TP2
script Shell

Préambule

- la première ligne d'un script doit être : #!/bin/bash
- il faut donner les droits d'exécution au fichier script (chmod) ;
- en phase de mise au point, on peut *tracer* ce que fait le shell en exécutant le script avec la commande : bash -x *script arguments*

Le but de cet exercice est d'écrire un script capable de supprimer des fichiers .class d'une arborescence de répertoires si certaine conditions sont remplies.

On fera l'hypothèse que si le *chemin*[1] d'un fichier class comprend un répertoire nommé bin alors le chemin du fichier source comprend un répertoire nommé src ; plus précisément, si le chemin d'un fichier class est de la forme ..../**bin**/xxx/yyy/f.class alors le chemin vers son fichier source est de la forme ..../**src**/xxx/yyy/f.java ; xxx et yyy sont facultatifs ; s'il sont présents, ces noms de répertoires correspondent à des paquetages.

Version initiale

La première étape consiste à transformer le chemin d'un fichier .class pour obtenir celui de son fichier source.

On supposera que le chemin à transformer est dans une variable :

- comment utiliser sed pour faire la transformation ?
- comment éviter de transformer Image**bin**aire en Image**src**aire ?
  - dans la documentation de bash (man bash), lisez la rubrique qui traite de *substitution de commande* ; (essayez */command substitution*) ; testez l'effet de cet opérateur ainsi : fichiers=$(ls) ou fichiers=`ls` (ls entre accents graves) puis affichez la variable fichiers.

Écrire le script detruire_v0.sh Il réalise les traitements suivant :

- prend en argument un chemin vers un fichier .class ;
- fabrique le chemin vers le fichier source correspondant ;
- vérifie la présence du fichier source ;
  - si absent, affiche un message d'erreur et termine le script ;
- si présent, vérifie si le fichier source contient la fonction main ;
  - si c'est le cas, supprime le fichier .class.

Attention : tester si une ligne contient la chaîne « main » ne suffit pas à affirmer que le fichier contient la *fonction* main.

Pour tester, appelez le script avec différents arguments :

- chemin vers le fichier .class conforme à l'hypothèse avec fichier source existant avec fonction main ;
- chemin vers le fichier .class conforme à l'hypothèse avec fichier source existant sans fonction main ;
- chemin vers le fichier .class conforme à l'hypothèse avec fichier source absent.

Ajouter des vérifications

script detruire_v1.sh : Même fonctionnement que la version précédente, mais le script doit faire les vérifications suivantes et émettre un message puis se terminer en cas d'erreur :

- nombre d'arguments correct (1) ;
- existence du fichier passé en argument ;
- le nom doit se terminer par .class (voir l'opérateur ==) ;
- le nom doit contenir un *répertoire* nommé bin (voir la commande grep) ;
- existence du fichier source ;
- présence de main dans le fichier source ;
- droits suffisants pour supprimer le fichier passé en argument (la commande dirname pourrait être utile).

Remarques :

- utilisez des noms de variable explicites ;
- pour tester les différents cas d'erreur, n'hésitez pas à créer des fichiers (même vides !), par exemple avec cp, cat, echo ou touch.

1. pathname in english

Traiter une liste de fichiers

On veut avoir la possibilité d'appeler le script avec un nombre de fichiers quelconque afin d'effectuer le traitement ci-dessus sur chacun d'eux.

script detruire_v2.sh : Commencez par mettre tout le code du script précédent dans une fonction.

La syntaxe de définition d'une fonction est :

```
# paramètres :
# première valeur à traiter
# deuxième valeur à traiter
# ...
function maFonction()
{
  # récupérer les arguments
  var1=$1
  var2=$2
#...
}
```

- Une fonction peut contenir toute commande ou instruction valide dans un script.
- Pour sortir d'une fonction avant la fin sans sortir du programme, on peut utiliser l'instruction return.
- La syntaxe d'appel d'une fonction est :

```
maFonction $variable1 $variable2
```

Remarque : une fonction doit etre définie avant toute instruction d'appel.

- Programmez une itération qui appelle cette fonction avec chaque argument du script en paramètre.
- Testez
  - avec un seul argument
  - avec plusieurs arguments
  - avec une liste vide

  en n'oubliant pas les différents cas du script précédent
- Trouvez comment appeler votre script avec la liste de tous les fichiers .class d'une arborescence, par exemple celle d'un tp de programmation du premier semestre.

Lecture sur l'entrée standard

script detruire_v3.sh : ajoutez au script précédent la fonctionnalité suivante : si le nombre d'arguments du script est nul, lire les noms de fichier sur l'entrée standard.

Vous pouvez vous aider de ces deux outils :

- read v : lit une valeur sur l'entrée standard et la place dans la variable v ; en cas d'échec de la lecture, par exemple s'il n'y a aucune donnée sur l'entrée standard, read renvoie un code d'erreur ;
- itération while

La syntaxe de l'itération while est :

```
while
  commande_1
  ...
  commande_n
do
  # corps de l'itération
done
```

Principe de fonctionnement :

- exécuter les commandes entre while et do ;
- si la dernière commande renvoie un code d'erreur, quitter l'itération ;
- si la dernière commande se termine sans erreur, exécuter les commandes du corps de l'itération (entre do et done) puis revenir au début.

- Pour tester, vous pouvez fournir les noms de fichier interactivement lors de l'exécution du script appelé sans argument ; pour terminer la saisie, il faudra taper Ctrl-D à la place d'un nom de fichier.
- Trouvez comment fournir sur l'entrée standard de votre script la liste de tous les fichiers .class d'une arborescence.