8 - Encodages, Grep

Laurent Tichit

5 avril 2011

Plan

Encodages et types de fichiers

Le dictionnaire

Les exemples porteront sur un fichier de données contenant un dictionnaire du Français (336531 lignes avec un mot par ligne).

```
à
abaissa
abaissable
abaissables
abaissai
abaissaient
abaissais
abaissait
abaissâmes
abaissant
abaissante
abaissantes
abaissants
abaissas
abaissasse
```

Les encodages

- L'anglais étant une langue sans caratères accentués, tous ses caractères sont encodés sur 7 bits (128 caractères différents) en utilisant la *table de codage ASCII*, associant un entier sur 7 bits à chaque symbole.
- Les autres langues doivent étendre l'ASCII. Il existe 2 systèmes principaux :
  - Les encodages sur 8 bits : Windows-CP-1511 et ISO-8859-5 (ou ISO-LATIN-5) pour le bulgare ; Windows CP-1512, ISO-8859-1 (ou ISO-LATIN-1) pour le français.
    Ce sont des tables d'encodage étendant l'ASCII.
  - L'unicode (UTF-32 sur 32 bits, UTF-16 sur 16/32 bits, UTF-8 sur 8/16/32 bits) permettant d'encoder tous les symboles du monde.
- Pour connaître les encodages supportés par votre système :

```
tichit@iml230:~$ locale
LANG=fr_FR.UTF-8
LC_CTYPE="fr_FR.UTF-8"
LC_NUMERIC="fr_FR.UTF-8"
LC_TIME="fr_FR.UTF-8"
LC_COLLATE="fr_FR.UTF-8"
```

Les encodages

- Selon l'encodage utilisé par votre système, vous utiliserez soit *dico-fr-L1.txt*, soit le fichier *dico-fr-UTF8.txt*
- La commande **file** permet souvent de connaître l'encodage utilisé par un fichier texte. Cependant, elle n'est pas infaillible.
- pour convertir un fichier d'un encodage vers un autre :
  iconv -f L1 -t UTF-8 fichier-en-latin-1 > fichier-en-UTF8
- **iconv -l** vous donne la liste des encodages connus.

Exercice

- Testez la commande **file** sur plusieurs fichiers à vous (fichiers C, java, images, texte, répertoire, etc.).
- Testez la commande **file -i** sur différents fichiers texte (code source ou texte) pour détecter leur encodage.

Expressions régulières

Les expressions régulières

- Une expression régulière est une manière compacte de définir un modèle de chaîne de caractères.
- Exemple : la recherche dans un texte donné d'un modèle commençant par un "T", suivi d'un certain nombre de lettres et finalement de la lettre "h" sera représentée par : `T.*h`
- Les expressions régulières sont disponibles dans de nombreux outils, comme `grep`, `more`, `less`, `find` (les types d'expressions régulières acceptés varient légèrement suivant les programmes).

Les expressions régulières

Dans une expression régulière, certains symboles ont une signification spéciale :

- `.` Tout caractère excepté newline (fin de ligne)
- `*` Facteur multiplicatif par zéro ou n
- `^` Début de ligne
- `$` Fin de ligne
- `[aet]` Ensemble de caractères (ici *a*, *e*, *t*)
- `[a-m]` Intervalle de caractères (ici entre *a* et *m*)
- `[^art]` Ensemble de caractères exclus
- `\` Le caractère suivant est considéré littéralement

La commande grep

La commande **grep modèle fichier.txt** retourne toutes les lignes du texte correspondant au modèle.
Il est recommandé de toujours écrire une expression régulière entre quotes (').

1. Quelles sont les lignes qui commencent par les lettres 'shu' ?

La commande grep

La commande **grep modèle fichier.txt** retourne toutes les lignes du texte correspondant au modèle.
Il est recommandé de toujours écrire une expression régulière entre quotes (').

1. Quelles sont les lignes qui commencent par les lettres 'shu' ?

   ```
   grep '^shu' dict
   ```

2. Quels sont les lignes qui finissent par les lettres 'ley' ?

La commande grep

La commande **grep modèle fichier.txt** retourne toutes les lignes du texte correspondant au modèle.
Il est recommandé de toujours écrire une expression régulière entre quotes (').

