# Introduction à Unix/Linux: Cours 4.

## Gestion des utilisateurs et des Processus.

Michaël Quisquater (Maître de Conférences,UVSQ)





## Première partie I

# Gestion des utilisateurs

# Notion d'utilisateur

- Pour qu'un utilisateur soit considéré comme tel, un compte utilisateur doit être créé sur le système. Il se voit attribué par l'administrateur un nom_de_compte (qui n'est pas encore attribué), UID et un groupe primaire d'appartenance. Il peut aussi faire partie d'autres groupes qui seront dits secondaires par rapport à cet utilisateur.
- La création d'un compte utilisateur nécessite de compléter les fichiers /etc/passwd et /etc/group.



Notion d'utilisateur et types
Les fichiers relatifs aux IUD's et GID's
Création de groupes et d'utilisateurs
Où se trouve le mot de passe ?
Changer d'utilisateur en connexion
Gestion des utilisateurs et des groupes en mode graphique
Permissions sur le système de fichiers
Root
Les utilisateurs non-privilégiés
Pseudo-utilisateurs
Comptes-service

# Notion d'utilisateur

- Unix considère un ensemble de groupes utilisateurs auxquels sont attribués un GID (group Id) lors de leur création.
- Chaque utilisateur doit appartenir à au moins un groupe.

# Attribution de l'UID et notion d'utilisateur

On distingue plusieurs types d'utilisateurs :

- root.
- Les utilisateurs non-privilegiés (conventionnels).
- Les pseudo-utilisateurs.
- Comptes-service

A partir des noyaux Linux 2.4, UID/GID est un entier positif de 32 bits. Afin d'assurer la compatibilité, il est conseillé de ne jamais aller au delà de 65534 (16 bits).





# Attribution des UID et GID conformément à la charte Débian : root

- L'UID/GID 0 a un rôle spécial : c'est le compte root (administrateur). Le nom peut être changé.

# Attribution des UID et GID conformément à la charte Débian

- Les UID's pour les utilisateurs non-privilégiés appartiennent souvent à une plage fixée (ex. entre 1000 et 29999 pour Débian/Ubuntu).





# Attribution des UID et GID conformément à la charte Débian

Les UID/GID's de 1 à 99 sont généralement réservés aux pseudo-utilisateurs non dynamiques. Ce sont utilisateurs (pseudo) qui n'ont pas besoin de tous les privilèges de root mais qui réalisent tout de même certaines tâches administratives. Ces pseudo-utilisateurs sont par exemple : daemon (UID/GID=1), bin (UID/GID=2), sys (UID/GID=3), games (UID/GID=5), man (UID/GID=6), lp (UID/GID=7), mail (UID/GID=8), news (UID/GID=9), uucp (UID/GID=10),... Ils disposent souvent de droits très restreints pour des raisons de sécurité.

# Attribution des UID et GID conformément à la charte Débian

- Les pseudo-utilisateurs :
  - **bin** : l'ancien propriétaire des commandes utilisateur (/bin). Aujourd'hui, c'est root qui les possède et ce compte n'est plus beaucoup utilisé.
  - **daemon :** compte utilisé pour les logiciels non privilégiés. Les fichiers et processus et qui n'ont pas besoin d'être détenus par root sont souvent détenus par daemon. Cependant, les démons sont de moins en moins utilisés au profit des comptes-services.



Notion d'utilisateur et types
Les fichiers relatifs aux IUD's et GID's
Création de groupes et d'utilisateurs
Où se trouve le mot de passe ?
Changer d'utilisateur en connexion
Gestion des utilisateurs et des groupes en mode graphique
Permissions sur le système de fichiers
Root
Les utilisateurs non-privilégiés
Pseudo-utilisateurs
Comptes-service

# Attribution des UID et GID conformément à la charte Débian

- Les pseudo-utilisateurs (suite) :
  - **nobody** : NFS (Network File System) utilise ce compte pour représenter le root pour le partage de fichiers. Afin que le root distant ait accès au partage de fichiers, son UID 0 doit être changé. Le compte nobody qui est utilisé pour cela. L'UID/GID 65534 est généralement réservé à cet utilisateur ne disposant pas ni de provilèges systèmes ni de fichiers.

