Le contrôle d'accès

Le modèle traditionnel Unix — Les permissions par défaut à la création — La commande **umask**

- Chaque fois qu'un utilisateur crée un nouveau fichier, des permissions par défaut lui sont attribuées.
- On peut les modifier avec la commande **`umask`** *(user mask)*
  - **`umask`** travaille avec les *droits à retirer* dans un masque
  - En standard, un nouveau fichier est créé avec les permissions **`rwx r-x r-x`**.
  - Pour ce cas, la commande est **`umask 0022`** :
    - Le premier 0 indique que les chiffres sont en octal.
    - Le second 0 indique qu'aucun droit pour l'utilisateur est retiré, donc **rwx**.
    - Le 2 indique que le droit **w** (**2**) est retiré pour le groupe et les autres, donc **`r-x`**.
  - Bien sûr, quand un programme crée un fichier, il peut restreindre encore les droits. Par exemple un éditeur va créer un fichier texte en enlevant le droit **x**.
  - Quand on modifie les permissions par défaut avec **`umask`** l'action est valable pour les fichiers crées ensuite, sans effet rétroactif sur les fichiers déjà créés.
  - Quand on appelle **`umask`** sans arguments la commande affiche le masque courant.

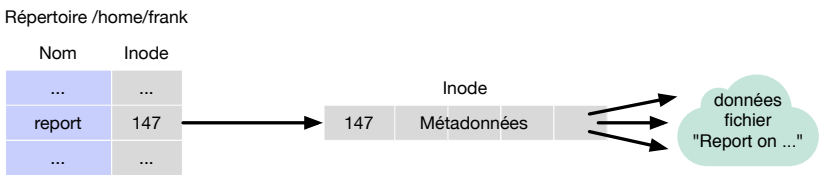
Le système de fichiers

Les liens — Introduction

- Un lien est similaire à un pointeur sur un fichier.
  - On part d'un fichier normal.
  - On ajoute un lien au fichier : Le lien apparaît comme un nouveau fichier, mais quand on accède à son contenu, on accède au fichier existant.
  - On peut modifier vers quel fichier le lien pointe.
- Il y a deux types de liens
  - les liens physiques / durs *(hard link)*
  - les liens symboliques *(symbolic link)*
- Les liens physiques sont utilisés rarement aujourd'hui, car limités à une partition.
  - Exception : La sauvegarde incrémentale

Le système de fichiers

Les liens — Anatomie d'un fichier

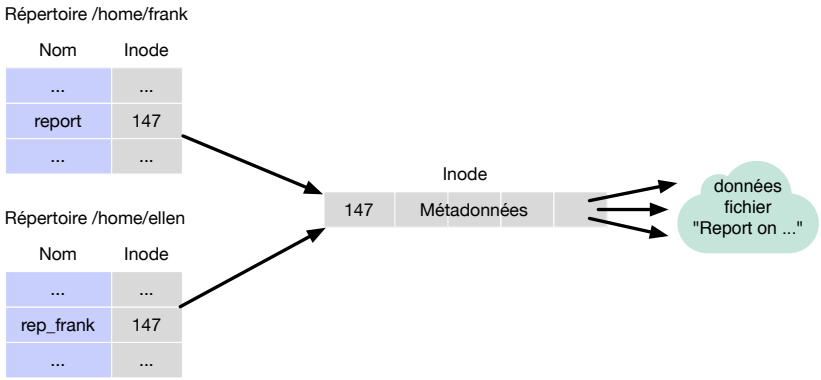


- Un fichier se divise en trois parties :
  - Une entrée dans un répertoire qui contient son nom et un numéro de *inode*
  - Un *inode* qui contient les métadonnées et des pointeurs vers les blocs de contenu
  - Les blocs de données stockant le contenu du fichier

Le système de fichiers

Les liens — Les liens physiques *(hard link)*

- Quand on crée un lien physique vers un fichier existant, on crée une nouvelle entrée dans un répertoire qui pointe vers le *inode* du fichier.



