Initiation à l'environnement Unix : TP6

novembre 2023 — Pierre Rousselin

1 Environnement

Exercice 1 :

1. Visualiser l'environnement du shell courant à l'aide de la commande `env`.
2. Créer le script suivant nommé `print_FOO` :

   ```
   #!/bin/sh
   printf 'FOO=%s\n' "$FOO"
   ```

   Le rendre exécutable, essayer de prévoir ce qui sera affiché lors de l'exécution et vérifier en l'exécutant.
3. Pour chacune des commandes ou suites de commandes suivantes, essayer de prévoir ce qui sera affiché, puis vérifier en les exécutant.
   a) `FOO=bar; echo $FOO; ./print_FOO`
   b) `FOO=bar; . print_FOO` remarque : la commande `.` (un point) demande au *shell courant*, et non à un nouveau processus, d'exécuter les commandes contenues dans un fichier.
   c) `export FOO=baz; ./print_FOO`
   d) `env -u FOO ./print_FOO`
   e) `FOO=bar ./print_FOO`

--- * ---

Correction de l'exercice 1 :

1.
2. `FOO` n'est pas définie, donc `$FOO` se développe en la chaîne vide.
3. Cette fois, `FOO=bar` est affiché car on est dans le shell courant, où `FOO` est bien définie.
4. le premier `bar` est affiché (car `FOO` contient `bar` dans le shell courant) mais le processus `./print_FOO` n'hérite pas de la variable normale `FOO` donc celle-ci est toujours non définie.
5. Cette fois, la variable est dans l'environnement, le processus créé par la commande `./print_FOO` en hérite et `FOO=baz` est affiché.
6. `env -u FOO` enlève `FOO` de l'environnement du processus enfant donc `FOO` y est non définie.
7. On met `FOO=bar` dans l'environnement du processus enfant. Donc `FOO=bar` est affiché.

--- * ---

Exercice 2 : Comportement influencé par l'environnement

1. Voir la liste des localisations disponibles sur la machine à l'aide de la commande `locale -a` et la localisation courante avec `locale`.
2. Écrire un script shell `hello_world` qui affiche
   a) `bonjour le monde !` si la variable d'environnement `LANG` contient une chaîne commençant par `fr_FR`;
   b) `hello, world!` si `LANG` contient `C` ou `C.UTF-8`
   c) `Ciao mondo!` dans les autres cas (pourquoi pas l'italien comme langue par défaut ?)

   Le tester en lui transmettant des environnements différents.
3. Visualiser et expliquer les différences entre
   — `ls` et `LANG=C ls`
   — `man ls` et `LANG=C man ls`

— `LANG=fr_FR.utf8 sort noms.txt` et `LANG=C sort noms.txt`, où le fichier noms.txt contient

```
alice
Bob
charlie
Daphne
Éléonore
Zoé
```

--- * ---

Correction de l'exercice 2 :

1. Sur les machines de Galilée, il y a les locales `POSIX` et `C` (qui sont les mêmes), `C.UTF-8` et `fr_FR.utf8`, qui devrait être la langue choisie.
2.
```
#!/bin/sh

case $LANG in
fr*)
        printf 'bonjour le monde !\n'
        ;;
C | C.UTF-8)
        printf 'hello, world!\n'
        ;;
*)
        printf 'Ciao mondo!\n'
esac
```
On peut l'essayer avec, par exemple
```
$ ./hello_world
$ LANG=C ./hello_world
$ LANG=bidule ./hello_world
```
3. Lorsque `LANG=C`, les caractères non ASCII ne sont pas affichés.
4. Langue française pour le manuel ou anglaise lorsque `LANG=C`.
5. Tri alphabétique dans un cas et selon la table ASCII dans l'autre. Le `é` n'est pas dans `ASCII`, mais est codé sur deux octets dont le premier commence par un `1`, donc passe derrière tous les autres.

--- * ---

2 Redirections

Exercice 3 : Redirection des sorties

1. Pour chacune des commandes suivantes, essayer de prévoir les contenus des fichiers dans lesquels il y a des redirections et vérifier :
   a) `printf '%s\n' "Salut les amis !" >message.txt`
   b) `echo hop >a.txt; echo plop >a.txt`
   c) `echo hop >a.txt; echo plop >b.txt; cat [ab].txt >c.txt`
   d) `echo hop >a.txt; echo plop >b.txt; cat [ab].txt >a.txt`
   e) `echo hop >a.txt; echo plop >>a.txt`
2. Mêmes consignes pour les commandes suivantes :
   a) `ls /bin/echo /bin/qsdfg /bin/sh >lsout.txt`
   b) `ls /bin/echo /bin/qsdfg /bin/sh 2>lserr.txt`

c) `ls /bin/echo /bin/qsdfg /bin/sh >lsout.txt 2>/dev/null`

3. Créer pour chaque question un fichier contenant :

a) les noms de tous les fichiers contenus dans le répertoire `/usr/include` ;

b) la concaténation des fichiers `/etc/hostname` et `/etc/group` ;

c) toutes les lignes du fichier `/etc/passwd` contenant `nologin` (utiliser la commande `grep`).

