Haute École de Bruxelles
École Supérieure d'Informatique

Bachelor en Informatique
Laboratoires Java - 1ère année

TDLinux - Prise en main de l'environnement, rappels, linux avancé

Résumé

Ce TD a pour but de vous rappeler les commandes `Linux`, des commandes de base aux commandes plus évoluées.

Nous vous invitons à relire le Guide Visuel avant de commencer.

1 Les commandes de base de Linux

Revoyons ici les commandes de base

1.1 Les commandes de base

- `passwd` permet de **changer le mot de passe** ;
- `exit` permet de **quitter** ;
- `nano nomDuFichier` permet d'**éditer** le fichier nommé *nomDuFichier* ;
- `cat nomDuFichier` permet d'**afficher à l'écran** le contenu du fichier nommé *nomDuFichier* ;
- `ls` permet de **lister** le contenu d'un répertoire ;
- `ls -a` permet de **lister** le contenu d'un répertoire, y compris les fichiers cachés ;
- `ls -l` permet de **lister** le contenu d'un répertoire au format long ;
- `mv nomDuFichier nouveauNomDuFichier` permet de **renommer** le fichier nommé *nomDuFichier* en un fichier nommé *nouveauNomDuFichier* ;
- `mv nomDuFichier nomDuDossier` permet de **déplacer** le fichier nommé *nomDuFichier* dans un répertoire nommé *nomDuDossier* ;
- `cp nomDuFichier nouveauNomDuFichier` permet de **copier** le fichier nommé *nomDuFichier* en un fichier copie nommé *nouveauNomDuFichier* situé dans le même répertoire ;
- `cp nomDuFichier nomDuDossier` permet de **copier** le fichier nommé *nomDuFichier* en un fichier copie nommé *nomDuFichier* situé dans un répertoire nommé *nomDuDossier* ;
- `rm nomDuFichier` permet de **détruire** le fichier nommé *nomDuFichier* ;
- `rmdir nomDuDossier` permet de **détruire** le dossier **vide** nommé *nomDuDossier* ;
- `mkdir nomDuDossier` permet de **créer un dossier vide** nommé *nomDuDossier* ;
- `rm -r nomDuDossier` permet de **détruire** le dossier **pas forcément vide** nommé *nomDuDossier* ;
- `pwd` permet **d'afficher le chemin du dossier courant** (celui où vous êtes en ce moment) ;
- `cd chemin/nomDuDossier` permet de changer le dossier courant en *chemin/nomDuDossier* ;
- `cd` permet de changer le dossier courant en votre dossier personnel ;

1.2 Les options dans la ligne de commande.

Nous avons déjà rencontré l'une ou l'autre option ; par exemple, les options `-a` ou `-l` de `ls` ; ou encore l'option `-r` de `rm`.

Ces options modifient le sens d'une commande.

On les reconnait car elles commencent par un `-` suivi d'une lettre ou `--` suivi d'un nom d'option.

Les options disponibles pour chaque commande se trouvent dans la page de manuel `man` de la commande en question.

1.3 Quelques commandes courantes

Les commandes de base

La commande pour :

- voir le contenu d'un dossier (la liste de ce qu'il contient) est ____
- voir le contenu d'un dossier (la liste de ce qu'il contient) au format long est _____
- voir le contenu d'un dossier (la liste de ce qu'il contient), y compris les fichiers cachés est _____
- éditer le contenu d'un fichier est _____
- changer son mot de passe est _____
- se déconnecter de linux1 est _____
- copier un fichier est ____
- renommer un fichier est ____
- déplacer un fichier est ____
- changer de dossier courant est ____
- créer un répertoire est _____
- visualiser le contenu d'un fichier sans l'éditer est ____
- voir quel est le dossier courant (son chemin) est ____
- détruire un fichier est ____
- détruire un dossier vide est _____
- détruire un dossier pas forcément vide est _____
- d'obtenir la liste des options de la commande rm est _____

La ligne de commande

- Qu'est-ce qui permet de distinguer / séparer les différentes parties d'une commande ? _____
- Dans la commande `nanotest`, combien y a-t-il de parties ? __
- Dans la commande `nano test`, combien y a-t-il de parties ? __

1.4 Ressources

2 Système de fichiers, chemin absolu et relatif

Nous allons maintenant nous rappeler ce qu'est le système de fichiers et ce que sont les chemins absolus et relatifs.

2.1 Le système de fichiers

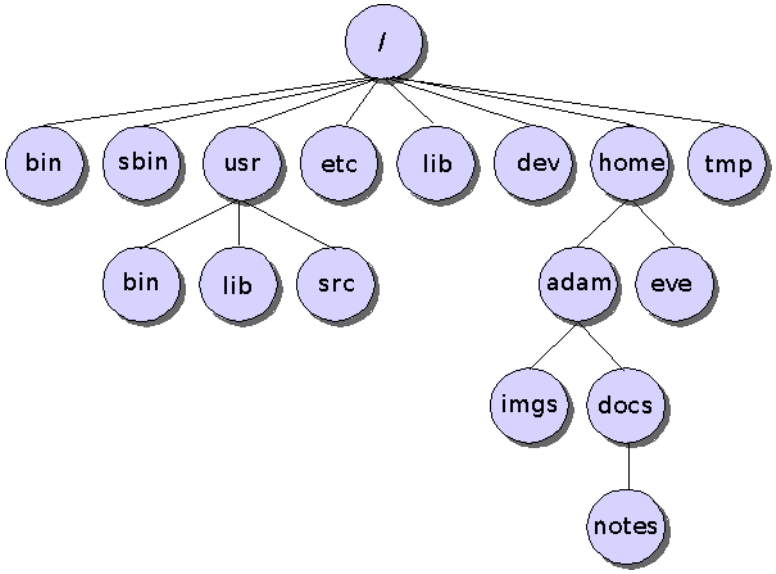

FIGURE 1 – arborescenceUnix.png

[source : franceftars.us.62-152-34-99.ppa.listkom.ru]

Chaque utilisateur dispose d'un dossier personnel, par exemple ici `adam` et `eve` qui se trouve dans un dossier appelé home, qui se trouve lui-même dans un dossier appelé / (la barre oblique).

/ est le dossier principal (on dit aussi la racine) du système de fichiers.

Avec Linux, tous les systèmes de fichiers sont assemblés en un seul système de fichiers dont le dossier principal est désigné par la barre oblique (/).

On peut voir que le dossier principal (`/`) contient le dossier `home` qui contient le dossier `adam`, la home de l'utilisateur `adam`.

Attention ! Le dossier appelé `home` ne contient pas que la home de l'utilisateur mais toutes les homes.

Ici, il contient aussi le dossier `eve` qui est la home de l'utilisateur `eve`.

Il y a des notations de répertoires particuliers :

- le répertoire `.` qui représente le dossier courant ;
- le répertoire `..` qui représente le dossier parent ;
- le répertoire `~` qui représente le dossier personnel, c'est un raccourci pour la home de l'utilisateur, c-à-d, par exemple `~` vaut `/home/adam` si l'utilisateur a pour home `/home/adam` .

Supposons être l'utilisateur `adam` dans le dossier courant `adam`,

- `ls ~` liste le contenu du répertoire `adam`, il est ici équivalent à la commande `ls` ;
- `ls .` liste le contenu du répertoire `adam`, il est ici équivalent à la commande `ls` ;
- `ls ..` liste le contenu du répertoire parent de `adam`, c-à-d `home`.

