Sous GNU/Linux, un grand nombre de tâches doivent être réalisées périodiquement pour assurer la cohérence du système. Parmi ces tâches répétitives, on peut citer par exemple la vérification des systèmes de fichiers, la sauvegarde des données utilisateur ou l'analyse des fichiers journaux du système.

D'autre part, il peut être nécessaire d'effectuer certaines tâches ponctuelles à des horaires précis pour éviter les interruptions de service ou répondre à des contraintes de fonctionnement de l'entreprise. Imaginons qu'une opération de maintenance sur le système d'information de l'entreprise nécessite l'arrêt du SGBD (Système de gestion de base de données) : il sera certainement moins pénalisant pour les utilisateurs de lancer cette tâche de nuit.

A Tâches Cron

Cron est un outil permettant d'automatiser l'exécution des tâches répétitives qui a été créé par Paul Vixie.

Il se présente sous la forme d'un processus démon **crond** utilisant un ensemble de fichiers consignant les travaux à effectuer avec la période d'exécution associée : les fichiers "crontab" ou "cron table" utilisateur.

1 Cron tables utilisateur

Chaque utilisateur possède son propre fichier crontab dans le répertoire */var/spool/cron*. Il n'est pas éditable directement pour des raisons de droits d'accès au répertoire, entre autres, et doit être modifié à l'aide de la commande **crontab -e** qui invoque pour cela l'éditeur défini par la variable **$EDITOR** ou **$VISUAL** (par défaut **vi**). Les commandes contenues dans la table seront alors exécutées sous l'identité du propriétaire du fichier.

De même, il est possible de consulter la "cron table" des utilisateurs avec :

```
crontab -l [-u <utilisateur>]
```

Bien sûr, seul l'administrateur est autorisé à consulter la liste des travaux planifiés par un autre utilisateur, et donc le seul à pouvoir spécifier le nom de l'utilisateur sur la ligne de commande.

Une autre action possible avec la commande **crontab** est la suppression complète de la cron table utilisateur avec l'option **-r** ; cette option est peu utilisée car il est possible, et souvent préférable, de commenter les travaux directement dans le fichier crontab. Néanmoins, on peut l'employer, après avoir sauvegardé la cron table, lors de la suppression d'un compte utilisateur.

2 Fichier crontab

Le fichier crontab utilisateur est composé de deux types de lignes :

- Les lignes d'initialisation de variables d'environnement : elles permettent de définir l'environnement dans lequel les travaux cron doivent être exécutés.
- Les lignes de commandes pour le démon **crond** : elles définissent les travaux à lancer périodiquement.

Les lignes commençant par le caractère "**#**" sont ignorées, ainsi que les lignes vides. Il n'est pas possible d'insérer un commentaire en fin de ligne d'affectation de variable ou d'instruction cron.

a Affectation de variables

Pour initialiser une variable d'environnement dans le fichier crontab, il suffit d'utiliser la syntaxe :

```
VAR=VALEUR
```

Les variables **LOGNAME**, **HOME** et **SHELL** sont prédéfinies et contiennent respectivement le nom du compte, le répertoire personnel indiqués dans `/etc/passwd` pour le propriétaire de la cron table, et `/bin/sh` comme interpréteur de commandes.

> *Les variables **HOME** et **SHELL** peuvent être surchargées, ce qui n'est pas le cas de **LOGNAME**.*

MAILTO est une autre variable remarquable utilisée par cron. Si celle-ci n'est pas définie (cas par défaut), les sorties standard et d'erreur des commandes de la cron table seront envoyées par mail au propriétaire de ce fichier. Il est possible d'affecter le nom d'un autre utilisateur à cette variable pour envoyer le message à celui-ci. Pour qu'aucun message ne soit envoyé, il faudra spécifier **MAILTO=""** ou rediriger explicitement les sorties des commandes.

b Commandes cron

Une ligne de commandes cron est constituée de deux informations :

- la périodicité d'exécution de la tâche ;
- la ligne de commandes shell à interpréter pour lancer la tâche.

Les cinq premiers champs de la ligne permettent donc de spécifier cette période, ils signifient dans cet ordre :

- Les minutes : les valeurs sont comprises entre **0** et **59**.
- Les heures : les valeurs sont comprises entre **0** et **23**.

