténation des fichiers indiqués

- `wc fic` : compte les lignes, mots et caractères du fichier

- Option `-l`, uniquement les lignes ; `-w`, les mots ; `-c`, les caractères

# À propos de cat et des commandes qui suivent

- Pour toutes les commandes qui suivent : si aucun fichier n'est donné en argument, la commande s'applique sur l'entrée standard
  - Rappel : `ctl-d` génère un code de fin de fichier (EOF)
- Par exemple :
  - `cat fic` : affiche le contenu de `fic`
  - `echo coucou | cat` : affiche `coucou`



# Extraire des parties de lignes

- `cut -c plage fic` : extrait des caractères de chaque ligne de `fic`

- `plage` : `num` **OU** `num1, num2, ...` **OU** `num1-num2`

Exemple : `cut -c3-7 fic.txt`

⇒ extrait les caractères 3 à 7 de `fic.txt`

- `cut -d car -f plage fic` : extraits des champs

- `-d car` : `car` = séparateur de champs (tabulation par défaut)
- `plage` comme dans le cas précédent

Exemple : `cut -d' ' -f2,4 fic.txt`

⇒ extrait les 2<sup>ième</sup> et 4<sup>ième</sup> mots de chaque ligne de `fic.txt`

# Supprimer ou transformer des caractères

- `tr s1 s2` : transforme chaque caractère de `s1` en ceux de `s2` (à partir de `stdin`)
  - Exemple : `cat fic | tr '\n ' 'ab'`  
⇒ transforme les retours à la ligne en a et les espaces en b
  - Exemple : `cat fic | tr '\n ' 'a'`  
⇒ transforme les retours à la ligne et espaces en a
- `tr -d s` : élimine chaque caractère de la chaîne `s`
  - Exemple : `cat fic | tr -d 'aeiouy'`  
⇒ élimine les voyelles de `fic`

# Trier les lignes de fichiers texte

## ■ `sort fic...`

- Par défaut, tri lexicographique
  - Option `-n` pour un tri numérique
- Par défaut, tri appliqué en tenant compte de toute la ligne
  - Option `-k x, x` pour un tri selon le champs `x`
    - `sort -k 2,2 fic` : tri selon le 2<sup>ème</sup> champ de chaque ligne
    - `sort -k 2,2 -k 3,3 fic` : tri selon les 2<sup>ème</sup> et 3<sup>ème</sup> champs
    - Remarque : pour un tri non numérique, `-k x, y` pour champs `x` à `y`
- Par défaut, le séparateur de champs est l'espace
  - Option `-t <caractère>` pour changer le séparateur
- Option `-r` pour inverser l'ordre du tri appliqué
- Peut s'appliquer sur un ensemble de fichiers
- D'autres options à consulter dans la page du manuel

# Recherche d'un motif dans un fichier texte

- `grep motif fichier fic ...`
  - Affiche les lignes des fichiers contenant le motif
  - Le motif est une expression régulière (ou rationnelle)
    - `grep` = *global regular expression print*
  - Pour CSC3102, seul un sous-ensemble d'expressions régulières GNU
    - Chaînes de caractères
    - **Attention! Les méta-caractères de `grep` sont différents de ceux de `bash` !**
      - `.` : n'importe quel caractère
      - `*` : répétition du caractère précédent 0 ou plusieurs fois
      - `[...]` (`/ [^...]`) : met en correspondance un caractère de (`/`hors) l'ensemble
    - **Attention : mettre le motif entre guillemets (« " »)**
  - Option `-v` affiche les lignes ne contenant pas le motif
  - Option `-r` permet de chercher récursivement le motif dans la sous-arborescence du répertoire donné en argument
  - D'autres options à consulter dans la page de manuel

# Recherche dans une arborescence

- `find` : recherche des entrées satisfaisants un ensemble de critères de sélection dans une arborescence
  - Parcourt récursivement et teste les critères à chaque étape
  - `find rep_de_recherche liste des critères`
    - `-name "<motif>"` : précise le nom des entrées à rechercher
      - `<motif>` est motif (expression régulière) différent de celui utilisé par `bash` ou `grep`
      - **Attention : mettre le motif entre guillemets (« " »)**
    - `-print` : permet l'affichage des résultats (par défaut)
    - Exemple : `find . -name core -print`
      - affiche les chemins des entrées nommées `core` de mon répertoire courant
    - `find /usr -name "*.c" -print`
      - affiche les chemins des entrées dont le nom se terminent par `.c` sous `/usr`

# Conclusion

## ■ Commandes clés :

- `more, less, head, tail, cat, wc`
- `cut, tr, sort, grep`
- `df, du, ls -lh`
- `tar`
- `find,`



# En route pour le TP!