Supposons être l'utilisateur `adam` dans le dossier courant `notes`,

- `ls ~` liste le contenu du répertoire `adam` ;
- `ls .` liste le contenu du répertoire `notes`, il est ici équivalent à la commande `ls` ;
- `ls ..` liste le contenu du répertoire parent de `notes`, c-à-d `docs`.

2.2 Chemin relatif

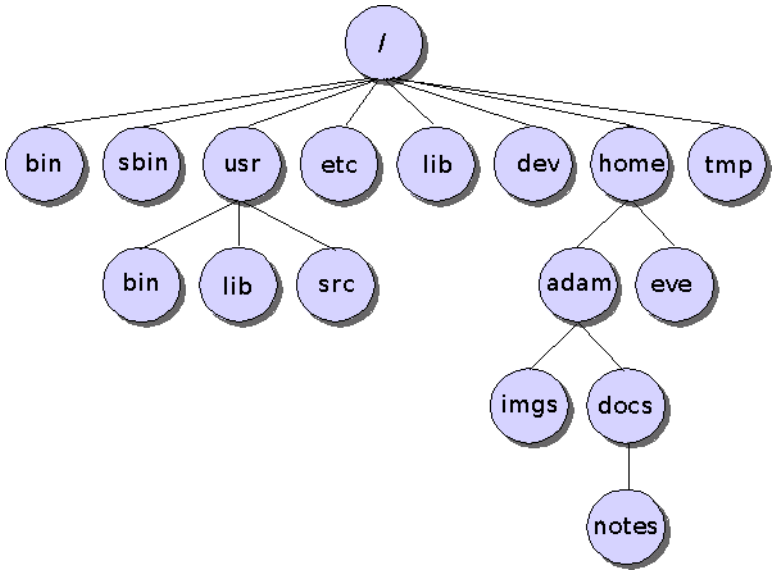

FIGURE 2 – arborescenceUnix.png

[source : franceftars.us.62-152-34-99.ppa.listkom.ru]

Si on se trouve dans la home de l'utilisateur `adam` et qu'on veut lister le contenu du répertoire `notes`,

- on peut le faire en 2 étapes en changeant de répertoire courant : `cd docs ls notes`
- on peut aussi le faire sans changer de répertoire courant, il faut alors lui indiquer le chemin pour arriver au fichier : `ls docs/notes`

On parle de **chemin relatif** car on va indiquer le chemin à suivre à partir du dossier courant, en effet, **à partir du dossier** `adam`, on va dans le dossier `docs`, puis, on va lister le dossier `notes`.

2.3 Chemin absolu

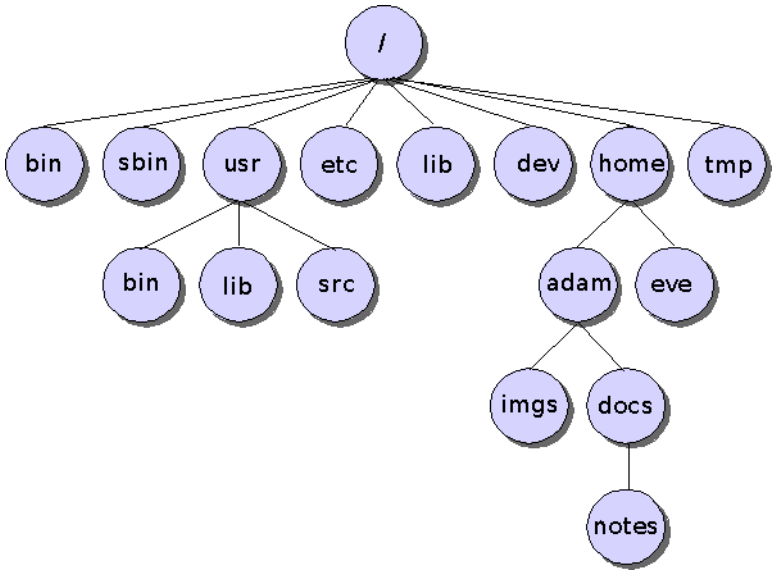

FIGURE 3 – arborescenceUnix.png

[source : franceftars.us.62-152-34-99.ppa.listkom.ru]

Si on se trouve dans la home de l'utilisateur `adam` et qu'on veut lister le contenu du répertoire `notes`,

- on peut aussi le faire sans changer de répertoire courant, en lui indiquant le chemin pour arriver au fichier à partir de la racine : `ls /home/adam/docs/notes`

On parle de **chemin absolu** car on va indiquer le chemin à suivre **à partir de la racine**, cette commande fonctionne donc peu importe le dossier courant.

2.4 Exercices

La racine du système de fichier sous Linux est

- ❍ .
- ❍ ..
- ❍ ~
- ❍ ~g12345
- ❍ /
- ❍ /home

Le(s)quel(s) de ces chemins est/sont un chemin absolu ?

- ❑ /home/g54321/tdLinux
- ❑ ~g54321/tdLinux
- ❑ g54321/tdLinux
- ❑ tdLinux

Le(s)quel(s) de ces chemins est/sont un chemin relatif ?

- ❑ tdLinux
- ❑ ../tdLinux
- ❑ ./../eCours/tdLinux
- ❑ /eCours/java/tds/tdLinux

[source : franceftars.us.62-152-34-99.ppa.listkom.ru]

La ligne de commande

Supposons que le répertoire courant est le dossier personnel `/home/adam`

- Quelle commande permet de supprimer le répertoire `imgs` et son contenu en utilisant un chemin absolu ? _____
- Quelle commande permet de supprimer le répertoire `imgs` et son contenu en utilisant un chemin relatif ? _____
- Quelle commande permet de créer un répertoire `imgs` dans le répertoire `eve` en utilisant un chemin relatif ? _____
- Quelle commande permet de créer un fichier `mesImages` dans le répertoire `imgs` du répertoire `eve` en utilisant un chemin absolu ? _____

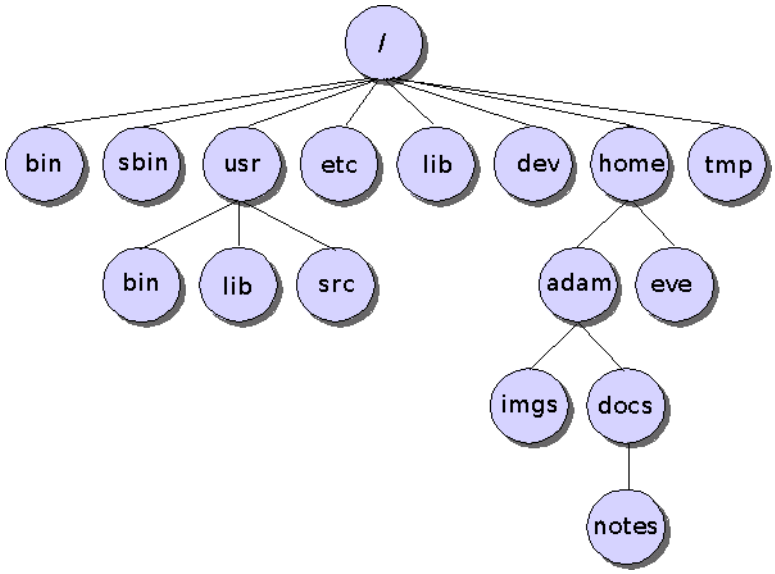

FIGURE 4 – arborescenceUnix.png

— Quelle commande permet de copier ce fichier `mesImages` que vous venez de créer dans le répertoire courant en utilisant que des chemins relatifs ? _____

