Franz Girardin

Technologies de l'internet

NOTES DE COURS

Département d'informatique et de Recherche Opérationnelle

Montréal 2025
16 janvier 2025

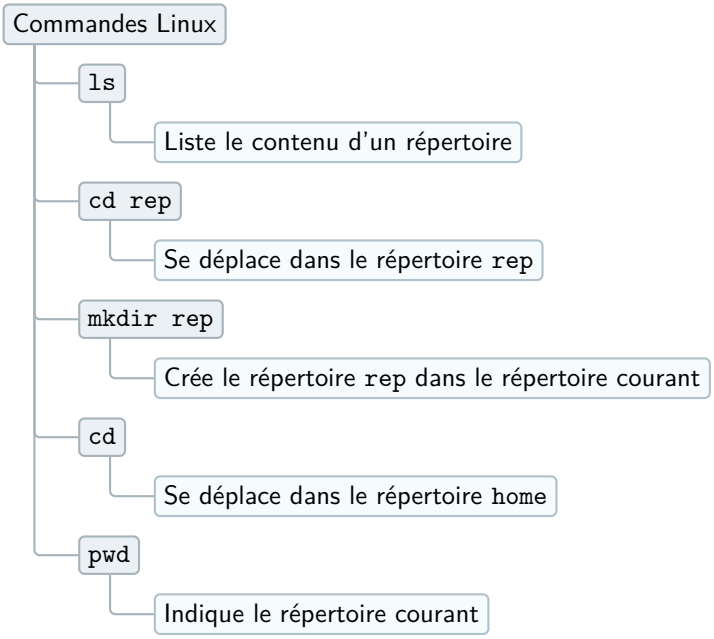
Table des matières

1 Scriptage

1.1 Commande de base

Note

La connaissance de commandes `UNIX` et une compréhension générale du langage `bash` évite souvent d'avoir à programmer des scripts éboré qui peuvent être remplacé par des commandes **simples mais sophistiquées**.



1.1.1 Pipeline et redirection

Concept 1.1 (Pipeline). La pipieline dénotée par la syntaxe `|` permet d'envoyer le `output` d'une première commande pour qu'elle serve de `input` à la commande subséquente.

Exemple 1 Utlisation d'une pipeline pour inverser l'affichage d'un string

```
echo "bonjour" | rev
```

La commande `rev` renverse le texte en `input` **ligne à ligne**.

Note

L'inverse n'est pas possible puisque `rev` lit uniquement sur des entrées standard (`input` d'une pipeline) ou un fichier, et ne prend pas directement d'argument de texte en ligne de commande :

```
rev "bonjour" | echo
```

L'exemple ci-haut **ne respecte pas la syntaxe**. D'ailleurs, `echo` s'attend à un argument qui le succède.

Exemple 2 Redirection vers un fichier

```
echo "bonjour" | rev > fichier.txt
```

La commande `>` permet d'enregistrer un enput dans le fichier `fichier.txt`.

Concept 1.2 (Redirection en `csh`). `>` Redirige la sortie standard vers un fichier. Si le fichier existe déjà, il est écrasé.

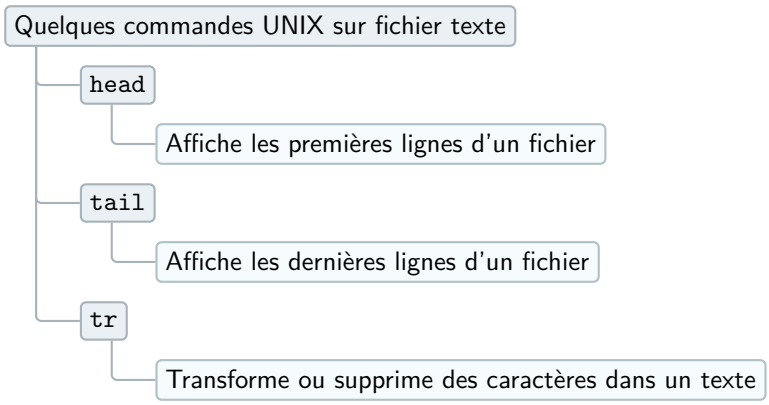
`>!` Force la redirection de la sortie standard vers un fichier. Cela écrase le fichier existant sans avertissement.

`»` Ajoute la sortie standard à la fin d'un fichier existant. Si le fichier n'existe pas, il est créé.

1.1.2 Obtenir de l'aide avec `man`

La commande `man` permet d'obtenir de l'information sur une commande `UNIX`.

1.1.3 Manipulation de fichier texte



Exemple 3 Affiche les trois premières lignes

```
head -n 3 zola1.txt
```

Cette commande affiche les trois premières lignes du fichier `zola1.txt`.

