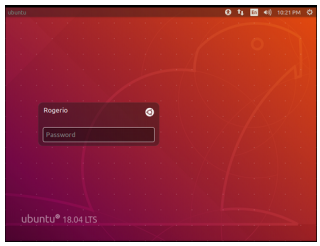
TP Linux 2 : la ligne de commandes

I Le standard POSIX

Lors du dernier cours, nous avons présenté les différents systèmes d'exploitation (Windows, Linux, Android, MacOs, iOs). Malgré leur différence, ils possèdent (sauf windows 10) quelquechose en commun : le standard POSIX. Ce standard définit pour tous les systèmes d'exploitation les programmes de base (lecture et écriture des fichiers, accéder au réseau, …) permettant d'utiliser le système. Le standard POSIX s'inspire largement des systèmes d'exploitation UNIX (Linux, MacOS, iOs, Android). Windows 10 n'est pas compatible POSIX car historiquement Windows est un dérivé du système d'exploitation MS-DOS et pas d'UNIX.

II Utilisateurs et groupes



Rogerio a oublié son mot de passe Ubuntu

Les systèmes POSIX sont conçus pour être **multi-utilisateurs** (comme au lycée où 1 ordinateur peut être utilisé par plusieurs élèves différents, chaque élève ayant ses dossiers et fichiers). Chaque utilisateur possède un **identifiant de connexion** (login) comme prénom.nom par exemple. Y est associé un **mot de passe** (comme *LoSC59 !*) permettant d'identifier l'utilisateur. Toutes ces données (identifiant, mot de passe, dossiers, fichiers) constituent **le compte** de l'utilisateur. Lorsq'un utilisateur s'identifie, le système démarre alors une interface utilisateur personnalisée. L'utilisateur commence alors **une session.** Le système associe un nombre appelé UID (User IDentifier) à l'identifiant.

Vérification : allez sur : https://cocalc.com/doc/terminal.html
puis faites : Run Terminal Now

A moins que vous ayez déjà un compte. Dans ce cas, créer un projet. Ensuite, faites New puis ->Terminal

Dans l'invite de commande, après le $, tapez : id
réponse : uid=2001(user) gid=2001(user) groups=2001(user)

L'utilisateur a pour UID : 2001.

GID signifie Groupe Identifier donc identifiant du groupe. L'administrateur de l'ordinateur (celui qui a tous les droits ou presque!) peut créer des groupes d'utilisateurs. Un utilisateur appartient à un groupe principal et à des groupes secondaires.

D'après la réponse précédente, j'appartiens seulement au groupe 2001(user).
Par contre, imaginons cette réponse :

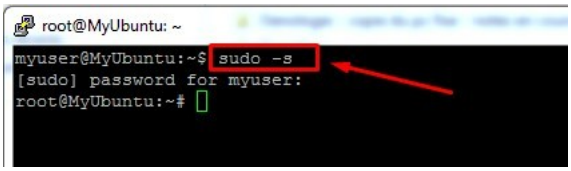
alice$ id
uid=1001(alice) gid=2002(nsi) groups=2002(nsi),2003(première)

Dans ce cas, Alice a pour groupe principal 2002. groups correspond à tous les groupes auxquels elle appartient donc 2002(nsi) et 2003(première)

Pourquoi l'administrateur définit-il des groupes d'utilisateurs de l'ordinateur ? On va voir qu'en procédant ainsi, l'administrateur peut attribuer des **permissions** (lire, écrire, exécuter) sur les dossiers et les fichiers à certains groupes et pas à d'autres.

L’administrateur sous Linux est appelé **root** dont l’UID et le GID vaut 0. C’est le super-utilisateur. Quand vous installez Linux sur un ordi, vous créez un mot de passe. Dans l’invite de commande, si l’on tape : " sudo -s " pour passer en mode super-utilisateur, le mot de passe est demandé : si vous l’avez alors le système ouvre une session root. Dans ce type de session, vous avez beaucoup de droits : en particulier, vous pouvez créer des groupes, ajouter des utilisateurs, en enlever, etc.

Vous pouvez essayez de taper sudo su dans le terminal de cocalc : vous serez remis à votre place de simple " user ".



III Permissions et propriétés des fichiers (ls -l)

Le système d’exploitation associe à chaque fichier ou dossier l’UID et le GID de son propriétaire. Le système permet aussi de définir pour un fichier ou dossier des permissions pour le propriétaire, le groupe propriétaire et les autres utilisateurs.

Comment voir les permissions d’un fichier ?