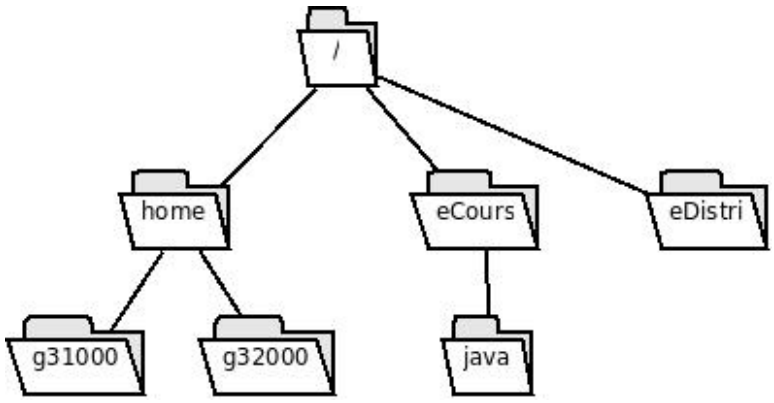

FIGURE 5 – fs.jpeg

La ligne de commande

Supposons que le répertoire courant est le dossier personnel `/home/g31000`

— Quelle commande permet de supprimer le répertoire `java` et son contenu en utilisant un chemin absolu ? _____

— Quelle commande permet de supprimer le répertoire `java` et son contenu en utilisant un chemin relatif ? _____

— Quelle commande permet de créer un répertoire `tds` dans le répertoire `g32000` en utilisant un chemin relatif ? _____
— Quelle commande permet de créer un fichier `Ex.java` dans le répertoire `tds` du répertoire `g32000` en utilisant un chemin relatif ? _____
— Quelle commande permet de copier ce fichier `Ex.java` que vous venez de créer dans le répertoire courant en utilisant que des chemins relatifs ? _____
— Quelle commande permet de lister au format long le dossier personnel en utilisant un chemin absolu ? _____

3 La ligne de commande

3.1 La ligne de commande

Parfois, vous devez entrer une commande assez longue parce que les noms de fichiers sont longs et/ou nombreux. Linux offre plusieurs facilités pour simplifier l'entrée de longues commandes.

La complétion de la commande

Lorsque vous appuyez sur la touche `TAB`, le shell tente de compléter le début de commande que vous avez déjà tapé. Si plusieurs possibilités existent, elles sont affichées si vous appuyez 2x sur `TAB`.

Expérience

Supposons que vous ne vous rappeliez plus très bien de la commande qui permet de modifier le mot de passe. Vous vous rappelez juste qu'elle commence par `pas`.

1. Tapez `pas` puis appuyez 2x sur la touche `TAB`.
2. Entrez un `s` puis appuyez à nouveau sur `TAB`.

La complétion des noms de fichiers

La touche de tabulation permet également de compléter un nom de fichier.

Exercice

1. Dans votre dossier td3, copiez le fichier `monfichieraunomtellementlongquilmeparaitpeuprobabledeletaper2xsanserreur` qui se trouve dans le dossier `/eCours/java/td/td3`.

2. Affichez le contenu de ce fichier en évitant de retaper son nom.

Joker

Il faut savoir que beaucoup de commandes linux qui traitent un fichier peuvent en traiter plusieurs à la fois ; il suffit de les indiquer tous.

Par exemple : `rm texte1 texte2 texte3 texte4` supprime les 4 fichiers indiqués.

On voudrait aller plus loin et éviter de taper explicitement chaque nom. Par exemple, on pourrait avoir envie de dire :

1. Supprime tous les fichiers qui commencent par «texte » ;
2. Supprime tous les fichiers Java (qui se terminent par .java) ;
3. ...

C'est possible grâce à la notion de «joker ». On ne va pas donner explicitement un nom de fichier mais un «motif », c'est-à-dire une description des noms de fichiers concernés.

Il y a essentiellement deux jokers :

— `?` : lorsque le bash rencontre un `?` dans un nom de fichier il sait qu'il peut le remplacer par n'importe quel caractère ;
— et `*` : lorsque le bash rencontre un `*'` dans un nom de fichier il sait qu'il peut le remplacer par n'importe quelle suite de caractères (0, 1 ou plusieurs caractères).

Exercices

1. Copiez dans votre répertoire td3 tous les fichiers du répertoire `/eCours/java/td/td3` dont la deuxième lettre est un '`x`'.
2. Copiez dans votre répertoire tdLinux tous les fichiers du répertoire `/eCours/java/td/td3` dont l'extension est `.java` (c'est possible sans passer par un `cd /eCours/java/td/td3`)
3. Listez le contenu des répertoires des étudiants (pour rappel, les répertoires des étudiants sont ceux qui se trouvent dans `/home` et qui commencent par un '`g`').
4. Listez le contenu des répertoires des professeurs (pour rappel, les répertoires des professeurs sont ceux qui se trouvent dans `/home` et qui sont composés de 3 lettres).

Sélection multiple

La commande `rm td*.java` supprime le(s) fichier(s) :

- ❏ td.java
- ❏ td2
- ❏ td2.java
- ❏ td3Prepa.java
- ❏ td3.java
- ❏ td10.java

La commande `rm td?.java` supprime le(s) fichier(s) :

- ❏ td.java
- ❏ td2
- ❏ td2.java
- ❏ td3Prepa.java
- ❏ td3.java
- ❏ td10.java

Revenir à une commande précédente

Il arrive souvent qu'il faille entrer une commande qu'on a déjà écrite il y a peu (ou en tout cas fort proche de ce qu'on a déjà écrit). C'est là que les flèches viennent à notre secours.

La flèche vers le haut permet de revenir aux commandes précédentes et de les modifier. À utiliser sans modération...

4 Les permissions

4.1 Les groupes d'un utilisateur

Les utilisateurs d'un système Linux sont groupés. Un groupe contient un ou plusieurs utilisateur(s) et un utilisateur appartient à un ou plusieurs groupe(s).

Sur Linux1,

- users : tous les utilisateurs sont dans ce groupe
- enseignants : tous les professeurs sont dans ce groupe
- etudiants : tous les étudiants sont dans ce groupe
- etudiants1 : tous les étudiants de première année sont dans ce groupe

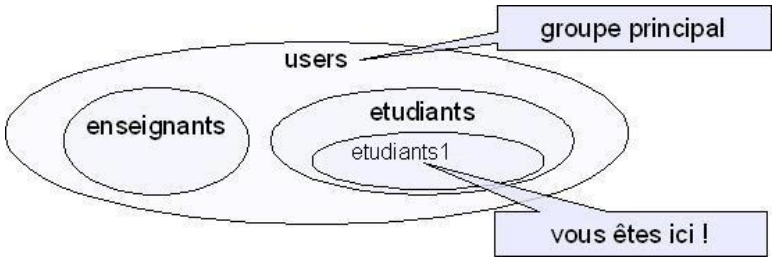

FIGURE 6 – groupesLinux1.jpg

La commande `groups` permet de connaitre tous les groupes auxquels l'utilisateur appartient.

Parmi tous les groupes d'un utilisateur, il y en a un qui est le groupe principal (c'est le premier donné par la commande `groups`). C'est celui qui sera utilisé par défaut lors de certaines opérations (par exemple, lors de la création d'un fichier)

4.2 Propriétaire et groupe d'un fichier

Tous les fichiers ont un propriétaire et appartiennent à un groupe. C'est la base pour définir les permissions sur ce fichier. On pourra ainsi donner des permissions différentes :

- au propriétaire du fichier ;
- aux utilisateurs qui appartiennent au même groupe que le fichier ;
- à tous les autres.

