Documents autorisés
Accès à internet autorisé sauf communication

Outils système – L2 info.

TP noté

(A. Scheuer, D-E. Amir, J-P. Jacquot, J. Etienne, Y. Coudert Osmont)

Jeudi 31 mars, 14 h – 16 h

Note importante : Vous déposerez un fichier par exercice, en respectant la consigne donnée dans chaque exercice.

Exercice 1 — Publipostage de notes (5 points)

Vour trouverez sur `arche` un fichier de tableur `notes.csv` au format `CSV` dont chaque ligne correspond à un étudiant et comporte quatre colonnes séparées par des point-virgules (« ; »), donnant son prénom, son nom, son adresse électronique et sa note.

On souhaite envoyer à chaque étudiant un courriel personnalisé pour lui donner sa note.

Le message à envoyer, contenu dans le fichier `gabarit.txt`, est le suivant :

```
1  Bonjour <PRENOM> <NOM>,
2
3  vous avez obtenu la note de <NOTE> a l'UE d'outils systemes. Je vous
4 rappelle que vous  avez la possibilite de consulter votre copie pour
5 comprendre cette note.
6
7  En vous souhaitant bonne reception,
8
9        l'equipe d'outils systeme.
```

Vous devez remplacer <PRENOM>, <NOM> et <NOTE> par leur valeur respective.

Dans une application réelle, on utiliserait la commande Unix `mail` qui permet d'envoyer un message en ligne de commande. Ici, vous utiliserez la commande `mock_mail` qui simule l'envoie d'un courriel. Celle-ci s'utilise comme :

`mock_mail -s "note outils systeme" adresse`

le texte du message étant ensuite donné sur l'entrée standard.

Écrivez la commande (ou le script) permettant d'envoyer ces messages. Le fichier à déposer doit s'appeler `publipostage.sh`. Si vous avez besoin de fichiers supplémentaires, ceux-ci s'appelleront `publi_help.1`, `publi_help.2`, etc.

.../...

Exercice 2 — Makefile (5 points)

Créez un fichier `Makefile` qui définira les règles suivante :

`Filtre` parcourt le dossier `DOS`, qui ne contient que des fichiers, situé dans le même répertoire que le fichier `makefile`, crée les dossiers `TEX`, `PDF`, `C` et `JAVA` dans le dossier `DOS`, puis déplace les fichiers d'extension `.tex`, `.pdf`, `.c` et `.java` du dossier `DOS` dans les dossiers correspondants et enfin supprime les autres.

`Codes` crée un répertoire `Codes` dans le même répertoire que le fichier `makefile` et déplace les fichiers d'extensions `.tex`, `.c` et `.java` du dossier `DOS` dedans.

`clean` supprime les fichiers du dossier `DOS` qui n'ont pas pour extension `.tex`, `.pdf`, `.c` ou `.java`.

La règle `Codes` doit être la règle par défaut (appelée lorsqu'aucun argument n'est donné à `make`). Votre fichier `Makefile` doit contenir une règle `.PHONY`.

Testez votre fichier `Makefile` avec le dossier `DOS` fourni sur `arche`.

Le fichier à déposer doit s'appeler `Makefile`.

Exercice 3 — Diagramme de classe (5 points)

Pour ce TP, il vous faudra récupérer le fichier `Boggle.zip` sur arche.

Martine, une étudiante consciencieuse, est sur le point de rendre son TP de Boggle. Avant de valider son dépôt, elle veux vérifier qu'elle n'a rien oublié dans son diagramme de classe. Afin de vérifier que son diagramme est correct, elle veut afficher pour chacun des fichiers l'endroit où il est utilisé.

Par exemple la classe du fichier `Observateur.java` (la classe correspond forcément au nom du fichier sans extension), est référencé dans : `VueInfos.java`, `VueLettres.java` et `VueMots.java`

Pour chacun des fichiers java de l'application, il est demandé de lister les noms de fichiers où la classe est utilisée. L'affichage du résultat est attendu sous la forme :

```
1 Main:
2 Boggle: Main VueInfos VueLettres VueMots
3 Dictionnaire:
4 Observateur: Boggle VueInfos VueLettres VueMots
5 VueInfos: Main
6 VueLettres: Main
7 VueMots: Main
```

