Chapitre 8
Planification de tâches

1. Introduction

Pour maintenir l'intérêt du travail et pour économiser du temps, il est important d'automatiser autant que possible les tâches répétitives. L'application la plus importante et la plus connue pour cela sur les systèmes Unix est *cron*.

2. Network Time Protocol (NTP)

Pour planifier les tâches de façon fiable, il est important que l'horloge du système soit précise. Le *NTP* (en anglais : *Network Time Protocol*, protocole de temps réseau) est un protocole qui permet de synchroniser l'horloge avec d'autres horloges dans le réseau. Ces horloges ne se trouvent pas nécessairement dans le même fuseau horaire : les serveurs et clients NTP utilisent l'*UTC* (*temps universel coordonné*) pour se synchroniser et, ensuite, le client corrige l'heure pour le fuseau horaire local. Un client peut synchroniser avec plusieurs serveurs ; la synchronisation avec plusieurs serveurs est même conseillée pour augmenter la fiabilité.

Le paquet qui livre un daemon et un client NTP s'appelle simplement *ntp*. Une solution alternative souvent utilisée s'appelle *OpenNTPD*, un logiciel développé pour OpenBSD qui fait également partie du catalogue de logiciels portés de FreeBSD et des dépôts de logiciels de Debian.

Pour pouvoir synchroniser l'horloge du serveur avec un serveur NTP sur internet, le port UDP 123 doit être ouvert pour le trafic sortant dans le pare-feu. Si le serveur doit également servir de serveur NTP pour d'autres ordinateurs, le port UDP 123 doit aussi être ouvert pour le trafic entrant. Cette dernière fonctionnalité n'est d'ailleurs pas traitée dans ce livre, mais elle s'active assez facilement avec l'aide de la page manuel du fichier de configuration.

Normalement, il n'est pas nécessaire de modifier le fichier de configuration, sauf si, pour des raisons de confidentialité ou autres, d'autres serveurs sont sélectionnés ou si le serveur ne sert pas seulement de client NTP, mais également de serveur NTP.

Plus d'informations sont disponibles dans `ntpd(8)` et les pages manuel référées par cette page.

2.1 FreeBSD

Sous FreeBSD, *ntp* fait partie du système de base et est donc déjà installé. Pour pouvoir lancer le daemon, *ntpd*, la ligne suivante doit être ajoutée au fichier `/etc/rc.conf` :

```
ntpd_enable="YES"
```

Cette ligne assure que le daemon NTP sera lancé au démarrage du serveur. Comme pour les autres daemons, la commande `service` sert au démarrage immédiat du daemon :

```
freebsd# service ntpd start
```

Le fichier de configuration s'appelle `/etc/ntp.conf`.

2.2 Debian

Sous Debian, le paquet *openntpd* est installé par défaut. Il n'est pas nécessaire de le remplacer par le paquet *ntp*.

Le fichier de configuration s'appelle `/etc/openntpd/ntpd.conf` et le service `openntpd` a déjà été lancé.

2.3 CentOS

Sous CentOS, aucun des deux paquets n'est installé par défaut ; le paquet *ntp* est disponible dans le dépôt de logiciels par défaut.

Après l'installation, le service n'est pas lancé automatiquement.

```
centos# yum install ntp
centos# systemctl enable ntpd
centos# systemctl start ntpd
```

Le fichier de configuration s'appelle `/etc/ntp.conf`.

3. Cron

Cron est un daemon qui vérifie chaque minute s'il y a des tâches à accomplir, et les exécute si c'est le cas. Le nom cron vient de χρόνος (*chronos*), le mot grec pour *temps* ou *heure*.

Une tâche planifiée avec *cron* s'appelle un *cronjob* et le fichier dans lequel sont définis les cronjobs s'appelle un *crontab* (en anglais : *cron table*, liste des crons). Des exemples des cronjobs courants sont la création de sauvegardes, la recherche des mises à jour, les scans périodiques des virus et des *rootkits*, l'envoi des mailings, etc.

