Préambule

Ce cours s'adresse aux Bac2 biotech indifféremment de l'option choisie, bioélectronique/ bioinformatique. Il présentera quelques commandes importantes, performantes issues du monde UNIX/LINUX.

Vous aurez donc besoin d'avoir accès à un "Linux". Même si les bioinformaticiens ont/auront inévitablement en cours d'année ce système sous la main, en début d'année ce ne sera pas le cas.

Il existe une instance de linux "sous" windows : WSL (Windows Servicing Linux".

Son installation sous windows est très aisée et sera réalisée en classe lors de la première leçon.
En résumé 👍

> Ouvrir un powershell administrateur
> Y introduire la commande : wsl install
> répondre aux deux questions posées; c-a-d:
> donner un nom d'utilisateur p.ex martin, uniquement des minuscules
> donner un mot de passe pour cet utilisateur (attention il ne s'affichera pas)

Normalement c'est tout ce qu'il y a à faire.
Ensuite vous lancez le programme "ubuntu" qui vient d'être installé.

Nous règlerons en classe les éventuels problèmes rencontrés, si il en échet.
Il s'agit en général de la non activation de la virtualisation dans le bios de vos machines, mais depuis plusieurs années maintenant cette fonction est activée par défaut. Il est cependant possible que certains d'entre vous dispose d'un ordinateur vieillot.

Si vous utilisez un ordinateur Apple, les versions actuelles utilisent un dérivé de FreeBSD, qui est un cousin proche de Linux. La plupart des commandes que nous découvrirons sont donc nativement présentes. Cependant elles le sont parfois dans des versions moins actuelles, ou personnalisées par Apple. Leur comportement pourra s'avérer être légèrement différent ou moins complet.

La Commande `grep` de Linux

Cette synthèse est conçue pour vous familiariser avec l'utilisation de la commande `grep` (Global Regular Expression Print) sous Linux. `grep` est un outil puissant pour la recherche de texte dans des fichiers et des flux de données. Pour cette session, nous allons explorer ses fonctionnalités à travers de qeulques exemples en utilisant un fichier simple que nous allons créer, une liste de courses, plutôt que de manipuler des fichiers système complexes ou des journaux.

Introduction à `grep`

La commande `grep` est un utilitaire de ligne de commande qui recherche des lignes contenant une correspondance pour un modèle donné et les affiche. Elle est extrêmement utile pour analyser des fichiers, trouver des informations spécifiques dans de grands ensembles de données, ou pour des tâches de scripting.

en un mot grep est un filtre

Syntaxe de base

La syntaxe générale de `grep` est la suivante :

```
grep [OPTIONS] MODÈLE [FICHIER...]
```

- **MODÈLE** : C'est la chaîne de caractères ou l'expression régulière que vous souhaitez rechercher.
- **FICHIER...** : Ce sont les fichiers dans lesquels vous souhaitez effectuer la recherche. Si aucun fichier n'est spécifié, `grep` lira depuis l'entrée standard (stdin).

Options courantes de `grep`

Voici un tableau des options les plus fréquemment utilisées avec `grep` :

Option	Description
`-i`	Ignore la casse (majuscules/minuscules) lors de la recherche.
`-v`	Inverse la correspondance ; affiche les lignes qui NE correspondent PAS au modèle.
`-n`	Affiche le numéro de ligne pour chaque correspondance trouvée.
`-c`	Affiche uniquement le nombre de lignes correspondantes.
`-w`	Recherche des correspondances de mots entiers uniquement.
`-E`	Interprète le modèle comme une expression régulière étendue (équivalent à `egrep`).
`-F`	Interprète le modèle comme une liste de chaînes fixes (non des expressions régulières).
`--color`	Met en évidence la partie correspondante de la ligne.
`-A NUM`	Affiche NUM lignes de contexte APRÈS la ligne correspondante.
`-B NUM`	Affiche NUM lignes de contexte AVANT la ligne correspondante.
`-C NUM`	Affiche NUM lignes de contexte AVANT et APRÈS la ligne correspondante.

Création du fichier exemple : `courses.txt`

Pour nos démonstrations, nous allons utiliser un fichier nommé `courses.txt` qui contient une liste de courses. Créez ce fichier dans votre répertoire de travail en le prenant sur le serveur BIG, en tapant ceci:

wget http://10.1.120.50/courses.txt (si vous avez installé wsl sur votre windows)
curl http://10.1.120.50/courses.txt -o courses.txt (si vous utilisez un Mac)

voila le contenu de ce fichier:

```
Pommes (bio)
Lait (frais)
Pain (complet)
Œufs (douzaine)
Yaourts (nature)
Céréales (chocolat)
Jus d'orange (sans pulpe)
Avocats (mûrs)
Tomates (grappe)
Fromage (chèvre)
Bouteilles d'eau
Vin (rouge)
Bières (locales)
Chips (paprika)
Chocolat (noir 70%)
Glace (vanille)
Citrons (jaunes)
Pâtes (spaghetti)
Riz (basmati)
Huile d'olive
```