- Le fichier est maintenant accessible depuis deux entrées de répertoire.
- Cela est possible seulement si le répertoire est sur le même *filesystem* que le *inode*.

Le système de fichiers

Les liens — Les liens symboliques *(symbolic link)*

- Pour traverser les limites des *filesystems*, Unix introduit les liens symboliques *(symbolic links).*
- Un lien symbolique a son propre inode et ses propres données fichier.
- Les données fichier contiennent le nom de chemin du fichier existant (absolu ou relatif).

Le système de fichiers

Les liens — Commande **ln** pour créer des liens

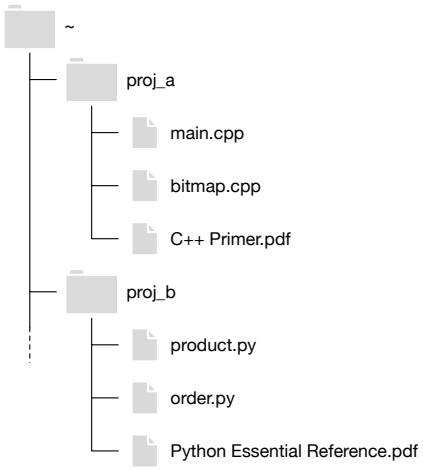
- La commande **`ln`** prend deux arguments :
  - le nom de chemin du fichier existant
  - le nom du lien à créer.
- Par défaut **`ln`** crée un lien physique.
  - Pour créer un lien symbolique on spécifie l'option **-s** ou **--symbolic**.
- Exemple : Frank est le propriétaire du fichier **`report`** dans le répertoire **`/home/frank`**. Ellen crée un lien symbolique vers ce fichier dans son répertoire **`/home/ellen`** sous le nom **`rep_frank`**.

```
$ pwd
/home/ellen
$ ln --symbolic /home/frank/report rep_frank
$ ls -l /home/frank/report rep_frank
-rw-rw-r--. 1 frank pubs 38 06-12 09:51 /home/frank/report
lrwxrwxrwx. 1 ellen pubs 13 06-12 09:52 rep_frank -> /home/frank/report
$ cat rep_frank
Report on sales by Frank
...
$
```

Le système de fichiers

Les liens — Exemple d'application

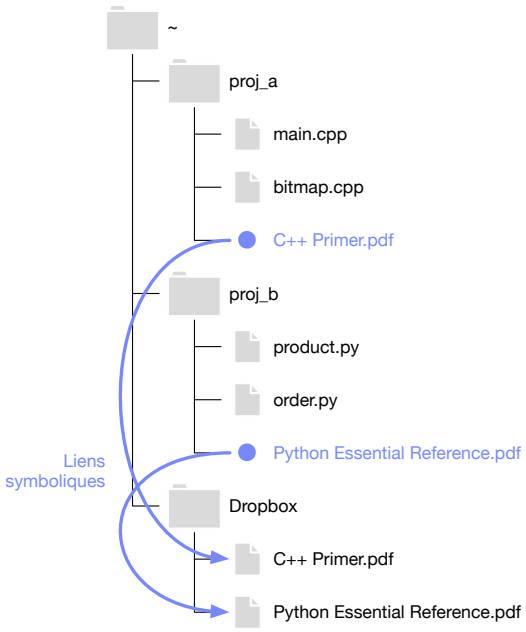
- Frank a acheté plusieurs livres électroniques. Il aime bien les utiliser comme référence quand il travaille sur divers projets.
  - Il place donc les fichiers des livres (fichiers PDF) dans les dossiers des projets.
- Maintenant Frank voudrait lire ses livres aussi sur sa nouvelle tablette. Il pourrait utiliser Dropbox pour partager et synchroniser les livres entre ordinateur et tablette. Mais il ne veut pas mettre tous les projets dans Dropbox. Comment faire ?



