



Administration Linux

Clément Weill



107

Tâches d'administration

107.1 Gestion des comptes utilisateurs et des groupes ainsi que des fichiers systèmes concernés

107.2 Automatisation des tâches d'administration par la planification des travaux

107.3 Paramètres régionaux et langues



107

Tâches d'administration

107.1.a Gestion des comptes utilisateurs et des groupes ainsi que des fichiers systèmes concernés



Ajouter un Compte Utilisateur

Pour ajouter un utilisateur, il faut utiliser la commande `useradd` avec les privilèges root.

Exemple:

```
# useradd michael
```

Lorsque que l'on exécute la commande `useradd`, les informations propres à l'utilisateur et au groupe sont mis à jour dans les bases de données groupe et mot de passe.

Si cela est spécifié, un répertoire maison est créé. Un groupe au nom de l'utilisateur est aussi créé.



Ajouter un Compte Utilisateur

Une fois le nouvel utilisateur créé, on peut lui attribuer un mot de passe avec la commande `passwd`.

On peut vérifier son User ID (*UID*) et son Group ID (*GID*) en utilisant les commandes `id` et `groups`.

Exemple:

```
# id michael
```

```
uid=1000(michael) gid=100(michael) groups=100(michael)
```

```
# groups michael
```

```
michael : michael
```



Ajouter un Compte Utilisateur

Remarque, chaque utilisateur peut vérifier son User ID (*UID*) et son Group ID (*GID*) en utilisant les commandes `id` et `groups`, ou changer son mot de passe avec `passwd`, mais seul un utilisateur avec les privilèges root peut modifier le mot de passe d'un autre utilisateur.

Les options les plus importantes de `useradd` sont:

- `-c` Crée un nouvel utilisateur avec des commentaires personnalisés.
- `-d` Crée un nouvel utilisateur avec un répertoire maison personnalisé.
- `-e` Crée un nouvel utilisateur avec une date spécifique où il sera supprimé.
- `-f` Crée un nouvel utilisateur en fixant un nombre de jours de validité du mot de passe, après quoi l'utilisateur devra le mettre à jour (ou perdre son compte).



Ajouter un Compte Utilisateur

- `-g` Crée un nouvel utilisateur avec un GID spécifique.
- `-G` Crée un nouvel utilisateur en l'ajoutant à multiple groupes secondaires.
- `-k` Crée un nouvel utilisateur en copiant les fichiers squelettes d'un répertoire spécifique (cette option ne fonctionne qu'en conjonction avec les options `-m` ou `--create-home`).
- `-m` Crée un nouvel utilisateur ainsi que son répertoire maison, s'il n'existe pas déjà.
- `-M` Crée un nouvel utilisateur sans répertoire maison.
- `-s` Crée un nouvel utilisateur avec un login shell spécifique.
- `-u` Crée un nouvel utilisateur avec un UID spécifique.

Essayez `man useradd` pour une liste complète des options



Modifier un Compte Utilisateur

Il est parfois nécessaire de modifier les attributs existant d'un utilisateur, comme le nom au login, le login shell, la date de péremption du mot de passe, etc. Pour cela on utilise la commande `usermod`.

Exemple:

```
# usermod -s /bin/tcsh micheal
```

```
# usermod -c "Micheal user account" micheal
```

Tout comme `useradd`, la commande `usermod` nécessite des privilèges root.

Dans l'exemple ci-dessus, on change le login shell de l'utilisateur `micheal` et on y ajoute un court descriptif.



Modifier un Compte Utilisateur

Les options les plus importantes de `usermod` sont:

- `-c` Ajoute des commentaires à l'utilisateur.
- `-d` Change le répertoire maison de l'utilisateur. En conjonction avec l'option `-m`, déplace le contenu de l'ancien répertoire dans le nouveau.
- `-e` Fixe une date de péremption pour l'utilisateur.
- `-f` Fixe un nombre de jours de validité du mot de passe, après quoi l'utilisateur devra le mettre à jour (ou perdre son compte).
- `-g` Change le groupe primaire de l'utilisateur, le groupe doit exister.
- `-G` Ajoute l'utilisateur à des groupes secondaires spécifiés, séparés par une virgule et sans espaces. Si utilisé seul, alors cela retire l'utilisateur de tout ses groupes, et si utilisé avec `-a` cela ajoute les groupes à ceux qui existe déjà.
- `-l` Change le nom de login de l'utilisateur.



Modifier un Compte Utilisateur

- ◆ `-L` Bloque l'utilisateur. Ca rajoute un point d'exclamation devant le mot de passe encrypté dans le fichier `/etc/shadow` ce qui bloque l'accès pour cet utilisateur.
- ◆ `-s` Change le login shell pour l'utilisateur.
- ◆ `-u` Change le UID pour l'utilisateur
- ◆ `-U` Débloque l'utilisateur. Ca enlève le point d'exclamation devant le mot de passe dans le fichier `/etc/shadow`.

Attention, quand on change le nom de login d'un utilisateur, il faut probablement aussi renommer le répertoire maison, ainsi que l'ensemble des objets lié à l'utilisateur. De même, quand on change le UID, il faut mettre à jour les droits de propriétés des fichiers et répertoires en dehors du répertoires maison.



Supprimer un Compte Utilisateur

Pour supprimer un compte utilisateur, il faut utiliser la commande `userdel`.

En particulier, cette commande met à jour les informations contenu dans la base de donnée des comptes, et supprime toutes les entrées liées à cet utilisateur.

L'option `-r` permet de supprimer le répertoire maison de l'utilisateur ainsi que tout ce qu'il contient. Les fichiers situées ailleurs devront être supprimé manuellement.

Exemple:

```
# userdel -r micheal
```

Tout comme `useradd` et `usermod`, il faut avoir des privilèges root pour supprimer un utilisateur.



Ajouter, Modifier et Supprimer des Groupes

De manière similaire à la gestion des utilisateurs, pour ajouter, modifier et supprimer des groupes, il faut utiliser les commandes `groupadd`, `groupmod`, et `groupdel`, avec des privilèges root.

Par exemple, pour créer un groupe nommé `developer`, on utilise la commande suivante:

```
# groupadd -g 1090 developer
```

L'option `-g` permettant de spécifier le GID du nouveau groupe.

Attention, quand on ajoute un nouvel utilisateur, les groupes primaires et secondaires auxquels il appartient *doivent* exister avant d'utiliser la commande `useradd`.



