Introduction aux systèmes d'exploitation (IS1)

TP 10 – Grep, Find et Expressions régulières

1 Expressions régulières

Dans ce TP, vous vous familiariserez avec la commande `grep`.
Cette commande recherche des chaînes de caractères, qui peuvent être un mot complet ("système"), une suite de lettres ("abcd"), ou une expression régulière. Quand elle a trouvé ce qu'on cherche, elle affiche par défaut la ligne correspondante.
Les *expressions régulières* sont des critères de recherche. On ne cherche pas un mot précis, mais des suites de caractères (un motif) correspondant aux critères demandés. Elles sont construites *comme des opérations arithmétiques*, en utilisant des opérateurs divers pour combiner des expressions.
Elles sont d'un usage fréquent avec `grep` bien sûr, mais aussi avec d'autres commande comme `sed` et `wk`, ainsi qu'avec certains éditeurs comme `vi` et `em cs`.
La liste des symboles utilisables pour `grep` (il y en a d'autres ! voir dans les pages de manuel pour plus de détails) :

`.`	un seul caractère quelconque
`*`	répétition, zéro ou plusieurs fois, du caractère précédent. `b*` représente l'ensemble { a,ab,abb,abbb,...}
+	répétition (au moins une fois) du caractère précédent `b+` représente l'ensemble { ab,abb,abbb,...}
?	l'élément précédent est facultatif `b?c` représente l'ensemble {ac,abc}
`^`	début de ligne `^` représente toutes les lignes qui commencent par 'a'
$	fin de ligne $ représente toutes les lignes qui finissent par 'a'
`[...]`	liste ou intervalle de caractères recherchés [abcd] représente l'ensemble {a,b,c,d} [a-d] représente l'ensemble {a,b,c,d}
`[^ b]`	listes des caractères interdits

Pour plus de détails, voir la page de manuel de `grep`.
tp9f.pdfExemples :

- `grep bc fichier` recherche la chaîne `bc` dans toutes les lignes du fichier. Les lignes où cette chaîne est trouvée sont envoyées sur la sortie standard.
- `grep " " fichier` recherche les lignes de `fichier` qui contiennent au moins un espace.

Exercice 1 – *Filtrage*

Lorsqu'on utilise la commande `ls`, on obtient souvent une longue liste de fichiers dans laquelle on doit retrouver l'information que l'on cherche. Le but de cet exercice est de voir comment on peut éviter de perdre du temps à chercher les lignes qui nous intéressent à l'aide de la commande `grep`.
En reprenant les fichiers de l'exercice 3 du TP8, on veut :

1. La liste des fichiers qui ont été créés en 2006.
2. La liste des fichiers qui n'ont pas été créés en 2005.
3. La liste de fichiers qui commencent par un "p".
4. La liste des fichiers dont l'utilisateur possède les droits de lecture et d'écriture.

Depuis votre répertoire personnel, on veut :

5. Lister tous les fichiers dont l'extension est `.j v .`
6. Lister tous les fichiers qui commencent par une majuscule.
7. Lister tous les répertoires.
8. Refaire les 2 questions précédentes en y incluant les sous-répertoires.

La commande `find` peut être aussi très utile pour la recherche de fichiers.

9. Cherchez tous les fichiers dont le nom commence par un "p" majuscule ou une minuscule dans toute l'arborescence de votre *home* ? (regardez dans la man en particulier l'option `-n me`)
10. Recherchez ensuite les fichiers dont le nom commence par une lettre entre p et z dans l'arborescence d'un sous-dossier de votre *home* ?
11. Limitez votre première recherche aux fichiers modifiés il y a plus de 30 jours ? Il y a 30 jours ? Il y a moins de 30 jours ?
12. Limitez maintenant votre recherche aux fichiers de type répertoire ?
13. Reprenez la première recherche en considérant uniquement les fichiers de taille supérieure à 10ko.
14. Utilisez `find` pour afficher le contenu de tous vos fichiers de sauvegarde (terminés par un ~) qui ont plus d'un mois.

Exercice 2 – *Recherche dans un fichier*

Le but de cet exercice est de rechercher des phrases dans une œuvre complète. Pour cela, récupérez le fichier Cyrano.txt dans le répertoire `/users/monitIS1/tp10/`.