# Attribution des UID et GID conformément à la charte Débian

- Les comptes-service ("service accounts") : ils sont créés quand le service est installé, p.ex : www-data, proxy, bind etc. Ces comptes ne doivent pas être modifiés et il n'est souvent pas possible de s'y loguer (in /etc/passwd, il y a /bin/false ce qui signifie qu'un Shell n'est pas disponible et donc qu'il n'est pas possible de s'y loguer).





# Les attributs d'un utilisateur UNIX

Les attributs d'un utilisateur UNIX sont spécifiés dans le fichier : /etc/passwd.

- Nom de compte utilisateur (login)
- Mot de passe
- Identifiant numérique (IUD)
- Groupe primaire d'appartenance (GID)
- Commentaire (appelée *gecos*) : le display manager utilise le nom du commentaire pour l'invite de login.
- Répertoire principal de l'utilisateur (home directory)
- Interpréteur de commandes (shell) par défaut

**Exemple :**
jean :x :1000 :1000 :commentaire :/home/jean/ :/bin/bash

# Les attributs d'un groupe utilisateur UNIX

Les attributs d'un groupe utilisateur UNIX sont spécifiés dans le fichier : /etc/group.

- Nom de groupe
- Mot de passe qui permet d'administrer le groupe (ajouter des membres au groupe etc)
- Identifiant numérique unique (GID)
- Liste de membres (vide ou contenir plusieurs nom de compte utilisateur séparés par un caractère ",")

**Exemple :** sport :x :jo :pierre,jean, jacques





# Création d'un groupe

- La création d'un groupe se fait simplement en invoquant la commande (en mode administrateur) :

  addgroup nom_groupe
- Cette commande ne fait qu'ajouter une ligne dans le fichier /etc/group.
- Ce fichier est destiné à définir les groupes secondaires des utilisateurs.
- De nombreuses options sont disponibles dans le man

**Remarque :** il existe un back-up de ce fichier : /etc/group-

# Suppression d'un groupe

- Pour retirer un groupe, il faut être en mode administrateur :

  delgroup groupe_utilisateur
- Notons que le groupe doit être vide pour le supprimer.





# Création et commande Adduser

- Créer un compte utilisateur consiste à :
  - Ajouter une ligne dans /etc/passwd et /etc/shadow si l'on souhaite que le mot de passe soit caché
  - Compléter ou ajouter une ligne dans /etc/group
  - Créer un répertoire principal pour l'utilisateur (/home/utilisateur)
  - Copier un certain nombre de fichiers standards depuis /etc/skel vers le répertoire principal de l'utilisateur et leur affecter les bonnes permissions.

# Création et commande Adduser

- Ces opérations sont généralement faites automatiquement au moyen de la commande (en mode adiministrateur) :

  adduser compte_utilisateur
- Cette commande se base sur le fichier /etc/adduser.conf qui réalise les opérations ci-dessus automatiquement et qui suit la charte Debian dans l'attribution des IUD's et GID's. Par défaut, la création d'un compteur_utilisateur crée un groupe primaire utilisateur ayant le même nom que le compte_utilisateur.





# Ajouter un utilisateur existant à un group existant

- Pour ajouter un utilisateur existant à un groupe existant (en mode administrateur) :

  adduser compte_utilisateur nom_groupe
- Il faut que l'utilisateur se reconnecte pour que le groupe soit pris en compte. De nombreuses options sont disponibles dans le man (notamment pour l'ajout d'utilisateurs système etc)

**Remarque :** il existe des back-up de ces fichiers : /etc/passwd-

# Suppression d'un utilisateur et commande deluser

- Pour retirer un utilisateur d'un groupe, il faut être en mode administrateur :

  deluser compte_utilisateur
- Notons que l'utilisateur que l'on supprime doit être déconnecté pour le supprimer.
- Le groupe primaire de l'utilisateur n'est pas supprimé.
- L'option –remove-home permet de supprimer le /home de l'utilisateur et la boite aux lettres.
- La commande se base sur le fichier deluser.conf





# useradd,usermod et groupadd,groupmod

- Les commandes useradd et groupadd permettent de créer des groupes et des utilisateurs en mode non-interactif
- Ces commandes disposent également de nombreuses options dans le man.