Ajouter, Modifier et Supprimer des Groupes

Si on souhaite renommer le groupe `developer` à `web-developer`, et changer son GID, on utilisera la commande suivante:

```
# groupmod -n web-developer -g 1050 developer
```

Remarque, si on change le GID d'un groupe, il faut aussi changer le GID de l'ensemble des fichiers et des répertoires qui doivent continuer d'appartenir à ce groupe.

Enfin, si on veut supprimer le groupe, il suffit d'utiliser :

```
# groupdel web-developer
```

On ne peut pas supprimer un groupe si c'est le groupe primaire d'un utilisateur. Il faut d'abord supprimer l'utilisateur, puis le groupe.



Le Répertoire Squelette

Lorsque l'on crée un nouvel utilisateur, ainsi que son répertoire maison, le nouveau répertoire maison est rempli en utilisant le contenu du répertoire squelette, par défaut situé à `/etc/skel`.

L'idée étant qu'un administrateur système peut personnaliser le répertoire squelette, et cela s'appliquera à l'ensemble des futurs utilisateurs.

Essayez : `ls -al /etc/skel`



Le Fichier `/etc/login.defs`

Linux utilise le fichier `/etc/login.defs` pour spécifier les paramètres de configurations qui contrôle la création des utilisateur et des groupes.

De plus, l'ensemble des commandes précédentes utilise les valeurs par défaut défini dans le fichier.

Les directives les plus importantes sont:

- `UID MIN` et `UID MAX`
 - le champs de User ID qui peuvent être assigné à un nouvel utilisateur ordinaire.
- `GID MIN` et `GID MAX`
 - le champs de Group ID qui peuvent être assigné à un nouveau groupe ordinaire.



Le Fichier `/etc/login.defs`

CREATE HOME

- Indique si un répertoire maison devrait être créé par défaut pour les nouveau utilisateur

USERGROUPS ENAB

- Indique si le système doit créer un nouveau groupe pour chaque nouvel utilisateur, et si supprimer l'utilisateur devrait supprimer le groupe du même nom si celui-ci est vide.

MAIL DIR

- Le repertoire où sont stocké les mails (*mail spool*).

PASS MAX DAYS

- Le nombre de jours maximum qu'un mot de passe peut être utiliser.

PASS MIN DAYS

- Le nombre de jours minimum autorisés entre deux changement de mot de passe.



Le Fichier `/etc/login.defs`



`PASS MIN LEN`

- La longueur minimum autorisée pour un mot de passe.



`PASS WARN AGE`

- Le nombre de jours d'alertes avant la péremption d'un mot de passe.



La Commande `passwd`

Cette commande est principalement utilisé pour changer le mot de passe d'un utilisateur.

N'importe quel utilisateur peut modifier son mot de passe, mais seul root peut modifier le mot de passe de n'importe quel utilisateur.

Ceci car la commande `passwd` a son bit SUID d'activé (il y a un `s` à la place du drapeau d'exécution pour le propriétaire), c'est à dire qu'il est exécuté avec les privilèges du propriétaires du fichier, ici root.

Exemple:

```
# ls -l /usr/bin/passwd
```

```
-rwsr-xr-x 1 root root 42096 mag 17 2015 /usr/bin/passwd
```



La Commande `passwd`

Les options les plus importantes de `passwd` sont:

- `-d` Supprime le mot de passe d'un utilisateur (ce qui bloque l'utilisateur).
- `-e` Force l'utilisateur à changer de mot de passe.
- `-i` Fixe un nombre de jours de d'inactivité après la péremption d'un mot de passe pendant lequel l'utilisateur doit mettre à jour son mot de passe (ou perdre son compte).
- `-l` Bloque un utilisateur (ajoute un point d'exclamation devant le mot de passe encrypté dans le fichier `/etc/shadow`).
- `-n` Définit le temps de vie minimum d'un mot de passe.
- `-S` Sort les informations à propos du statut du mot de passe d'un utilisateur.



La Commande `passwd`

- `-u` Débloque un utilisateur (retire le point d'exclamation devant le mot de passe encrypté dans le fichier `/etc/shadow`).
- `-x` Définit le temps de vie maximum d'un mot de passe.
- `-w` Définit le nombre de jours d'alertes avant la péremption d'un mot de passe, durant lesquels l'utilisateur est informé qu'il doit changer de mot de passe.

Remarque: Un groupe peut aussi avoir un mot de passe, défini avec la commande `gpasswd`. Un utilisateur peut temporairement rejoindre le groupe s'il en connaît le mot de passe avec la commande `newgrp`. De plus, `gpasswd` est aussi utilisé pour ajouter et retirer des utilisateurs d'un groupe, ainsi qu'en définir les administrateurs.



La Commande `chage`

La commande `chage` (*change age*) est utilisé pour changé les information relative au vieillissement d'un mot de passe.

La commande est réservé à l'utilisateur root, sauf avec l'option `-l` qui permet d'avoir les informations concernant le vieillissement de son propre mot de passe.

Les autres options de la commande `chage` sont:

- `-d` Fixe la date où le mot de passe a été changé la dernière fois.
- `-E` Fixe la date d'expiration de l'utilisateur.
- `-I` Fixe le nombre de jours d'inactivité, après qu'un mot de passe ait dépassé la date de fin de validité, avant que le compte ne soit bloqué.



La Commande `chage`

- `-m` Fixe le temps de vie minimum d'un mot de passe pour un utilisateur.
- `-M` Fixe le temps de vie maximum d'un mot de passe pour un utilisateur.
- `-W` Fixe le nombre de jours d'avertissement avant la péremption d'un mot de passe, pendant lesquels l'utilisateur est informé qu'il doit modifier son mot de passe.



Exercices





107

Tâches d'administration

107.1.b Gestion des comptes utilisateurs et des groupes ainsi que des fichiers systèmes concernés



Introduction

Les commandes vu dans le chapitre précédent, ainsi que les applications graphiques fournis par les différentes distributions qui effectuent les mêmes tâches, mettent à jour une série de fichiers qui stockent les informations sur les utilisateurs et les groupes.