1. Quelles sont les lignes qui commencent par les lettres 'shu' ?

   ```
   grep '^shu' dict
   ```

2. Quels sont les lignes qui finissent par les lettres 'ley' ?

   ```
   grep 'ley$' dict
   ```

3. Quelles sont les lignes qui contiennent un 'a' entre deux traits d'union ?

La commande grep

La commande **grep modèle fichier.txt** retourne toutes les lignes du texte correspondant au modèle.
Il est recommandé de toujours écrire une expression régulière entre quotes (').

1. Quelles sont les lignes qui commencent par les lettres 'shu' ?

   ```
   grep '^shu' dict
   ```

2. Quels sont les lignes qui finissent par les lettres 'ley' ?

   ```
   grep 'ley$' dict
   ```

3. Quelles sont les lignes qui contiennent un 'a' entre deux traits d'union ?

   ```
   grep '.*-a-.*' dict
   ```

4. Quelles sont les lignes qui contiennent la chaine 'aer' suivie de deux caractères quelconques suivi d'un 'y' ?

La commande grep

La commande **grep modèle fichier.txt** retourne toutes les lignes du texte correspondant au modèle.
Il est recommandé de toujours écrire une expression régulière entre quotes (').

1. Quelles sont les lignes qui commencent par les lettres 'shu' ?

   ```
   grep '^shu' dict
   ```

2. Quels sont les lignes qui finissent par les lettres 'ley' ?

   ```
   grep 'ley$' dict
   ```

3. Quelles sont les lignes qui contiennent un 'a' entre deux traits d'union ?

   ```
   grep '.*-a-.*' dict
   ```

4. Quelles sont les lignes qui contiennent la chaine 'aer' suivie de deux caractères quelconques suivi d'un 'y' ?

   ```
   grep 'aer..y' dict
   ```

La commande grep

5 Trouver tous les mots qui contiennent l'une des trois lettres 'g', 'j' ou 'p' suivie des deux caractères 'er' puis de deux caractères quelconques puis d'un 'y'

La commande grep

5 Trouver tous les mots qui contiennent l'une des trois lettres 'g', 'j' ou 'p' suivie des deux caractères 'er' puis de deux caractères quelconques puis d'un 'y'

```
grep '[gjp]er..y' dict
```

[0-9]permet de définir l'ensemble des chiffres. Dans la définition d'un intervalle il est indispensable de respecter l'ordre de votre encodage (ASCII, latin-1, etc.).

6 Trouver tous les mots qui commencent par une lettre de l'intervalle 'a-d' ('a', 'b', 'c' ou 'd') puis qui contienne les deux lettres 'hr' puis une lettre pris dans l'intervalle 'e-i' ou dans l'intervalle 'w-z' ('e', 'f', 'g', 'h', 'i', 'w', 'x', 'y' ou 'z')

La commande grep

5 Trouver tous les mots qui contiennent l'une des trois lettres 'g', 'j' ou 'p' suivie des deux caractères 'er' puis de deux caractères quelconques puis d'un 'y'

```
grep '[gjp]er..y' dict
```

[0-9]permet de définir l'ensemble des chiffres. Dans la définition d'un intervalle il est indispensable de respecter l'ordre de votre encodage (ASCII, latin-1, etc.).

6 Trouver tous les mots qui commencent par une lettre de l'intervalle 'a-d' ('a', 'b', 'c' ou 'd') puis qui contienne les deux lettres 'hr' puis une lettre pris dans l'intervalle 'e-i' ou dans l'intervalle 'w-z' ('e', 'f', 'g', 'h', 'i', 'w', 'x', 'y' ou 'z')

```
grep '^[a-d]hr[e-ix-z]' dict
```

La commande grep

7 Trouver tous les mots qui commencent par la lettre 'k' suivie de au moins deux voyelles.

La commande grep

7 Trouver tous les mots qui commencent par la lettre 'k' suivie de au moins deux voyelles.

```
grep '^k[aeiou][aeiou][aeiou]*' dict
```

Il faut préfixer un symbole par un '\' si on souhaite que ce symbole soit interprété littéralement :

8 Trouver toutes les lignes qui contiennent le symbole '$'.

La commande grep

7 Trouver tous les mots qui commencent par la lettre 'k' suivie de au moins deux voyelles.

```
grep '^k[aeiou][aeiou][aeiou]*' dict
```

Il faut préfixer un symbole par un '\' si on souhaite que ce symbole soit interprété littéralement :

8 Trouver toutes les lignes qui contiennent le symbole '$'.

```
grep '\$' data
```

9 Trouver toutes les lignes commençant par '*' et se terminant par '\$' et comportant au moins un caractère entre les deux.