# chown et chgrp

- La modification de groupe propriétaire d'un fichier peut être réalisée par le propriétaire d'un fichier vers un groupe auquel il appartient, sans les droits root :

  chgrp nouveau_groupe fichier
- Pour toutes les autres opérations sur la propriété des fichiers et des groupes, il faut les droits administrateurs :

  chgrp nouveau_groupe fichier

  chown nouveau_prop fichier





# Mot de passe et attributs d'utilisateurs et groupe utilisateur UNIX

- A l'origine le hashé d'un mot de passe Unix étaient stocké dans /etc/passwd.
- Ce fichier doit être accessible en lecture pour tous les utilisateurs (pour trouver le compte_utilisateur à partir de l'UID p.exemple).
- Certains outils tels que John the Ripper ou Ophcrack (P. Oechslin) permettent de retrouver beaucoup de mots de passe à partir de leur hashé.

$\Rightarrow$ le hashé du mot passe doit être mis dans un fichier qui n'est pas accessible lecture.

# Mot de passe et attributs d'utilisateurs et groupe utilisateur UNIX (suite)

Trois situations peuvent se présenter (dans /etc/passwd et /etc/group)

- le mot de passe hashé dans /etc/passwd et /etc/group
- x : signifie que le mot de passe (s'il existe) se trouve dans /etc/shadow, resp. /etc/gshadow
- * : interdire l'usage d'un mot de passe



Notion d'utilisateur et types
Les fichiers relatifs aux IUD's et GID's
Création de groupes et d'utilisateurs
Où se trouve le mot de passe ?
Changer d'utilisateur en connexion
Gestion des utilisateurs et des groupes en mode graphique
Permissions sur le système de fichiers

# Mot de passe et attributs d'utilisateurs et groupe utilisateur UNIX (suite)

- Le fichier /etc/shadow (resp. gshadow) n'est uniquement accessible en écriture par root (et son groupe) et lisible par le groupe shadow (qui est vide par défaut).
- La commande passwd <compte_utilisateur >est chargée au cryptage du mot de passe dans /etc/shadow. Option : -d pour supprimer le mot de passe (plus nécessaire pour se connecter), -l (vérouiller le compte), -u (dévérouiller le compte).

**Remarque :** il existe des back-up de ces fichiers : /etc/shadow- et /etc/gshadow-

Michaël Quisquater (Maître de Conférences,UVSQ) Introduction à Unix/Linux: Cours 4.

# sudo et /etc/sudoers

- Les sudoers sont des utilisateurs à qui root a délégué/donné des droits administrateurs.
- Le fichier qui définit les droits des sudoers est /etc/sudoers.
- Il faut utiliser un éditeur de texte particulier pour modifier le fichier : visudo -s





# Changer d'utilisateur en connexion

- La commande **su** : - (pour changer vers root), nom_compte (pour utiliser le nom_compte).
- Graphiquement : il suffit de cliquer sur "déconnexion" et puis "changer d'utilisateur"

# Quelques commandes

| Commande | Fonction | compte |
|---|---|---|
| id | Affiche l'UID et les GID des groupes auxquels appartient un utilisateur | user |
| groups | Affiche les groupes auxquels appartient un utilisateur | user |
| w | Affiche des informations sur l'utilisateur connecté | user |
| finger | Affiche des informations sur l'utilisateur connecté | user |
| who | Affiche la liste des utilisateurs connectés (qui ont ouvert une session) | user |





# L'utilitaire users-admin

- L'utilitaire graphique pour la gestion des groupes est disponible dans le paquet : gnome-system-tools
- L'utilisation est relativement intuitive.