Ces fichiers sont situés dans `/etc/`, et sont:

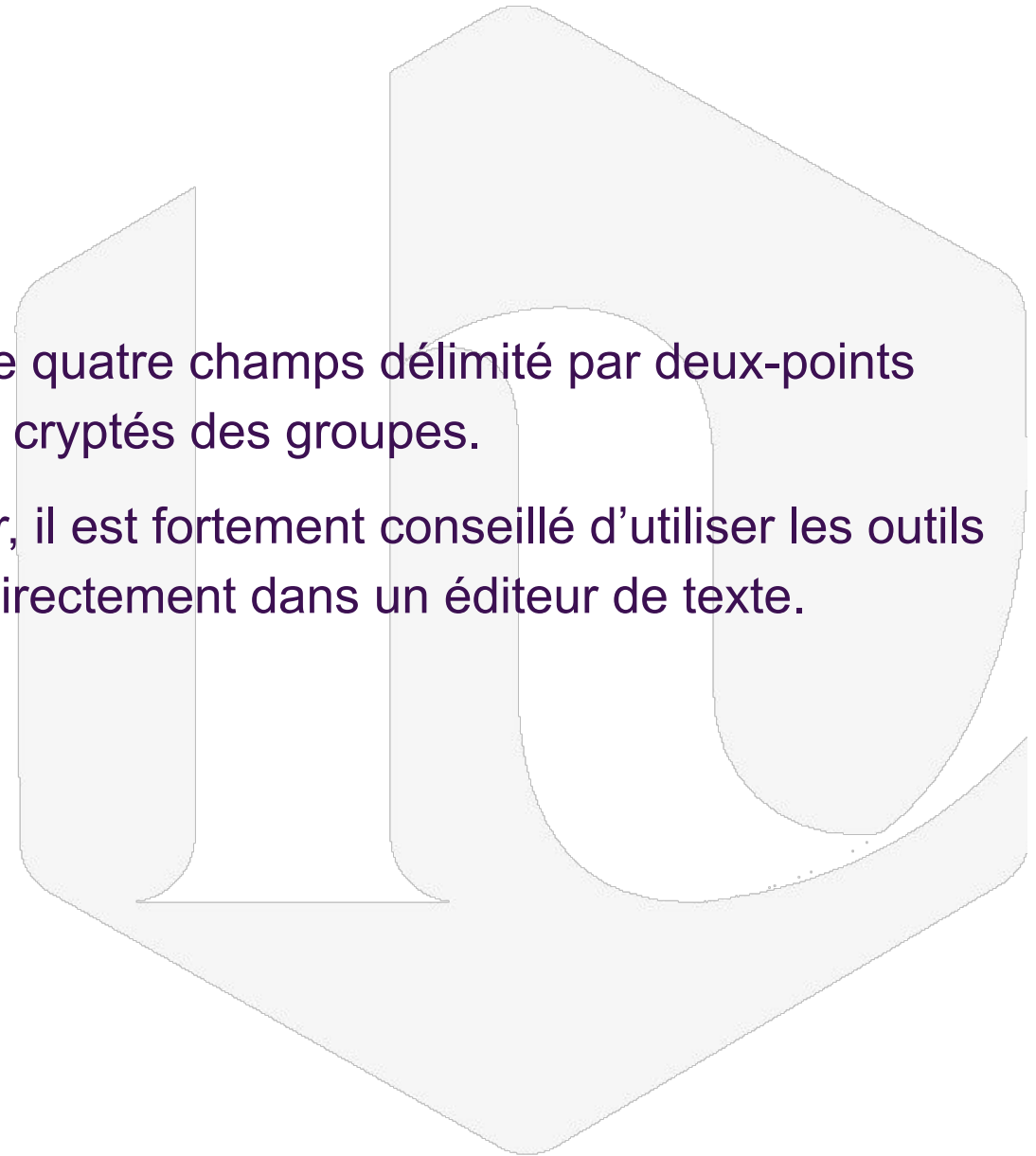
- `/etc/passwd` Un fichier de sept champs délimité par deux-points contenant les informations basiques sur les utilisateurs.
- `/etc/group` Un fichier de quatre champs délimité par deux-points contenant les informations basiques sur les groupes.
- `/etc/shadow` Un fichier de neuf champs délimité par deux-points contenant les mots de passes cryptés des utilisateurs.



Introduction

- `/etc/gshadow` Un fichier de quatre champs délimité par deux-points contenant les mot de passes cryptés des groupes.

Même si ces fichiers sont en clair, il est fortement conseillé d'utiliser les outils fournis pour les modifier et non directement dans un éditeur de texte.





`/etc/passwd`

Ce fichier texte lisible de tous, contient une liste d'utilisateur, chacun sur une ligne séparé.

Chaque ligne comprend sept champs, séparé par deux-points, qui sont:

◆ **Username**

- Le nom utilisé quand l'utilisateur se connecte au système.

◆ **Password**

- Le mot de passe crypté (ou un `x` si les mots de passes shadow sont utilisés).

◆ **User ID (UID)**

- Le nombre de l'ID assigné à l'utilisateur dans le système.

◆ **Group ID (GID)**

- Le nombre de l'ID du groupe primaire de l'utilisateur dans le système.



`/etc/passwd`

◆ **GECOS**

- Un champs optionnel qui contient des information supplémentaires sur l'utilisateur (comme le nom complet). Ce champs peut recevoir plusieurs entrées séparées par des virgules.

◆ **Home Directory**

- Le chemin absolue du répertoire maison de l'utilisateur.

◆ **Shell**

- Le chemin absolue du programme a lancé automatiquement quand l'utilisateur se connecte au système (souvent un shell interactif comme `/bin/bash`).



/etc/group

Ce fichier texte lisible de tous, contient une liste des groupes, chacun sur une ligne séparé.

Chaque ligne comprend quatre champs, séparé par deux-points, qui sont:

◆ **Group Name**

- Le nom du groupe.

◆ **Group Password**

- Le mot de passe crypté du groupe (ou un `x` si les mot de passes shadow sont utilisés).

◆ **Group ID (GID)**

- Le nombre de l'ID assigné au groupe dans le système.

◆ **Member List**

- Une liste d'utilisateurs, séparé par des virgules, qui appartient au groupe, excepté les utilisateurs dont c'est le groupe primaire.



/etc/shadow

Ce fichier texte accessible qu'à root, contient une liste de mot de passes cryptés, chacun sur une ligne séparé.

Chaque ligne comprend neuf champs, séparé par deux-points, qui sont:

◆ **Username**

- Le nom utilisé par l'utilisateur quand il se connecte au système.