Le système de fichiers

Les liens — Exemple d'application

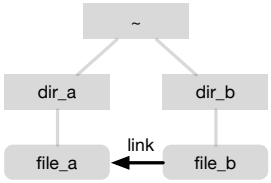
- Solution :
  - Frank déplace tous les livres dans le dossier Dropbox pour les partager entre ordinateur et tablette.
  - Dans les dossiers projet il crée des liens symboliques vers les livres qui sont stockés dans Dropbox.



Le système de fichiers

Les liens — Exercice

- a) Répétez la procédure suivante en utilisant une fois un lien physique et une fois un lien symbolique. Que constatez-vous ?
  - Créez un premier répertoire `dir_a` avec un fichier nommé `file_a` qui contient le texte `111`.
  - Créez un deuxième répertoire `dir_b` avec un lien nommé `file_b` qui pointe sur `file_a`.
  - Accédez à `file_b`.
  - Supprimez `file_a`.
  - Accédez à `file_b`.
  - Re-créez `file_a`, mais contenant le texte 222.
  - Accédez à `file_b`.
- b) Avec la même situation initiale que a) faites les manipulations suivantes. Que constatez-vous ?
  - Déplacez `file_a` dans un autre répertoire.
  - Accédez à `file_b`.



Le système de fichiers

Les liens — Comparaison liens physiques / liens symboliques

	Liens physiques	**Liens symboliques**
Un lien peut pointer sur un fichier dans un autre *filesystem*		
Un lien peut pointer sur un répertoire		
On peut distinguer le fichier original et le lien		
Un lien peut pointer sur un autre lien		
Déplacer le fichier original ne casse pas le lien		
Supprimer et recréer le fichier original ne casse pas le lien		

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Souvent on a besoin de trouver dans une arborescence (ou sur toute la machine) des fichiers qui remplissent certains critères, par exemple
  - trouver des fichiers orphelins qui ont un propriétaire qui n'a plus de compte sur la machine
  - trouver des répertoires qui sont ouverts en écriture à tout le monde
  - trouver des fichiers qui n'ont pas été modifiés depuis plus de cinq ans.

- La commande `find` a été conçue pour ce genre de recherche. Elle est plus adaptée au scripting que la commande `ls` :
  - L'affichage de `ls -l` peut varier de système en système. Il est déconseillé de l'utiliser pour un script.
  - `find` permet de formuler une requête et obtenir une liste complète des fichiers qui correspondent

- Par défaut `find` ne se limite pas aux fichiers d'un répertoire comme `ls`, mais descend de manière récursive une arborescence complète (comme `ls -R`).

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Syntaxe :
  - `find chemin expression_booléenne`
  - `chemin` : Chemin du répertoire de départ. `find` va parcourir les fichiers de ce répertoire et de manière récursive tous les répertoires qu'il contient.
  - `expression_booléenne` : Sert à tester différents propriétés du fichier : nom, type, permissions, date de la dernière modification, etc. Permet de combiner les tests par les opérateurs booléens ET, OU et NON.
- Fonctionnement :
  - `find` parcourt systématiquement tous les fichiers et répertoires dans l'arborescence
  - Pour chaque fichier / répertoire rencontré
    - Évalue les tests dans l'expression booléenne.
    - Si le résultat est vrai, affiche le fichier (ou exécution de l'action spécifiée).
- Expression booléenne :
  - Est évaluée avec court-circuit (comme les opérateurs && et || en C)

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Options de recherche
  - Type de fichier : `-type` *code*
    - Par défaut `find` trouve tout type de fichier. Ce critère permet de chercher un type spécifique. Le code peut être `f` pour un fichier ordinaire, `d` pour un répertoire, `l` pour un lien symbolique, ...
  - Nom de fichier : `-name` *motif*
    - Le *motif* peut être le nom complet recherché (`-name building.jpg`) ou peut contenir les métacaractères `*`, `?` et `[ ]` . Dans ce dernier cas protéger le motif (`-name "*.jpg"`)
    - Si l'on veut ignorer la casse utiliser la variante `-iname` *motif*
  - Propriétaire du fichier : `-user` *utilisateur*
    - On peut spécifier l'utilisateur par nom (`-user bob`) ou par UID (`-user 1002`)
  - Groupe auquel le fichier est associé : `-group` *groupe*
    - On peut spécifier le groupe par nom (`-group proj_a`) ou par GID (`-group 1001`)