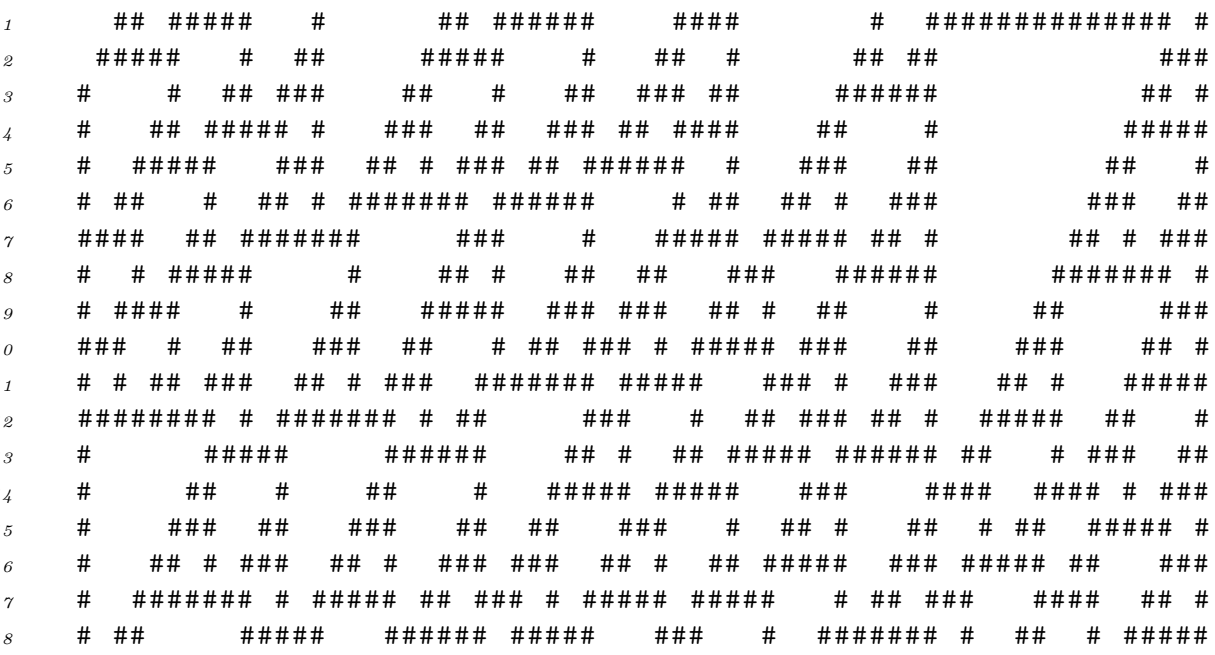
Écrivez un script bash qui parcourt les fichiers/dossiers de `Boggle` et affiche toutes les dépendances des fichiers java trouvés.

Note :
- la commande `find ./*` permet de lister tous les fichiers et dossiers présents.
- l'option `-n` de la commande `echo` empêche le retour à la ligne
- regarder les options (General Output Control) de la commande `grep` peut vous faire gagner du temps

Le fichier à déposer doit s'appeler `diagramme.sh`.

Exercice 4 — Automate Cellulaire (5 points)

Dans cet exercice, on s'intéresse à l'implantation d'un automate cellulaire avec la commande `sed`. Un automate cellulaire est une grille dont chaque cellule contient un état et qui évolue au cours du temps. Ici on s'intéressera à un automate défini sur une grille 1D avec deux états 0 et 1. Un automate peut donc être représenté par une chaîne de caractères contenant uniquement des 0 et des 1 (ex : `"00111101101"`). On représentera l'évolution au cours du temps en affichant une ligne par étape, comme ceci :



Où chaque 0 est remplacé par un espace `' '` et chaque 1 est remplacé par `#`.

Règles Les règles de l'automate décrivent son évolution dans le temps. On peut représenter ces règles par un tableau :

Pattern initial	111	110	101	100	011	010	001	000
Nouvel état central	0	1	1	0	1	1	1	0

La nouvelle valeur d'une cellule dépend de ses deux voisins. Par exemple si un 1 est adjacent à deux 1 il devient un 0 mais si son voisin de droite est un 0 alors sa valeur reste à 1. Il faut considérer que les cellules en dehors de l'automate valent 0 (en rouge dans l'exemple suivant). Voici un exemple d'un automate sur la première ligne suivit de son évolution après application des règles.

0001111011010
01100111111

.../...

Consignes Sur Arche vous trouverez un script Bash `automate.sh` dont la fonction `simulation` est à compléter. Elle doit simuler une étape d'application des règles précédentes en modifiant la variable `s` qui est une chaîne remplie de 0 et 1 représentant l'automate. Pour se faire vous utiliserez des commandes `sed`. On rappelle que la commande peut être utilisée pour effectuer des substitutions comme ceci :

```
s=$(echo "$s" | sed -E 's/expression_reguliere/nouvelle_chaine/g)'
```

Cette commande remplace toutes les occurrences de `expression_reguliere` par `nouvelle_chaine`. L'exécution du script doit afficher dans le terminal l'évolution temporelle donnée plus haut.

Indices

- On peut n'appliquer que les règles qui modifient l'état central du pattern. Il y en a 3 : `111 -> 0`, `101 -> 1` et `001 -> 1`.
- Pour se rappeler de l'ancienne valeur des états durant les substitutions il est possible de définir deux nouveaux caractères `a` et `b` représentant successivement les états qui sont passé de 0 à 1 et 1 à 0. Sur le précédent exemple cela donnerait :

  0001111011010
  0a1bb1a11a1

  Dans ce cas, n'oubliez pas de remplacer dans vos expressions régulières certains `0` par des `([0a])` et une expression similaire pour certains `1`. On rappelle que vous pouvez y faire référence dans `nouvelle_chaine` avec `\n` où `n` signifie que l'on fait référence à la `n`-ème parenthèse. Par exemple :

  ```
  echo "xy" | sed -E 's/(.)(.)/\2\1/'
  ```

  affiche `yx`. Vous pourrez alors remplacer les `a` et les `b` par des `1` et des `0` à la fin de la fonction `simulation`.
- N'oubliez pas le cas particulier des extrémités de l'automates.
- Faites plusieurs appels à `sed`. N'essayez pas de tout faire en une seule commande !

Le fichier à déposer est le fichier `automate.sh` complété.