◆ **Encrypted Password**

- Le mot de passe crypté de l'utilisateur (si il commence par **!**, c'est que l'utilisateur est bloqué).

◆ **Date of Last Changed Password**

- La date du dernier changement de mot de passe, en jours depuis le 01/01/1970 (une valeur de 0 implique que l'utilisateur doit changer son mot de passe à sa prochaine connection).



/etc/shadow

◆ **Minimum Password Age**

- Le nombre de jours minimum qui doivent s'écouler avant que l'utilisateur puisse changer de mot de passe à nouveau.

◆ **Maximum Password Age**

- Le nombre de jours maximum qui doivent s'écouler avant qu'un changement de mot de passe soit requis.

◆ **Password Warning Period**

- Le nombre de jours d'avertissement avant la péremption d'un mot de passe, pendant lesquels l'utilisateur est informé qu'il doit modifier son mot de passe.



/etc/shadow

◆ Password Inactivity Period

- Le nombre de jours d'inactivité, après qu'un mot de passe ait dépassé la date de fin de validité pendant lesquels l'utilisateur doit mettre à jour son mot de passe, après quoi le compte est bloqué.

◆ Account Expiration Date

- La date exprimée, en jours depuis 01/01/1970, à laquelle l'utilisateur sera bloqué. Un champs vide indique qu'il n'y a pas de date de péremption.

◆ A Reserved Field

- Un champs réservé pour une application dans le future.



`/etc/gshadow`

Ce fichier texte accessible qu'à root, contient une liste de mot de passes cryptés, chacun sur une ligne séparé.

Chaque ligne comprend quatre champs, séparé par deux-points, qui sont:

◆ **Group Name**

- Le nom du groupe.

◆ **Encrypted Password**

- Le mot de passe crypté du groupe (si il commence par `!`, c'est qu'aucun utilisateur ne peut rejoindre le groupe avec `newgrp`).



/etc/gshadow

◆ **Group Administrators**

- Une liste, délimité par des virgules, des administrateurs du groupe (Ils peuvent modifier le mot de passe du groupe, ainsi qu'ajouter ou retirer des membres du groupe avec la commande `gpaswd`).

◆ **Group Members**

- Une liste, séparé par des virgules, des membres du groupe.



Filtrer les Bases de Données

Il est souvent nécessaire d'accéder à des informations stocké dans ces quatre fichiers. Pour cela, le plus simple est d'utiliser la commande `grep` ou de concaténé `cat` et `grep`.

Exemple:

```
# grep emma /etc/passwd
```

```
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
```

```
# cat /etc/group | grep db-admin
```

```
db-admin:x:1050:grace,frank
```



Filtrer les Bases de Données

Une autre façon d'accéder à ces bases de données est d'utiliser la commande `getent`.

En général, cette commande affiche les entrées des bases de données qui prennent en charge la librairie *Name Service Switch (NSS)*.

Exemple:

```
# getent passwd emma
```

```
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
```

```
# getent group db-admin
```

```
db-admin:x:1050:grace,frank
```



Filtrer les Bases de Données

Si le deuxième argument de `getent` est vide, la commande affiche l'ensemble des entrées dans la base de données fournies en première argument.

La commande `getent` ne requiert pas d'être root, mais juste que la base de données soit lisible.

Remarque, La commande `getent` ne peut accéder qu'aux bases de données qui sont configuré dans le fichier `/etc/nsswitch.conf`.



Exercices





107

Tâches d'administration

107.2.a Automatisation des tâches d'administration par la planification des travaux



Planifications des Tâches avec Cron

Linux permet la planifications de tâches en utilisant le programme `cron`.

C'est un daemon qui tourne continuellement et qui se réveille toutes les quelques minutes pour vérifier un ensemble de tableaux et y trouver des tâches à exécuter. Ces tableaux sont appelé *crontabs*, et ils contiennent les *cron jobs*.

Chaque utilisateur peut aussi utiliser `cron` et à son propre `crontab`, et l'utilisateur root qui maintient les `crontab` du système.

Remarque, `cron` ne fonctionne que si le système est en marche, pour contourner ce problème, Linux permet l'usage de `anacron` qui n'est utilisable que par root, et qui lance les tâches qui n'ont pas pu être exécuté, si la machine était éteinte, au démarrage du système.



Crontabs Utilisateur

User crontabs sont des fichiers texte qui maintiennent la planification des cron jobs définis par l'utilisateur.

Elles sont toujours nommé d'après l'utilisateur qui les a créé, mais leurs localisation varie selon les distributions (souvent un sous-répertoire de `/var/spool/cron`).

Chaque ligne d'un crontab utilisateur comprends six champs séparé par des espaces:

- La minute d'une heure (0-59).
- L'heure d'une journée (0-23).
- Le jour d'un mois (1-31).



Crontabs Utilisateur

- ◆ Le mois d'une année (1-12).
- ◆ Le jour de la semaine (0-7, avec Dimanche=0 ou Dimanche=7).
- ◆ La commande à exécuter.









Pour les mois et les jours de la semaine, on peut aussi utiliser les trois premières lettres du nom au lieu du chiffre correspondant.

Les cinq premiers champs indiquent quand doit être exécuté la commande qui se trouve dans le sixième champ, et peuvent contenir plusieurs valeurs.



Crontabs Utilisateur

En particuliers, on peut spécifier plusieurs valeurs en utilisant:

-   **(asterisks)**
 - Fait références à n'importe quelle valeurs.
-   **(virgule)**
 - Spécifie une liste de valeurs possibles.
-   **(tiret)**
 - Spécifie une étendue de valeurs possibles.
-   **(slash)**
 - Spécifie des valeurs incrémentés.

La plupart des distributions fournissent un exemple dans le fichier `/etc/crontab` qui permet de bien comprendre la syntaxe.



Crontabs Utilisateur

Exemple de `/etc/crontab:`

```
SHELL=/bin/sh
```

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

```
# Example of job definition:
```

```
# .----- minute (0 - 59)
```

```
# | .----- hour (0 - 23)
```

```
# | | .----- day of month (1 - 31)
```

```
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
```

```
# | | | | .---- day of week (0 - 7) OR sun,mon,tue,wed,thu,fri,sat
```

```
# | | | | |
```

```
# * * * * * [user-name] command to be executed
```



Crontabs Système

System crontabs sont des fichiers texte qui maintiennent la planifications des cron jobs du système.

