420-PPS-TT

Automatisation des tâches réseau

Planification des tâches (service crond)

5 septembre 2024

Table des matières

1 Introduction

Sous Linux, des tâches peuvent être configurées pour s'exécuter automatiquement pendant une période de temps donnée et à des dates données. Un administrateur système peut utiliser des tâches automatisées pour effectuer des copies de sauvegarde périodiques, surveiller le système, exécuter des scripts personnalisés, etc.

2 CRON

cron est un service qui permet d'automatiser l'exécution des tâches répétitives. Il se présente sous la forme d'un démon **crond** utilisant un ensemble de fichiers consignant les travaux à effectuer avec la période d'exécution associée : les fichiers "**crontab**".

La période d'exécution est une combinaison de l'heure, du jour du mois, du mois, du jour de la semaine et de la semaine.

cron suppose que le système est allumé en continu. Si le système n'est pas allumé au moment où une tâche doit être exécutée, l'exécution n'a pas lieu. Pour configurer des tâches basées sur des périodes au lieu d'heures précises, utiliser **anacron**.

Pour utiliser le service **cron**, le paquetage **cronie** doit être installé. Pour savoir si le paquetage est installé, utilisez la commande :

```
[root@localhost ~]# dnf list installed cron*

Installed Packages
cronie.x86_64              1.5.2-4.el8               @anaconda
cronie-anacron.x86_64      1.5.2-4.el8               @anaconda
crontabs.noarch            1.11-17.20190603git.el8   @anaconda
```

cron recherche dans le répertoire **/var/spool/cron**, des fichiers de **crontab** ayant des noms existants dans **/etc/passwd**. Les fichiers trouvés sont chargés en mémoire.

Ensuite, **cron** se réveille toutes les minutes, examine les **crontab** mémorisés, et vérifie chaque commande pour savoir s'il doit la lancer dans la minute à venir. Lors de l'exécution d'une commande, toute sortie est envoyée par mail au propriétaire de la **crontab** (ou à l'utilisateur dont le nom est mentionné dans la variable d'environnement **MAILTO** si elle existe).

De plus, **cron** vérifie chaque minute si la date de modification de son répertoire de stockage a changé. Si c'est le cas, **cron** examinera les dates de modifications de chaque fichier **crontab**, et rechargera ceux qui ont été changés. Ainsi, **cron** n'a pas besoin d'être redémarré après la modification d'un fichier **crontab** (Linux).

2.1 CONFIGURATION D'UNE TÂCHE CRON

Prenons comme exemple le fichier **crontab** suivant :

```
MAILTO=hakimb
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
HOME=/home/hakimb
# Commentaire

55 23 31 12 * /etc/jobs/year.sh
00 20  1  * * /etc/jobs/month.sh
30  8  *  * 1 /etc/jobs/week.sh
20 12  *  * * /etc/jobs/day.sh
40  *  *  * * /etc/jobs/hour.sh
*   *  *  * * /etc/jobs/minute.sh   %chaque minute
```

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

DÉFINITION DES VARIABLES

Les lignes d'initialisation de variables d'environnement. Elles permettent de définir l'environnement dans lequel les travaux **cron** doivent être exécutés.

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 modifiées, ce qui n'est pas le cas de **LOGNAME**.

MAILTO est une autre variable utilisée par **cron**. Si celle-ci n'est pas définie, les sorties standard et d'erreur des commandes de la **cron** table seront envoyées par mail au propriétaire de la **crontab**. 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 les sorties des commandes.

DÉFINITION DES COMMANDES

Les lignes de commandes pour le service **crond** définissent les travaux à lancer périodiquement.

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.

Chacune des lignes du fichier **crontab** a le format :

```
minute    hour day  month          dayofweek command
```

<table>
<tr><td>minute</td><td>nombre entier entre 0 et 59</td></tr>
<tr><td>hour</td><td>nombre entier entre 0 et 23</td></tr>
<tr><td>day</td><td>nombre entier entre 1 et 31</td></tr>
<tr><td>month</td><td>nombre entier entre 1 et 12 (ou le nom ANSI sur trois lettres).