Exemples d'utilisation avec `courses.txt`

Recherche simple

Rechercher toutes les occurrences du mot "Lait" :

grep "Lait" courses.txt

- **Résultat attendu** : `Lait (frais)`

Recherche insensible à la casse

Rechercher "pommes" ou "Pommes" :

grep -i "pommes" courses.txt

- **Résultat attendu** :
  `Pommes (bio)`

Afficher les lignes qui ne correspondent pas

Afficher toutes les lignes qui ne contiennent PAS le mot "bio" :

grep -v "bio" courses.txt

- **Résultat attendu** :

```
Lait (frais)
Pain (complet)
Œufs (douzaine)
Yaourts (nature)
Céréales (chocolat)
Jus d'orange (sans pulpe)
Avocats (mûrs)
Tomates (grappe)
Fromage (chèvre)
Bouteilles d'eau
Vin (rouge)
Bières (locales)
Chips (paprika)
Chocolat (noir 70%)
Glace (vanille)
Citrons (jaunes)
Pâtes (spaghetti)
Riz (basmati)
Huile d'olive
```

Afficher le numéro de ligne

Trouver les "Œufs" et afficher les numéros de ligne :

grep -n "Œufs" courses.txt

- **Résultat attendu** : `4:Œufs (douzaine)`

Compter les correspondances

Compter le nombre de lignes contenant "(bio)" ou "(frais)" :

grep -c "(bio)\\|(frais)" courses.txt

- **Résultat attendu** : 2

Recherche de mots entiers

Rechercher le mot exact "Pain" (et non "Pains") :

grep -w "Pain" courses.txt

- **Résultat attendu** : `Pain (complet)`

Afficher le contexte autour des correspondances

Utilisation de `-A`, `-B`, `-C` pour afficher des lignes avant, après ou les deux.

Afficher 1 ligne après (After) "Yaourts" :

grep -A 1 "Yaourts" courses.txt

- **Résultat attendu** :

```
Yaourts (nature)
Céréales (chocolat)
```

Afficher 1 ligne avant (Before) "Glace" :

grep -B 1 "Glace" courses.txt

- **Résultat attendu** :

```
Chocolat (noir 70%)
Glace (vanille)
```

Afficher 1 ligne avant et 1 ligne après (Context) "Tomates" :

grep -C 1 "Tomates" courses.txt

- **Résultat attendu** :

```
Avocats (mûrs)
Tomates (grappe)
Fromage (chèvre)
```

Expressions régulières avec `grep`

`grep` tire sa puissance de l'utilisation des expressions régulières.

Caractères spéciaux courants

- `.` : Correspond à n'importe quel caractère unique (sauf le retour à la ligne).
- `*` : Correspond à zéro ou plusieurs occurrences du caractère ou de l'expression précédente.
- `^` : Correspond au début de la ligne.
- `$` : Correspond à la fin de la ligne.
- `[abc]` : Correspond à un seul caractère parmi 'a', 'b' ou 'c'.
- `[^abc]` : Correspond à un seul caractère qui n'est PAS 'a', 'b' ou 'c'.
- `[a-z]` : Correspond à un seul caractère minuscule de 'a' à 'z'.
- `\b` : Correspond à une limite de mot.
- `\d` : Correspond à un chiffre (équivalent à `[0-9]`). (Nécessite `-E`)
- `+` : Correspond à une ou plusieurs occurrences du caractère ou de l'expression précédente (Nécessite `-E`).
- `?` : Correspond à zéro ou une occurrence du caractère ou de l'expression précédente (Nécessite `-E`).

Exemples d'expressions régulières

- **Commence par "Pâtes"** :

  grep "^Pâtes" courses.txt

  Résultat attendu : `Pâtes (spaghetti)`

- **Se termine par "d'eau"** :

  grep "d'eau$" courses.txt

  Résultat attendu : `Bouteilles d'eau`

- **Contient un mot de 4 lettres** (utilisez `-E` pour les expressions régulières étendues) :

  grep -E "\b[A-Za-z]{4}\b" courses.txt

  Résultat attendu :

  ```
  Lait (frais)
  Pain (complet)
  Jus d'orange (sans pulpe)
  Chocolat (noir 70%)
  ```

- **Contient des parenthèses avec n'importe quoi à l'intérieur** :

  grep -E "\(.+\)" courses.txt

  Résultat attendu : Toutes les lignes avec des parenthèses.