`/etc/crontab` et tous les fichiers dans le répertoire `/etc/cron.d` sont les crontabs du système.

La plupart des distributions utilisent les répertoires `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, et `/etc/cron.monthly`, contenant des scripts à exécuter aux fréquences respectives.

Si on souhaite exécuter un script tous les jours, il suffit de le placer dans `/etc/cron.daily`.

Attention, certaines distributions utilisent plutôt `/etc/cron.d/hourly`, `/etc/cron.d/daily`, `/etc/cron.d/weekly`, et `/etc/cron.d/monthly`.



Crontabs Système

La structure des crontabs système est similaire à celle des utilisateur avec un champs en plus pour indiquer quel utilisateur va exécuter la commande:

- La minute d'une heure (0-59).
- L'heure d'une journée (0-23).
- Le jour d'un mois (1-31).
- Le mois d'une année (1-12).
- Le jour de la semaine (0-7, avec Dimanche=0 ou Dimanche=7).
- Le nom de l'utilisateur à utiliser pour exécuter la commande.
- La commande à exécuter.

On peut aussi utiliser les même paramètres que pour les crontabs utilisateur, c'est à dire , , , et .



Spécifications Temporelles Particuliers

Lors de la modification de fichier crontab, il existe des raccourcis spéciaux pour les cinq premières colonnes:

- `@reboot`
 - Exécute la tâche au redémarrage du système.
- `@hourly`
 - Exécute la tâche une fois par heure, au début de l'heure.
- `@daily` (ou `@midnight`)
 - Exécute la tâche une fois par jour, à minuit.
- `@weekly`
 - Exécute la tâche une fois par semaine, à minuit le dimanche.



Spécifications Temporelles Particuliers



`@monthly`

- Exécute la tâche une fois par mois, à minuit sur le premier jour du mois.



`@yearly` (ou `@annually`)

- Exécute la tâche une fois par an, à minuit le 1er janvier.



Variables Crontab

Dans un fichier crontab, il y a parfois des attributions de variables avant que la planification des tâches soit déclaré.

Les variables d'environnement les plus communes sont:



HOME

- Le répertoire ou `cron` invoque les commandes, par défaut le répertoire maison de l'utilisateur.



MAILTO

- Le nom de l'utilisateur, ou l'adresse mail, à qui envoyer les sorties standards et les erreurs. Plusieurs valeurs séparé par une virgule sont autorisé, et un champs vide indique qu'aucun mail de doit être envoyé.



Variables Crontab



`PATH`

- Le chemin où les commandes peuvent être trouvées.



`SHELL`

- Le shell à utiliser, par défaut `/bin/sh`.





Création de Cron Jobs Utilisateur

La commande `crontab` est utilisé pour le maintien des fichiers crontab pour un utilisateur.

En particulier, `crontab -e` permet d'éditer son fichier crontab, ou de le créer si celui ci n'existe pas.

Par défaut, la command `crontab` ouvre l'éditeur indiqué dans les variables d'environnement `VISUAL` ou `EDITOR`, pour permettre la modification du fichier crontab dans votre éditeur de prédilection.



Création de Cron Jobs Utilisateur

Exemple:

```
$ crontab -e
```

```
no crontab for frank - using an empty one
```

```
Select an editor. To change later, run 'select-editor'.
```

```
1. /bin/ed
```

```
2. /bin/nano < ---- easiest
```

```
3. /usr/bin/emacs24
```

```
4. /usr/bin/vim.tiny
```

```
Choose 1-4 [2]:
```



Création de Cron Jobs Utilisateur

Si l'on souhaite exécuter le script `foo.sh` situé dans le répertoire maison chaque jour à 10:00, on peut ajouter la ligne suivante à notre crontab:

```
0 10 * * * /home/frank/foo.sh
```

Considérons les entrées suivantes :

```
0,15,30,45 08 * * 2 /home/frank/bar.sh
```

```
30 20 1-15 1,6 1-5 /home/frank/foobar.sh
```

La première ligne nous indique que le script `bar.sh` sera exécuté tous les mardis à 8:00, 8:15, 8:30, et 8:45.

La deuxième ligne nous indique que le script `foobar.sh` sera exécuté à 8:30, du lundi au vendredi pour les quinze premiers jours des mois de Janvier et Juin.



Création de Cron Jobs Utilisateur

Remarque, même si les fichiers crontab peuvent être édités manuellement, il est fortement recommandé d'utiliser la commande `crontab`. Souvent, les permissions sur les fichiers crontab ne permettent de les éditer qu'à partir de la commande `crontab`.

Les options les plus utiles de `crontab` sont:

- `-l`
 - Affiche le crontab courant dans la sortie standard.
- `-r`
 - Retire le crontab courant.
- `-u`
 - Spécifie le nom de l'utilisateur dont on souhaite modifier le crontab. Ne peut être utilisé que par root, et permet à root d'accéder au crontab des utilisateurs.



Création de Cron Jobs Système

Contrairement aux crontabs utilisateur, les crontabs système sont modifiées directement avec un éditeur de texte, sans passer par la commande `crontab`, en modifiant `/etc/crontab` ou les fichiers dans `/etc/cron.d`.

Par exemple, si l'on souhaite exécuter le script `barfoo.sh` situé dans le repertoire `/root` chaque jour à 1:30, on peut rajouter la ligne suivante à `/etc/crontab`:

```
30 01 * * * root /root/barfoo.sh >>/root/output.log  
2>>/root/error.log
```

L'output standard est ajouté au log `/root/output.log`, et les erreur au log `/root/error.log`.



Création de Cron Jobs Système

Remarque, A moins que l'output soit redirigé vers un fichier comme dans l'exemple précédent (ou que la variable `MAILTO` soit vide), toutes les outputs du cron job seront envoyé par mail.

Une pratique courante est de rediriger la sortie standard vers `/dev/null` et de ne pas rediriger la sortie erreur.

Ainsi, l'utilisateur est notifié par mail immédiatement en cas de problèmes.



Configurer l'Accès à la Planification

Avec Linux, les fichiers `/etc/cron.allow` et `/etc/cron.deny` sont utilisés pour fixer des restrictions sur `crontab`.

En particulier, ils autorisent ou bloquent certains utilisateurs de planifier des cron jobs.

Si `/etc/cron.allow` existe, alors seuls les utilisateurs non-root listés dedans peuvent planifier des tâches en utilisant la commande `crontab`.

