

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`
- `tar -tf fic.tgz` : liste le contenu de 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
- Option `-t`, pour lister

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 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`
 - `cat > fic` : écrit ce qui est tapé en entrée standard dans `fic`

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 avec `-c`

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

⇒ extrait les 2^{ième} et 4^{ième} mots de chaque ligne de `fic.txt`

Supprimer ou transformer des caractères

- `tr s1 s2` : transforme chaque caractère de l'ensemble `s1` en ceux de l'ensemble `s2`

(la commande ne prend pas de fichier en argument : toujours 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^{ème} champ de chaque ligne
 - `sort -k 2,2 -k 3,3 fic` : tri selon les 2^{ème} et 3^{ème} 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 du texte (1/3)

■ `grep motif fic1 fic2 ...`

- Affiche les lignes des fichiers qui contiennent le motif
 - Peut aussi lire l'entrée standard : `cat fic | grep 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
 - `^ / $` : ancre le motif au début / à la fin de la ligne
 - **Attention : mettre le motif entre guillemets simples (« ' »)**

Recherche d'un motif dans du texte (2/3)

■ Options :

- `-v` : inverse le motif (affiche les lignes qui ne le contiennent pas)
- `-r` : traite récursivement les fichiers dans le dossier passé en argument
- `-i` : ignore la casse
- `-q` : n'affiche rien, seul le code de retour indique le résultat
 - Utile pour seulement tester la présence du motif
 - Code de retour 0 \Leftrightarrow motif trouvé

■ D'autres options à consulter dans la page du manuel

Recherche d'un motif dans du texte (3/3)

■ Exemples :

- `grep warning *.log`
 - Affiche les lignes contenant `warning` de tous les fichiers `.log` du dossier courant
- `grep -i warning *.log`
 - Comme ci-dessus, mais en ignorant la casse
- `grep -v '^[mn]' fic`
 - Affiche les lignes de `fic` ne commençant pas par `m` ou `n`
- `grep '(.*)$' fic`
 - Afficher les lignes qui se terminent par des caractères quelconques entre parenthèses

Recherche dans une arborescence

- `find` : recherche des entrées satisfaisants un ensemble de critères de sélection dans une arborescence
 - Parcourt récursivement et teste les critères sur chaque entrée
 - `find rep_de_recherche liste des critères`
 - `-name "<motif>"` : précise le nom des entrées à rechercher
 - `<motif>` est motif conforme à `bash` à l'exception des crochets [...]
 - **Attention : mettre le motif entre guillemets (« " »)**
 - `-type <type>` : précise le type des entrées à rechercher
 - `f` : fichier normal ; `d` : dossier ; `l` : lien symbolique
 - `-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!