

**Exercice 1.** Écrire une commande qui affiche un à un ses arguments sous la forme :

```
$ essai toto bonjour titi
L'argument 1 est toto
L'argument 2 est bonjour
L'argument 3 est titi
```

**Exercice 2.** Écrire une commande **oui-non** qui demande à l'utilisateur de saisir **oui** ou **non** jusqu'à ce que celui-ci ait effectué une saisie correcte.

**Exercice 3.** Écrire une commande **qui** qui affiche une et une seule fois le login de tous les utilisateurs connectés sur la machine à un instant donné. (Idée : voir **who** et **sort**).

**Exercice 4.** Écrire une commande **qui-ou** qui affiche le login de tous les utilisateurs connectés avec la provenance de la connection (une et une seule fois chaque couple nom/provenance). L'affichage final doit présenter les données sous la forme **login**, deux caractères espace et **provenance**.

**Exercice 5.** Lorsque vous ouvrez un fichier que vous avez sauvegardé sous dos, vous voyez apparaître un symbole  $\text{^M}$  à la fin des lignes (essayez avec **vi**). Ceci est dû au fait que les fins de lignes sont marquées par **retour-chariot/nouvelle-ligne** alors que sous unix les fins de lignes sont marquées par **nouvelle-ligne**. Écrire une commande **dos2unix <source> <destination>** qui transforme un fichier texte **source** enregistré au format dos en un fichier texte **destination** enregistré au format unix. Si le nombre de paramètres est incorrect, afficher **dos2unix <source> <destination>**. **Remarque :** le caractère **retour-chariot** peut être obtenu par **contrôle-q** **contrôle-m** et en **bash** par **contrôle-v** **contrôle-m** Attention, si les noms des fichiers **source** et **destination** sont identiques, vous devez utiliser un fichier temporaire en prenant garde de ne pas écraser un fichier déjà existant.

**Exercice 6.** Exercice inverse du précédent. Écrire la commande **unix2dos** avec les mêmes contraintes que dans l'exercice précédent. Chercher dans **man 7 regex** comment on désigne la fin de ligne par une expression régulière.

**Exercice 7.** Écrire une commande qui permet de renommer une liste de fichiers du répertoire courant. Le premier argument est un préfixe et les arguments les noms des fichiers à renommer. Les nouveaux noms des fichiers doivent être créés à partir du préfixe et d'un numéro d'ordre. Par exemple :

```
$ renomme old toto.txt essai.c fichier.java
```

renommara respectivement les fichiers **toto.txt**, **essai.c** et **fichier.java** en **old1**, **old2** et **old3**. La commande ne doit renommer que les fichiers réguliers, si elle n'a qu'un seul argument, elle doit traiter tous les fichiers réguliers du répertoire. En cas de conflit de nom (le nom choisit existe déjà), on ajoute des occurrences du préfixe jusqu'à trouver un nom qui n'existe pas encore.

**Exercice 8.** Modifier la commande `jeter` que vous avez déjà écrite afin d'ajouter quelques améliorations :

- ne pas écraser les fichiers déjà dans la poubelle, si on veut jeter un fichier de nom `fichier` et qu'un fichier de même nom se trouve déjà dans la poubelle, il faut renommer le nouveau fichier à `jeter en fichier#1`, ou `fichier#2` si le fichier `fichier#1` est déjà présent dans la poubelle et ainsi de suite ;
- transformer l'option `-r` en `-v` (`v` pour vider) ;
- ajouter une option `-s` (`s` pour sortir), `jeter -s fichier chemin` sort le fichier `fichier` de la poubelle et le place à l'emplacement indiqué par `chemin` ;
- ajouter une option `-r` (`r` pour restaure), `jeter -r fichier` restaure le fichier `fichier` contenu dans la poubelle à son *emplacement* initial avec son *nom* initial. Par exemple :

```
[~ryl/essai/tp] jeter -l
toto.txt
[~ryl/essai/tp] jeter toto.txt
[~ryl/essai/tp] jeter -l
toto.txt toto.txt#1
[~ryl/essai/tp] cd ../test
[~ryl/essai/test] jeter -r toto.txt#1
[~ryl/essai/test] cd ../tp
[~ryl/essai/test] ls
toto.txt
```

Si lors de la restauration, il existe déjà un fichier de même nom à l'emplacement visé, demandez confirmation avant de l'écraser.

**Exercice 9.** Écrire une commande `verifier-chemin` qui prend un chemin en argument et qui vérifie si le chemin est valide. La commande doit afficher `OK` si le chemin est valide et le chemin jusqu'au premier répertoire non valide si celui-ci n'est pas valide. Par exemple `verifier-chemin /home/etu/toto/tp_systeme` affichera `verifier-chemin /home/etu/toto` si le répertoire `toto` n'existe pas (`/home` et `/home/etu` existent).

**Exercice 10. Enquête.** Les machines ne doivent pas être arrêtées violemment, c'est en particulier désagréable pour les utilisateurs connectés au moment de l'arrêt. La commande `enquete` va permettre d'analyser un tel crime. Utilisez la commande `last` pour prendre connaissance des dernières connections.

- `enquete -crimes` affiche les dates (mois, jour et heure) des derniers événements `reboot`.
- `enquete -victimes` affiche le nom des victimes des crimes, c'est-à-dire le nom des personnes qui étaient connectées avant l'un des crimes et qui ont vu leur connection violemment coupée.
- `enquete -crime` affiche la ligne correspondant au dernier crime.
- `enquete -temoins` affiche la liste des témoins potentiels du dernier crime. Est témoin potentiel toute personne qui c'est connectée sur la machine dans les 30 minutes qui suivent le crime. Il faut évidemment que la personne se soit connectée « physiquement » sur la machine et non pas à distance.