En principe, chaque utilisateur peut avoir son propre crontab, mais l'administrateur système peut limiter l'accès à *cron* à l'aide des fichiers nommés `allow` (*permettre*) et `deny` (*refuser*). Si le fichier `allow` existe et qu'il ne contient pas le nom d'un certain utilisateur, ce dernier ne peut pas se servir de *cron*. Si le fichier `allow` n'existe pas, mais que le fichier `deny` existe et qu'il contient le nom d'un certain utilisateur, cet utilisateur ne peut pas se servir de *cron*. Si aucun des fichiers `allow` et `deny` n'existe, tous les utilisateurs peuvent se servir de *cron*.

Sous FreeBSD, le fichier `allow` est `/var/cron/allow` et le fichier `deny` est `/var/cron/deny`. Sous Linux, ce sont respectivement `/etc/cron.allow` et `/etc/cron.deny`.

La commande pour afficher et modifier le crontab est `crontab`. Le paramètre `-l` (en anglais : *list*, lister) sert à l'affichage du crontab et le paramètre `-e` (en anglais : *edit*, modifier) sert à sa modification. L'utilisateur root peut se servir du paramètre `-u` pour afficher ou modifier le crontab d'un autre utilisateur.

```
$ crontab -e

# crontab -l -u diane
```

Un crontab peut contenir des cronjobs et des définitions de variables, un cronjob ou une définition par ligne. Les lignes vides et les lignes commençant par un dièse (#) sont ignorées. Normalement, les cronjobs sont lancés avec seulement trois variables d'environnement : *SHELL*, *LOGNAME* et *HOME*. Les variables *LOGNAME* et *HOME* sont prises de la ligne de cet utilisateur dans le fichier `/etc/passwd` et la variable *SHELL* est définie comme `/bin/sh`. Il n'est pas possible de modifier la variable *LOGNAME* dans le crontab. À part ces variables, le propriétaire du crontab peut définir toutes les variables nécessaires à l'exécution des cronjobs. Il en existe une spéciale : MAILTO ; si cette variable a été définie, tous les résultats des cronjobs dans le crontab seront envoyés à cette adresse.

La ligne d'un cronjob comporte six champs, séparés par des espaces libres : les cinq premiers champs définissent l'heure à laquelle la commande doit être exécutée et le reste de la ligne est la commande à exécuter. Les cinq champs pour définir l'heure sont les suivants, dans cet ordre :

minute

Valeurs possibles : 0 - 59

heure

Valeurs possibles : 0 - 23

jour du mois

Valeurs possibles : 1 - 31

mois

Valeurs possibles : 1 - 12

jour de la semaine

Valeurs possibles : 0 - 7 (0 et 7 signifient le dimanche)

En plus des valeurs simples, il est également possible de spécifier des séries (*1-5*), des collections (*1,3,5*) et des ensembles des deux (*0,2-5,7,9-11*). Une valeur spéciale est `*`, ce qui signifie chaque valeur possible. Un cronjob ayant une étoile dans le champ jour-du-mois sera exécuté chaque jour du mois (si l'exécution n'est pas limitée par la valeur d'un autre champ). D'autres manières pour spécifier l'heure sont décrites dans `crontab(5)`.

Voici un exemple d'un crontab :

```
# Envoyer les résultats éventuels à Dimitri.
MAILTO=dimitri@example.com

# Chercher les commandes dans ces trois répertoires.
PATH=/bin:/usr/bin:/usr/local/bin

# Sauvegarder le répertoire compta toutes les nuits à 3h15.
15 3 * * * tar -cf /data/svg.$(date +%F) /data/compta

# Envoyer un courriel à tout le monde, les vendredis à 16h.
30 16 * * 5 echo "Bon week-end!" | mail tous@example.com

0 7 1,15 * * echo "Jours 1 et 15 de tous les mois, à 7h"

0 13 1 1,5,9 * echo "1 janvier, mai, septembre à 13h"

15 8 13 * 5 echo "Tous les vendredis 13 à 8h15"
```

À part les crontabs des utilisateurs, il existe également les crontabs système. Ces crontabs sont installés dans le fichier `/etc/crontab` et le répertoire `/etc/cron.d`.