--- * ---

Correction de l'exercice 3 :

1.a) Ouverture en écriture du fichier `message.txt`, le fichier est tronqué s'il existait et la ligne `Salut les amis !` (avec caractère fin de ligne) est écrite dedans.

b) Premier `echo`, idem : ouverture en écriture et tronquature éventuelle de `a.txt`, mais le second fait la même chose, donc le fichier `a.txt` ne contient que la ligne `plop`.

c) Le développement de noms de chemins dans la dernière commande (le `cat`) donne `a.txt b.txt` lesquels sont donc concaténés et écrits dans `c.txt`.

d) Idem, mais cette fois il y a une subtilité importante. Dans la commande `cat`, le fichier `a.txt` est ouvert en écriture et tronqué *avant d'être lu par* `cat` il est alors vide et seul le contenu de `b.txt` va être copié dans `a.txt`. Remarque : utiliser une redirection *append* donne une erreur, comme on pourrait s'y attendre.

e) Grâce à la redirection *append*, `a.txt` contiendra les deux lignes `hop` et `plop`.

2.a) Le fichier `lsout.txt` ne contient que `/bin/echo` et `/bin/sh`. L'autre écriture (quelque chose qui dit que `/bin/qsdfg` n'existe pas) se fait sur la sortie des erreurs.

b) Cette fois le fichier `lserr.out` ne contient que le message d'erreur.

c) Les messages d'erreurs sont envoyés dans le trou noir (`/dev/null`), le fichier `lsout.txt` ne contient que les sorties non erronées.

3.a) `ls /usr/include/* >liste_includes.txt`

b) `cat /etc/hostname /etc/group >hostgroup.txt`

c) `grep nologin /etc/passwd >passnolog.txt`

--- * ---

Exercice 4 : groupements de commandes et redirections

En shell, on peut grouper les commandes en utilisant des accolades ou des parenthèses. Exemple :

```
$ (a=hop; printf '%s\n' "$a"; date)
hop
mar. 23 nov. 2021 17:09:21 CET
$ echo $a # a est défini seulement dans le sous-shell
```

Autre exemple (attention, espace ou fin de ligne après `{` et le `;` suivi d'un espace ou fin de ligne avant `}`).

```
$ { a=hop; printf '%s\n' "$a"; date; }
hop
mar. 23 nov. 2021 17:09:21 CET
$ echo $a
hop
```

La différence entre les deux groupements vient du fait qu'un sous-shell ne change jamais l'état du shell courant.

En utilisant les deux méthodes suivantes :

— plusieurs redirections (`>` et/ou `>>`) ou
— une seule redirection mais un regroupement de commandes (dans un sous-shell ou non, comme vous préférez)

écrire :

1. dans le fichier `infos.txt` la date et l'heure, le nom de la machine (*hostname*), le nom de l'utilisateur et le chemin absolu de son répertoire personnel ;
2. dans le fichier `binaires.txt` la liste au format « long » (avec permissions, taille, propriétaire, etc) des fichiers contenus dans le répertoire `/usr/bin` puis la taille totale de ce répertoire au format *human-readable* avec `du -sh`.

--- * ---

Correction de l'exercice 4 : C'est bien les groupements de commandes :

1. ```
   $ date >infos.txt; hostname >>infos.txt; printf '%s\n' "$USER" >>infos.txt;
   $ printf '%s\n' ~ >> infos.txt
   ```
   contre :
   ```
   $ { date; hostname; printf '%s\n' "$USER"; printf '%s\n' ~; } >infos.txt
   ```
   ou dans un sous-shell
   ```
   $ (date; hostname; printf '%s\n' "$USER"; printf '%s\n' ~) >infos.txt
   ```
2. ```
   $ ls -l /usr/bin >binaires.txt; du -sh /usr/bin >>binaires.txt
   ```
   contre
   ```
   $ { ls -l /usr/bin; du -sh /usr/bin; } >binaires.txt
   ```
   ou dans un sous-shell
   ```
   $ (ls -l /usr/bin; du -sh /usr/bin) >binaires.txt
   ```
   ou même (en profitant du fait que les changements dans un sous-shell sont locaux) :
   ```
   $ (cd /usr/bin; ls -l; du -sh) >binaires.txt
   ```

--- * ---

Exercice 5 : Redirection de l'entrée standard

Créer un fichier `prenoms.txt` contenant les 6 lignes suivantes (avec un retour à la ligne à la fin de la 4ème ligne) :

```
alice
Bob
charlie
Daphne
Éléonore
Zoé
```

puis essayez de prévoir les sorties des commandes ou suites de commandes suivantes (et vérifier) :

1. `tr a-z A-Z <prenoms.txt`
2. `tr -d '\n' <prenoms.txt`
3. `tr '\n' ' ' <prenoms.txt`
4. `tr aeiouAEIOU X <prenoms.txt`

--- * ---

Correction de l'exercice 5 :

1. Changement des minuscules en majuscules (petite remarque pas rigolote du tout : `GNU tr` ne sait pas gérer l'utf8, mais ici on n'a pas de caractère hors ASCII).

2. Suppression (`-d` comme *delete*) de tous les caractères fin de ligne, donc tous les prénoms sont concaténés.

3. Changement des fins de lignes en espaces.

4. Toutes les voyelles sont changées en `X`.

`--- * ---`