Si `/etc/cron.allow` n'existe pas, mais que `/etc/cron.deny` existe, alors les utilisateurs non-root listés dedans ne pourront pas planifier de tâches en utilisant `crontab`.



Une Alternative à Cron

En utilisant `systemd` en tant que manager système et service, on peut fixer des *timers* comme alternative à `cron` pour planifier des tâches.

Les Timers sont des fichiers `systemd` unité identifié par le suffixe `.timer`, et pour chacun, il faut un fichier unité correspondant qui décrit l'unité à activé quand le timer se termine. Par défaut, un `timer` active le service du même nom, mais sans le suffixe.

Un timer inclus une section `[Timer]` qui précise quand la tâche doit être exécuté.

Grâce à l'option `OnCalendar=`, on peut définir des *real-time timers* qui fonctionnent de la même manière que les cron jobs.



Une Alternative à Cron

La syntaxe de `OnCalendar=` est la suivante:

```
DayOfWeek Year-Month-Day Hour:Minute:Second
```

avec `DayOfWeek` étant optionnel. Les opérateurs `*`, `/`, et `,` fonctionnant comme pour les cron jobs, et `..` servant à indiquer des plages continues.

Remarque, On peut aussi définir des *timers monotoniques* qui se déclenche une fois qu'un certains temps c'est écoulé depuis un moment donné (par exemple 5min apres le demarrage).



Une Alternative à Cron

Par exemple, si l'on veut exécuter le service `/etc/systemd/system/foobar.service` à 5:30 le premier lundi de chaque mois, on peut ajouter les lignes suivantes au fichier unité correspondant `/etc/systemd/system/foobar.timer`:

```
[Unit]
```

```
Description=Run the foobar service
```

```
[Timer]
```

```
OnCalendar=Mon *-*-1..7 05:30:00
```

```
Persistent=true
```

```
[Install]
```

```
WantedBy=timers.target
```



Une Alternative à Cron

Une fois le timer créé, il faut l'activer et le lancer en tant que root:

```
# systemctl enable foobar.timer
```

```
# systemctl start foobar.timer
```

On peut changer la fréquence de notre tâche en modifiant la valeur de `OnCalendar` et en tapant la commande `systemctl daemon-reload`.

Pour connaître l'ensemble des timers actifs, on peut utiliser la commande `systemctl list-timers`. On peut rajouter l'option `--all` pour voir aussi les timers inactifs.

Remarque, les timers sont logués dans le journal de systemd, et peuvent être accédés avec `journalctl`. De plus, si nous ne sommes pas root, il faut rajouter l'option `--user` pour les commandes `systemctl` et `journalctl`.



Une Alternative à Cron

Au lieu d'utiliser la syntaxe complète de `OnCalendar`, on peut utiliser des expressions spéciales pour des fréquences particulières:

- `hourly`
 - Exécute la tâche une fois par heure, au début de l'heure.
- `daily`
 - Exécute la tâche une fois par jour, à minuit.
- `weekly`
 - Exécute la tâche une fois par semaine, à minuit le lundi.
- `monthly`
 - Exécute la tâche une fois par mois, à minuit sur le premier jour du mois.
- `yearly`
 - Exécute la tâche une fois par an, à minuit le 1er janvier.

Pour plus d'informations, voir `man 5 systemd.timer`



Exercices





107

Tâches d'administration

107.2.b Automatisation des tâches d'administration par la planification des travaux



Planification avec `at`

La commande `at` permet de planifier une tâche ponctuelle dans le future.

Après avoir entré `at` dans l'invite de commande suivie du temps spécifique où l'on souhaite exécuter la commande, on entre dans l'invite de commande de `at`, où l'on peut spécifier les commandes à exécuter.

On en sort avec la séquence clavier `Ctrl+D`.

Exemple:

```
$ at now +5 minutes
```

```
at> date
```

```
at> Ctrl+D
```



Planification avec `at`

De manière similaire à `cron`, les sorties standard et erreur sont envoyé par mail à l'utilisateur.

Remarque, Il est nécessaire que le daemon `atd` soit entrain de tourné pour utiliser la planification `at`.

Remarque, la commande `batch` est similaire à `at`, néanmoins les tâches `batch` ne sont exécutés que si la charge sur le système n'est pas trop importante.



Planification avec `at`

Les options les plus importantes de `at` sont:

- `-c`
 - Renvoie les commandes d'une tâche spécifiée par son job ID dans la sortie standard.
- `-d`
 - Supprime la tâche spécifiée par son job ID. C'est un alias pour `atrm`.
- `-f`
 - Lis la tâche depuis un fichier au lieu d'utiliser l'entrée standard.
- `-s`
 - Liste l'ensemble des tâches en attente pour un utilisateur. Si root, alors list l'ensemble de toutes les tâches en attente. C'est un alias pour `atq`.



Planification avec `at`



`-m`

- Envoie un mail à la fin de la tâche même si il n'y a pas de sortie.



`-q`

- Utilise une file qui est désignée par une lettre unique, dans l'intervalle `a - z`, et `A - Z`. La file `a` est la file d'attente par défaut pour `at` tandis que la file `b` est celle par défaut pour `batch`. Plus les files ont une lettre importante, plus les travaux seront exécutés avec une valeur de gentillesse élevée.



`-v`

- Affiche les heures de lancement programmées avant de lire le job.



Lister les Tâches Planifiées avec `atq`

Planifions deux tâches avec `at`: La première exécute le script `foo.sh` à 9:30, la deuxième exécute le script `bar.sh` après deux heures.

```
$ at 09:30 AM
```

```
at> ./foo.sh
```

```
at> Ctrl+D
```

```
job 13 at Tue Mar 5 09:30:00 2024
```

```
$ at now +2 hours
```

```
at> ./bar.sh
```

```
at> Ctrl+D
```

```
job 14 at Tue Mar 5 11:10:00 2024
```



Lister les Tâches Planifiées avec `atq`

Pour lister les tâches en attente, on utilise la commande `atq` qui liste chacune des tâches ainsi : *job ID, job execution date, job execution time, queue, et username.*

```
$ atq
```

```
13      Tue Mar 5 09:30:00 2024 a frank
```

```
14      Tue Mar 5 11:10:00 2024 a frank
```

Remarque, si on exécute `atq` en tant que root, on récupère la liste de toutes les tâches en attente.



Suppression de Tâche avec `atrm`

Si on veut supprimer une tâche `at`, on peut utiliser la commande `atrm` suivi du numéro de la tâche.