Il faut utiliser la commande ls avec l’option -l. Vérifiez que vous êtes bien dans /home/user en tapant pwd après $ pour avoir le répertoire courant. Tapez donc : ls -l après $

Vous voyez cette réponse :

```
drwxr-xr-x 2 user user  2 May 29 13:34 permissions
-rw-r--r-- 1 user user  0 May 31 13:22 permissions.term
```

Vous avez sans doute un autre nom que " permissions ".

Sur le dossier " permissions ", les permissions sont ce charabia : drwxr-xr-x
d signifie qu’il s’agit d’un dossier (sinon – signifie que c’est un fichier). Ensuite, il faut lire les symboles suivants (r,w ou x) par bloc de 3 : rwx r-x r-x

r : droit de lire (read)
w : droit d’écrire (write)
x : droit d’exécuter
- : permission refusée

le **premier groupe** de 3 symboles indique les permissions du **propriétaire** sur le fichier ou dossier. " rwx " pour cet exemple, le proriétaire a tous les droits.

Le **deuxième groupe** de 3 symboles indique les permissions du **groupe** propriétaire sur le fichier ou dossier. " r-x " pour cet exemple, le groupe proriétaire peut lire et exécuter ce dossier mais pas d’y écrire (pas le droit d’y créer des fichiers ou d’en supprimer ou de les renommer).

Le **troisième groupe** de 3 symboles indique les permissions pour tous les autres utilisateurs sur le fichier ou dossier. " r-x " pour cet exemple, les autres utilisateurs peuvent lire et exécuter ce dossier.

IV Application : permissions sur un fichier python

Nous allon créer un programme bidon en python et voir quelles sont les permissions sur ce fichier. Nous allons exécuter ce programme python depuis la console de cocalc.

Toujours dans /home/user du terminal tapez : nano. L’éditeur de texte s’ouvre. Tapez ce super programme :

```
#!/usr/bin/python
print("cool linux")
```

puis **Ctrl O** puis écrire le nom du fichier : **cool.py** puis **entrée** au clavier puis **Ctrl X**

Remarque : la ligne #!/usr/bin/python permet d'exécuter ce programme en console (si on en a le droit!) en donnant le chemin vers le dossier où se trouve python.

Vérifiez que votre fichier a bien été créé et affichons les permissions, tapez : ls -l
-rw-r--r-- 1 user user 39 May 31 14:23 cool.py

C'est bien un fichier (- au début). Pour le propriétaire : permissions de lire et écrire, pas d'exécuter. Pour le groupe et les autres, permisssions : lire seulement. C'est dommage ! On ne peut pas exécuter notre programme en console. Essayons quand même. Tapez : ./cool.py qui correspond au RUN de windows !
Réponse : bash: ./cool.py: Permission denied

Bon, c'est confirmé : on n'a pas la permission de l'exécuter. On en reste pas là ! On va se l'octroyer ce droit !

V Modifier les permissions CHMOD
1) Exemple

Une commande Linux permet de modifier les permissions, il s'agit de chmod (change mode). Par exemple, ici on veut ajouter (+) à l'utilisateur (u) la permission d'exécuter (x) le fichier cool.py. Il faut taper :
chmod u+x cool.py

Essayez ! Puis faites ls -l. Normalement, les permissions sont devenues :
-rwxr--r-- 1 user user 39 May 31 14:23 cool.py

Le x est bien apparu. Donc, maintenant : ./cool.py devrait marcher. Essayez. S'affiche :
cool linux

2) Généralisation

La commande chmod s'écrit : chmod cmp fichier-dossier

c étant soit u (utilisateur) soit g (groupe) soit o (autres) soit a (tous)
m étant soit + (ajouter des droits) soit – (retirer des droits)
p étant soit r soit w soit x

Exemple : si vous faites :
~$ chmod g+w,o-r cool.py
Vous obtenez :
-rwx rw- --- 1 user user 39 May 31 14:23 cool.py

On a ajouté le droit d'écrire le fichier cool.py au groupe et retiré le droit de lire aux autres.

VI Encore quelques commandes (cp, mv et redirection >)

Pour **copier** un fichier dans un dossier : cp nom_fichier nom_dossier
Pour **renommer** un fichier : mv nom_fichier nouveau_nom_fichier

Pour **rediriger** le résultat d'une commande dans un fichier texte :
ls /home/user/*[.py] > fichiers_python.txt.
Traduction:fait la liste des noms de fichiers dont le nom contient n'importe quel quel caractère suivi de " .py " et écrit cette liste dans le fichier fichiers_python.txt. Essayez et vérifiez avec nano.