TP 5 - Un peu de lecture (2)

Exercice 1 (Apprendre la lecture en chinois)**.** Écrivez un script `classification_hsk.sh` prenant en paramètre un fichier texte contenant du chinois et permettant d'évaluer sa difficulté pour un étudiant étranger, à l'aide du classement 汉语水平考试 (abrégé en HSK pour Hanyu Shuiping Kaoshi), qui contient 6 niveaux et sert à évaluer le niveau des étrangers qui apprennent le chinois. Par exemple, sur le texte disponible ici:

```
$ wget https://www.gutenberg.org/files/24225/24225-0.txt
```

on obtiendra le résultat suivant:

```
$ ./classification_hsk.sh 24225-0.txt
Le fichier 24225-0.txt contient 1463 caractères uniques, dont:

        * 132 caractères de niveau HSK 1
        * 108 caractères de niveau HSK 2
        * 124 caractères de niveau HSK 3
        * 185 caractères de niveau HSK 4
        * 179 caractères de niveau HSK 5
        * 172 caractères de niveau HSK 6

Il reste 563 caractères non classés.
```

Vous aurez besoin:

1. de la commande suivante, qui extrait d'un fichier texte uniquement les caractères chinois (sauf la ponctuation), et en affiche un par ligne:

   ```
   $ pcregrep -ou "\p{Han}" fichier.txt
   ```

   Le `-o` signifie la même chose que pour `grep`; le `-u` précise que l'on travaille avec des caractères Unicode (UTF-8), indispensables pour traiter le chinois.

2. des fichiers HSK, disponibles ici:

   ```
   $ wget http://igm.univ-mlv.fr/~alabarre/teaching/shnu/linux/hsk.zip
   ```

   L'archive contient six fichiers: le fichier `hsk_i_sorted_2` contient tous les caractères du niveau `i` (triés, un par ligne, pas de doublon, et sans les caractères des niveaux précédents).

Exercice 2 (青出于蓝，而胜于蓝)**.** Le chinois présente d'autres difficultés que les nombreux caractères à connaître, par exemple les 成语. Nous allons écrire un script qui recense les 成语 apparaissant dans un texte, en utilisant la première colonne de la liste suivante[1]:

[1] Merci à 黄子杰 pour cette ressource.

`https://raw.githubusercontent.com/thunlp/THUOCL/master/data/THUOCL_chengyu.txt`

1. Écrivez un script `chengyu1.sh` prenant en paramètre un fichier texte et affichant le nombre de 成语 qu'il contient, en cherchant chaque 成语 de notre liste dans le texte donné. N'oubliez pas de supprimer les retours à la ligne dans le texte puisqu'ils peuvent couper des 成语 et empêcher leur détection. Triez aussi la liste des 成语 avant votre recherche, car nous voudrons comparer le résultat avec ce que nous écrirons dans la question suivante. Le résultat attendu est le suivant (attention, le résultat peut prendre quelques dizaines de secondes à s'afficher):

```
$ ./chengyu1.sh 24225-0.txt
J'ai détecté les 17 成语 différents suivants:

        * 下回分解
        * 不共戴天
        * 不明不白
        * 十有八九
        * 可望而不可即
        * 名山大川
        * 弄假成真
        * 急流勇退
        * 恍然大悟
        * 手舞足蹈
        * 明明白白
        * 易如反掌
        * 有福同享
        * 望梅止渴
        * 毛骨悚然
        * 神情恍惚
        * 英雄本色
```

   Il vous sera utile de pouvoir rajouter des éléments à la fin d'un tableau, comme ceci: avec l'initialisation `tableau=(1 2 3)`, l'instruction `tableau+=(4)` rajoutera `4` en dernière position (attention aux parenthèses!).

2. La lenteur de notre script est due au fait qu'il nécessite autant de recherches qu'il y a de 成语. Il serait plus efficace d'extraire toutes les séquences qui pourraient correspondre à un 成语, puis de comparer toutes ces séquences à notre liste avec `comm`. Procédons par étapes:

   (a) Écrivez une commande permettant de stocker dans un tableau les longueurs triées et sans doublon des 成语 de la liste donnée.
   (résultat attendu: `(4 5 6 7 8 9)`)

   (b) Pour chacune des longueurs récupérées, générez la liste des séquences de longueur `n` du fichier texte de départ, et stockez le résultat trié dans un fichier séparé. Par exemple, si la chaîne est 一二三四五 et `n` vaut 3, alors on veut obtenir 一二三, 二三四, 三四五.

   Attention: nous aurons besoin pour cela du *lookahead*, une notion avancée des expressions régulières. Illustrons-la sur les caractères ASCII: par exemple, si `fichier` contient uniquement la chaîne `"abcde"`, alors l'expression ci-dessous fonctionne:

```
$ pcregrep -o1 -o2 '(\w)(?=(\w{2}))' fichier
abc
bcd
cde
```

Le premier groupe `(\w)` contient une seule lettre, et le second groupe `(\w{2})` contient toutes les autres. L'expression demande à `pcregrep` de trouver les groupes d'une lettre qui sont suivis d'un groupe de deux lettres; la syntaxe `(?=(\w{2}))` permet de ne pas "consommer" les caractères du groupe 2. Sans cela, on écrirait `(\w)(\w{2})`, et seul `'abc'` serait dans le résultat: en effet, l'extraction de `'bc'` forcerait `pcregrep` à continuer sa recherche à `'d'` au lieu de `'b'`, et à partir de `'d'`, il n'est plus possible de trouver une séquence de 3 lettres.

(c) Enfin, écrivez un script `chengyu2.sh` réalisant la même chose que `chengyu1.sh`, mais en comparant cette fois les tables de sous-séquences extraites avec le fichier de départ. Comparez les temps d'exécution des deux scripts avec `time`.