1. Comptez le nombre de lignes du fichier en utilisant la commande `wc`.
2. Comptez le nombre de lignes qui commencent par une majuscule.
3. Comptez maintenant le nombre de lignes du fichier sans utiliser la commande `wc`.
4. Comptez le nombre d'occurrences du mot *nez*.
5. Trouvez et affichez les lignes contenant le mot *nez*.

6. Faites la même chose en affichant en plus le numéro de chaque ligne trouvée.
7. Trouvez les lignes contenant *cap* mais pas *capitaine*.
8. Comptez le nombre de scènes de la pièce. Comptez le nombre d'actes.
9. Affichez la liste des scènes et des actes dans l'ordre.
10. Comptez le nombre de répliques de Cyrano et de Roxane.
11. Quelle est la proportion de répliques de Cyrano par rapport à celles de Roxane ?
12. Vérifiez que la fameuse réplique des Nez apparaît bien dans la pièce. (*C'est un roc ! c'est un pic ! C'est un cap ! Que dis-je, c'est un cap ? C'est une péninsule !*)
13. Trouvez tous les mots de la pièce commençant par "Sc". Connaissez-vous la signification de chacun d'entre eux ?

Exercice 3 – *Coup de pouce aux mots croisés*

Dans tout cet exercice, nous manipulerons le fichier '/usr/share/dict/words' qui est le dictionnaire de base d'Unix. Le but de ce petit exercice est de faire un script qui reproduit le comportement des petites machines que l'on trouve dans le commerce pour faire des mots croisés.

Écrivez un petit script qui demande à l'utilisateur de rentrer un mot incomplet sous la forme d'un mélange de lettre et de "?" et donne à l'utilisateur tous les mots (du fichier words) qu'il peut écrire en remplaçant chaque "?" par une lettre.

Exercice 4 – *Recherche dans des ensembles de fichiers*

Le but de cet exercice est de recherche des motifs dans un ensemble de fichiers. Vous récupérerez le fichier `sources.t r.gz` dans le répertoire `/users/monitIS1/tp10/`. Après avoir extrait les fichiers contenus dans l'archive, vous trouverez un certain nombre de fichiers écrits en JAVA dans le répertoire `sources`. Toutes les questions devront être faites sans utiliser la commande `wc`.

1. Comptez le nombre de fichiers java contenus dans le répertoire `sources`.
2. Comptez le nombre de lignes de code.
3. Comptez le nombre de commentaires.
4. Comptez le nombre de classes définies dans ces fichiers.
5. Dans quel fichier se trouve la classe `IS1` ?
6. Nous voulons trouver la méthode `System.out.println`, mais pas `System.err.println`. Combien de fois apparaît-elle dans tous ces fichiers et dans quelles lignes ?
7. Quelles sont les méthodes publiques de la classe `Printf.j v` ? Dans quelles classes sont-elles utilisées ? Avec quels arguments ?
8. Donnez la liste de tous les tableaux définis dans ces classes (on demande les noms des variables, pas les noms des méthodes qui renvoient un tableau).
9. Donnez la liste des méthodes qui renvoient un entier.

Exercice 5 – *Remplacement*

Le but de cet exercice est de remplacer certains motifs dans un fichier par d'autres. On demande de faire un script qui prend en argument une expression régulière, une chaîne de caractères (qui va remplacer les motifs trouvés) et un fichier. La sortie de ce script sera un fichier dont le nom sera composé du nom original auquel on rajoutera l'extension `.out`.

1. Faites une copie répertoire bin créé lors du TP9 et remplacez, dans tous vos scripts du TP9, la chaîne `/usr/loc l/bin/b sh` par `/bin/b sh`.
2. *(Difficile)* À partir du fichier `SliceText.j v` , créez un fichier ne contenant que la documentation du programme (c'est-à-dire l'ensemble des commentaires).
3. En reprenant les mêmes fichiers, enlevez tous les commentaires.
4. *(Difficile : à faire si vous avez fait tout le reste)* Écrivez un script qui extrait du texte Cyrano de Bergerac l'ensemble des répliques de Cyrano.