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Options de recherche (suite)
  - Permissions du fichier : `-perm` *pmode*
    - On spécifie pour *pmode* un tiret `-` suivi d'un mode de permission symbolique (comme dans chmod). Exemple : `-perm -g+w` cherche tous les fichiers qui peuvent être écrits par le groupe
  - Date de dernière modification du contenu du fichier : `-mtime` *jours*
    - Ce critère ne travaille qu'avec des jours (périodes des 24 heures). On spécifie 0 pour une modification le jour même, 1 hier, 2 avant-hier, etc.
    - Les signes + ou - permettent de tester une inégalité
      - `-mtime 2` : fichiers modifiés il y a deux jours (entre 48 et 72 heures)
      - `-mtime -2` : fichiers modifiés il y a moins de deux jours
      - `-mtime +2` : fichiers modifiés il y a plus de deux jours
  - Taille du fichier : `-size` *taille*
    - La syntaxe de *taille* est particulière car elle travaille par défaut en blocs d'une taille de 512 octets. Ajouter k pour spécifier des Ko, M pour des Mo et G pour de Go (puissances de 2).
    - Les signes + ou - permettent de tester une inégalité
      - `-size` +10k : fichiers d'une taille de plus de 10 Ko
      - `-size` -10k : fichiers d'une taille de moins de 10 Ko

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Opérateurs booléens
  - ET logique : `-a`, `-and`
    - Inséré par défaut entre les options de recherche, peut être omis
  - OU logique : `-o`, `-or`
    - Priorité plus faible que le ET logique, peut nécessiter des parenthèses `( )`. Les parenthèses doivent être protégées de l'interprétation du shell.
    - Exemple : Trouver des fichiers ordinaires qui se terminent en `.jpg` ou `.png` :

```
-type f \( -iname '*.jpg' -o -iname '*.png' \)
```

  - NON logique : !

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Commandes
  - Généralement on termine l'expression booléenne par une ou plusieurs commandes. Les commandes sont traités par `find` comme des options de recherche qui retournent toujours la valeur booléenne *vrai*.
  - Afficher le nom du fichier : `-print`
    - C'est la commande par défaut.
    - Attention : `-print` utilise un espace pour séparer les noms de fichier. Si les noms contiennent des espaces et on veut traiter les noms dans un script, préférer la commande `-print0`
  - Afficher le nom du fichier en séparant les noms par le caractère nul : `-print0`
    - Cette commande est généralement utilisée avec la commande `xargs -0` pour passer les noms de fichiers en argument à une commande.

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Commandes (suite)
  - Exécuter une commande : `-exec` *commande arguments* `\;`
    - `-exec` va exécuter la commande située juste après pour chaque fichier trouvé.
    - La commande doit se terminer par un point-virgule `;` Ce caractère doit être protégé pour ne pas être interprété par le shell.
    - Pour passer comme paramètre pour la commande le fichier trouvé par `find`, il faut écrire `{}`. Ces accolades seront substituées par le chemin du fichier.
    - Exemple : supprimer le fichier trouvé : `-exec rm {} \;`
    - La valeur booléenne de `-exec` correspond au code de retour de la commande — vrai pour 0, faux dans le cas contraire.