Exemple 4 Renverser les trois premières lignes

```
head -n 3 zola1.txt | rev
```

Cette commande affiche les trois premières lignes du fichier `zola1.txt`, chaque ligne étant renversée.

1.2 La commande `tr`

Définition 1.3 (Commande `tr`). La commande `tr` est utilisée pour transformer ou supprimer des caractères dans une entrée standard.

1.2.1 Utilisation de `tr` et `-s`

Exemple 5 Découpe en mots une ligne avec ponctuation

```
head -n 1 zola1.txt | tr '[[:punct:]]' ' '
                    | tr '[[:space:]]' '\n'
                    | tr -s '[[:space:]]'
```

Cette commande découpe la première ligne de `zola1.txt` en mots en remplaçant les ponctuations par des espaces, divisant chaque mot sur une nouvelle ligne et éliminant les espaces redondants.

`Pourquoi ce choix ?` Cette méthode différencie correctement un mot suivi d'une virgule ou d'un point (par exemple, `Bourse,` devient simplement `Bourse`).

Note

L'option `-s` supprime les **répétition successives** des caractères spécifiés dans l'ensemble donné. Ainsi :

```
echo "aaa   bbb    ccc" | tr -s ' '
```

engendre le `output "aaa bbb ccc"` en supprimant les répititon d'espace (' ')

Exemple 6 Découpe les mots alphabétiques uniquement

```
head -n 1 zola1.txt | tr -sc '[[:alpha:]]' '\n'
```

Cette commande découpe la première ligne de `zola1.txt` en mots, remplaçant tout caractère non alphabétique (y compris ponctuation) par des sauts de ligne.

`Pourquoi mieux ?` Cette méthode est plus concise et efficace. Elle ne nécessite pas de multiples appels à `tr` et évite les mots collés à une ponctuation, comme `Bourse,`, qui devient directement `Bourse`.

1.2.2 Utilisation de `tr` et `-sc`

Note

```
echo "abc123@@@def456" | tr -sc '[:alpha:]' '\n'
```

`-sc` combine deux fonctionnalités: `-c` prend le complément de lensemble spécifié (`[:alpha:]`, soit **tout sauf les lettres alphabétiques**), et `-s` compresse les répétitions des caractères du complément.

Cette commande transforme `"abc123@@@def456"` en:

```
abc
def
```

Les caractères non alphabétiques (`123@@@456`) sont remplacés par des sauts de ligne compressés.

1.3 La commande `grep`

Définition 1.4 (Commande `grep`). La commande `grep` est utilisée pour rechercher des lignes correspondant à un motif dans un fichier ou une entrée standard.

1.3.1 Utilisation de `grep` et `i`

Exemple 7 Recherche insensible à la casse avec `grep -i`

```
grep -i "bonjour" fichier.txt
```

Cette commande recherche toutes les lignes contenant `bonjour` dans `fichier.txt`, sans tenir compte de la casse. Par exemple, elle trouvera `"Bonjour"` ou `"BONJOUR"`.

Exemple 8 Recherche du motif `ven` insensible à la casse

```
head -n 60 zola1.txt | tr '[[:space:]]' '\n' | grep -i ven
```

Cette commande extrait les 60 premières lignes du fichier `zola1.txt`, divise chaque mot sur une nouvelle ligne en remplaçant les espaces par des sauts de ligne, puis recherche toutes les occurrences du motif `ven` sans tenir compte de la casse.

Exemple 9 Recherche avec surlignage du motif `ven`

```
head -n 60 zola1.txt | tr '[[:space:]]' '\n' | grep --colour -i ven
```

Cette commande effectue la même recherche que l'exemple précédent, mais utilise l'option `-colour` pour surligner en couleur toutes les occurrences du motif `ven`, rendant les résultats plus visibles.

1.4 La commande awk

Définition 1.5 (Commande `awk`). La commande `awk` est un langage de traitement de texte utilisé pour analyser et manipuler des fichiers ou des flux de données structurés, ligne par ligne, en fonction de motifs et d'actions spécifiés.

1.5 Utilisation de `awk` `uniq` `grep` et `sort`

Exemple 10 Extraction et tri des mots commençant par ven

```
head -n 200 zola1.txt | tr '[[:space:]]' '\n'
                      | grep -i '^ven'
                      | sort | uniq -c
                      | sort -k1,1nr | awk '{print $2}'
```

Cette commande effectue les étapes suivantes: `head -n 200` extrait les 200 premières lignes de `zola1.txt`.
`tr '[[:space:]]' '\n'` divise chaque mot sur une nouvelle ligne.
`grep -i 'ven'` filtre les mots commençant par ven, insensible à la casse.
`sort` trie les mots par ordre alphabétique.
`uniq -c` compte les occurrences de chaque mot unique.
`sort -k1,1nr` trie les mots par fréquence, en ordre décroissant.
`awk '{print $2}'` affiche uniquement les mots (2 colonne), sans leur fréquence.