Par exemple, pour supprimer la tâche précédente avec l'ID 14, il suffit de lancer:

```
$ atrm 14
```

On peut supprimer plusieurs tâches en indiquant plusieurs ID.

Remarque, `atrm` est un alias de `at -d`.



Configurer l'Accès à la Planification

De manière similaire à `cron`, les fichiers `/etc/at.allow` et `/etc/at.deny` sont utilisés pour fixer des restrictions sur `at`.

En particulier, ils autorisent ou bloquent certains utilisateurs de planifier des cron jobs.

Si `/etc/at.allow` existe, alors seuls les utilisateurs non-root listés dedans peuvent planifier des tâches en utilisant la commande `at`.

Si `/etc/at.allow` n'existe pas, mais que `/etc/at.deny` existe, alors les utilisateurs non-root listés dedans ne pourront pas planifier de tâches en utilisant `at`.



Spécification du Temps

On peut spécifier le temps d'exécution d'une tâche `at` au format `HH:MM`, et optionnellement suivi de AM ou PM pour les format sur 12 heures.

Si le l'heure est déjà passé, l'heure du lendemain est utilisé.

Si on veut préciser le jour où la tâche doit être exécuté, il faut rajouter la date après l'heure au format : `month-name day-of-month, month-name day-of-month year`, `MMDDYY`, `MM/DD/YY`, `DD.MM.YY` and `YYYY-MM-DD`.

Les mot-clés suivant sont aussi accepté: `midnight`, `noon`, `teatime` (16h), et `now`. On peut aussi les faire suivre d'un plus (+) et d'un temps à rajouter (minutes, heures, jours, et semaine).

Enfin, on peut préciser aujourd'hui ou demain, avec `today` ou `tomorrow`.



Spécification du Temps

Par exemple, `at 07:15 AM Jan 01` exécute la tâche le 1er janvier à 7:15, et `at now +5 minutes` pour exécuter la tâche dans 5 minutes.

Pour plus d'information, le fichier `timespec` dans `/usr/share` contient des définitions exactes des spécifications du temps.



Une Alternative à `at`

On peut aussi utiliser `systemd` pour planifier des tâches ponctuelles, en utilisant la commande `systemd-run`.

Elle crée une unité timer transitoire pour que la commande soit exécutée sans avoir à créer de fichier service.

Par exemple:

```
# systemd-run --on-calendar='2024-03-05 11:30' date
```

Si on veut exécuter le script `foo.sh`, après deux minutes, on peut utiliser :

```
# systemd-run --on-active="2m" ./foo.sh
```

Pour plus d'information, consulter `man 1 systemd-run`



Exercices





107

Tâches d'administration

107.3 Paramètres régionaux et langues



Fuseaux Horaires

Les fuseaux horaires sont désigné soit par le nom d'une ville ou d'un pays qui occupe le fuseau horaire, soit par référence au premier méridien :

◆ UTC +X

- *Coordinated Universal Time*

◆ GMT +X

- *Greenwich Mean Time*

Il s'agit du **format Posix TZ**.

La commande `date` affiche aussi le fuseau horaire utilisé:

```
$ date
```

```
Tue Mar 19 10:45:21 +01 2024
```

Le décalage par rapport à UTC est donné par `+01` ce qui signifie *GMT+1*



Fuseaux Horaires

La commande `timedatectl`, qui est disponible dans les distributions qui utilisent systemd, affiche avec plus de détails le fuseau horaire utilisé:

Essayez: `timedatectl`

Le fuseau horaire par défaut est conservé dans le fichier `/etc/timezone`. De plus, pour les fuseaux horaires donnés par référence au décalage à *UTC*, il est nécessaire de commencer par `Etc`.