Lister au format long le contenu du répertoire nous permet de voir les infos suivantes :

FIGURE 7 – permissions.jpg

Sur le schéma ci-dessus, on peut lire, par exemple, que le fichier `Max.java` appartient à `g32671` et a été placé dans le groupe users.

Nous vous disions qu'il existait un groupe principal (le premier donné par la commande `groups`) et que c'est celui qui serait utilisé par défaut lors de certaines opérations.

C'est le cas par exemple ici, quand l'utilisateur `g32671` a créé son fichier `Max.java`, ce fichier s'est retrouvé dans le groupe par défaut `users`.

Or,

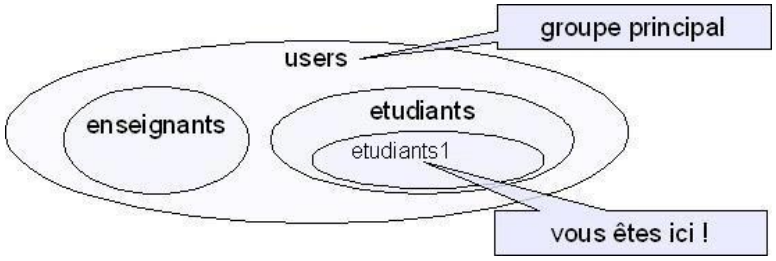

FIGURE 8 – groupesLinux1.jpg

le groupe `users` contient à la fois le groupe `étudiants` / `étudiants1` et aussi le groupe `enseignants`.

Comme nous avons vu que nous pouvions donner des permissions différentes :

- au propriétaire du fichier ;
- aux utilisateurs qui appartiennent au même groupe que le fichier ;
- à tous les autres.

Si le groupe du fichier est `users`, les utilisateurs qui appartiennent au même groupe que le fichier sont à la fois des `étudiants` / `étudiants1` et des `enseignants`.

On ne pourrait donc pas distinguer les permissions des étudiants de celles des enseignants.

Pour faire la distinction entre étudiants et enseignants, il faudrait changer le groupe auquel appartient le fichier.

`chgrp etudiants1 Max.java` indique que le fichier `Max.java` doit être placé dans le groupe `etudiants1` (le propriétaire du fichier peut exécuter cette commande mais il doit obligatoirement indiquer un groupe auquel il appartient).

Il existe aussi une commande que seul l'administrateur peut exécuter `chown g32000 Max.java` qui change le propriétaire du fichier `Max.java` à `g32000`.

Exercices

1. Visualisez le propriétaire des fichiers de votre dossier personnel.
2. Créez un répertoire `tdLinux` dans votre dossier personnel ;
3. Visualisez le propriétaire des fichiers de votre dossier `tdLinux`.

Exercices

1. Visualiser les groupes auxquels vous appartenez.
2. Visualiser le groupe auquel appartient votre dossier `tdLinux`.
3. Quel est votre groupe principal ?
4. Quels sont les groupes auxquels appartient votre professeur ?
5. Avez-vous un groupe en commun avec lui ?
6. Quel(s) groupe(s) Linux avez-vous en commun avec les autres étudiants de votre groupe ESI ?
7. Changez le groupe de votre dossier `tdLinux` pour que les enseignants puissent avoir des permissions différentes de celles des étudiants .

Exercices

1. Visualisez vos fichiers et déterminez à quel groupe ils appartiennent.
2. Créez un fichier de test et modifiez le groupe auquel il appartient.

FAQ

Les fichiers dans mon dossier personnel ne sont pas automatiquement à moi ?

Non. En pratique c'est généralement le cas, mais on peut très bien trouver dans un dossier personnel un fichier qui appartient à quelqu'un d'autre.

4.3 Les permissions

Les permissions sur un fichier

Vous savez qu'un fichier appartient à un utilisateur et est placé dans un groupe.

Vous savez aussi que vous pouvez donner des permissions différentes :

— au propriétaire du fichier ;
— aux utilisateurs qui appartiennent au même groupe que le fichier ;
— à tous les autres.

Voyons maintenant quelles sont ces permissions. Il existe 3 types de permissions : en lecture, en écriture et en exécution.

— Lecture : si une personne a le droit en lecture sur un fichier, il peut en voir le contenu. Par exemple, avec cat, more ou encore view.
— Écriture : Si une personne a le droit en écriture sur un fichier, il peut en modifier le contenu. Par exemple, avec nano.
— Exécution : Cette permission concerne les exécutables. Un exécutable est un fichier qui contient un programme en langage machine directement exécutable par le processeur.

Ces 3 permissions peuvent évidemment être combinées (exemple : on donne l'accès en lecture ET en écriture).

En pratique, on ne veut pas donner les mêmes droits à tout le monde. On pourra préciser :

— les droits du propriétaire d'un fichier ;
— les droits des utilisateurs qui appartiennent au même groupe que le fichier ;
— les droits pour toutes les personnes non reprises dans une catégorie ci-dessus (les autres).

Lister au format long le contenu du répertoire nous permet de voir les infos suivantes :

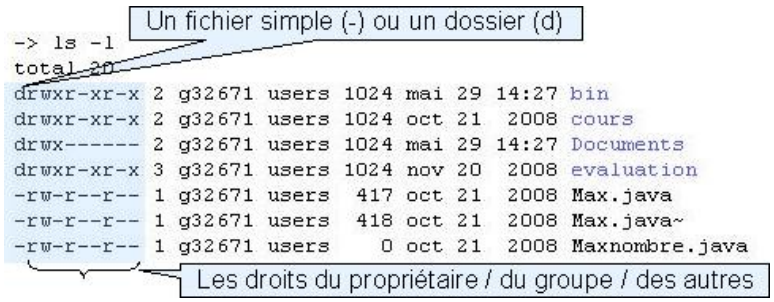

FIGURE 9 – permissions2.jpg

Les permissions sont indiquées avec des lettres (r pour la lecture, w pour l'écriture et x pour l'exécution) dans cet ordre en mettant un tiret (-) si la permission n'est pas donnée.

Il y a 3 blocs de 3 données pour les droits du propriétaire, des utilisateurs du groupe auquel le fichier est attribué et enfin, des autres (dans cet ordre).

Sur l'exemple ci-dessus, on peut voir que le fichier `Max.java` :

— est en lecture/écriture pour le propriétaire (g32671) ;
— est en lecture seule pour tous les utilisateurs du groupe users (en fait tout le monde sur linux1) ;

— est en lecture seule pour les autres (en fait personne sur linux1).

Les permissions sur un répertoire.

Tout comme pour les fichiers, on peut donner des permissions au dossier.

La situation est sensiblement la même :

— un dossier a un propriétaire et un groupe ;
— on indique les mêmes permissions r, w et x ;
— on spécifie des permissions pour le propriétaire, le groupe et les autres.

Ce qui change, c'est la signification des permissions. Que veut dire «exécuter »un dossier par exemple ?

Explicitons à présent le sens de chaque droit :

— lecture : on a le droit de lister le contenu du dossier, de voir ce qu'il-contient. On peut par exemple faire un ls du dossier.
— écriture : on peut modifier le contenu du dossier. On peut ainsi effacer un fichier (via la commande rm). On remarque ainsi que pour effacer un fichier il ne faut pas de droit en écriture sur le fichier mais bien sur le dossier qui le contient.
— exécution : on peut «ouvrir »le dossier / entrer dedans. On peut ainsi en faire son dossier courant (cd td1) ou le traverser dans un chemin (cat td1/monTexte1).

