R1.04 – TP 2

Récupérez l'archive `tp2.tgz` et décompactez la obligatoirement dans votre répertoire personnel.

Table des matières

1 Édition de commande

Il nous arrive de répéter une commande. Bash conserve en mémoire l'historique des dernières commandes. Cet historique est chargé depuis le fichier `~/.bash_history`) au moment du lancement de bash et recopié dans le même fichier à la fermeture du bash. Cet historique est consultable avec la commande interne `history` qui le présente numéroté. Pour rappeler une commande, plusieurs solutions :

1. ↑ et ↓ permettent de naviguer dans l'historique (proche)
2. ctrl+r permet de rechercher un motif dans l'historique. Le prompt affiche alors : (`reverse-i-search`)`':
   Une répétition de ctrl+r permet de remonter vers une commande plus ancienne contenant le motif.
3. `!n` rappelle la commande de numéro `n`, `!motif` rappelle la dernière commande commençant par `motif`.

Question.

— Consultez l'aide de `history`
— Sauvegardez votre historique dans un fichier `monHisto.txt`. Vérifiez votre fichier.
— Supprimez votre historique. Vérifiez en consultant `~/.bash_history`.
— Ajoutez à votre historique celui fourni dans le fichier `Un/histo.txt`

Résultat attendu :

```
history
...
  85  pwd
  86  ls
  87  touch unFichier.txt
  88  ls
  89  cd ..
  90  cd
  91  ls
  92  cd ..
  93  pwd
  94  history
```

Question. En utilisant ctrl+r, rappelez la 2° commande contenant le mot `touch`. Vous devriez obtenir un fichier `photo.jpg` vide

Question. Rappelez la commande de numéro 72. Vous devriez obtenir un répertoire `Images`

Question.

- Saissisez `pwd`, puis `ls`
- En utilisant ↑ rappelez l'avant dernière commande.
- En utilisant `!-`, rappelez l'avant dernière commande.

Autre aide fourni par bash, la complétion obtenue par →|. L'appui sur cette touche complète chemins ou commandes selon le contexte. Bash complète ce qui est sans ambiguïté. S'il y a ambiguïté, un deuxième →| montre les complétions possibles (rien, si aucune n'existe).

Question. Saisissez `his`→|.

Question. Saisissez `he`→|→|. Quelles commandes connaissez-vous déjà ?

Question. Saisissez `cd /u`→|`b`→|. Saisissez `cd -` pour revenir à votre répertoire précédent.

ctrl+l (ou la commande clear) efface l'écran du terminal ctrl+c interrompt (stoppe définitivement) la commande courante ou l'édition d'une commande et vous affiche le prompt.

Question. Effacez votre terminal.

Question. Exécutez la commande `sleep 100`. Interrompez cette commande.

2 Caractères spéciaux

Bash met à notre disposition trois caractères ayant une signification spéciale dans la désignation d'objets du système de gestion de fichier. Ces trois caractères sont les suivants :

1. `*` correspond à n'importe chaîne de caractères, y compris la chaîne vide
2. `?` correspond à n'importe quel caractère (unique)
3. `[...]` correspond à n'importe lequel des caractères entre les `[]` (un seul). Sauf si le premier caractère est^ ou `!` auquel cas, correspond à n'importe quel caractère sauf ceux entre les `[]`. On peut également définir des intervalles avec `-`.

Dans une commande, ces caractères sont remplacés par les correspondances trouvées avant exécution de la commande. Ce mécanisme est appelé *globbing*.

```
$ ls
alex.txt    alexandre.txt   alexis.txt  clemence.txt    eline.txt   ezio.txt    inaya.txt   jordan.txt
loane.txt   ruben.txt
$ ls a*
alex.txt    alexandre.txt   alexis.txt
$ ls ale ?.txt
alex.txt
$ ls [ei]*
eline.txt   ezio.txt    inaya.txt
$ ls [i-l]*
inaya.txt   jordan.txt  loane.txt
$ ls [ !ae]*
clemence.txt   inaya.txt   jordan.txt  loane.txt   ruben.txt
```

Question. Placez vous dans le répertoire `Deux`. Listez les fichiers dont le nom contient dans cet ordre, `'i'` et `'o'`. Résultat attendu :

```
elio.txt    eliot.txt   ezio.txt    lino.txt    livio.txt   marilou.txt marion.txt  ninon.txt
```

Question. Placez vous dans le répertoire `Deux`. Listez les fichiers dont le nom commence par `'a'` et comporte un `'l'` en quatrième position. Résultat attendu :

```
amelie.txt
```

Question. Listez les fichiers du répertoire `/bin` dont le nom ne commence pas par une lettre minuscule comprise entre `'h'` et `'z'` mais dont le nom contient un `'h'` ou un `'z'`.

3 Variables : utilisateur et d'environnement

il est possible de définir des variables à l'aide d'une simple affectation. La valeur de la variable est ensuite consultable avec son nom préfixé par `$`, éventuellement délimité par des . Exemple :

```
$ a=3
$ echo $a
3
$ echo $a1

$ echo ${a}1
31
```

Ces variables ne sont visibles que depuis le bash qui les définit.

Question.
- Testez l'exemple.
- Ouvrez un nouveau terminal et consultez la valeur de la variable `a`.
- Depuis le premier terminal lancez un autre bash (tapez simplement bash).
- Consultez la valeur de la variable `a`.
- Fermez le bash supplémentaire en tapant ctrl+d.