1.6 Utilisation de `awk`, `uniq`, `grep`, et `sort`

Exemple 11 Filtrage des mots fréquents commençant par ven

```
cat zola1.txt | tr '[[:punct:]]' ' '
              | tr '[[:space:]]' '\n'
              | grep -i '^ven'
              | sort | uniq -c
              | sort -k1,1nr
              | awk '$1 > 3 {print $0}'
```

Cette commande effectue les étapes suivantes: `cat zola1.txt` lit tout le contenu du fichier `zola1.txt`.
`tr '[[:punct:]]' ' '` remplace toutes les ponctuations par des espaces.
`tr '[[:space:]]' '\n'` divise chaque mot sur une nouvelle ligne.
`grep -i 'ven'` filtre les mots commençant par ven, insensible à la casse.
`sort` trie les mots par ordre alphabétique.
`uniq -c` compte les occurrences de chaque mot unique.
`sort -k1,1nr` trie les mots par fréquence, en ordre décroissant.
`awk '$1 > 3 {print $0}'` affiche uniquement les mots avec plus de 3 occurrences, avec leur fréquence.

1.7 Commande `wc`

Définition 1.6 (Commande `wc`). La commande `wc` (word count) est utilisée pour compter les lignes, les mots ou les caractères dans une entrée standard ou un fichier.

1.7.1 Compter le nombre de mots uniques dans un texte

Exemple 12 Compter les mots uniques

```
cat zola1.txt | tr '[[:punct:]]' ' '
              | tr -s '[[:space:]]'
              | tr '[[:space:]]' '\n'
              | sort | uniq -c | wc -l
```

Cette commande effectue les étapes suivantes: `cat zola1.txt` lit le contenu complet de `zola1.txt`.
`tr '[[:punct:]]' ' '` remplace toutes les ponctuations par des espaces.
`tr -s '[[:space:]]'` compresse les espaces consécutifs en un seul espace.
`tr '[[:space:]]' '\n'` divise chaque mot sur une nouvelle ligne.
`sort` trie les mots par ordre alphabétique.
`uniq -c` compte les occurrences de chaque mot unique.
`wc -l` compte le nombre total de mots uniques.

1.7.2 Trouver les deux mots les plus fréquents

Exemple 13 Les deux mots les plus fréquents

```
cat zola1.txt | tr '[[:punct:]]' ' '
              | tr -s '[[:space:]]'
              | tr '[[:space:]]' '\n'
              | sort | uniq -c
              | sort -k1,1nr | head -n 2
```

Cette commande effectue les étapes suivantes: `cat zola1.txt` lit le contenu complet de `zola1.txt`.
`tr '[[:punct:]]' ' '` remplace toutes les ponctuations par des espaces.
`tr -s '[[:space:]]'` compresse les espaces consécutifs en un seul espace.
`tr '[[:space:]]' '\n'` divise chaque mot sur une nouvelle ligne.

`sort` trie les mots par ordre alphabétique.
`uniq -c` compte les occurrences de chaque mot unique.
`sort -k1,1nr` trie les mots par fréquence, en ordre décroissant.
`head -n 2` affiche les deux mots les plus fréquents avec leur fréquence.

1.8 Environnement

Concept 1.7 (Utilité de `ls -l`, `chmod u+x`, et exécution de scripts)**.** `ls -l` Affiche les permissions, le propriétaire, la taille et d'autres métadonnées des fichiers dans le répertoire courant.

`chmod u+x` Ajoute les droits d'exécution (`x`) à l'utilisateur (`u`) pour un fichier ou un script, permettant son exécution comme un programme.

Pour exécuter un script : `nomscript` Si `./` est dans le `PATH`, exécute directement le script.

`./nomscript` Exécute un script du répertoire courant, même si `./` n'est pas dans le `PATH`.

`csh ./nomscript` Exécute un script en utilisant explicitement le shell `csh`, utile pour les scripts spécifiques à ce shell.

1.8.1 Variables d'environnement

Concept 1.8 (Importance de `env`, `echo $SHELL`, `echo $PATH`, et `env | grep LANG`). `env` Affiche toutes les variables d'environnement du shell actif, utiles pour diagnostiquer ou configurer l'environnement de travail.

`echo $SHELL` Affiche le shell utilisé par défaut (ex. `/bin/bash`, `/bin/zsh`), ce qui peut être utile pour vérifier la configuration du système.

`echo $PATH` Liste les répertoires dans lesquels le shell recherche les exécutables. Permet de vérifier et modifier le chemin d'accès pour exécuter des programmes.

`env | grep LANG` Filtre les variables d'environnement liées à la langue et l'encodage, comme `LANG` ou `LC_ALL`. Utile pour s'assurer que les paramètres régionaux sont correctement définis.