Le système de fichiers

Trouver des fichiers à travers leurs méta-données — **find**

- Exemples
  - Trouver dans le répertoire personnel tous les fichiers plus grands que 5 Mo
    - `find ~ -size +5M`
  - Trouver dans le répertoire personnel tous les répertoires appartenant au groupe `proj_a` ou au groupe `proj_b`
    - `find ~ -type d \( -group proj_a -o -group proj_b \)`
  - Trouver dans le répertoire personnel tous les fichiers se terminant en ~ (les fichiers de sauvegarde) et les supprimer
    - `find ~ -type f -name '*~' -exec rm {} \;`
    - Avant de lancer cette commande on fera un essai avec `find ~ -type f -name '*~' -exec echo rm {} \;`
  - Trouver dans le répertoire personnel tous les fichiers de texte qui contiennent la chaîne de caractères `'lab report'` et les copier dans le répertoire `/tmp/reports`
    - `find ~ -type f -name '*.txt' -exec grep 'lab report' {} \; -exec cp {} /tmp/reports \;`

Le compte super-utilisateur
Introduction

- Le compte super-utilisateur *(superuser account)* est l'utilisateur omnipotent du système Unix
  - Sa caractéristique essentielle est d'avoir un UID 0.
  - Par convention on lui donne le nom root.
  - Sur les systèmes Unix le super-utilisateur peut effectuer toute opération valide sur tout fichier ou processus.
- Exemples d'opérations réservées au super-utilisateur
  - Régler l'horloge du système
  - Fixer le nom de la machine
  - Configurer des interfaces réseau
  - Arrêter le système
  - Changer des quotas d'utilisation et des priorités de processus
  - ...

Le compte super-utilisateur
Devenir super-utilisateur

- Il y a trois manières de devenir super-utilisateur
  - Se connecter au système directement comme root
  - Changer de compte en étant connecté sur un autre compte utilisateur et devenir root avec la commande `su`.
  - Exécuter une commande avec les droits super-utilisateur à travers la commande `sudo`.
- Se connecter comme root
  - Par convention l'invite de commande change de **$** à **#**.

```
Ubuntu 14.04 LTS ubuntu tty1

ubuntu login: root
Password: (not echoed)
Last login: Mon May 5 01:15:12 PDT...
Welcome to Ubuntu 14.04 LTS...
root@ubuntu:~#
```

- Changer de compte avec `su`
  - Taper `su -`. Le système demandera le mot de passe de root.
  - En cas de succès on obtient l'invite de commande de root.

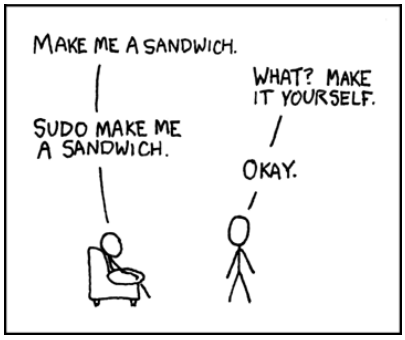
```
marcel@ubuntu:~$ su -
Password: (not echoed)
root@ubuntu:~#
```

Le compte super-utilisateur

Devenir super-utilisateur

- Exécuter une commande avec `sudo`
  - `sudo` permet d'exécuter une commande comme utilisateur root sans savoir le mot de passe de root
    - Par exemple la commande `shutdown -h now` éteint le système, mais peut seulement être exécutée par le super-utilisateur.
  - `sudo` nécessite seulement le mot de passe de l'utilisateur, pas celui de root
  - Après avoir donné son mot de passe, on peut continuer pendant un certain temps à exécuter des commandes sans devoir répéter de nouveau le mot de passe.
  - Un fichier de configuration détermine quels utilisateurs peuvent utiliser `sudo` : `/etc/sudoers`

```
$ shutdown -h now
shutdown: Need to be root
$ sudo shutdown -h now
Password: (not echoed)
The system is going down for system
halt NOW!
[...]
```



Source: http://xkcd.com/149/

Le compte super-utilisateur

Devenir super-utilisateur

- Une connexion au compte root procure des pouvoirs illimités.
  - Une petite erreur peut avoir des conséquences catastrophiques.
    - Par exemple la commande `rm -rf /`
  - Un terminal laissé sans surveillance peut être abusé.
- En règle générale on préfère utiliser des pouvoirs super-utilisateurs **seulement quand c'est nécessaire**.
- Beaucoup de distributions Linux viennent avec une configuration par défaut qui encourage l'utilisation de `sudo` :
  - Le compte root est bloqué pour la connexion directe (son mot de passe est aléatoire et secret)
  - Lors de l'installation un compte utilisateur personnel est créé.
  - Ce compte est configuré pour autoriser la commande `sudo`.

