

# Licence Professionnelle Système d'exploitation N°1 : Introduction

- Fichiers d'environnement

- Scripts

# Linux

- Pré-requis :
  - RAN : Être capable d'exploiter un poste sous Linux
  - Commandes, Système de fichiers, Filtres
- Supports : <http://moodle.univ-lille1.fr> (LP CGIR - RAN clé : *lpcgir*)
  - *proxy* : *cache-etud.univ-lille1.fr:3128*
  - *Inscription* : *http://ent.univ-lille.fr*
- Objectif : Scripting bash
  - Être capable de :
    - Créer une bibliothèque de scripts permettant d'automatiser certaines tâches d'administration système
    - Débugger, faire évoluer des scripts existants

# Gestion des comptes

- Commandes de base **useradd groupadd userdel groupdel...**
- Commandes interactives pour la création de comptes :
  - **adduser, addgroup, deluser, delgroup**  
utilisables sans interaction en spécifiant des options
- Commandes de traitement par lots :
  - Données depuis l'entrée standard ou un fichier  
format de ligne (username:passwd:uid:gid:gecos:homedir:shell)
  - **newusers** : créer plusieurs utilisateurs
  - **chpasswd** : modifier les mots de passe pour un ensemble d'utilisateurs
    - cryptage par défaut PAM (sinon option *-c sha-512*)
- Autres commandes
  - **crypt** : crypter une chaîne
  - **pwck, gpck** contrôler l'intégrité des fichiers passwd et group

# Valeurs par défaut

---

- Le comportement des commandes de création d'utilisateur est régi par des fichiers d'environnement qui renseignent des variables d'environnement :
  - adduser => fichier **/etc/adduser.conf**
  - useradd => fichier **/etc/login.defs**
- Exemples de variables pour **adduser.conf** ( *équivalents pour login.defs*)
  - **FIRST\_UID, LAST\_UID, FIRST\_GID, LAST\_GID** : N° automatiques
  - **SKEL** : choix des fichiers d'environnement
  - **DHOME** : répertoire d'accueil des répertoires de connexion
  - **NAME\_REGEX** : format des **identifiants** acceptés
  - ...

(format des mots de passe : PAM => /etc/pam.d/passwd)

# Fichiers silencieux

---

- Le répertoire de connexion contient des fichiers de **personnalisation de l'environnement de travail**
  - Ils sont appelés **fichiers silencieux**
  - Leur nom commence par un « . »
    - *ne sont pas listés lors d'un ls simple => **ls -a***
- Fichiers pour la configuration de Linux
- Fichiers/répertoires silencieux pour la configuration des applications
  - configuration de l'environnement graphique : .gnome
  - navigateur firefox : .mozilla
  - poubelle de l'utilisateur : .local/share/Trash
  - ...

# Les fichiers d'environnement bash

- Lors de la connexion bash interprète :

**/etc/profile** (commun)

- Puis à chaque **nouvel interpréteur bash** lancé (donc au **login** aussi) :

1) **/etc/bash.bashrc** (commun)

+ lors la connexion uniquement, le 1<sup>er</sup> des fichiers trouvés parmi:  
**~/.bash\_profile** ou **~/.bash\_login** ou **~/.profile**

2) Puis **~/.bashrc**

- A la déconnexion le shell interprète **~/.bash\_logout**

*Debian utilise par défaut .profile .bashrc et .bash\_logout*

*Les fichiers d'environnement de root sont dans /root*

*Le comportement pour root diffère un peu*

# Fichiers modèles

---

- Lors de la création d'un utilisateur des fichiers silencieux "modèles" sont copiés dans son répertoire de connexion
  - ces modèles se trouvent dans le répertoire **/etc/skel**
    - **.profile, .bashrc , .bash\_logout**
  - Ils sont utilisés par défaut lorsqu'un utilisateur est créé par la **useradd**
- Il est possible par exemple de définir un ensemble d'**alias** de commandes qui sera commun à tous les utilisateurs :
 

```
alias lister="ls -Ali"    # listage long + inode + silencieux (sans . et ..)
```

# Changement d'identité

- **su** changer d'identité en fournissant le **mot de passe de l'utilisateur cible**
  - *retour à l'identité d'origine par déconnexion (Ctrl+D ou exit)*
  - **su** : sans utilisateur spécifié vous devenez **root**
  - **su user** : vous devenez l'utilisateur **user**
  - **su - user** : force la **relecture du fichier d'environnement** *bash\_profile*
  - **su -c commande user** : exécute **une seule commande** sous l'identité spécifiée
- **sg** permet de changer de groupe
- **sudo** permet d'exécuter une commande avec les droits d'un autre utilisateur  
*par défaut root, sinon préciser l'option -u user*
  - réclame **le mot de passe du demandeur** *reste valide pendant 15mn*  
*sudo cat /etc/shadow (réclame le mot de passe)*  
*sudo cat /etc/gshadow (juste après, ne réclamera plus le mot de passe)*



# /etc/sudoers

---

- L'usage de sudo selon le contenu de **/etc/sudoers**
  - *sudo -l* permet de lister les opérations possibles pour l'utilisateur
- Chaque ligne du fichier spécifie quel(s) utilisateur(s), peut utiliser quelle(s) commande(s), depuis quelle(s) machine(s) :

**utilisateur      machine = (identite) commande**

**ALL** désigne, selon le contexte : tous les utilisateurs, toutes machines, ou toutes commandes

Exemples :