Le format des crontabs système ressemble aux crontabs d'utilisateurs. Par contre, comme le crontab système ne contient pas les cronjobs d'un seul utilisateur mais qu'il peut contenir ceux de plusieurs utilisateurs, un champ supplémentaire est ajouté entre la définition de l'heure et la commande à exécuter. Ce champ contient le nom d'utilisateur du propriétaire du cronjob en question.

Le crontab système pourrait contenir, par exemple, les cronjobs suivants :

```
15 12 5 * * diane echo "Les 5 du mois à 12h15, pour Diane"

30 8 * * 1-5 root echo "Jours de travail à 8h30, pour root"
```

Il existe aussi sur Linux les répertoires `/etc/cron.daily`, `/etc/cron.hourly`, `/etc/cron.monthly` et `/etc/cron.weekly` dans lesquels des scripts peuvent être installés pour être exécutés respectivement tous les jours, toutes les heures, tous les mois et toutes les semaines.

De la même manière, il existe sur FreeBSD les répertoires `/etc/periodic` et `/usr/local/etc/periodic`, et leurs sous-répertoires respectifs. Les scripts dans ces répertoires peuvent être activés et désactivés ; le fichier `/etc/defaults/periodic.conf` indique quels scripts sont activés par défaut. Pour activer d'autres scripts, les variables correspondantes sont définies dans le fichier `/etc/periodic.conf` qui peut être créé s'il n'existe pas encore (le fichier `/etc/defaults/periodic.conf` pourrait être écrasé par une mise à jour). Les variables à définir se trouvent dans les scripts correspondants.

4. Anacron

Une variante de *cron* qui doit être évoquée est *anacron*. Ce daemon a, en principe, les mêmes fonctionnalités que *cron* mais il part du principe que le serveur n'est pas toujours allumé. Si un cronjob a été raté parce que le serveur était éteint, avec *cron* cette tâche ne sera plus exécutée ; *anacron,* par contre, exécutera cette tâche quand le serveur sera rallumé.

Comme le serveur décrit dans ce livre est censé être allumé tout le temps, *anacron* ne sera pas détaillé ici.

5. At

La commande `at` (*à*) sert à la planification des tâches uniques. Sous Debian et CentOS, cette application ne fait pas partie de l'installation de base ; sous Debian, le daemon est lancé automatiquement après l'installation.

```
debian# apt install at

centos# yum install at
centos# systemctl enable atd
centos# systemctl start atd
```

Pour planifier l'exécution d'une commande, celle-ci est envoyée en texte vers le flux *STDIN* de la commande `at` ; l'heure de l'exécution est spécifiée comme argument de ligne de commande. La commande suivante peut ainsi servir à planifier un redémarrage du système samedi matin prochain à 3h30 :

```
# echo "reboot" | at 03:30 Saturday
```

La commande qui sera exécutée le samedi matin à 3h30 n'est alors pas `echo "reboot"`, mais `reboot` ; `echo` est employée seulement pour envoyer la commande à `at`.

Une autre utilisation pratique est la suivante, par exemple :

– L'administrateur système a créé des nouvelles règles pour le pare-feu, mais il n'est pas tout à fait sûr qu'elles soient bonnes.

– Avant de charger ces nouvelles règles dans le pare-feu, l'administrateur planifie une tâche *at* pour recharger les anciennes règles fonctionnelles dans le pare-feu dans cinq minutes.

```
# echo "nft -f /root/pare-feu.ok" | at now + 5 minutes
```

– Maintenant l'administrateur peut charger les nouvelles règles en toute confiance, sachant que même s'il s'exclut lui-même, dans cinq minutes, les anciennes règles seront rechargées, libérant de nouveau l'accès.

La commande `atq` (en anglais : *at queue*, file d'attente de at) sert à l'affichage des tâches planifiées de l'utilisateur ou de tous les utilisateurs si la commande est exécutée par root.