Par exemple:

```
Etc/GMT+3
```

```
America/Sao_Paulo
```

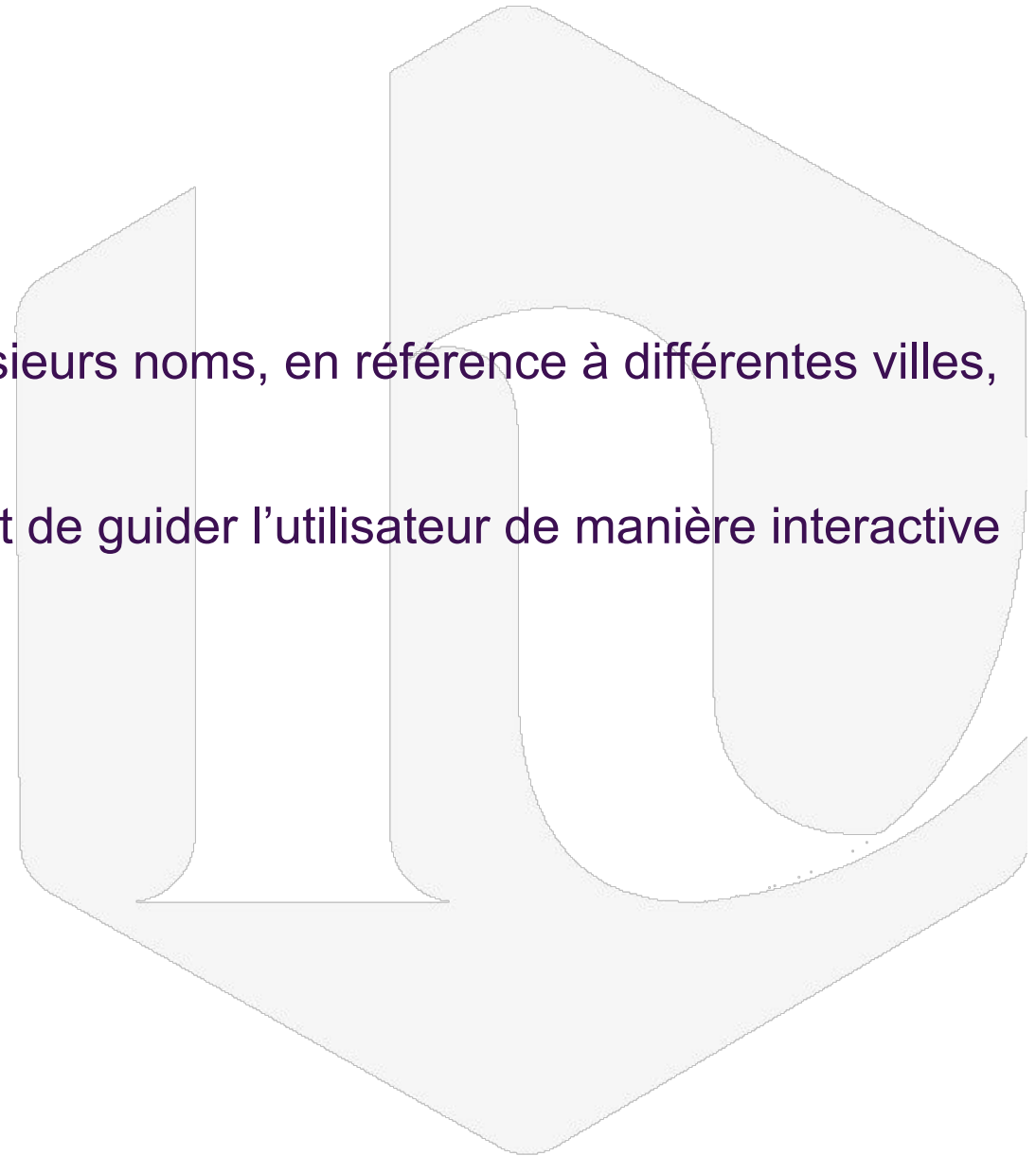


Fuseaux Horaires

Un fuseau horaire peut avoir plusieurs noms, en référence à différentes villes, ou à différents pays.

La commande `tzselect` permet de guider l'utilisateur de manière interactive vers le bon fuseau.

Essayez: `tzselect`





Fuseaux Horaires

Comme la commande `tzselect` l'indique, c'est la variable d'environnement `TZ` qui définit le fuseau horaire pour la session shell, qu'importe le fuseau horaire par défaut du système.

En ajoutant `TZ='America/Sao paulo'; Export TZ` au fichier `~/.profile`, cela va définir le fuseau horaire pour toutes les sessions futures de l'utilisateur.

On peut aussi utiliser `env` pour lancer un *sub-shell* avec un fuseau horaire différent, par exemple:

```
$ env TZ='Africa/Cairo' date
```



Heure d'Été (*Daylight Savings Time*)

Le fichier `/etc/localtime` contient l'ensemble des données utilisé par le système pour ajuster l'horloge correctement à l'heure d'été.

Les systèmes Linux standards ont des fichiers pour chaque fuseau horaire dans le répertoire `/usr/share/zoneinfo`, donc `/etc/localtime` est un lien symbolique vers le fichier correspondant dans le répertoire.

Les fichiers dans `/usr/share/zoneinfo` sont organisé par les noms des fuseaux horaires correspondants.

Par exemple, l'informations pour `Europe/Paris` est situé dans `/usr/share/zoneinfo/Europe/Paris`.



Langues et Encodage des Caractères

Les systèmes Linux fonctionnent avec une grande diversité de langues et de Encodages des caractères, qui sont appelé *locales*.

Le plus basique des configurations *locales* est la variable d'environnement `LANG`, qui permet à la plupart des programmes shell de savoir quelle langue utilisée.

La variable `LANG` est au format `ab CD`, avec `ab` le code langue au format ISO-639 et `CD` le code région au format ISO-3166.

Essayez: `echo $LANG`

Comme vous pouvez le constatez, `LANG` contient aussi le encodage des caractères qui doit être utilisé par le système.



Langues et Encodage des Caractères

Quelques standards de encodages de caractères:

- ◆ ASCII
 - *American Standard Code for Information Interchange*
- ◆ Standard Unicode UTF-8
 - *Universal character set Transformation Format - 8 bits*
- ◆ Standards ISO
 - par exemple ISO-8859-1

Les paramètres régionaux pour l'ensemble du système sont définis dans le fichier `/etc/locale.conf`.

Les utilisateurs peuvent changer les paramètres régionaux en redéfinissant la variable d'environnement `LANG`.



Langues et Encodage des Caractères

Sur les systèmes utilisant *systemd*, la commande `localectl` peut aussi être utilisé pour redéfinir la langue.

Par exemple:

```
localectl set-locale LANG=en_US.UTF-8
```

Il existe d'autres variables d'environnement pour les paramètres régionaux:

◆ `LC_COLLATE`

- Définit l'ordre alphabétique.

◆ `LC_CTYPE`

- Définit comment le système interprète certain ensemble de caractères. Par exemple, c'est lui qui définit la case majuscule et minuscule.



Langues et Encodage des Caractères

- ◆ `LC_MESSAGES`
 - Définit la langue pour les affichages de message programme (principalement les programmes GNU).
- ◆ `LC_MONETARY`
 - Définit le symbole et le format de la monnaie.
- ◆ `LC_NUMERIC`
 - Définit le format numérique. Principalement, le séparateur des milliers et des décimales.
- ◆ `LC_TIME`
 - Définit le format pour l'heure et la date.
- ◆ `LC_PAPER`
 - Définit la taille standard d'une feuille de papier.
- ◆ `LC_ALL`
 - Outrepasse les autres variables, y compris `LANG`.



Langues et Encodage des Caractères

La commande `locale` permet d'afficher l'ensemble des variables dans la configuration actuelle.

Essayez:

```
locale
```

Normalement, la seule variable non-définie est `LC_ALL`, qui peut être utilisé pour outrepasser les autres.

Essayez:

```
$ date
```

```
$ env LC_ALL=en_US.UTF-8 date
```



Langues et Encodage des Caractères

Pour éviter certaines surprises lors d'une comparaison alphabétique, il est recommandé d'utiliser le locale `C`, ou locale *POSIX*, c'est à dire, `LANG=C`.

Il s'agit juste d'une comparaison au niveau des bits, elle est donc plus rapide.



Conversion d'Encodage

Un texte est inéligible si l'encodage utilisé à sa création n'est pas le même que celui du système.

La commande `iconv` peut résoudre cela, en convertissant d'un encodage à un autre.

Par exemple:

```
$ iconv -f ISO-8859-1 -t UTF-8 original.txt > convert.txt
```

avec `-f ISO-8859-1` (ou `--from-code ISO-8859-1`) l'encodage source, et `-t UTF-8` (ou `--to-code UTF-8`) l'encodage destination.

L'ensemble des encodages supporté par `iconv` est disponible avec `iconv -l` ou `iconv --list`.



Exercices

