université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

vendredi 13 septembre 2024

Introduction à l'informatique
L1–Informatique

TP n° 1

Consignes les exercices ou questions marqués d'un ⋆ devront être d'abord rédigés sur papier (afin de se préparer aux épreuves écrites du partiel et de l'examen). En particulier, il est recommandé d'être dans les mêmes conditions qu'en examen : pas de document ni de calculatrice. Tous les TPs se font sous Linux.

1 TP sous Linux avec la session de secours

En mode « dégradé » les salles de TP ne possèdent qu'un utilisateur local à la machine dont le login est « `secours` ». Le mot de passe vous sera donné en début de séance. Il convient de sauvegarder régulièrement votre travail, soit sur une clé usb, soit sur un espace en ligne.
Conseil : Les deux premiers TPs vont vous permettre de découvrir le shell Unix et de vous familiariser avec des concepts et commandes de base. Il est **vivement recommandé** de garder une trace des commandes que vous avez testées, soit sur une feuille, soit en les copiant dans un fichier texte que vous sauvegarderez. Vous pourrez ainsi réviser efficacement en ne vous contentant pas que du corrigé.

2 Commandes de base

⋆ On considère la séquence de commandes ci-après, rentrées les unes après les autres dans un terminal. Décrire l'effet de chaque commande (création de répertoire, changement de répertoire, affichage dans le terminal, erreur, ...). Les commandes sont décrites page 26 et suivantes du support de cours 1.

1. `cd ~`
2. `mkdir IntroInfo`
3. `mkdir IntroInfo/TP1`
4. `cd IntroInfo/TP1`
5. `cd ..`
6. `ls`
7. `chmod 700 TP1`

Réponse: Durée estimée de l'exercice : 10 minutes de recherche au brouillon puis donner la correction. On peut interroger les étudiants dans l'ordre de la salle, 1 réponse par étudiant. Leur laisser 5 à 10 minutes pour tester ces commandes dans le terminal après la correction.

1. `cd ~` : fait du répertoire utilisateur le répertoire courant
2. `mkdir IntroInfo` : crée un répertoire `IntroInfo` dans le répertoire courant (qui est le répertoire utilisateur). Une erreur peut se produire si le répertoire existe déjà.
3. `mkdir IntroInfo/TP1` : crée un répertoire `TP1` dans le répertoire `IntroInfo`. Une erreur peut se produire si le répertoire existe déjà.
4. `cd IntroInfo/TP1` : `IntroInfo/TP1` devient le répertoire courant.
5. `cd ..` : le répertoire parent de vient le répertoire courant (on se trouve donc dans `IntroInfo`).
6. `ls` : affiche le contenu du répertoire courant (donc `TP1`).
7. `chmod 700 TP1` : change les permissions de `TP1`. Il devient accessible en lecture écriture et "traversable" (on peut rentrer dedans) pour l'utilisateur et ni lisible, ni écrivable ni traversable par les autres.

Ouvrir un terminal et effectuer les commandes ci-dessus. Constatez que tout se déroule comme prévu. **Attention** : il est possible que la toute première commande échoue si vous utilisez un poste sur lequel l'exercice a déjà été fait (les sessions de secours sont nétoyées régulièrement, mais pas à chaque déconnexion).

Dans ce cas, vous commencerez par exécuter la commande `mv IntroInfo IntroInfo_old` (en vérifiant avant que le répertoire `IntroInfo_old` n'existe pas déjà et en adaptant le suffixe (`_old2`, `_old3`, ...) si c'est le cas).

3 Motifs *glob*

⋆ pour chacun des motifs *glob* ci-dessous, donner une suite de caractères de longueur au moins 1 reconnue par le motif.

1. `*txt`
2. `+(txt)`
3. `[0-9]*`
4. `+([0-9])`
5. `@(a.txt|b.txt|c.txt)`
6. `+([^0-9])+([0-9])+([^0-9]).bak`
7. `????`
8. `?*?`

Réponse: Durée estimée de l'exercice : 15 20 minutes de recherche au brouillon puis donner la correction. On peut interroger les étudiants dans l'ordre de la salle, 1 réponse par étudiant. Leur laisser 5 à 10 minutes pour tester ces commandes dans le terminal après la correction.

