ntroduction aux systèmes d'exploitation (S1)

T 8 – C rep, ind, Sed et xpressions régulières

1 Expressions régulières et recherche de motifs

ans ce T , vous vous familiarisere avec les commandes `grep`, `find` et `sed`.
a commande `grep`, par laquelle nous allons commencer, recherche des chaînes de caractères dans du texte, par exemple la chaîne "système" ou bien encore la chaîne "abcd". orsque la chaîne de caractères en question est trouvée, par défaut, `grep` affiche la ligne correspondante.
xemples :

- `grep abc fichier` recherche la chaîne `abc` dans toutes les lignes du fichier. es 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.

n fait, `grep` ne permet pas seulement de rechercher des chaînes de caractères fixées, mais aussi des chaînes de caractères remplissant certaines conditions. Ces conditions sont exprimées par des *expressions régulières*. ne expression régulière est construite comme une expression arithmétique : on forme une expression complexe en combinant des expressions simples à l'aide d'opérateurs.
oici quelques opérateurs utilisables dans les expressions régulières fournies à `grep` (il y en a d'autres ! voir dans les pages de manuel pour plus de détails) :

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

xemples :

- 'expression régulière $a*b+c?[^c]$ sélectionne les chaînes de caractères `bbbca` et `aaba` mais pas la chaîne `bcc`.
- 'expression régulière $a.*b$ sélectionne exactement les chaînes de caractères qui commencent par un `a` et finissent par un `b`.

Exercice 1 – *Filtrage*

orsqu'on utilise la commande `ls`, on obtient souvent une longue liste de fichiers dans laquelle on doit retrouver l'information que l'on cherche. e but de cet exercice est de voir comment on peut ne sélectionner que les lignes qui nous intéressent à l'aide de la commande `grep`.
n reprenant les fichiers de l'exercice 1 du T , on veut :

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

epuis votre répertoire personnel, on veut :

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

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

9. Cherche tous les fichiers dont le nom commence par une minuscule ou un " " majuscule dans toute l'arborescence de votre *home* ? (renseigne vous, dans le manuel, sur l'option `-name`)
10. echerche ensuite les fichiers dont le nom commence par une lettre entre p et dans l'arborescence d'un sous-dossier de votre *home* ?
11. imite votre première recherche aux fichiers modifiés il y a plus de 30 jours ? l y a 30 jours ? l y a moins de 30 jours ?
12. imite maintenant votre recherche aux fichiers de type répertoire ?
13. eprene la première recherche en considérant uniquement les fichiers de taille supérieure à 10ko.
14. tilise `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*

e but de cet exercice est de rechercher des phrases dans une œuvre complète. our cela, récupére le fichier Cyrano.txt dans le répertoire `tp10` du compte `monitIS1`.

1. Compte le nombre de lignes du fichier en utilisant la commande `wc`.
2. Compte le nombre de lignes qui commencent par une majuscule.
3. Compte maintenant le nombre de lignes du fichier sans utiliser la commande `wc`.
4. Compte le nombre d'occurrences du mot *nez*.
5. Trouve et affiche les lignes contenant le mot *nez*.
6. aites la même chose en affichant en plus le numéro de chaque ligne trouvée.
7. Trouve les lignes contenant *cap* mais pas *capitaine*.
8. Compte le nombre de scènes de la pièce. Compte le nombre d'actes.
9. Affiche la liste des scènes et des actes dans l'ordre.
10. Compte le nombre de répliques de Cyrano et de oxane.
11. Quelle est la proportion de répliques de Cyrano par rapport à celles de oxane ?
12. érifie que la fameuse réplique des e 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. Trouve tous les mots de la pièce commençant par "Sc". Connaisse -vous la signification de chacun d'entre eux ?

Exercice 3 – *Recherche dans des ensembles de fichiers*

e but de cet exercice est de rechercher des motifs dans un ensemble de fichiers. ous récupérere le fichier `sources.tar.gz` dans le répertoire `tp10`. Après avoir extrait les fichiers contenus dans l'archive, vous trouvere un certain nombre de fichiers écrits en A A dans le répertoire `sources`. Toutes les questions devront être faites sans utiliser la commande `wc`.

1. Compte le nombre de fichiers java contenus dans le répertoire `sources`.

2. Compte le nombre de lignes de code.

3. Compte le nombre de commentaires.

4. Compte le nombre de classes définies dans ces fichiers.

5. ans quel fichier se trouve la classe `IS1` ?

6. ous 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.java` ? ans quelles classes sont-elles utilisées ? Avec quels arguments ?

8. onne 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. onne la liste des méthodes qui renvoient un entier.

2 Remplacement de motifs

Exercice 4 – *Edition automatique d'un fichier*

e but de cet exercice est d'utiliser la commande `sed` pour automatiser certains traitements sur un fichier texte, comme ajouter/supprimer du texte et des lignes selon certains critères.

`sed -e commande`$_1$ `-e commande`$_2$ `...-e commande`$_k$ `fic`

effectue les k commandes spécifiées, dans l'ordre, pour chacune des lignes du fichier `fic`. ne commande s'écrit

`adresse(s) fonction arguments`

où `adresse(s)` est une liste de 0, 1 ou 2 adresses et `fonction` est le nom d'une fonctionnalité offerte par `sed`. ne commande n'est appliquée à la ligne en cours de traitement que si l'adresse de cette ligne concorde avec le champ `adresse(s)` de la commande. Consulte le manuel pour découvrir les types d'adresses utilisées par `sed` ainsi que les fonctionnalités qu'il offre. orsqu'on utilise une séquence d'expressions régulières dans une commande, elles doivent être délimitées par le caractère *slash* : `/expr`$_1$`/.../expr`$_k$`/`.

es questions qui suivent doivent être traitées indépendamment sur le fichier `Cyrano.txt` que l'on conservera intact, la sortie étant écrite dans le fichier `Cyrano.txt.mod`. Chaque question doit être résolue en une unique commande bash.

1. Change , dans tout le fichier, les C A O en Cyrano.

2. emplace , dans tout le fichier, les 'a' par des 'A'. Cela a-t-il vraiment été fait partout ? Comment pallier à cela ?

3. Change les 'e' en ' ' dans le deuxième acte uniquement.

4. ouble les mots commençant par 's' et finissant par 'e', en prenant soin d'insérer un espace entre l'original et sa copie.

5. emplace les mots commençant par 's' et finissant par 'e' par la séquence de caractères *s&e*.

6. Change les "le" en "la" et réciproquement. Quelle est la difficulté et comment la surmonter ?

7. Supprime toutes les lignes vides du fichier.

8. e conserve que les lignes contenant la séquence de lettres *amour*.

9. Triple toutes les lignes contenant le mot oxane.

10. éécrive le fichier en ne laissant qu'un mot par ligne.