L'ensemble des variables définies est consultable avec `set`. Il est possible de supprimer une variable avec `unset`.

Pour rendre les variables visibles ailleurs, il faut utiliser la commande `export`. Ces variables deviennent ainsi visibles depuis tout shell lancé depuis le shell courant.

Question.
- Définissez une variable `a` de valeur `3` et une variable `b` de valeur `2`.
- Exportez `a`.
- Depuis ce terminal lancez un autre shell (tapez simplement `bash`).
- Consultez les valeurs des variables `a` et `b`
- Fermez le bash supplémentaire.

Ce mécanisme est utilisé par bash lui même pour définir des **variables d'environnement**. Ces variables d'environnement sont initialisées à la connexion de l'utilisateur, avec des noms prédéfinis qui permettent de configurer l'environnement de l'utilisateur. Par convention les variables d'environnement sont en majuscules. Le shell utilise et modifie ces variables.

Vous pouvez consultez l'ensemble des variables d'environnement avec `env`.

Parmi les variables d'environnement, nous trouvons en particulier :

1. `PATH` : concaténation de répertoires, séparés par `:` où seront cherchées les commandes
2. `HOME` : répertoire personnel
3. `SHELL` : shell utilisé

Question.
- Consultez la valeur de la variable d'environnement `PWD`.
- Positionnez vous dans le répertoire `/tmp`.
- Consultez les valeurs des variables d'environnement `PWD` et `OLDPWD`
- Exécutez la commande `cd -`
- Consultez les valeurs des variables d'environnement `PWD` et `OLDPWD`

Question. Nous souhaitons (en modifiant une variable d'environnement), changer le comportement de la commande `cd` (sans argument ou `cd ~`). Cette commande doit nous positionner dans le répertoire `Tp2/Trois`. Essayez en utilisant un chemin relatif et un chemin absolu. Dans chacun de ces deux cas, testez la commande `cd` depuis différentes positions dans le système de fichiers. Quelle solution est préférable ?

Question.
- Consultez la valeur de la variable d'environnement `PATH`.
- Positionnez vous dans le répertoire `Trois`.
- Saisissez `ok.sh` (pas `./ok.sh`). Ceci ne doit pas fonctionner.
- Concaténez le répertoire `.` au `PATH` existant.
- Saisissez `ok.sh`.

Question.
- Consultez la valeur de la variable d'environnement `PS1`

— Consultez le man de bash et recherchez la section `INVITES` ou `PROMPTING` si votre man est en anglais.
— Modifiez la valeur de la variable d'environnement PS1 pour avoir une invite de la forme : `user@machine (11:00:00) $`
— Restaurez la valeur de la variable d'environnement `PS1`

Comme vous le comprenez, il est donc utile de définir ces variables d'environnement le plus tôt. Le shell nous fournit des fichiers dont le contenu est automatiquement exécuté, soit à la connexion (fichiers *profile*), soit au démarage de bash (fichiers *bashrc*).

Question.
— consultez le fichier `~/.profile`.
— consultez le fichier `~/.bashrc`. La version de l'iut a la particularité de charger le fichier `~/.bashrc_iut` s'il existe.
— créez le fichier `~/.bashrc_iut` avec une commande `echo` affichant le message suivant : `Bonjour, vous utilisez /bin/bash en version XXX` où bien sûr le shell et sa version proviennent des variables d'environnement.
— Lancez un nouveau terminal

4 Protection des caractères spéciaux

Nous venons de voir qu'il existait des caractères spéciaux. Dans certaines circonstances, il peut être nécessaire d'ôter à ces caractères leur particularité. Les protections permettent de forcer l'interpréteur à ignorer la signification spéciale de certains caractères (par ex. `*`) ou mots (par exemple `$HOME`). Il existe trois mécanismes de protection :

1. caractère d'échappement `\` : pour la non interprétation d'un seul caractère, par exemple : `\*`.
2. apostrophe (simple quote) `'`. : un contenu délimité par des quotes n'est jamais interprété.
3. guillemets (double quote) `"` : préserve la valeur littérale de chacun des caractères sauf `$`, et `.` Le backslash ne conserve sa signification que lorsqu'il est suivi par `$`, `"`, `\` ou une fin de ligne.

Question. Créez un fichier `*.txt` contenant quelques mots.

Question. Créez un fichier `*$HOME*.txt` contenant quelques mots.

Question. Essayez la commande `echo ls $HOME *.txt`.

Question. Essayez la commande `echo 'ls $HOME *.txt'`.

Question. Essayez la commande `echo "ls $HOME *.txt"`.

Question. Expliquez les différences de comportement.

5 Pour aller plus loin

Question. Consultez la page de man de `glob` : `man 7 glob`

Il est possible de définir des motifs plus riche en activant une option du bash. L'activation se fait par

la commande suivante : `shopt -s extglob` (vous pouvez consulter le résultat avec la commande `shopt`).

Question. Consultez le man de bash et recherchez la section `Motifs génériques` ou `Pattern Matching` si votre man est en anglais.

Question. Dans le répertoire `Deux`, en utilisant un seul motif, affichez les noms des fichiers qui commencent par `ga` ou `ma` et finissent par `l` ou `n`.

Résultat attendu :

```
gabriel.txt  marion.txt  martin.txt
```