1. `*txt` : L'expression reconnait n'importe quelle mot finissant par `txt` (ex : toto.txt, footxt, txt, ...). Le plus court est `txt`.
2. `+(txt)` : L'expression reconnait une répétition de `txt` (au moins une fois, donc `txttxttxttxt...`). Le mot le plus court est `txt`.
3. `[0-9]*` : reconnait n'importe quel mot qui commence par un chiffre. Les plus courts sont `0`, `1`, ..., `9`.
4. `+([0-9])` : reconnait un mot composé uniquement de chiffres, de longueur au moins 1.
5. `@(a.txt | b.txt | c.txt)` : reconnait exactement l'un des trois choix `a.txt`, `b.txt` ou `c.txt`.
6. `+([^0-9])+([0-9])+([^0-9]).bak` : reconnait un mot constitué d'une suite non-vide de non-chiffres, suivi d'une suite non-vide de chiffres, suivi d'une suite non-vide de non-chiffres, suivi de `.bak`. Par exemple : `abc1def.bak`
7. `????` : reconnait n'importe quel mot de 4 caractères (`toto`, ...)
8. `?*?` : reconnait n'importe quel mot d'au moins 2 caractères (un au début, un à la fin et une chaine éventuellement vide au milieu).

Tester les réponses dans le terminal. Pour cela, se placer dans le répertoire `TP1` créé lors de l'exercice précédent, créer des fichiers vides au moyen de la commande « `touch` *nomdefichier* » et tester l'expression au moyen de « `ls` *expression* ». Le fichier que vous venez de créer doit être listé (au moins lui). Exemple :

```
$ touch toto.txt
$ ls *txt
```

Affiche `toto.txt`.

Si les expressions de type `@(...)`, `*(...)`, `+(...)`, ou `?(...)`, ne fonctionnent pas, il faut activer le support pour les motifs globs étendus en entrant la commande :

```
$ shopt -s extglob
```

dans le terminal où sont fait les tests.

4 Permissions

⋆ On suppose pour cet exercice que le répertoire courant est le répertoire personnel. De plus, on suppose que les répertoires `IntroInfo` et `IntroInfo/TP1` existent (cf. exercice 1).

Donner la commande permettant de mettre les permissions demandées. On utilisera des permissions numériques et on justifiera en montrant la représentation binaire.

1. le répertoire personnel possède tous les droits pour l'utilisateur et uniquement le droit d'exécution pour le groupe et les autres
2. les répertoires `IntroInfo` et `IntroInfo/TP1` possèdent tous les droits pour l'utilisateur et les droits de lecture et d'exécution pour le groupe et les autres (une commande par répertoire)
3. le fichier `lisible.txt` du répertoire `IntroInfo/TP1` possède les droits de lecture/écriture pour l'utilisateur et uniquement les droits de lecture pour le groupe et les autres
4. le fichier `secret.txt` du répertoire `IntroInfo/TP1` possède les droits de lecture/écriture pour l'utilisateur et aucun droit pour le groupe et les autres

Mettre les permissions sur tous les fichiers et répertoires comme indiqué ci-dessus (vous pouvez créer 2 fichiers `lisible.txt` et `secret.txt` au moyen d'un éditeur de texte). Demander à un voisin (sur une autre machine) de tester les permissions de vos répertoires et fichiers (en essayant de rentrer dans les repertoires et de lire les fichiers). En particulier constater la différence entre droit en exécution et droit en lecture sur un répertoire. Enfin, supprimer les droits pour les autres et le groupe sur le répertoire personnel. Constater que plus personne n'a accès à ce répertoire à part le propriétaire.

Réponse: 10 à 15 minutes pour les permissions et idem pour tester les commandes.

1. On suppose que le répertoire utilisateur est le répertoire courant (comme dit dans l'énoncé).

   ```
   chmod 711 .
   ```

   On veut `rwx` pour l'utilisateur et uniquement `x` pour les autres : 111 001 001 (en binaire) donne 711 en octal (permissions numériques). Quelqu'un d'autre que l'utilisateur peut faire « `cd` » dans ce répertoire mais pas « `ls` » (pas de droit en lecture du répertoire). On peut cependant rentrer dans un sous-répertoire si on en connait le nom (par exemple `IntroInfo`).

2.

   ```
   chmod 755 IntroInfo
   chmod 755 IntroInfo/TP1
   ```

   On veut `rwx` pour l'utilisateur et `rx` pour les autres, donc 111 101 101 donne 755. On constate que n'importe qui peut faire `ls` dans ce répertoire (droit en lecture)

3.

   ```
   chmod 644 IntroInfo/TP1/lisible.txt
   ```

   Similaire au précédant, on veut `rw` pour l'utilisateur et `r` pour les autres : 110 100 100 donnent 644. Le contenu de ce fichier est lisible par d'autres (par exemple avec `cat lisible.txt`).

4.

   ```
   chmod 600 IntroInfo/TP1/secret.txt
   ```

   Similaire au précédant, on veut `rw` pour l'utilisateur et rien pour les autres : 110 000 000 donnent 600. Le contenu de ce fichier n'est pas lisible (par exemple `cat secret.txt` renvoie un message d'erreur pour quelqu'un autre que le propriétaire).

