TP noté

Remarque importante. Le sujet est très long. Il comprend 4 exercices, mais *il n'y a pas besoin de faire tous les exercices pour avoir 20.* Regardez tout le sujet puis travaillez sur les questions que vous savez faire le mieux dans chaque exercice.

Documents papier et en ligne autorisés. L'accès à toute autre ressource, ainsi que la communication entre vous ou avec des tiers, sont interdits.

Exercice 1. Vous trouverez à télécharger sur Arche un fichier de données sur les pays du monde (`countries.csv`)[1]. Les champs sont séparés par des virgules.

(a) Écrivez un script awk qui affiche la population totale de chaque continent en 2010. Attention à la première ligne du fichier.

(b) Écrivez un script awk qui vérifie que chaque ligne a exactement 6 champs et que les 3 derniers sont numériques (utilisez la fonction `isnum` de awk). Si c'est le cas, votre script ne doit rien afficher ; autrement, il doit afficher un message et appeler `exit(1)`.

(c) Écrivez un script awk qui affiche les mêmes données que `countries.csv` mais avec une colonne supplémentaire : la différence de population entre 2020 et 2000. Il doit également y avoir une ligne supplémentaire au début contenant seulement «`# BEGIN #`» et une ligne supplémentaire à la fin contenant seulement «`# END #`».

(d) *Question bonus.* Écrivez un script awk qui vérifie que chaque ligne de l'entrée a le même nombre de champs que la première ligne. Il doit également vérifier que chaque champ numérique sur la première ligne est aussi numérique sur toutes les autres lignes. Enfin, il doit ignorer les lignes qui commencent par un #.

Exercice 2. Dans cet exercice nous vous demandons d'écrire des fonctions dans un script bash `exo2.sh`.

Conseil : pour tester une fonction, ajoutez une ligne `mafonction "$@"` à la fin du script et remplacez `mafonction` par le nom de la fonction à tester. Vous pourrez alors exécuter `exo2.sh` avec les arguments voulus depuis le terminal pour exécuter la fonction.

Vous trouverez sur Arche une archive de fichiers HTML (`recettes.zip`) à télécharger ; chaque fichier contient une recette de cuisine[2].

(a) Écrivez une fonction `unIngredient` dans `exo2.sh` qui prend des ingrédients en argument et affiche chaque ligne de chaque fichier HTML qui contient au moins l'un des ingrédients. Ainsi, «`unIngredient carotte poireau`» affiche toutes les lignes qui contiennent «carotte» ou «poireau» dans tout fichier HTML du dossier courant.

(b) Écrivez une fonction `fichIngredient` dans `exo2.sh` qui prend des ingrédients en argument et affiche une ligne «`fichier:ingrédient`» pour chaque fichier qui contient l'ingrédient. Par exemple, «`fichIngredient carotte pomme`» affiche :

```
BarbunyaPilaki.html:carotte
Bibimbap.html:carotte
BoeufBourguignon.html:carotte
...
BeignetsDePommes.html:pomme
BoeufBourguignon.html:pomme
...
```

1. Source : banque mondiale, licence Creative Commons By.
2. Source : M. Damien Desfontaines, licence Creative Commons Zero.

Une même ligne peut apparaître plusieurs fois si un ingrédient est mentionné plusieurs fois dans un même fichier.

Indication : voyez l'option `-H` de `grep` ; vous avez aussi le droit d'utiliser `awk`.

(c) Écrivez un script `fichIngredient1` dans `exo2.sh` qui fait comme `fichIngredient`, sauf que chaque ligne n'apparaît qu'une seule fois en sortie.

Indication : vous pouvez utiliser les commandes `sort` et `uniq`.

(d) Écrivez une fonction `numIngredient` dans `exo2.sh` qui prend en argument un nombre n et des ingrédients, qui affiche les noms des fichiers qui contiennent au moins n ingrédients parmi ceux demandés. Ainsi, «`numIngredient 2 poireau carotte`» affiche :

```
HachisParmentierVegetarien.html
PastaEFagioli.html
PotAuFeu.html
SaumonSafrane.html
SoupeVerteDePaques.html
```

Vous avez toujours le droit d'appeler `awk`.

(e) Écrivez une fonction `tousIngredients` dans `exo2.sh` qui prend en argument des ingrédients (pas de nombre), et qui affiche les noms de fichiers qui contiennent *tous* les ingrédients demandés.

Exercice 3. Dans cet exercice, on travaille encore sur les fichiers de l'archive `recettes.zip`.

(a) Écrivez un script `exo3-a.sh` qui prend en entrée un fichier HTML et qui affiche le même fichier, avec les modifications suivantes :

- Toutes les lignes avant `<div id="contenu">` (incluse) doivent être supprimées.
- La ligne `\documentclass{article}\begin{document}` doit être ajoutée au début.
- Toutes les lignes après `</div>` (incluse) doivent être supprimées ; on parle du `</div>` qui se trouve après `<div id="contenu">`.
- La ligne `\end{document}` doit être ajoutée à la fin.

Votre script a le droit d'appeler `awk`.

(b) Écrivez un script `exo3-b.sh` qui appelle *uniquement* `sed` (éventuellement plusieurs fois) et qui effectue les transformations suivantes sur son entrée.

- Les `<ul>` et `</ul>` sont remplacés par `\begin{itemize}` et `\end{itemize}`.
- Les `<ol>` et `</ol>` sont remplacés par `\begin{enumerate}` et `\end{enumerate}`.
- Les lignes `<li>blablabla</li>` doivent être remplacées par `\item blablabla`, où `blablabla` peut représenter n'importe quoi.
- Les lignes `<p>blablabla</p>` doivent être remplacées par `blablabla` tout court, où `blablabla` peut représenter n'importe quoi.
- Les `<a href="blabla">bliblibli</a>` doivent être remplacés par `bliblibli`, où `blablabla` et `bliblibli` peuvent représenter n'importe quoi.
- Les `<img blablabla>` doivent être supprimés, où `blablabla` peut représenter n'importe quoi.

(c) Écrivez un `makefile` qui compile tout fichier `.html` en un fichier `.tex` en le passant à `exo3-a.sh` puis `exo3-b.sh`. (Voir ci-après pour un exemple de fichier LaTeX.) Puis il compile chaque fichier `.tex` en un fichier `.pdf` grâce à la commande `pdflatex`. Ajoutez une règle `clean` qui supprime tous les fichiers `.aux` et `.log`.

Exercice 4. Écrivez un script bash `exo4.sh` qui prend en entrée un fichier dont certaines lignes contiennent «`BEGIN`» et «`END`», et qui vérifie que les `BEGIN` et `END` s'enchaînent correctement. Par exemple, `BEGIN` puis `END` puis `END` est incorrect car le deuxième `END` ne «referme» pas un `BEGIN`.

Annexe : exemple de fichier LaTeX

```
\documentclass{article}\begin{document}
  \begin{itemize}
  \item Un point dans une liste
  \item Un autre point
  \item Encore un autre point
  \end{itemize}
  \begin{enumerate}
  \item Liste numérotée, cette fois
  \item Deuxième point
  \item Troisième point
  \end{enumerate}
\end{document}
```