Changer les permissions.

Lorsqu'un fichier est créé, il l'est avec des permissions par défaut définies par l'administrateur du système.

On peut évidemment les changer via la commande `chmod`.

Il existe 2 notations fort différentes pour indiquer les permissions : avec un nombre et avec des lettres.

Changer les permissions avec un nombre.

Pour changer les permissions avec un nombre, vous donnez d'un coup toutes les permissions que vous autorisez

— d'abord, au propriétaire du dossier/fichier ;
— ensuite, aux utilisateurs qui appartiennent au même groupe que le dossier/fichier ;
— enfin, à tous les autres.

Vous avez vu qu'il y a une permission de lecture, d'écriture et d'exécuter/de traverser. Si vous avez fait un peu attention, vous aurez remarqué que chacune de ces permissions se situe dans un ordre bien précis :

— d'abord une permission de lecture (r) ;
— ensuite une permission d'écrire (w) ;
— enfin une permission d'exécuter/de traverser (x).

`rwx` avec à chaque fois la possibilité ou non d'avoir cette permission.

On pourrait

— avoir uniquement un droit de lecture : `r--` qu'on pourrait écrire `100` et convertir en 4 (100 en base 2)
— avoir uniquement un droit d'écriture : `-w-` qu'on pourrait écrire `010` et convertir en 2 (010 en base 2)
— avoir uniquement un droit d'exécution/de traverser : `--x` qu'on pourrait écrire `001` et convertir en 1 (001 en base 2)
— avoir un droit de lecture et d'écriture : `rw-` qu'on pourrait écrire `110` et convertir en 6 (110 en base 2)
— avoir un droit de lecture et d'exécution/de traverser : `r-x` qu'on pourrait écrire `101` et convertir en 5 (101 en base 2)
— avoir un droit d'écriture et d'exécution/de traverser : `-wx` qu'on pourrait écrire `011` et convertir en 3 (011 en base 2)
— avoir tous les droits (de lecture, d'écriture et d'exécution/de traverser) : `rwx` qu'on pourrait écrire `111` et convertir en 7 (111 en base 2)
— avoir n'avoir aucun droit (ni de lecture, ni d'écriture ni d'exécution/de traverser) : `---` qu'on pourrait écrire `000` et convertir en 0 (000 en base 2)

C'est ce nombre trouvé pour les permissions qu'on donnera

— d'abord, au propriétaire du dossier/fichier ;
— ensuite, aux utilisateurs qui appartiennent au même groupe que le dossier/fichier ;
— enfin, à tous les autres.

On combine alors les nombres obtenus pour les 3 sortes d'utilisateurs.

`chmod 640 monFichier.java` donne

— au propriétaire le droit en lecture/écriture ;
— aux utilisateurs dans le même groupe que le fichier le droit en lecture uniquement ;
— et aucun droit aux autres.

Changer les permissions avec des lettres.

Pour changer les permissions avec des lettres, vous dites les permissions que vous ajoutez/retirez, à qui vous les ajoutez/retirez et ce,sans toucher aux autres droits.

- Il faut d'abord dire à qui on modifie les droits :
  - `u` pour le propriétaire *(user)*
  - `g` pour le groupe auquel le fichier/dossier appartient *(group)*
  - `o` pour les autres *(other)*
  - `a` pour tous (le propriétaire, le groupe auquel le fichier/dossier appartient, les autres) *(all)*
- ensuite, spécifier
  - `+` si on ajoute des droits ;
  - `-` si on en retire.
- On indique enfin quel(s) droit(s) on ajoute ou enlève
  - r
  - w
  - x

`chmod a+w monFichier.java`
- ajoute
- le droit d'écriture
- à tous

pour le fichier `monFichier.java`.

`chmod go-wx monFichier.exe`
- retire
- les droits d'écriture et d'exécution
- aux utilisateurs du même groupe que celui auquel appartient le fichier et aux autres (c-à-d à tout lemonde sauf le propriétaire)

pour le fichier `monFichier.exe`.

Déterminez les bonnes permissions

Remplissez les blancs avec la permission correcte (r, w, x ou -). Il s'agit de trouver la permission minimale à mettre pour répondre à la demande.

- Pour un fichier Java, la permission la plus adéquate est __ __ __
- Pour la version compilée (le bytecode), la permission la plus adéquate est __ __ __
- Le fichier qui contient (l'exécutable de) la machine virtuelle a probablement comme permisson __ __ __

Exercice

Soit le fichier `Max.java` de la capture d'écran ci-dessous.

Est-ce qu'un professeur peut l'éditer ?

(la réponse est disponible dans la version en ligne)

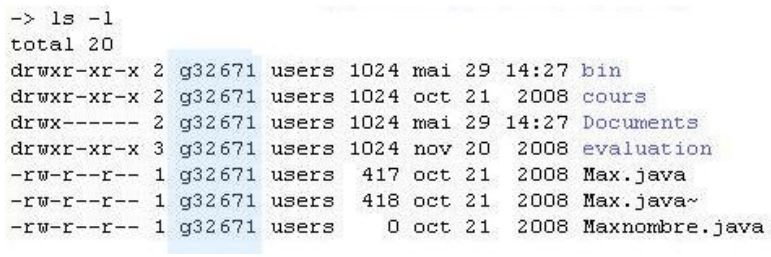

FIGURE 10 – Contenu détaillé d'un dossier

Déterminez les bonnes permissions

Soit le fichier "Max.java" de la capture d'écran ci-dessus.

On voudrait que l'étudiant `g32671` puisse travailler normalement, que les autres étudiants ne puissent pas tricher sur lui mais que les professeurs puissent lire son travail.

— Quel groupe faut-il donner au fichier ?

_____

— Quelle commande permet de donner ce groupe au fichier ?

_____ _____ _____

— Quelles permissions minimales donner au fichier ?

_ _ _ _ _ _ _ _ _

— Quelle commande permet de donner ces permissions au fichier ?

_____ ____ _____

Exercice

Reprenez les permissions affichées dans la capture d'écran ci-dessous et exprimez-les avec un nombre de 3 chiffres.

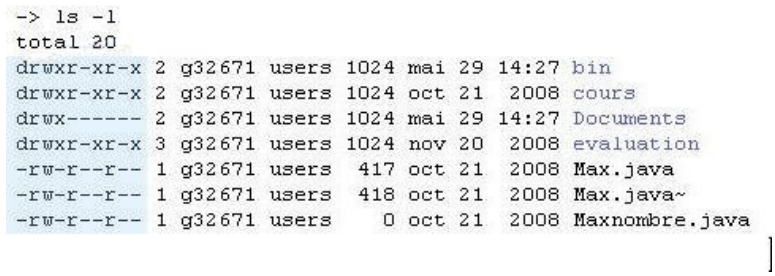

FIGURE 11 – Contenu détaillé d'un dossier

Permissions par défaut

1. Si ce n'est pas encore fait, créez un dossier "tdLinux".
2. Créez-y un fichier vide.
3. Demandez les détails du fichier (propriétaire, groupe, permission)

On constate qu'un nouveau fichier appartient à celui qui l'a créé (on s'en doute) et au groupe principal du créateur. Il y a aussi des permissions par défaut (plutôt permissives dans notre cas).

Modifier les permissions

Vous savez que la commande qui permet de modifier les permissions d'un fichier est `chmod`.