<table>
<tr><td>1</td><td>jan</td><td>janvier</td></tr>
<tr><td>2</td><td>feb</td><td>février</td></tr>
<tr><td>3</td><td>mar</td><td>mars</td></tr>
<tr><td>4</td><td>apr</td><td>avril</td></tr>
<tr><td>5</td><td>may</td><td>mai</td></tr>
<tr><td>6</td><td>jun</td><td>juin</td></tr>
<tr><td>7</td><td>jul</td><td>juillet</td></tr>
<tr><td>8</td><td>aug</td><td>août</td></tr>
<tr><td>9</td><td>sep</td><td>septembre</td></tr>
<tr><td>10</td><td>oct</td><td>octobre</td></tr>
<tr><td>11</td><td>nov</td><td>novembre</td></tr>
<tr><td>12</td><td>dec</td><td>décembre</td></tr>
</table></td></tr>
<tr><td>dayofweek</td><td>nombre entier entre 0 et 7, 0 ou 7 désigne dimanche

(ou le nom ANSI sur trois lettres)

<table>
<tr><td>0</td><td>sun</td><td>dimanche</td></tr>
<tr><td>1</td><td>mon</td><td>lundi</td></tr>
<tr><td>2</td><td>tue</td><td>mardi</td></tr>
<tr><td>3</td><td>wed</td><td>mercredi</td></tr>
<tr><td>4</td><td>thu</td><td>jeudi</td></tr>
<tr><td>5</td><td>fri</td><td>vendredi</td></tr>
<tr><td>6</td><td>sat</td><td>samedi</td></tr>
<tr><td>7</td><td>sun</td><td>dimanche</td></tr>
</table></td></tr>
<tr><td>command</td><td>commande ou script à exécuter.</td></tr>
</table>

Comme vous pouvez le voir dans le fichier **crontab** précédent, le service **cron** est utilisé pour exécuter les jobs des scripts suivants :

/etc/jobs/year.sh	le 31 décembre de chaque année à 23h55
/etc/jobs/month.sh	le 1ier de chaque mois à 20h
/etc/jobs/week.sh	chaque lundi à 8h30
/etc/jobs/day.sh	chaque jour à midi 12h20 (midi 20)
/etc/jobs/hour.sh	toutes les heures 40 min (8h40, 9h40, 10h40…)
/etc/jobs/minute.sh	toutes les minutes

Pour les valeurs ci-dessus :

- Un astérisque (*****) peut être utilisé pour indiquer toutes les valeurs valides. Par exemple, un astérisque utilisé pour la valeur du mois signifie une exécution de la commande tous les mois (avec respect des restrictions des autres valeurs).
- Un trait d'union (**-**) placé entre deux nombres entiers indique une gamme de nombres entiers. Par exemple **1-4** correspond aux nombres entiers **1, 2, 3, 4**.
- Une liste de valeurs séparées par des virgules (**,**) correspond à une liste. Par exemple, **3, 4, 6, 8-12** correspond aux valeurs : **3, 4, 6, 8, 9, 10,11, 12**.
- La barre oblique (**/**) peut être utilisée pour spécifier des valeurs échelonnées. Pour passer un nombre entier dans une gamme, faites-la suivre de /<nombre entier>. Par exemple, **0-59/2** permet de définir une minute sur deux dans le champ des minutes (**0,2,4,…58**). Ces valeurs échelonnées peuvent également être utilisées avec un astérisque. Par exemple, la valeur ***/3** peut être utilisée dans le champ des mois pour passer un mois sur trois.
- Les lignes commençant par un signe dièse (#) correspondent à des commentaires et ne sont pas traitées.
- Le démon **cron** vérifie le répertoire **/var/spool/cron** toutes les minutes pour détecter les changements éventuels. S'il trouve des changements, ceux-ci sont chargés en mémoire. Ainsi, le démon n'a pas besoin d'être redémarré si un fichier **crontab** est modifié.

