# Unix Rappels TD1
# scripts shells

Rappels sur les variables, les structures de contrôle, les redirections, les pipes, les paramètres positionnels, etc

### Un premier script

Ecrivez le script shell « `salut.sh` » qui produit le résultat :

```
> ./salut.sh
Bonjour à tous
```

### Améliorations

Ecrivez un script « horloge1`.sh` » qui produit le résultat :

```
> ./horloge1.sh
Nous sommes le 14 Septembre 2012, il est : 9h30.
```

Modifiez le script précédent pour que l'heure corresponde au premier argument du script :

```
> ./horloge2.sh 9h30
Nous sommes le 14 Septembre 2012, il est : 9h30.
```

Modifiez le script précédent pour que l'heure corresponde au contenu d'un fichier `horaire`:

```
> cat horaire
```

9h30

```
> ./horloge3.sh
Nous sommes le 14 Septembre 2012, il est : 9h30.
```

Modifiez le script précédent pour que le nom du fichier contenant l'heure corresponde à l'argument 1

### Tests

A quoi correspond `$0` ? Modifiez le script `horloge2.sh` pour qu'il affiche un message d'erreur dans le cas ou l'argument n'est pas spécifié :

```
> ./horloge2.sh
usage: horloge2.sh <heure>
```

Modifiez le script `horloge 3.sh` pour qu'il vérifie l'existence du fichier passé en paramètre et affiche un message d'erreur si le fichier n'existe pas :

```
> ./horloge3.sh vide
fichier « vide » non trouvé
```

### Expressions et boucles

Modifiez le script `horloge3.sh` pour qu'il affiche les dates et horaires pour divers fuseaux horaires (Time Zone), chaque zone étant stockée dans un fichier `<Capitale>.TZ` (ex : Paris.TZ, Londres.TZ, NewYork.TZ, etc.)

```
> ./horloge3.sh
Nous sommes le 14 Septembre 2012, il est : 9h30 à Paris.
Nous sommes le 14 Septembre 2012, il est : 8h30 à Londres.
Nous sommes le 14 Septembre 2012, il est : 3h30 à New York
```

Ecrivez un script `moyenne.sh` qui analyse un fichier « `notes` » dont chaque ligne est composée d'un nom, d'un prénom d'élève et d'une série de notes (de longueur arbitraire, mais non nulle), et liste la moyenne de chaque élève (ainsi que son identité).

**Utilitaire déplacements de fichiers :**
Ecrire un shell script permettant de supprimer, dans un répertoire donné, tous les liens
- correspondant à des fichiers ordinaires ;
- dont le propriétaire est le propriétaire de la session et possède le droit d'écriture.

Le script admettra comme argument le nom du répertoire à traiter. Par défault ce répertoire sera /tmp. On prévoira par ailleurs une option –i assurant une suppression interactive.Utilitaire adduser
Ecrire un shell script permettant de rajouter un utilisateur au système.
(comme vous n'avez pas les droits root, vous ferez une copie du fichier /etc/passwd dans votre répertoire puis vous créerez un répertoire dans votre arborescence pour recevoir les répertoires home des nouveaux utilisateurs).
La fonction suivante, d'affichage de l'usage du script, vous donnera le format souhaité :

```
Usage()
{
echo "Usage : "
echo –n "$0 login [-g groupe] [-r répertoire]
echo "[-s shell] [-i info] [-u uid] "
exit
}
```

Ce script devra :
- vérifier que le nombre d'arguments est correct (au moins un : login),
- récupérer les arguments,
- vérifier que le login :
- commence bien par une lettre : [a-z]*
- ne contient pas de caractère spéciaux : *[a-zA-Z0-9]*
- ne compte pas plus de 8 caractères : ? ? ? ? ? ? ? ?
- n'est pas déjà utilisé (grep),
- que le groupe existe bien dans /etc/group,
- que le répertoire parent existe bien,
- que le champ info ne contient pas le caractère ' :' : *[ ! :]*

L'extraction, du nom du propriétaire du fichier, peut se faire au moyen de la commande sed ou bien plus naturellement avec la commande cut :

```
Var=`ls –l $i | tr –s ' ' | cut –d ' ' –f 3 `
```

**Utilitaire adduser :**
Ecrire un shell script permettant de rajouter un utilisateur au système.
(comme vous n'avez pas les droits root, vous ferez une copie du fichier /etc/passwd dans votre répertoire puis vous créerez un répertoire dans votre arborescence pour recevoir les répertoires home des nouveaux utilisateurs).
La fonction suivante, d'affichage de l'usage du script, vous donnera le format souhaité :

```
Usage()
{
echo "Usage : "
echo –n "$0 login [-g groupe] [-r répertoire]
echo "[-s shell] [-i info] [-u uid] "
exit
}
```

Ce script devra :
- vérifier que le nombre d'arguments est correct (au moins un : login),
- récupérer les arguments,
- vérifier que le login :

- commence bien par une lettre : [a-z]*
- ne contient pas de caractère spéciaux : *[a-zA-Z0-9]*
- ne compte pas plus de 8 caractères : ? ? ? ? ? ? ? ?
- n’est pas déjà utilisé (grep),
- que le groupe existe bien dans /etc/group,
- que le répertoire parent existe bien,
- que le champ info ne contient pas le caractère ‘ :’ : *[ ! :]*