Prenez le temps de **lire** la page de **manuel** de cette commande.

Exercices

1. Créez un fichier `brol` dans le dossier `tdLinux` avec quelques mots.
2. Faites en sorte que personne d'autre ne puisse en voir le contenu.
3. Faites en sorte que tout le monde puisse voir son contenu mais pas le modifier.
4. Faites en sorte que les autres étudiants ne puissent pas voir son contenu mais les professeurs bien. Attention, pour ce faire, il faut pouvoir distinguer les étudiants des enseignants ; et donc, distinguer les groupes.

Exercices

Modifiez les droits de votre dossier `tdLinux` et, si nécessaire, des fichiers qui s'y trouvent pour que tout le monde puisse

1. voir quels fichiers s'y trouvent mais sans pouvoir lire le contenu de ces fichiers ;
2. modifier le contenu d'un des fichiers mais pas supprimer ce fichier ;
3. supprimer un fichier mais pas modifier son contenu.

Les commandes linux

Quelle commande linux permet de faire l'action suivante ?
— se déconnecter _____

— changer son mot de passe _____
— nettoyer l'écran _____
— voir le contenu d'un dossier au format long ____ ____
— créer un répertoire _____
— déplacer un fichier ____
— connaitre les groupes d'un utilisateur _____
— modifier le groupe d'un fichier _____
— modifier le propriétaire d'un fichier (par le super-utilisateur !) _____
— retirer la permission de traverser le répertoire `tds` à tous _____ ____ ____
— placer le fichier `Hello.java` dans le groupe `etudiants1` _____ _____ _____

Les permissions

Remplissez les blancs avec la permission minimale correcte (r, w, x ou -),

1. pour que le répertoire `/home/gxxxxx/td3` permette à un autre étudiant d'y créer le fichier `/home/gxxxxx/td3/fichier`

   — — — — — — — — —
2. pour que le répertoire `/home/gxxxxx/td3` permette à un autre étudiant d'accéder au fichier `/home/gxxxxx/td3/fichier` dont il connait le chemin

   — — — — — — — — —

Modifiez les permissions

— pour que le fichier `/home/gxxxxx/td3/fichier` puisse être lu et modifié par votre professeur et vous même mais seulement lu par les autres étudiants

  _____ __ __ __ _____

— À quel groupe ce fichier doit-il appartenir ?

  _____

— Quelle commande permet de modifier le groupe du fichier afin de l'adapter à ce qui est demandé ci-dessus ?

  _____ _____

Sélection multiple

Parmi les propositions suivantes, lesquelles représentent des chemins absolus ?

❏ `/usr/local/java/`
❏ `/home/g31000/td3`

- ❏ `g31000/td3`
- ❏ `~/td3`
- ❏ `td3`
- ❏ `~g31000/td3`

5 Compter (wc)

5.1 Compter (wc)

Vous vous demandez peut-être combien de lignes fait un programme que vous avez écrit ou encore combien de lignes vous avez écrites aujourd'hui. La commande `wc` peut vous répondre ; elle indique le nombre de lignes, de mots et de caractères contenus dans les fichiers donnés.

Syntaxe : `wc fichier...`

Exemple

La commande : `wc Ex2.java` affiche le nombre de lignes, mots et caractères contenus dans le fichier `Ex2.java`.

Exercices

1. Comment afficher le nombres de lignes de tous les fichiers Java de votre dossier courant.
2. Examinez les options de la commande et trouvez comment n'afficher **que** le nombre de lignes et pas le nombre de mots et de caractères.
3. Une convention d'écriture Java indique de ne pas dépasser la colonne 80 dans les programmes. Trouvez l'option qui permet de vérifier que tous vos programmes actuels vérifient cette convention.

6 Recherche dans des fichiers (grep)

6.1 Recherche dans des fichiers (grep)

Il est parfois difficile de s'y retrouver dans les fichiers. Vous allez être amenés à vous poser des questions du genre :

— Dans quel fichier ai-je écrit un programme qui vérifie si une année est bissextile ?
— Dans quel fichier ai-je déjà utilisé un switch ?

La commande `grep` peut venir à votre secours. Dans son utilisation la plus simple, elle permet d'extraire de fichiers toutes les lignes qui contiennent un certain texte (appelé *pattern*).

Syntaxe : `grep pattern fichier...`

Exemple

Pour trouver dans quel fichier vous avez utilisé une variable nommée "bissextile", vous pouvez écrire :

```
grep bissextile *.java
```

Exercices

1. Comment trouver les programmes Java du TD4 où vous avez déjà utilisé un "switch" ?
2. Comment trouver, parmi **tous** les programmes Java que vous avez déjà écrits, ceux qui utilisent des booléens ?

7 Recherche de fichier (find)

7.1 Recherche de fichier (find)

`find` est une commande linux très puissante qui vous fera gagner beaucoup de temps. Elle permet de rechercher dans une arborescence de fichiers ceux qui correspondent à un critère donné (taille, droits, nom, dates...). Elle

permet également d'appliquer une commande à chacun des fichiers ainsi trouvés.

Attention à ne pas la confondre avec la commande `grep` qui va examiner le contenu des fichiers.

Exemple

```
find ~ -name Ex.java
```

Recherche, chez vous (`~`), un fichier nommé `Ex.java`

Exercice

Trouvez avec la commande `find` tous les fichiers Java que vous avez déjà écrits.

Nous avons écrit pour vous une classe `Color` mais nous ne savons plus très bien où nous l'avons stockée. Nous nous rappelons juste l'avoir mise quelque part dans `/eCours`. Pouvez-vous la retrouver pour nous ?

8 Redirections

8.1 Entrées et sorties standards

Tout programme qui s'exécute dispose de trois fichiers ouverts d'office par le système pour lui : l'entrée standard, la sortie standard et la sortie d'erreur, identifiés respectivement par les numéros 0, 1 et 2.
En Java, on retrouve ces trois fichiers :

- `System.in` pour 0 (entrée standard) qu'on retrouve dans la déclaration `Scanner clavier = new Scanner(System.in);`
- `System.out` pour 1 (sortie standard) qu'on retrouve dans l'instruction `System.out.println();`
- `System.err` pour 2 (erreur standard) qu'on retrouve dans l'instruction `System.err.println();`

Cela peut vous paraitre bizarre de dire que le clavier et l'écran sont des fichiers mais c'est bien ainsi que le programme les voit. Et c'est pratique,

car nous allons pouvoir *rediriger* ces entrées et ces sorties vers de vrais fichiers de façon tout à fait transparente pour le programme ; il ne sera pas nécessaire de le modifier.

8.2 Rediriger la sortie

Il est possible, au moment où on lance un programme, de rediriger sa sortie. Tout ce que le programme enverra sur sa sortie standard (par exemple avec un `System.out.println()` en Java) ne sera pas visible à l'écran mais sera envoyé dans, par exemple, un fichier.

Une redirection de sortie standard se note «>»ou «1>»lors du lancement du programme. Ces redirections sont réalisées par le shell avant l'exécution de la commande et sont transparentes pour cette commande.

Exemple

```
ls −l > liste
```

ou

```
ls −l 1> liste
```

Ces deux commandes sont équivalentes ; elles n'affichent pas le résultat à l'écran, mais l'écrivent dans le fichier `liste` créé ici ou écrasé s'il préexistait. Par défaut `ls -l` affiche le résultat à l'écran. Nous avons *redirigé* cette sortie standard vers un fichier, en l'occurrence `liste`.