La commande grep

7 Trouver tous les mots qui commencent par la lettre 'k' suivie de au moins deux voyelles.

```
grep '^k[aeiou][aeiou][aeiou]*' dict
```

Il faut préfixer un symbole par un '\' si on souhaite que ce symbole soit interprété littéralement :

8 Trouver toutes les lignes qui contiennent le symbole '$'.

```
grep '\$' data
```

9 Trouver toutes les lignes commençant par '*' et se terminant par '\$' et comportant au moins un caractère entre les deux.

```
grep '^\\\*..*\\\$' Fichier
```

Explication :

- \\ l'antislash
- * l'étoile
- . tout caractère
- .* tout caractère (répété entre 0 et n fois)
- \\ l'antislash,
- \$ le dollar

La commande grep

10 Quels sont les mots qui commencent par 'quo' et se terminent par 't' ?

La commande grep

10 Quels sont les mots qui commencent par 'quo' et se terminent par 't' ?

```
grep '^quo[a-z]*t$' dict
```

ou mieux :

```
grep '^quo.*t$' dict
```

11 Résoudre la vieille devinette qui consiste à trouver tous les mots contenant les voyelles 'a', 'e', 'i', 'o', 'u', dans cet ordre, les lettres n'étant pas forcément adjacentes.

La commande grep

10 Quels sont les mots qui commencent par 'quo' et se terminent par 't' ?

```
grep '^quo[a-z]*t$' dict
```

ou mieux :

```
grep '^quo.*t$' dict
```

11 Résoudre la vieille devinette qui consiste à trouver tous les mots contenant les voyelles 'a', 'e', 'i', 'o', 'u', dans cet ordre, les lettres n'étant pas forcément adjacentes.

```
grep '[a-z]*a[a-z]*e[a-z]*i[a-z]*o[a-z]*u' dict
```

ou mieux

```
grep '.*a.*e.*i.*o.*u' dict
```

12 L'option `-v` de `grep` permet d'indiquer le complémentaire. Afficher toutes les lignes non vides d'un texte

La commande grep

10 Quels sont les mots qui commencent par 'quo' et se terminent par 't' ?

```
grep '^quo[a-z]*t$' dict
```

ou mieux :

```
grep '^quo.*t$' dict
```

11 Résoudre la vieille devinette qui consiste à trouver tous les mots contenant les voyelles 'a', 'e', 'i', 'o', 'u', dans cet ordre, les lettres n'étant pas forcément adjacentes.

```
grep '[a-z]*a[a-z]*e[a-z]*i[a-z]*o[a-z]*u' dict
```

ou mieux

```
grep '.*a.*e.*i.*o.*u' dict
```

12 L'option `-v` de `grep` permet d'indiquer le complémentaire. Afficher toutes les lignes non vides d'un texte

```
grep -v '^$'
```

Commandes **tr**, **sort**, **uniq** et **grep**

tr "[A-Z]" "[a-z]" < fichier | tr -cs "[a-z]" "[\n]" | grep -v '^.\$' | sort | uniq

- **tr "[A-Z]" "[a-z]" < fichier1** Rediriger le fichier *fichier1* comme entrée de la commande **tr** et substituer à toute lettre majuscule la lettre minuscule correspondante.
- **tr -cs "[a-z]" "[\n]"** Remplacer tout caractère qui ne fait pas partie de l'ensemble des lettres minuscules (-**c**) par un retour à la ligne. Si plusieurs substitutions successives sont à opérer, n'en effectuer qu'une (-**s**).
- **grep** On verra cette commande en détail plus tard. Ici, elle ne conserve que les lignes contenant pluis d'un caractère.
- **sort** Trier le fichier par ordre alphabétique.
- **uniq** Supprimer les lignes consécutives équivalentes