- **Rechercher des articles ayant une description entre parenthèses** :

  grep -E "^\w+\s*\(.+\)$" courses.txt

  Résultat attendu :

  ```
  Pommes (bio)
  Lait (frais)
  Pain (complet)
  Œufs (douzaine)
  Yaourts (nature)
  Céréales (chocolat)
  Jus d'orange (sans pulpe)
  Avocats (mûrs)
  Tomates (grappe)
  Fromage (chèvre)
  Vin (rouge)
  Bières (locales)
  Chips (paprika)
  Chocolat (noir 70%)
  Glace (vanille)
  Citrons (jaunes)
  Pâtes (spaghetti)
  Riz (basmati)
  ```

- **Rechercher des articles qui ne contiennent PAS de parenthèses** :

  grep -vE "\(.+\)" courses.txt

  Résultat attendu :

Bouteilles d'eau Huile d'olive

Combinaison de `grep` avec d'autres commandes

`grep` est souvent utilisé en combinaison avec d'autres commandes Linux via des "pipes" (`|`).

`cat` et `grep`

Pour rechercher des "Pommes" dans notre liste de courses en utilisant `cat` pour lire le fichier :

cat courses.txt | grep "Pommes"

- **Résultat attendu** : `Pommes (bio)`

`grep` et `wc -l`

Pour compter le nombre d'articles "bio" dans la liste :

grep "(bio)" courses.txt | wc -l

- **Résultat attendu** : 1

Il aurait été plus simple d'utiliser l'option de comptage (count) -c

grep -c "(bio)" courses.txt

Exclusion multiple avec `grep -v`

Pour afficher les lignes qui ne contiennent ni "Lait" ni "Œufs" :

grep -v "Lait" courses.txt | grep -v "Œufs"

- **Résultat attendu** :

```
Pommes (bio)
Pain (complet)
Yaourts (nature)
Céréales (chocolat)
Jus d'orange (sans pulpe)
Avocats (mûrs)
Tomates (grappe)
Fromage (chèvre)
Bouteilles d'eau
Vin (rouge)
Bières (locales)
Chips (paprika)
Chocolat (noir 70%)
Glace (vanille)
Citrons (jaunes)
Pâtes (spaghetti)
Riz (basmati)
Huile d'olive
```

Filtrer et compter des mots spécifiques

Pour simuler le filtrage d'un journal et compter les articles "frais" ou "nature" :

grep -E "(frais|nature)" courses.txt | wc -l

- **Résultat attendu** : 2

Ressources supplémentaires

Pour approfondir vos connaissances sur `grep` et les expressions régulières, vous pouvez consulter les pages de manuel :

- `man grep`
- `man regex`

N'hésitez pas à pratiquer avec ce fichier `courses.txt` ou d'autres fichiers que vous créez pour maîtriser cet outil essentiel du monde Linux.

Insistons un instant sur les répétitions d'éléments recherchés ; celles-ci peuvent être spécifiées avec des accolades `{}`.

- `{n}` : Correspond exactement à `n` occurrences du caractère ou de l'expression précédente.
- `{n,}` : Correspond à au moins `n` occurrences du caractère ou de l'expression précédente.
- `{n,m}` : Correspond à entre `n` et `m` occurrences du caractère ou de l'expression précédente.

Celles-ci peuvent intéresser plus prioritairement celles et ceux qui se dirigeront vers la bioinformatique.
Si vous recherchez dans une séquence d'ADN ; un A suivi d'un T, suivi d'un nombre variable allant de 2 à 5 de G ou de C suivi de T et A:

grep -E "at[gc]{2,5}ta"

fera immédiatement le travail, exemple suivant sur un fichier d'ADN "35n", avec numérotation des lignes ou le motif est retrouvé. C'est évidemment imparfait car cette recherche se limite aux lignes de texte, et ne trouvera pas par exemple un motif qui commence en fin d'une ligne et se poursuit en début de la ligne suivante.

```
dcsysd ~ # grep -nE "at[gc]{2,5}ta" 35n
6:atccatcgcgatgggtattatatgaaacataacactcgtacatggtgattgaaaataata
9:taatgccgcatcgcctacgaagagcgtgaaggcaaccactaaaaattcgtccacgaataa
11:taactatactaccattctgcaacatttggcaaacgaccatcctaatatactgagggagaa
28:agttggaaggccaaaattgagcattatacaaacaggccttttgattttgcaatgccgtag
37:aagcattgaacaggtattgaaactggcaaagccgttgcaactgaaattaagagaatggta
42:ttggtataactgctcaactggaaatctgatgctaattggaacatttgctgccctgctgta
71:ttccctgctcattatatcacataagatgtaaattagaggatgcgtaaaatagttcaactg
77:ctatcctacgtcagccagtcagttactagacggcaaattagatgaatttctaatgctgtc
79:aataggcattaataaaactatgtacgaattagaaatgctaaggtctcgagcac
dcsysd ~ #
```

Cette synthèse n'est qu'un bref aperçu, il existe énormément de variations et combinaisons que nous n'avons fait qu'explorer.