On rappelle que pour trouver le répertoire utilisateur d'un autre utilisateur, on peut utiliser l'alias `~toto` où `toto` est le login de l'utilisateur en question.

5 Permissions 2

Le découpage des permissions entre l'utilisateur, le groupe et les autres n'est parfois pas assez fin. Par exemple, on souhaiterait laisser l'accès à un fichier à tous les utilisateurs connaissant un certain mot de passe, quel que soit leur groupe.

1. ⋆ rappeler brièvement ce qu'autorisent les droits en lecture et en exécution sur des *répertoires*.
2. En se basant la dessus, imaginer quel droits mettre sur votre repertoire personnel pour autoriser tous les utilisateurs connaissant « le mot de passe » `0CtHTp22` à accéder à un fichier `partage.txt`.

Remarque : en pratique cette technique n'est pas vraiment utilisée et permet d'illustrer les différences de permissions sur les répertoire. Un système moderne utilise les ACL (*access control list*), un mécanisme permettant de donner des droits différents à des ensembles arbitraires d'utilisateurs (mais ça dépasse le cadre du cours).

Réponse:

1. Le droit en lecture permet de lister le contenu du répertoire (faire `ls` dedans). Le droit en exécution permet de traverser le répertoire (faire `cd` dedans).
2. Il suffit de créer le répertoire et de mettre les permissions suivantes :

   ```
   mkdir 0CtHTp22
   chmod 711 ~
   chmod 755 0CtHTp22
   ```

   On peut ensuite placer le fichier `partage.txt` dans le répertoire `0CtHTp22`.

   Les gens qui connaissent le mot de passe peuvent faire

   `cd ~kim/0CtHTp22/`

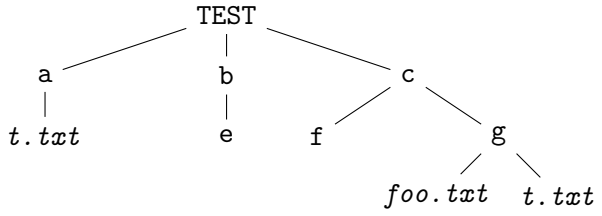
   (on remplacera `kim` par le login de l'utilisateur qui partage le fichier). Ceux qui ne connaissent pas le mot de passe ne peuvent pas faire `cd kim` puis `ls` pour regarder le nom des répertoire se trouvant dans le répertoire personnel, car celui-ci est illisible.

6 Arborescence et chemins

On suppose que l'on se trouve dans un répertoire `TEST`, que ce dernier est vide et que l'on exécute les sept commandes suivantes. Dessiner l'arborescence finale des fichiers et répertoires (on utilisera `TEST` comme racine de l'arborescence).

1. `mkdir a b c d`
2. `touch a/t.txt d/foo.txt`
3. `cd c`
4. `mkdir ../b/e f g`
5. `cd ..`
6. `cp */*.txt c/g`
7. `rm -rf d`

Réponse: Les noms en italique sont des fichiers, ceux en police droite des répertoires. On pourra aussi utiliser un dessin orienté de droite à gauche, comme dans la figure **??**.



7 Recherche de fichiers (bonus)

Si vous avez fini votre feuille en avance, vous pouvez tentez l'exercice suivant, en lisant la fin des supports du cours 1

Se placer dans le répertoire `/usr/share/doc`. Ce dernier contient les fichiers de documentation de tous les paquets logiciels installés sur la machine. Utiliser la commande `find` pour trouver les fichiers correspondant aux critères ci-dessous. En cas de doute, consulter le cours ou la page de manuel au moyen de la commande `man find`.

1. les fichiers dont le nom est `README.txt`

2. les fichiers d'une taille supérieure à 3 Mo
3. les fichiers d'une taille inférieure à 4 Mo (attention, lire attentivement l'aide de l'option `-size` de la page de manuel)
4. les fichiers dont la taille est comprise entre 3 et 4 M
5. les répertoires dont le nom commence par `a` ou `A`
6. les fichiers étant des liens symboliques et, pour chacun de ces fichiers, appeler la commande `ls -l` sur le fichier.

Réponse:

1. `find . -name 'README.txt'`
2. `find . -size +3M`
3. `find . -size -5M` **attention**, le `-` signifie « strictement inférieur à » dans **l'unité demandée**. donc `-4M` voudrait dire, tout ce qui est de taille au plus 3 Mo.
4. `find . -size -5M -a -size +3M`
5. `find . -name '[aA]' -a -type d`
6. `find . -type l -exec ls -l {} \;`