# Permissions sur le système de fichiers

- Les droits sur un fichier sont écrits dans l'inode.
- A chaque fichier est attribué un utilisateur propriétaire et un groupe propriétaire.
- Les droits sont attribués à trois catégories d'utilisateurs :
  - les droits associés à l'utilisateur propriétaire (u)
  - les droits associés au groupe propriétaire (g)
  - les droits associés aux autres (o)





# Permissions sur le système de fichiers (lecture)

A chacune de ces trois catégories, on associe trois droits "rwx" (- si on n'a pas le droit) :

- Droits en lecture :
  - Droit de lire un fichier régulier
  - Droit de lire les noms des fichiers d'un répertoire
  - Droit de récupérer des données sur un socket
  - Droit de récupérer le signal émis par un périphérique

# Permissions sur le système de fichiers (écriture)

- Droits en écriture :
  - Droit de modifier, renommer, supprimer un fichier régulier
  - Droit de renommer ou supprimer le contenu d'un répertoire
  - Droit d'émettre un signal vers un périphérique
  - Droit d'émettre du trafic réseau





# Permissions sur le système de fichiers (exécution)

- Droits en exécution :
  - Droit d'exécuter un fichier régulier
  - Droit d'entrer dans un répertoire (ce qui peut-être utile si on veut traverser un répertoire)

# Set-UID, Set-GID et sticky bit

- Le Set-IUD (Set-GID) permet d'exécuter des fichiers avec les droits du propriétaire du fichier. Cela permet en particulier de modifier des fichiers sur lesquels on n'a pas de droit en écriture (mais bien la personne qui met à disposition une fonction avec un Set-IUD ou SET-GID)
- Un fichier dispose d'un Set-UID (ou Set-GID) lorsque le droit en exécution x est remplacé par s.





# Set-UID, Set-GID et sticky bit (suite)

- Le sticky bit est utilisé pour manier de façon plus subtile les droits d'écriture d'un répertoire. Il empêche de supprimer un fichier d'un répertoire (même s'il est possible de le modifier s'il y a un droit en écriture sur le répertoire.)
- Le sticky bit est représenté par "t" à la place du droit en exécution des autres utilisateurs que le propriétaire ou le groupe propriétaire.

# Commandes pour modifier les droits d'un fichier

Avec la commande chmod (syntaxe explicite) :

- On définit la portée : u (utilisateur propriétaire), g (grope propriétaire), o (autre), a (tout le monde)
- On précise si on ajoute ou on retire le droit (+ ou -)
- On précise de quel droit il s'agit (r,w,x,s,t).

**Exemples** :

- chmod u+r toto.txt : on rajoute le droit en lecture à l'utilisateur propriétaire.
- chmod g+rwx toto.txt : on rajoute le droit en lecture/éciture/exécution au groupe propriétaire.
- chmod ug+rwx toto.txt : on rajoute le droit en lecture/éciture/exécution à l'utilisateur propriétaire et au groupe propriétaire.





# Commandes pour modifier les droits d'un fichier

Avec la commande chmod (syntaxe octale : 4 chiffres de 0 à 7) :

- Le chiffre des milliers est utilisé pour activer le Set-UID (4),Set-GID (2) et sticky bit (1).
- Le chiffre des centaines définit le droit de l'utilisateur propriétaire, celui des dizaines du groupe propriétaire et celui des unités celui des autres utilisateurs.
- A l'intérieur de chaque centaine, dizaine et unité, il suffit de convertir le chiffre en binaire pour définir les droits (1 pour droit et 0 pour absence de droit).

**Exemple :** chmod 744 toto.txt = Utilisation : rwx ; Groupe r– ; Autre ; r –.

Deuxième partie II

# Gestion des processus





## Notion de processus

- Un processus est un programme en cours d'exécution sur un processeur.
- En Linux, un seul code exécutable tourne sur un processeur à la fois. Si il y a plusieurs processeurs, il est possible de d'exécuter plusieurs processus simultanément.
- Le premier code exécutable mis en mémoire est le noyau muni de ses modules et pilotes (mode noyau).
- Le noyau possède une procédure permettant d'appeler d'autres codes afin qu'ils deviennent des processus en exécution.

# Notion de processus

- Ces processus vont à leur tour appeler d'autres processus (en invoquant le noyau) et ainsi que de suite.
- Cet arborescence de processus (qui bouge avec le temps car certains processus se terminent) constitue LE code exécutable (et exécuté) de la machine.
- Un processus peut être de type noyau ou de type utilisateur, suivant le mode du processeur.