Gestion des utilisateurs
Introduction

- Les *comptes utilisateurs* sont le mécanisme par lequel
  - les utilisateurs se présentent au système
  - prouvent qu'ils sont ceux qu'ils prétendent être
  - sont autorisés ou non d'accéder à l'information et aux ressources disponibles sur le système.
- Du point de vue du système, un utilisateur n'est pas nécessairement un individu. C'est une entité qui peut exécuter des programmes et être le propriétaire de fichiers.
  - Il y a des comptes utilisateurs qui sont crées seulement pour exécuter les processus d'un sous-système ou service : les *pseudo-utilisateurs*
- Tout utilisateur est caractérisé par
  - un nom
  - un numéro d'utilisateur (UID)
  - un ou plusieurs numéros de groupe
  - un mot de passe
  - un shell
- L'information des comptes utilisateurs est stockée dans plusieurs fichiers
  - Les utilisateurs sont définis dans `/etc/passwd` (et `/etc/shadow`)
  - Les groupes d'utilisateurs sont définis dans `/etc/group`

Gestion des utilisateurs
Fichier **/etc/passwd**

- Fichier maître de l'information des utilisateurs
- Chaque compte utilisateur a un enregistrement. Un enregistrement est une seule ligne qui contient :
  - le nom utilisateur *(username)*
  - le mot de passe chiffré ou x (quand le mot de passe est stocké séparément)
  - le numéro d'utilisateur *(UID)*
  - le numéro de groupe d'utilisateurs *(GID)*
  - le nom complet, numéro de téléphone, bureau et autre information pour consommation humaine (champ « GECOS »)
  - le répertoire personnel *(home directory)*
  - le *login shell*
- Exemple :

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
[...]
marcel:x:1000:1000:Marcel Graf,,,:/home/marcel:/bin/bash
$
```

Gestion des utilisateurs
Utilisateurs spéciaux

- Traditionnellement un système Unix contient des utilisateurs spéciaux. Parmi ceux-ci :
  - `root` : super-utilisateur (uid = 0)
  - `daemon` : utilisateur fictif des démons
  - `bin` : propriétaire de /bin et de /usr/bin
  - `sys` : utilisateur système (System V)
  - `adm` : propriétaire des fichiers de comptabilité
  - `uucp` : utilisateur pour les connexions UUCP
  - `lp` : utilisateur administrateur de l'impression

Gestion des utilisateurs
Le fichier *shadow*

- Le fichier `/etc/passwd` est en lecture pour tous
- Le mot de passe chiffré n'est pas déchiffrable ...
- ... mais une attaque de force brute à base de dictionnaires peut aboutir (exemple : outil *crack*)
- Sous certains systèmes (principalement System V et Linux), la liste des utilisateurs est décomposée en deux fichiers :
  - `/etc/passwd` (sans mots de passe) lisible par tous
  - `/etc/shadow` (avec mots de passe) lisible par le super-utilisateur uniquement
- Le champ du mot de passe peut contenir un seul caractère
  - par exemple `x` : reflétant une redirection vers un autre moyen d'authentification, ou
  - `*` : empêchant l'authentification.

Gestion des utilisateurs

Fichier **/etc/shadow**

- Chaque ligne contient :
  - le nom de l'utilisateur
  - le mot de passe chiffré
  - la date du dernier changement de mot de passe
  - le nombre minimum de jours entre deux changements du mot de passe
  - le nombre maximum de jours de validité du mot de passe
  - le nombre de jours avant l'expiration du mot de passe à partir duquel l'utilisateur est averti
  - le nombre de jours pendant lesquels le compte peut être inutilisé
  - la date d'expiration du compte
  - un champ 'réservé'

Gestion des utilisateurs

Définition des groupes, fichier **/etc/group**

- Un groupe peut être défini de deux manières :
  - Implicitement, par GID : chaque fois qu'un nouveau GID apparaît dans la colonne 4 de `/etc/passwd`.
  - Explicitement, par nom et GID : en faisant un enregistrement dans le fichier `/etc/group`.
- Un enregistrement dans `/etc/group` consiste d'une seule ligne. La ligne contient :
  - le nom du groupe *(group name)*
  - le mot de passe chiffré du groupe (utilisé par `newgrp`)
  - le numéro du groupe *(GID)*
  - la liste des utilisateurs supplémentaires appartenant au groupe (en plus des utilisateurs listés dans `/etc/passwd`)
- Exemple :