Faites l'essai et vérifiez le contenu du fichier créé.

Exercice

1. Reprenez votre programme qui affiche des suites de nombres et plus précisément celui qui affiche la suite appelée : "le pas croissant". Exécutez-le pour afficher les 1000 premiers nombres de cette suite.
2. Sauvez le résultat dans un fichier pour pouvoir l'examiner à votre aise. **Rappel** : pour examiner le contenu d'un fichier, inutile de passer par un éditeur, la commande `more` suffit.
3. Est-ce que le nombre 15007 en fait partie ? (aide : vous vous rappelez de la commande `grep` ?)

Note

Avec la simple redirection en sortie, si le fichier existe déjà, il est écrasé. Nous pouvons choisir de ne pas l'écraser, mais de le compléter (ajouter du contenu à la fin du fichier) via la double redirection (en sortie) notée `>>`.

8.3 Rediriger l'entrée

L'«entrée standard»peut être associée à un fichier au lieu du clavier. Cela permet d'utiliser les données à partir du fichier au lieu de les entrer au clavier. C'est ce qu'on appelle une redirection de l'entrée.

La redirection d'entrée se note «<».

Exemple

```
commande < data
```

Les lectures de la commande se feront dans le fichier `data` et pas au clavier.

Expérimentation

Nous avons écrit pour vous une petite classe `Multiples5` qui lit une série de nombres et n'affiche que ceux qui sont des multiples de 5. À la fin, elle affiche le nombre de multiples de 5.

1. Copiez-la chez vous ; vous la trouverez quelque part dans `/eCours/java`.
2. Lancez l'application et entrez des nombres au clavier. La combinaison de touches `Ctrl-d` est l'équivalent de la marque «fin de fichier»pour le clavier, c'est ainsi que vous terminerez l'acquisition de la série de nombres au clavier.
3. Il ne faut pas confondre `Ctrl-d` et `Ctrl-c` qui tue le processus. Comment mettre en évidence la différence ?
4. Exécutez la classe en associant le clavier à un petit fichier texte où vous aurez écrit les nombres au préalable, séparés par des blancs ou des caractères équivalents (pas de `Ctrl-d ici` ;-)).

Exercice

On vous demande d'afficher, parmi les 1000 premiers nombres de la suite des pas croissants, tous ceux qui sont des multiples de 5. Combien y en a-t-il ?

8.4 Les tubes (pipes en anglais)

Pour résoudre l'exercice de la section précédente, vous avez dû créer un fichier temporaire qui n'a servi qu'à ça. Vous avez redirigé la sortie de la première commande dans un fichier et redirigé ce fichier comme entrée de la seconde commande.

Le symbole «|»permet de chainer des commandes ; la sortie de l'une sert d'entrée à la suivante. On parle de "pipe" en anglais et de "tube" en français.

C'est une situation qui se présente souvent, surtout en Linux qui propose de nombreuses commandes qui font une seule chose (plutôt bien) et qu'on combine pour obtenir un résultat plus conséquent. Des commandes comme `more`, `grep`, `wc`... peuvent prendre leurs données sur l'entrée standard.

Exemple

La commande `more` permet d'afficher un message page par page. Si on ne lui donne pas de nom de fichier, il pagine les données reçues sur l'entrée standard.

On peut donc remplacer

```
ls /home >temp
more temp
```

par

```
ls /home | more
```

Exercice

À vous d'utiliser des pipes.

1. Utilisez un pipe pour afficher parmi les 1000 premiers nombres de la suite des pas croissants, tous ceux qui sont des multiples de 5.
2. Supprimez du programme `Multiples5` la ligne finale qui affiche le nombre de multiples trouvés.
3. Relancez votre commande de l'exercice précédent. Vous ne voyez plus, à la fin, le nombre de multiples, ce qui est normal. Quelle enchainement de commandes permet d'afficher ce nombre (et uniquement ce nombre) ? Rappelez-vous, il existe une commande Linux qui "compte".
4. Affichez, parmi les 1000 premiers nombres de la suite des pas croissants, tous ceux qui contiennent un 0.

8.5 Rediriger les erreurs

On a vu qu'il est possible de rediriger la sortie d'un programme. Il est aussi possible de rediriger ses erreurs.

Attention ! C'est au programme à décider ce qui est un message d'erreur (en utilisant `err` au lieu de `out`).

Une redirection de la sortie d'erreur se note «2>».

Exemple

Supposons que vous ayez écrit une classe `Malfaite` qui provoque beaucoup d'erreurs à la compilation. Pour les regarder à votre aise, vous pouvez les rediriger dans un fichier `erreur` via : `javac MalFaite.java 2>erreur`. Vous pouvez alors examiner les erreurs en ouvrant le fichier `erreur`, par exemple via un `more erreur`.

Exercice

Les professeurs se demandent combien d'étudiants ont déjà copié chez eux le fichier `Multiple5.java`. Pouvez-vous indiquer la (suite de) commande(s) qui permet de répondre à la question ?

9 Les filtres Linux

9.1 Les filtres Linux

De nombreuses commandes Linux sont basées sur le principe `KISS` (Keep It Simple, Stupid). Elles font peu, mais le font bien et surtout, peuvent facilement coopérer pour, au final, obtenir un résultat bluffant. Parmi toutes les commandes, les filtres sont à mettre en évidence.

Un `filtre` est une commande Linux qui acquiert des données sur l'entrée standard et les envoie vers la sortie standard après les avoir éventuellement transformées.

Nous allons nous concentrer sur les filtres suivants : `tr`, `cut`, `cat`, `sort`, `head`, `tail`, `split`, `uniq`, `grep`, `more`, `less`, `wc`, `grep`, ... Vous en connaissez déjà ; voyez-en les pages de manuel respectives et l'aide (`--help`) pour en connaitre le détail.

Exemple

On voudrait connaitre le nombre d'utilisateurs qui sont connectés à linux1 pour le moment.

Voyons, étape par étape, comment on peut obtenir le résultat.

1. La première étape est de penser à la commande `who` qui affiche toutes les connexions actives.
2. C'est gagné, il suffit de compter le nombre de lignes, direz-vous ! Allons ! Ne nous arrêtons pas là ; l'ordinateur peut compter pour nous, ce qui donne :

```
who | wc -l
```

3. Cette fois, ça y est, on a la réponse ! Ben, non ! Il y a un piège car la commande `who` ne donne pas les utilisateurs mais les connexions ce qui n'est pas tout à fait la même chose ; un utilisateur peut avoir ouvert plusieurs fenêtres "putty".

   La commande `uniq` peut venir à notre secours en supprimant les doublons mais il faut que les lignes soient parfaitement identiques et contigües.

   Parfait ! La commande `cut` peut ne garder que certaines colonnes et la commande `sort` peut trier les lignes. On obtient alors :

   ```
   who | cut -f 1 -d '␣' | sort | uniq | wc -l
   ```

   Je ne me réjouis pas trop vite ; je suis sûr que vous allez encore trouver une faille, n'est-ce pas ? Non, pas cette-fois ; on y est !

Exercice 1 - Nombre de connexions d'un utilisateur

Trouvez un enchainement de commandes qui permet de donner le nombre de connexions d'un utilisateur donné.

Il existe de nombreuses façons de résoudre cet exercice. Celle à laquelle nous pensons fait intervenir : `grep`, `wc`, et `who`.

Exercice 2 - Nombre de PC connectés

Trouvez un enchainement de commandes qui permet de donner le nombre de PC connectés à linux1. Ce n'est pas exactement le nombre d'utilisateurs car un utilisateur pourrait être connecté sur plusieurs machines.