# Processus, Groupe de Processus et Session

- Le premier processus créé par le noyau est "init" qui est identifié par un PID (Process IDentifier) qui vaut 1.
- Celui-ci initie par sa création également une session et un groupe de processus qui sont identifiés par un SID (Session IDentifier) et d'un PGID (Process Group IDentifier).
- Ce premier processus est appelé leader de la Session et du Groupe de processus dont le SID et le PGID prennent la valeur de son PID.
- Le processus init a donc PID=1, SID=1 et PGID=1.

# Processus, Groupe de Processus et Session (suite)

La création de l'arborescence de processus se déroule de façon récursive. Soit "A" un processus de l'arborescence. Alors :

- "A" est peut initier la création d'un processus léger (Thread) qui a un TID (thread IDentifier) identique au PID de "A" si un seul thread , sinon tous les threads issus d'un même processus ont même PID mais différents TID.
- "A" peut aussi initier un autre processus "B". "A" est dit processus parent du processus enfant *B*. Le PPID (Parent Process IDentifier) de "B" est le PID de "A". Le processus enfant "B" héritent du SID et du PGID de "A".



Notion de processus
Commandes relatives aux processus
Session et TTY
Définition
Hiérarchie des processus

# Processus, Groupe de Processus et Session (suite)

- "A" peut devenir leader d'un nouveau groupe ou bouger de groupe de processus qui doit être dans la même session. Dans les deux cas, il change de PGID.
- Si "A" n'est pas un leader de groupe de processus, il peut devenir leader d'une nouvelle session et donc devenir leader d'un nouveau groupe de processus. Il change de SID et PGID.

# Processus, Groupe de Processus et Session (suite)

**Remarques :**

- un groupe de processus n'a pas toujours de leader même si elle en a toujours un à sa création.
- un thread peut initier d'autres threads.
- A chaque processus est associé des "variables" d'états (priorité, état, utilisation mémoire, commande qui l'a lancé, ...). Ces valeurs sont stockées dans le repertoire *PID* du système de fichier virtuel /proc et sont utilisées et déterminées par le noyau.



Notion de processus
Commandes relatives aux processus
Session et TTY
Définition
Hiérarchie des processus

# Processus, Groupe de Processus et Session (suite)

- Concrètement, les nouveaux noyaux linux n'implémentent que les processus et émulent les threads comme des processus avec un TID=PID et qui peuvent communiquer entre-eux.

# Lister et modifier les priorioritées des processus

| Commande | Action |
|---|---|
| ps | Affiche les informations d'une sélection de processus actifs (PID,TTY). L'option -axjf donne beaucoup plus de processus et d'information. |
| pidof <commande > | Donne le PID de la <commande >en exécution |
| top (ou htop ) | Donne de l'information en temps réel des processus et thread. |
| nice -n Number <commande > | Attribue la priorité Nombre à la <commande >(Nombre=-20 : le plus favorable à 19 le moins favorable). |
| renice -n <nombre >PID | Attribue la priorité <nombre >au processus indexé par PID |





# Envoyer un signal à un processus

- Il est possible d'envoyer un signal à un processus
- Le signal SIGKILL permet de tuer un processus. Le signal SIGSTOP permet d'arrêter le processus et SIGCONT permet de le reprendre. SIGTERM permet de demander au shell le nettoyage et la fermeture de ses fichiers. SIGTERM peut être ignoré par le shell. SIGTERM est souvent suivi par un signal SIGKILL qui lui force l'extinction du processus.

# Envoyer un signal à un processus

- Un processus ne peut être tué que par l'utilisateur qui l'a lancé ou root
- On envoie un signal à un processus en mentionnant son PID ou le nom de l'application liée à ce PID. Il faut aussi spécifier le signal que l'on veut envoyer.





