Pratique des machines

TP4 : code octal, script bash, redirections

am@up8.edu

Octobre 2025

Dans ce TP :

- écriture de scripts bash
- redirection avec |

→ **Avant de commencer** : créer un répertoire TP4 dans le répertoire dédié à ce cours sur votre session.

1 Permissions

Protection de votre espace personnel

Vous **devez** protéger votre espace personnel en :

- changeant votre mot de passe (c'est impératif, même pour celles et ceux qui n'utilisent pas les machines du bocal),
- et en restreignant les accès en lecture ou en exécution à votre répertoire personnel aux autres étudiant·es.

Rappel : on utilise la commande `chmod` pour gérer les permissions.

- Pour **ajouter une permission** en écriture (*write*) aux autres utilisateurs (**o**) sur le fichier `file`, il faut faire :

```
$ chmod o+w file
```

- Pour **enlever une permission**, on met un "-" au lieu du "+". Pour changer la permission de l'utilisateur au lieu du groupe, on utilise `u` au lieu de `o`, la lettre `g` pour les autres utilisa-

teurs du groupe, et `a` pour tout le monde (all). On peut déclarer plusieurs changements en les séparant par des virgules, par exemple :

```
$ chmod u-r,o-x dirs
```

Une autre façon de faire pour indiquer les droits sur un fichier consiste à utiliser un masque avec une combinaison des codes octaux correspondant aux permissions à donner. Un code octal est composé de 3 chiffres :

1. le premier code les droits de l'utilisateur,
2. le deuxième du groupe et
3. le troisième des autres.

1.1 Code octal et changement de droits

Lettre	Droit si fichier	Droit si dossier	Code octal
-	pas de permission	pas de permission	0
r	lecture	lister les éléments et copier	4
w	écriture	créer/supprimer éléments	2
x	exécuter	entrer dans le dossier	1

1.1.1 Lecture seule, écriture seule et exécution seule

Par exemple, le code 421 signifie : lecture seule (4) pour l'utilisateur, écriture seule (2) pour le groupe et exécution seule (1) pour les autres.
On l'utilise avec :

```
$ chmod 421 file
```

→ créez un fichier `421.txt`, vérifiez les droits par défaut des différents types d'utilisateurs, puis utilisez la commande ci-dessus pour lui accorder les droits correspondant à la combinaison 421. Vérifiez que les droits ont été changés en vérifiant ses métadonnées avec la commande `ls -l`. Que remarquez-vous par ailleurs ?

1.1.2 Avantages du code octal : combinaison de permissions

On peut plusieurs types de droits en même temps, on peut *combiner* les codes en les additionnant. Par exemple :

- le code 3 = 1 + 2 signifie écriture (2) + exécution (1)
- le code 7 = 1 + 2 + 3 signifie lecture (3) + écriture (2) + exécution (1)

→ à quoi correspondent les codes 5 et 6 ?
→ que signifie le code `753` ? Assurez-vous en en créant un fichier `753.txt`, en changeant ces droits avec `chmod`, puis en vérifiant ses métadonnées avec la commande `ls -l`. Quelle est la combinaison de lettres correspondante ?
→ quelle commande permet d'ajouter le maximum de droits à tous les types d'utilisateur·ices ?

1.2 Changer de propriétaire

Il est aussi possible de changer l'utilisateur propriétaire d'un fichier avec :

```
$ chown utilisateur fichier
```

ou le groupe propriétaire :

```
$ chgrp groupe fichier
```

2 Script bash

Pour l'instant, nous n'avons exécuté d'instructions bash qu'en `ligne de commande`. Nous allons voir ici comment les stocker dans des scripts (fichiers `.sh`) pour pouvoir les réutiliser.

1. **Démarrage** :
   (a) Écrire un script `mon_premier_script.sh` de 2 lignes qui crée un fichier de nom `fichier.txt` et liste le contenu du répertoire courant. On peut faire ceci en ligne de commande avec :

```
$ echo "touch fichier.txt
$ > ls" > mon_premier_script.sh
```

   Rq : le premier signe ">" ligne 2 a été ajouté par le terminal pour signaler que la chaîne de caractères n'est pas terminée. Vous ne devez pas écrire ce caractère vous même.

(b) Vérifier le contenu du fichier à l'aide de la commande `cat`.

(c) Lancer le script à l'aide de l'interpréteur :

```
$ bash mon_premier_script.sh
```

→ Les commandes présentes dans le script se sont-elles exécutées ?