À nouveau, il existe de nombreuses façons de résoudre cet exercice. Celle à laquelle nous pensons fait intervenir la commande `tr -s ' '` qui supprime plusieurs occurences consécutives d'un même caractère facilitant ainsi la sélection par colonne de la commande `cut`.

Exercice 3 - Droits sur les dossiers personnels

Trouvez un enchainement de commandes qui permet de donner le nombre de professeurs qui ont donné le droit à ceux qui ne font pas partie de leur groupe d'entrer dans leur dossier personnel.

À votre imagination...

10 Transfert de fichiers

10.1 Transfert de fichiers

Il est bon de pouvoir récupérer ce que vous avez déjà fait sur linux1.

Il existe plusieurs méthodes décrites dans l'aide-mémoire (dans le répertoire *aide*).

- La solution la plus complète : la commande DOS ftp. Pour ce faire : Ouvrir une console DOS (Start/Run.../cmd) et entrer la commande ftp linux1, s'identifier (login/password), commandes put pour envoyer un fichier de DOS vers Linux et get pour l'opération inverse commandes mput et mget pour envoyer/recevoir plusieurs fichiers quit pour quitter.
- 1. Ouvrez l'explorateur de fichier Windows (par exemple en cliquant sur l'icône "My Computer").
  2. Dans le champ d'adresse, tapez l'adresse `ftp://linux1`
     (une capture d'écran est disponible dans la version en ligne)
  3. Une boite de dialogue vous demande votre login et mot de passe (sur *linux1*).
  4. Vous voyez apparaitre un dossier "linux1" qui correspond à votre dossier sur *linux1*.
  5. Vous pouvez y prendre/déposer des fichiers comme vous le feriez pour un dossier Windows. Vous pouvez par exemple les mettre sur une **clé USB**.
- La solution la plus moderne : un logiciel de transfert ftp Il existe moult logiciels faisant ce genre de travail. Celui installé à l'école s'appelle FileZilla. Pas besoin d'explication, dès que vous serez connecté -et après l'installation- vous verrez apparaitre votre répertoire local et votre répertoire distant ... un glisser-déplacer (drag and drop) devrait faire l'affaire.

Il est bon de pouvoir récupérer ce que vous avez déjà fait sur linux1.

Il existe plusieurs méthodes décrites dans l'aide-mémoire (dans le répertoire *aide*).

En voici une.

1. Ouvrez l'explorateur de fichier Windows (par exemple en cliquant sur l'icône "My Computer").
2. Dans le champ d'adresse, tapez l'adresse `ftp://linux1`
   (une capture d'écran est disponible dans la version en ligne)
3. Une boite de dialogue vous demande votre login et mot de passe (sur *linux1*).

4. Vous voyez apparaitre un dossier "linux1" qui correspond à votre dossier sur *linux1*.
5. Vous pouvez y prendre/déposer des fichiers comme vous le feriez pour un dossier Windows. Vous pouvez par exemple les mettre sur une **clé USB**.

11 Gestion des processus

11.1 Gestion des processus

Les programmes contenant des boucles peuvent poser quelques soucis. Par exemple, une boucle mal écrite peut donner un processus qui tourne indéfiniment. Ou encore un processus peut prendre beaucoup de temps et ralentir les autres processus tournant sur la même machine. Voyons comment gérer les processus.

- La commande `ps` affiche une image statique de l'état des processus en cours.
- La commande `kill` envoie un signal à un processus. (Un signal est un message simple envoyé à un processus. Les signaux ont un nom et un ou plusieurs numéro(s) selon le système d'exploitation).
- Tout processus est identifié par un `PID` (Processus Id)

Exemple

```
[~] ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mcd       3408  0.0  0.3  16016  3780 pts/0    Ss   13:43   0:00 -bash
mcd       3536  0.0  0.3  15856  3572 pts/1    Ss   14:19   0:00 -bash
mcd       3599  100  0.1  12832  1448 pts/0    R+   14:27   0:02 /bin/bash ./Infini
mcd       3600  0.0  0.0   4684   964 pts/1    R+   14:27   0:00 ps u
[~] kill 3599
[~] ps u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mcd       3408  0.0  0.3  16016  3780 pts/0    Ss+  13:43   0:00 -bash
mcd       3536  0.0  0.3  15856  3572 pts/1    Ss   14:19   0:00 -bash
mcd       3601  0.0  0.0   4684   960 pts/1    R+   14:27   0:00 ps u
```

FIGURE 12 – ps et kill

- `kill 3599` envoie au processus de PID `3599` le signal n°15 (SIGTERM, signal par défaut), qui demande au processus de se terminer.
- `kill -SIGKILL 3599` ou `kill -9 3599` envoie explicitement le signal n°9 (SIGKILL) au processus `3599` ; ceci le tuera.
- La touche clavier `Ctrl-c` envoie le signal n°2 (SIGINT) au processus lié au terminal et a comme effet de le **terminer**.

— La touche clavier `Ctrl-z` envoie le signal n°20 (SIGTSTP) au processus lié au terminal et a comme effet de le **suspendre**. Ce dernier reste donc dans le système.
— Pour obtenir la liste de tous les signaux : `kill -l` ou encore `man 7 signal`.

Expérimentation - La boucle infinie

1. Écrivez un programme minimal contenant une boucle infinie

```
while(true){}
```

2. Visualisez vos processus en cours en utilisant la commande `ps u`.
3. Ouvrez une seconde fenêtre putty, et exécutez-y votre boucle infinie. Exécutez à nouveau la commande `ps u` dans la première fenêtre.
4. Retrouvez le processus correspondant au programme qui cycle (son PID) et tuez-le en utilisant la commande kill avec les bons paramètres. Quel nom a le programme à tuer ?

   Sur linux1, le système tue les processus après un temps défini d'utilisation du CPU (timeout). Il se pourrait donc qu'il s'arrête avant l'effet de votre action ; ce n'est pas le moment de s'endormir ;-).
5. Lancez une deuxième exécution et suspendez votre programme par `Ctrl-z`. Vérifiez l'état du processus stoppé par la commande `ps` (`man ps` et recherchez la signification du champ `STAT`).
6. Reprenez le processus interrompu en envoyant le signal SIGCONT (via la commande kill) et vérifiez son nouvel état avant qu'il ne soit éjecté par le système à cause du «timeout».
7. Une deuxième manière de reprendre un processus suspendu est de taper la commande `fg num` (faites un `man bash`), cela doit être fait dans la console dans laquelle vous avez tapé `Ctrl-z`. Le numéro `num` est fourni par le système lorsque le processus a été suspendu par `Ctrl-z`.

   Essayez aussi `fg` pour reprendre le dernier processus suspendu.

12 Conclusion

12.1 Félicitations

Vous êtes arrivés au bout de ce premier TD.

Avant de quitter le laboratoire, n'oubliez pas de quitter proprement la connexion avec `linux1` (`exit`). et d'éteindre l'ordinateur ou de vous déloguer.

Attention, afin d'arriver au laboratoire dans les meilleures conditions, il est bien de revoir la matière qui sera mise en pratique. C'est pourquoi nous vous fournissons quelques **exercices préparatoires** à faire à la maison pour vous permettre d'évaluer si vous êtes prêt. Afin de vérifier que vous préparez bien ces exercices, une **interrogation** sera faite avant de démarrer chaque labo.

À la semaine prochaine et soyez à l'heure !