# Tuer un processus

| Commande | Action |
|---|---|
| kill -l | Affiche les différents signaux que l'on peut envoyer à un processus. |
| kill [option] PID | Permet d'envoyer un signal défini par l'option au processus PID. L'option -SIGKILL permet de tuer un processus. Si on remplace PID=-1 envoie le signal à tous les processus sauf init (PID=1) et le processus kill lui-même (une invite de session sera redémarrée automatiquement). |
| killall <nom> | Similaire à kill si ce n'est que l'on donne le <nom>du processus au lieu du PID |
| xkill | Permet de tuer un processus qui est lié à une fenêtre sans avoir à rechercher son PID. IL suffit de cliquer sur la fenêtre correspond au processus à tuer après avoir lancé la commande xkill. |

# Session et TTY

- Le système UNIX a été conçu pour que de nombreux utilisateurs se connectent à distance sur la même machine.
- Le port utilisé était RS-232 qui est un port de communication série.
- Le noyau établit un lien entre le port série (anciennement utilisé pour la communication) et les fichiers spéciaux "/dev/ttyx" (où x est remplacé par 1,2 etc).





# Session et TTY (suite)

- Aujourd'hui : port série RS-232 n'est plus utilisé
- Le mécanisme a été conservé et le fichier spécial /dev/ttyx sert à envoyer ce qui doit être affiché à l'écran et tapé au clavier (+mouvement de la souris).
- Afin de pouvoir utiliser plusieurs terminaux quand on n'a pas de fenêtre (en mode console). Au démarrage de la machine, plusieurs sessions (associées au système ; l'UID=0) sont lancées, chacune ayant accès à un "/dev/ttyx" différent. L'utilisateur peut avoir accès à ces sessions par la combinaison de touches :
  - Sous Mac : Pomme Fx
  - Sous PC normal : Crtl + Alt +Fx
  - Sous VirtualBox : Host (Pomme pour Mac et Crtl de droite sous Dos) +Fx

# Session et TTY (suite)

- Sur un système Debian, l'utilisateur a accès aux sessions correspondant aux devices /dev/ttyx (avec x=1..7). La session correspondant au /dev/tty7 est réservée à l'interface graphique, les autres au shell en mode console.
- Lorsque l'utilisateur se logue sur la session associée au device /dev/ttyx, il crée un groupe de processus au sein de cette session dont l'UID est celui de l'utilisateur.



Notion de processus
Commandes relatives aux processus
Session et TTY
Notion de TTY
Contrôle des TTY's
Contrôle de Jobs

# Session et TTY (suite)

- Afin de disposer de plusieurs Shell en mode graphique, l'émulateur de terminaux va reprendre le même principe en associant des fichiers spéciaux /dev/pts/0 et /dev/pts/1 (pour root) et un /dev/pts/ (avec $x \geq 2$) par Shell lancé par un utilisateur.
- La commande **tty** donne le fichier spécial associé au terminal courant.

# Groupe de processus et Contrôle de Jobs

- Un job est un concept utilisé par le shell.
- Tous les programmes lancés à partir du shell que l'on ne détache pas (ex. les démons) est un job.
- On peut lancer un job en avant-plan ou en arrière plan.
  - Un job en avant-plan est un processus qui peut lire et écrire sur le terminal. En particulier, il apprtient à un groupe de processus ayant la particularité des signaux SIGINT provenant du terminal. Ceux-ci permettent de communiquer avec celui-ci via le terminal. Cela signifie en particulier que l'invite de commande n'est pas disponible à l'utilisateur pour lancer une autre commande.





# Groupe de processus et Contrôle de Jobs (suite)

- Les autres jobs, dits "en arrière plan" font partie d'un autre groupe de processus (PGID différent). Et leur tentative d'écrire sur le clavier engendre bien souvent un signal une supension du processus par le shell.
- Il est possible de faire passer un job de l'avant-plan à l'arrière plan et inversement.

# Groupe de processus et Contrôle de Jobs (suite)

| Commande | Action |
|---|---|
| jobs | Liste les jobs en exécution en donnant les numéros de job entre crochets et le PID du processus. |
| <commande>& | Lance la <commande>en arrière-plan |
| fg %<nb_job> | Place le processus correspondant à la <nb_job>en avant-plan (foreground) |
| CTRL-Z | Stoppe (sans le tuer) le job en avant-plan et donne accès au clavier. Il faut ensuite utiliser les commandes fg, bg ou kill pour le processus stoppé |
| bg %<nb_job> | Place le processus correspondant à la <nb_job>en avant-plan (foreground) |
| kill %<nb_job> | Tue le processus correspondant au <nb_job>. |