*user* ALL=(root) /sbin/halt => Autorise *user* à arrêter la machine

ALL = (ALL) ALL => autorise tout utilisateur, depuis n'importe quelle machine à prendre n'importe quelle identité, pour exécuter n'importe quelle commande

# Modifier /etc/sudoers

- Commencer par créer des alias pour regrouper les éléments  
*liste des directives => man 5 sudoers*

- ▶ Exemple :

**User\_Alias** STOPPEURS=boul,bill

**Host\_Alias** PCMAISON=192.168.0.2, 192.168.0.3

**Cmnd\_Alias** STOP=/sbin/halt

STOPPEURS PCMAISON=(root):STOP

#boul et bill peuvent exécuter /sbin/halt

- ▶ **Attention** : protéger les **redirections**

- ▶ sudo -u user cat ~user/file # sans redirection pas de protection

- ▶ sudo -u user **bash -c** "cat ~user/file > text.txt" # redirection lancer un shell pour l'exec

- ▶ sudo -u user **-s** # changer d'identite équivalent su  
cat ~user/file>text.txt  
Ctrl+D

# Script shell

---

- shell : langage pour l'écriture de SCRIPTS d'administration
- Indispensable à la fonction d'administrateur système:
  - Automatisation des tâches répétitives ou fastidieuses
  - Exécution de tâches différées, périodiques...
- Un shell-script est un fichier texte contenant une liste de commandes shell  
*l'extension (sh) est facultative, mais permet une identification rapide*
- Le shell permet :
  - Accès aux variables d'environnement
  - Accès aux commandes du shell
  - Structuration de l'exécution : (*commandes internes*)
    - Conditions : **if**, **case**
    - Boucles : **for**, **while**
    - Fonctions : **function**

# Programmation

---

- Décomposition du traitement à réaliser :
  - création d'un ensemble de scripts élémentaires
  - bibliothèque de scripts, utilisés comme de nouvelles commandes
  
- Règles pour faciliter la maintenance, la réutilisation des scripts :
  - en-tête décrivant la fonction et la manière d'utiliser le script,
  - commenter les différentes sections,
  - robustesse, vérification des arguments, existence des ressources...
  - suivi de l'exécution des commandes => trace, messages d'erreur...
  
- A utiliser pour l'administration système quand :
  - peu de données structurées : tableaux...
  - peu d'arithmétique
  - sécurité non cruciale
  - peu d'accès aux ressources

# Fichier script

- La 1<sup>ère</sup> ligne du script spécifie le shell utilisé (*sha-bang*) : **#!/bin/bash**  
*la 1<sup>ère</sup> ligne spécifie le programme pour interpréter les commandes*
- Commentaires : le **#** dans une ligne **commente** le reste de la ligne
  - s'il n'est pas dans le sha-bang,
  - s'il n'est pas protégé par ' ou " ou \ ou {}...
- Fin d'exécution : commande **exit nn**
  - Toujours préciser un code de retour :
    - valeur retournée en fin d'exécution
    - 0 : ok,
    - 1 .. 255 : code d'erreur
    - **echo \$ ?** : affiche le code de retour de la dernière commande exécutée

# Exécution

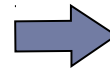
- Edition du script *MonScript.sh* :

```
#!/bin/bash
date +%d/%m
```

- Exécution de *MonScript.sh* :

```
source MonScript.sh #shell courant
```

```
bash MonScript.sh #nouveau shell
```



14/09

- Rendre exécutable :

```
chmod a+rx MonScript.sh
```

```
./MonScript.sh
```

*le droit r est indispensable en plus du x  
appliqué ici pour tous UGO*

*indispensable de faire précéder d'un chemin pour l'invocation  
(./ si script local)*

- Un script placé dans */usr/local/bin* , sera accessible depuis toute l'arborescence


# echo

- La commande **echo** permet d'envoyer des messages vers la sortie standard
  - **echo "message à afficher"** # guillemets facultatifs mais conseillés
  - options
    - **-n** : pas de retour à la ligne
    - **-e** : interprétation des séquences d'échappement
      - **\n** : retour à la ligne, **\t** : tabulation, **\b** : retour arrière...
- Remarques :
  - **echo \*** : affiche les noms du répertoire courant
  - **echo {1..5}** : affiche 1 2 3 4 5
  - **echo {a..d}** : affiche a b c d
  - la commande **clear** efface l'écran

# Séquences d'échappement

- Couleur
  - modification des couleurs en console : `\033[nnm`
    - *nn* entre 30 et 37 couleur de l'encre
    - *nn* entre 40 et 47 couleur du fond
  
- Caractères spéciaux
  - accès aux caractères unicode UTF-8, en hexadécimal : `\xnn\xnn\xnn`
    - exemple : E296B6 => `\xE2\x96\xB6`
  
- Exemple : texte en rouge entre *triangles couchés* E296B6 et E29780
 

```
echo -en "\033[31m\xE2\x97\x80"
echo -n "Bonjour"
echo -en "\xE2\x96\xB6\033[0m"
```





# printf

- Ecriture formatée avec la commande **printf**
  - Contrôle l'affichage comme la fonction des langages de programmation, *mais sans les parenthèses*
  - repose sur des séquences prédéfinies
    - **%s** (chaîne),
    - **%d** (décimal),
    - **%f** (flottant)...
    - protection du "%" : **%%**
- **printf** "\ta%sgsh %2.2f\n" "rr" "3,1416" => affiche : arrgh 3,14
  - écriture avec parenthèses possibles
  - affiche la suite si plus long que le modèle