```
$ cat /etc/group
root:x:0:
daemon:x:1:
[...]
marcel:x:1000:
$
```

Gestion des utilisateurs
Outils de gestion

- Édition du fichier `/etc/passwd` : `vipw`
  - Lance un éditeur et charge le fichier.
  - Crée un verrou avant pour éviter des éventuelles modifications concurrentes.
  - Commande correspondante pour le fichier `/etc/group` : `vigr`
- Changement de mot de passe : `passwd [utilisateur]`
- Changement de nom complet, numéro de téléphone, bureau, etc. : `chfn [utilisateur]`
- Changement du shell : `chsh [utilisateur]`
- Modification de paramètres dans `/etc/passwd` et `/etc/shadow`
  - Linux : `usermod`
  - Solaris : `passmgmt`

Gestion des utilisateurs
Création d'un compte utilisateur

- La création d'un compte utilisateur comprend les tâches suivantes :
  - Affecter à l'utilisateur un nom, un UID et un groupe principal, et décider de quels autres groupes il doit devenir membre. Cette information est enregistrée dans les fichiers de configuration des comptes.
  - Affecter un mot de passe au nouveau compte.
  - Créer un répertoire personnel pour l'utilisateur.
  - Copier des fichiers de configuration initiaux *(skeleton files)* dans le répertoire personnel.
  - Utiliser chown et/ou chgrp pour faire du nouvel utilisateur le propriétaire de son répertoire et des fichiers.
  - Fixer d'autres paramètres du compte (vieillissement du mot de passe, date d'expiration du compte, ...).
  - Ajouter l'utilisateur à d'autres services (service email, impression, ...).
  - Tester le nouveau compte.

Gestion des utilisateurs

Création d'un compte utilisateur

- Ajouter un compte utilisateur manuellement :
  - Ajout dans `/etc/passwd` (et dans `/etc/group` éventuellement) : `vipw` (et `vigr`)
  - Enregistrement du mot de passe : `passwd utilisateur`
  - Création du répertoire personnel (généralement dans /home) :
    ```
    mkdir répertoire
    chown utilisateur répertoire
    chgrp groupe répertoire
    ```
  - Copie initiale des fichiers de configuration : `cp -a /etc/skel/* /etc/skel/.[!.]* répertoire`
- Ajouter un compte utilisateur automatiquement (Linux) :
  - La commande `useradd` fait tout cela.
  - Exemple :
    - ```
      useradd -u 1008 -c "Albert Einstein" -p amMcXKoJqL7S. \
              -d /home/aeinstein -s /bin/bash -m -g staff \
              aeinstein
      ```


Gestion des utilisateurs

Suppression d'un compte utilisateur

- Supprimer un compte manuellement :
  - Invalidation du compte :
    - remplacement du mot de passe chiffré par `*` ou `**No Login**`
    - remplacement du shell par `/bin/false`
  - Suppression du compte :
    - suppression des informations dans `/etc/passwd` (et dans `/etc/group` éventuellement)

Text

  - Suppression des fichiers appartenant au compte :
    - suppression du répertoire personnel
    - suppression de tous les fichiers de l'utilisateur :
      - mailbox
      - crontab
      - etc ...
- Supprimer un compte automatiquement (Linux) :
  - La commande `userdel` fait tout cela.
  - Exemple :
    - `userdel -r aeinstein`
      ```