- Les jours : les valeurs sont comprises entre **1** et **31**. Si les jours **29**, **30** ou **31** sont spécifiés alors que le mois ne contient pas autant de jours, les tâches ne seront tout simplement pas exécutées. On préfère donc généralement utiliser des valeurs inférieures à **29**.
- Les mois : les valeurs peuvent être soit le numéro du mois dans l'année (**1** à **12**), soit son nom ANSI sur trois lettres (**jan**, **feb**, **mar**, **apr**...).
- Le jour de la semaine : les valeurs admises vont de **0** à **7**, où **0** et **7** sont équivalentes et désignent dimanche. De même que pour le mois, il est possible d'utiliser le nom ANSI du jour sur trois lettres (**sun**, **mon**, **tue**...).

Ces champs acceptent aussi les caractères suivants :

- le caractère "*****" qui signifie "quel que soit" ;
- la virgule "**,**" qui permet de séparer plusieurs valeurs ;
- le tiret "**-**" entre deux valeurs qui désigne un intervalle, éventuellement suivi de "**/**" pour spécifier l'incrément dans cet intervalle.

Par exemple :

```
32        7    *    *    1          tous_les_lundis_à_7h32
0-59/5    *    *    *    *          toutes_les_5_minutes
4         3    *    *    sat,sun    tous_les_samedis_et_dimanches_à_3h04
```

> *Les champs du fichier crontab sont séparés par un ou plusieurs espaces et/ou tabulations.*

Le sixième champ, quant à lui, permet d'indiquer la ligne de commandes que **crond** doit exécuter.

Étant donné que le but de **crond** est d'exécuter la tâche même si l'utilisateur n'est pas connecté, les commandes spécifiées dans ces fichiers ne doivent pas dépendre d'un terminal, notamment en attendant une entrée de la part de l'utilisateur.

Si une sortie était retournée par la commande, nous avons vu qu'elle serait renvoyée par courrier à l'utilisateur en fonction de la valeur de la variable **MAILTO**.

Une erreur classique dans l'emploi de Cron consiste à utiliser des chemins relatifs pour les noms de commandes ; il faut considérer que ***crond*** *ne possède pas le même environnement que l'utilisateur. Par conséquent, les variables telles que* ***PATH*** *ne sont pas définies. Ceci est valable aussi pour le contenu des scripts shell pouvant être lancés à partir des crontabs.*

3 Cron table système

En plus des cron tables utilisateur, il existe une notion de crontab système. Celle-ci permet de planifier des tâches d'administration comme le nettoyage régulier de */tmp* ou la mise à jour automatique des paquetages du système.

La différence avec la cron table de **root**, qui permet d'obtenir les mêmes résultats, est la possibilité de spécifier dans celle-ci le nom de l'utilisateur sous lequel doit s'exécuter la tâche.

Le fichier crontab système est */etc/crontab*. Sa syntaxe est la même que les cron tables utilisateur, avec la possibilité de spécifier le nom d'utilisateur juste avant la ligne de commandes à lancer (en sixième champ).

Par exemple :

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root nice -n 19 run-parts --report /etc/cron.hourly
02 4 * * * root nice -n 19 run-parts --report /etc/cron.daily
22 4 * * 0 root nice -n 19 run-parts --report /etc/cron.weekly
42 4 1 * * root nice -n 19 run-parts --report /etc/cron.monthly
```

On remarque dans ce fichier les entrées faisant référence aux répertoires */etc/ cron.hourly*, */etc/cron.daily*, */etc/cron.weekly* et */etc/cron.monthly* ; c'est en fait un moyen simple pour l'administrateur d'ajouter des tâches devant s'exécuter respectivement toutes les heures, jours, semaines et mois, en plaçant les scripts correspondants dans ces répertoires.

En effet, la commande **run-parts <répertoire>** exécute tous les scripts et programmes présents dans le répertoire spécifié.

> *L'option **--report** de la commande **run-parts** affiche le nom des scripts exécutés produisant des sorties ; le rapport envoyé (suivant la variable **MAILTO**) est alors plus explicite. De plus, la commande **nice -n 19** permet de baisser la priorité de ces tâches planifiées (cf. chapitre 1 - Gestion de la mémoire et des ressources système).*

4 Démon crond

Comme tout démon, un script d'initialisation permettant de piloter le service est présent dans le répertoire */etc/rc.d/init.d*.

On saisira :

```
/etc/rc.d/init.d/crond [start|stop|restart]
```

Le démon **crond** vérifie alors toutes les minutes si une tâche doit être effectuée.

Lors de la modification d'un fichier crontab, il faut envoyer le signal SIGHUP au démon **crond** pour que celui-ci prenne les modifications en compte.

> *Le processus associé à ce démon appartenant à **root**, un utilisateur ordinaire ne pourra pas envoyer le signal SIGHUP à **crond** ; la commande **crontab** qui a été invoquée pour modifier le fichier cron table utilisateur se chargera automatiquement de cette tâche.*