# Projet Shell : Gestion de versions d'un fichier
# Système et programmation système

Julien BERNARD
Université de Franche-Comté

Licence 2 Informatique
2014 – 2015

Le but de ce projet est de réaliser un utilitaire de gestion de versions appelé `version.sh`. Comme son nom l'indique, `version.sh` est implémenté en langage shell. Son rôle est de gérer les différentes versions d'un fichier qu'on va pouvoir sauvegarder au fur et à mesure. L'utilitaire s'utilise toujours de la même manière :

```
$ version.sh <commande> <fichier> [options]
```

`commande` représente une commande de l'utilitaire, `fichier` est le fichier concerné, tandis qu'`options` représente les options éventuelles de la commande. Nous proposons d'implémenter les commandes suivantes : `add`, `rm`, `commit`, `revert`, `diff`, `log` et `checkout`. Le détail de ces commandes sera expliqué par la suite.

## Spécification

Pour les fichiers versionnés, c'est-à-dire gérés par le gestionnaire de versions, les différentes versions seront sauvegardées dans un répertoire caché nommé `.version`. La structure de ce répertoire est imposée par le projet.

Dans le répertoire `.version`, pour chaque fichier versionné, on trouvera :

— La première version du fichier, avec l'extension `.1` ;
— La dernière version du fichier, avec l'extension `.latest` ;
— Un ensemble de patchs entre les versions successives, avec l'extension `.N` où `N` est la `N`$^{e}$ version du fichier.

Par exemple, si on a déjà 3 versions successives du fichier `example.txt`, le répertoire `.version` contient `example.txt.1` qui est une copie de la première version du fichier ; `example.txt.2` et `example.txt.3` qui sont des patchs entre les versions 1 et 2, et 2 et 3 respectivement ; `example.txt.latest` qui est une copie de la dernière version du fichier. Donc, `example.txt.1` et `example.txt.latest` sont des copies du fichier à un instant donné, tandis que les autres fichiers sont des patchs.

Le répertoire `.version` doit toujours être cohérent par rapport à cette spécification. Il est nécessaire de faire toutes les vérifications utiles pour ne pas le rendre incohérent. Pour chaque appel à l'utilitaire `version.sh`, quand cela a un sens, il faudra notamment :

— vérifier la présence du répertoire `.version` ;
— vérifier qu'un fichier est sous contrôle du gestionnaire de version.

Il est très important de mettre un maximum de messages d'information à destination de l'utilisateur pour que celui-ci puisse vérifier le bon fonctionnement ou les erreurs survenues. Les messages seront **en anglais**. Des exemples sont donnés dans les questions qui suivent, mais tous les messages ne sont pas donnés.

Le script devra pouvoir être appelé depuis n'importe quel répertoire, pour n'importe quel fichier dans le système de fichiers. Le répertoire `.version` doit toujours se trouver dans le même répertoire que le fichier versionné. Vous pourrez pour cela vous aider des commandes `dirname(1)` et `basename(1)`.

# Réalisation

## Exercice 1 : `diff(1)` et `patch(1)`

Si vous ne l'avez pas déjà fait, faites l'exercice sur `diff(1)` et `patch(1)`.

## Exercice 2 : Gestion globale des commandes

**Question 2.1** Vérifier qu'il y a bien deux arguments (la commande et le nom du fichier) et sinon, afficher un message d'aide.

```
$ ./version.sh
Usage: version.sh <cmd> <file> [option]
where <cmd> can be: add checkout commit diff log revert rm
```

**Question 2.2** Vérifier que le fichier existe (indice : `test(1)`).

```
$ ./version.sh add unknown.txt
Error! 'unknown.txt' is not a file.
```

**Question 2.3** Gérer les différentes commandes grâce à une structure de contrôle `case ... esac` (indice : `dash(1)`). Si la commande n'existe pas, afficher un message d'erreur.

```
$ ./version.sh unknown example.txt
Error! This command name does not exist: 'unknown'
```

## Exercice 3 : Commandes `add` et `rm`

**Question 3.1** Implémenter la commande `add` qui met un fichier sous contrôle du gestionnaire de version, et sauvegarde la version 1, qui est également la dernière version dans ce cas.

```
$ ./version.sh add example.txt
Added a new file under versioning: 'example.txt'
```

**Question 3.2** Implémenter la commande `rm` qui supprime toutes les versions d'un fichier sous contrôle. Demander une confirmation de la part de l'utilisateur. Supprimer le répertoire `.version` s'il est vide.

```
$ ./version.sh rm example.txt
Are you sure you want to delete 'example.txt' from versioning? (yes/no) yes
'example.txt' is not under versioning anymore.
```

## Exercice 4 : Commande `commit`

**Question 4.1** Implémenter la commande `commit` qui ajoute une nouvelle version d'un fichier sous contrôle. Indiquer le numéro de la version qui vient d'être «commitée». Si le fichier courant est identique à la dernière version, on ne commitera rien (indice : `cmp(1)`).

```
$ ./version.sh commit example.txt
Committed a new version: 2
```

## Exercice 5 : Commandes `revert` et `diff`

**Question 5.1** Implémenter la commande `revert` qui annule tous les changements en cours et revient à la dernière version commitée.

```
$ ./version.sh revert example.txt
Reverted to the latest version
```

**Question 5.2** Implémenter la commande `diff` qui affiche la différence entre la version en cours du fichier et la dernière version commitée. Le format de la différence est le format unifié.

```
$ ./version.sh diff example.txt
--- ./.version/example.txt.latest       2012-02-29 18:56:02.000000000 +0100
+++ example.txt 2012-02-29 19:12:36.000000000 +0100
@@ -1 +1 @@
-foo
+bar
```

## Exercice 6 : Commande `checkout`

**Question 6.1** Implémenter la commande `checkout` qui récupère la version indiquée en paramètre. Vérifier qu'il y a bien un troisième paramètre. Vérifier que la version existe. Partir de la première version et appliquer les patchs successivement (indice : `seq(1)`).

```
$ ./version.sh checkout example.txt 2
Checked out version: 2
```

## Exercice 7 : Synonyme de commande

**Question 7.1** Implémenter les alias de commande suivants :

- `ci` pour `commit`
- `co` pour `checkout`

## Exercice 8 : Ajout d'un log et commande `log`

Nous allons maintenant ajouter un log. Pour chaque commit, on associe un commentaire d'une seule ligne qui sera sauvegardé dans un fichier `.log` dans le répertoire `.version`. Par exemple, le log pour le fichier `example.txt` sera sauvegardé dans `example.txt.log`. Le commentaire sera préfixé par la date et l'heure du jour au format RFC-2822 (voir `date(1)`).

**Question 8.1** Modifier la commande `commit` pour ajouter un paramètre, un commentaire pour ce log.

```
$ ./version.sh commit example.txt "Add an example"
Committed a new version: 2
```

**Question 8.2** Implémenter la commande `log` qui affiche le log des versions déjà commitées. (Indice : `nl(1)`)

```
$ ./version.sh log example.txt
1: Mon, 09 Mar 2015 17:48:21 +0100 First commit
2: Mon, 09 Mar 2015 17:52:04 +0100 Add an example
3: Mon, 09 Mar 2015 17:56:33 +0100 Add another example
```

# Travail à rendre

Vous devrez rendre un seul fichier, `version.sh`, avant le :

**vendredi 25 mars 2015 à 18h00**.

Vous commenterez intelligemment votre fichier de manière à expliquer ce que vous faites. Vous indiquerez les commandes que vous avez implémentées et, pour les autres, le script affichera : `Not implemented`.

Vous indiquerez votre nom dans un commentaire.