Users, groupes, permissions

Users

Les systèmes UNIX sont multi-users.

L'user `root` a tous les droits ("super-user"). Son identifiant est `0`.

```
man adduser deluser usermod chage
```

```
/etc/passwd
/etc/shadow
```

- l'UID est la référence (si vous permutez deux noms par rapport à leurs UID), par exemple, c'est lui qui est stocké dans les inodes ou les processus, le nom n'apparaît que dans `/etc/passwd` pour faciliter la manipulation humaine.
- un user ne correspond pas forcément à une personne (utilisateur/utilisatrice).

Users

Que se passe-t-il quand on crée un user ?

Demo.

Groupes

```
man addgroup

/etc/group
/etc/gshadow
```

- GID, mêmes remarques que pour UID.
- Un user est relié à un groupe, un groupe est relié à un ensemble (éventuellement vide) d'users, il est parfois utile de s'en tenir à ce niveau formel, sans interprétation excessive.
- Il existe des groupes sans user !
- Deux groupes avec le même ensemble d'users n'ont pas forcément les mêmes permissions, un groupe n'est pas déterminé par les users qui lui sont liés.
- Un user peut ne pas se trouver dans le groupe du même nom https://wiki.debian.org/SystemGroups

Champs gecos/finger

Des questions sur l'identité de l'utilisateur·ice sont posées lors de l'utilisation de la commande `adduser`. Les réponses seront ajoutées au champ de commentaires de `/etc/passwd`. Elles peuvent être utilisées dans les méta-données de certains formats de fichiers (par exemple pour indiquer automatiquement le nom de l'auteur du document). L'utilisateur·ice peut ne pas en avoir conscience.

Attention à ne pas trahir les utilisateur·ices à leur insu !

Pensez à utiliser l'option `--gecos ''`

Retour sur "le home"

Si un user s'appelle `tartempion`, son répertoire personnel (en: homedir) sera le répertoire `/home/tartempion/` par défaut. Ce chemin est appelé le HOME de l'user `tartempion`.

Le fichier `/etc/passwd` définit la variable d'environnement `${HOME}` qui définit le raccourci `~`.

Remarque sur la relativité relative des chemins.

Fichiers et processus

Les fichiers et les processus ont un user propriétaire et un groupe propriétaire.

Certains services sont exécutés en tant qu'un user dédié (par exemple, un serveur web n'est pas exécuté en tant que `root` mais en tant que `www-data`), de sorte à limiter ses accès. Ces users système ont un ID inférieur à 999.

Permissions d'accès au fichier

- modification de certaines permissions d'un fichier :

```
$ chmod {o,g,u,a}{+,-,=}{r,w,x} <fichier>
```

- écrasement de toutes les permissions d'un fichier :

```
$ chmod <nombre_octal> <fichier>
```

- changement de l'user propriétaire d'un fichier :

```
$ chown
```

- changement du groupe propriétaire d'un fichier :

```
$ chgrp
```

Signification des permissions pour les répertoires

Rappel: un répertoire est un fichier dont le contenu est une suite de liens (<nom>, <numéro d'inode>).

- lecture : accéder aux noms des liens
- écriture : ajouter ou supprimer des liens
- exécution : connaissant le nom, accéder au numéro d'inode correspondant

Pour qu'un user puisse accéder à un fichier identifié par un chemin dans l'arborescence, il faut que tous les répertoires du chemin absolu entre la racine et son nom soient exécutables (on parle de "traverser" les répertoires).

Explications au tableau et démos.

Permissions d'accès au fichier

En plus des 9 flags précédents, il y en a 3 qui répondent à des besoins particuliers :

Introduction : comment changer son mot de passe ?

Pourquoi pouvoir lire `/etc/passwd` donne un avantage par rapport à `su`, même si les mots de passe sont hachés ? Attaques par dictionnaires sans attendre que l'OS redonne la main.

```
setuid (fichiers, ex passwd, sudo)
setgid (fichiers et répertoires, ex site web et www-data)
sticky bit (répertoires, ex /tmp)

notation SST our les superposer aux 9 flags visibles
majuscule lorsqu'il y a rien derrière, minuscule sinon.
```

Explications au tableau et démos

Permissions étendues : ACL

Les "Access Control Lists" permettent de définir des permissions sur un fichier plus finement que selon 3 entités (user, group, other).

Ce sont des attributs étendus (`xattrs`) qui sont eux aussi stockés dans les inodes.

```
man setfacl
man getfacl
```

Heptakosioiheptekontaheptaphobie

Relire cette partie du cahier de vacances.

Remarque : la commande `chmod -R 777 /` semble donner toutes les permissions et donc résoudre tous les problèmes, mais ils créent des problèmes de sécurité et certains programmes refusent de faire confiance à des ressources insuffisamment protégées.

C'est par exemple le cas pour votre clef privée SSH, si elle est trop lisible, votre client SSH refusera de l'utiliser car elle est accessible à trop de monde (et donc plus vraiment privée).

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/id_rsa
```

Sudo, su, runuser

Quelques blagues à rajouter au cahier de vacances :

- https://www.luc-damas.fr/hop/jacques-a-dit
- https://xkcd.com/149/
- https://turnoff.us/geek/sudo-prank/
- https://me.me/t/sudo
- http://infotrux.free.fr/wp-content/uploads/2013/05/commandes_linux.png
- https://me.me/i/godeheaven-rm-rf-devil-permission-denied-god-heaven-rm-rf-devil-858b4f30c8cd448f810b95f64deb446a
- https://xkcd.com/838/ (cf TP)

Et tous les problèmes que vous avez rencontré avec `sudo ssh-keygen`, `sudo ssh`, etc

La commande `runuser` permet de faire exécuter à `root` des commandes en tant qu'user moins privilégié (et en particulier, faire prendre moins de risques au système).