(d) **Recopiez** le contenu ci-dessous dans un autre script `helloworld.sh`

```
#!/usr/bin/bash
VAR="Hello world"
echo "$VAR"
```

shebang

Le shebang #! indique au système le chemin de l'interpréteur à utiliser (il peut y en avoir plusieurs sur votre machine), ici `usr/bin/bash`.

Rq : la commande

```
$ which bash
```

vous indique le chemin absolu vers l'interpréteur bash utilisé par défaut par votre terminal.

(e) `Exécutez le script` grâce à la commande suivante :

```
$ ./helloworld.sh
```

Quel est le message d'erreur que vous recevez ?

(f) Pour que le système puisse exécuter ce script, vous devez `autoriser l'exécution` du script graĉe à la commande :

```
$ chmod u+x helloworld.sh
```

Exécutez le script à nouveau (étape (e)) et vérifiez que celui-ci s'exécute bien.

Exécution de script et chemins

Lorsque vous exécutez :

```
$ ./helloworld.sh
```

Vous indiquez à votre système que le script que vous souhaitez exécuter se trouve dans votre répertoire de travail ("`./`"), et non dans le(s) répertoire(s) par défaut où sont stockés les exécutables. Par défaut, votre système va chercher

les exécutables dans les dossiers qui s'affichent lorsque vous exécutez :

```
$ echo "$PATH$
```

Si vous souhaitez exécuter un script qui se trouve ailleurs que dans votre espace de travail, vous pouvez indiquer à votre système le chemin vers le script :

```
$ ./dossier-qui-contient-mon-script/helloworld.sh
```

→ Exécutez :

```
$ ls -l /usr/bin
```

→ quels sont les exécutables qui s'y trouvent ?

2. **Modification** du script
   Le but est maintenant de créer un script qui demande le prénom de l'utilisateur et lui dit bonjour.
   D'abord, changer la valeur initiale donnée à VAR pour qu'elle soit votre prénom.
   On peut utiliser `echo` pour afficher en même temps une chaîne de caractère constante et la valeur de la variable :

```
$ echo "Bonjour $VAR"
```

   → faites la modification et vérifiez si le script a le comportement attendu. → renommez le script en helloyou.sh

   Pour demander la valeur d'une variable à l'utilisateur (mode interactif), on utilise la commande `read` :

```
$ read VAR
```

   → Placer cette ligne à la place de l'assignation de VAR. Ajouter une ligne avant celle-ci pour afficher le message "Quel est votre nom ?".
   → Exécuter le script.

3. **Amélioration** : on souhaite que "Quel est votre nom ?" s'affiche la même ligne sur la même ligne où on doit écrire le nom.
   En utilisant le manuel sur les commandes `echo` et `read`, trouvez deux manière différentes de faire ça et notez-les dans un fichier.

3 Redirections

Rappel : les opérateurs suivants permettent de rediriger la sortie standard vers :

- un fichier en écrasant son contenu : avec l'opérande "chevron droit" `>`
- un fichier en concaténant la sortie au contenu existant avec "double chevron droit" `>>`
- une nouvelle commande avec l'opérande "pipe" : `|`

On s'intéresse ici à l'opérande `|`, qui permet d'enchaîner des commandes.

→ essayez les commandes suivantes :

```
$ ls -l
$ ls -l | grep sh
$ ls -l | grep txt
```

→ qu'observez-vous ? qu'en concluez vous sur la commande `grep` ?

4 Pour la semaine prochaine : apprendre à utiliser le manuel

$ man [OPTIONS] [[SECTION] PAGE ...] ...

man - Interface de consultation des **man**uels de référence du système

Pour avoir de l'aide sur une commande, il est possible de consulter son manuel d'utilisation avec la commande `man` (pour « manual »), par exemple :

```
$ man cp
```

vous donne le manuel de la commande correspondant à la `copie de fichier`.

→ À quoi servent les commandes suivantes ?

- `wc`
- `sort`
- `uniq`

→ Combien d'arguments prennent les commandes suivantes ?

- `pwd`
- `find`

- `cp`
- `grep`

→ Quelles sont les options à utiliser

- pour afficher sur la sortie standard un texte dont les lignes ont été classées par ordre décroissant (`sort`)
- pour afficher sur la sortie standard un texte dont les lignes ont été classées aléatoirement (`sort`)
- pour lister les fichiers du dossier courant en affichant leurs tailles (`ls`)
- pour lister les fichiers du dossier courant en affichant leurs tailles de manière « lisible par un humain » (`ls`)