2.2 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.

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 d'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 de l'utilisateur juste avant la ligne de commandes à lancer.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# For details see man 4 crontabs
# 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 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

La commande **run-parts** exécute tous les scripts et programmes présents dans le répertoire passé en argument.

2.3 CRÉATION D'UN CRONTAB

Les utilisateurs peuvent configurer des tâches **cron** à l'aide de l'utilitaire ***crontab***.

La commande suivante permet à **bob** d'éditer son fichier **crontab** :

```
[bob@localhost ~]$ crontab -e

* * * * * touch /home/bob/$$
```

Le fichier **crontab** de chaque usager est stocké dans le répertoire **/var/spool/cron** et porte le même nom que l'usager.

Par exemple le fichier **crontab** de **bob** est **/var/spool/cron/bob**

```
[root@localhost ~]# ls -l /var/spool/cron/
total 4
-rw------- 1 bob bob 25 Feb 18 09:31 bob
```

Le fichier **crontab** est édité à l'aide de l'éditeur déterminé par la variable d'environnement **VISUAL** ou **EDITOR** dans l'ordre respectif.

Chaque utilisateur dispose de sa propre table **crontab**, et bien que celles-ci se trouvent dans **/var/spool/cron**, elles ne sont pas conçues pour être éditées directement.

```
[bob@localhost ~]$ ls -ltr

total 0
-rw-r--r-- 1 bob bob 0 Feb 18 10:00 2524
-rw-r--r-- 1 bob bob 0 Feb 18 10:01 2557
-rw-r--r-- 1 bob bob 0 Feb 18 10:02 3724
-rw-r--r-- 1 bob bob 0 Feb 18 10:03 3841
-rw-r--r-- 1 bob bob 0 Feb 18 10:04 3892
```

```
[root@localhost ~]# crontab -l

* * * * * rm -f /tmp/*
```

```
[root@localhost ~]# mail

Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/root": 6 messages 6 unread
>U  1 (Cron Daemon)         Fri Feb 18 10:17  30/1273  "Cron <root@localhost> rm -f /tmp/*"
 U  2 (Cron Daemon)         Fri Feb 18 10:18  30/1273  "Cron <root@localhost> rm -f /tmp/*"
 U  3 (Cron Daemon)         Fri Feb 18 10:19  30/1273  "Cron <root@localhost> rm -f /tmp/*"
 U  4 (Cron Daemon)         Fri Feb 18 10:20  30/1273  "Cron <root@localhost> rm -f /tmp/*"
 U  5 (Cron Daemon)         Fri Feb 18 10:21  30/1273  "Cron <root@localhost> rm -f /tmp/*"
 U  6 (Cron Daemon)         Fri Feb 18 10:22  30/1273  "Cron <root@localhost> rm -f /tmp/*"
```

2.4 COMMANDE crontab

L'option -l permet d'imprimer la table crontab en cours sur la sortie standard :

```
[root@localhost ~]# crontab -l

* * * * * touch /root/$$
```

Si l'option -u est indiquée, elle permet de préciser le nom de l'utilisateur dont la crontab doit être manipulée.

Si l'option -u n'est pas indiquée, crontab affichera votre table cron.

Si aucune option n'est indiquée, la commande crontab sert à installer une nouvelle table crontab, en utilisant le fichier indiqué ou l'entré standard.

```
[root@localhost ~]# crontab   jobs
```

L'option -r supprime la table crontab en cours.

```
[root@localhost ~]# crontab   -r
```

L'option -e permet d'éditer la table en cours, une fois que vous sortez de l'éditeur, la table modifiée sera installée automatiquement.

```
[root@localhost ~]# crontab   -e
```

2.5 LANCEMENT ET INTERRUPTION DU SERVICE

Pour lancer le service **crond**, utilisez la commande :

```
[root@localhost ~]# systemctl start crond
```

Pour vérifier le statut du service crond, utilisez la commande :

```
[root@localhost ~]# systemctl status crond

● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; vendor
prese>
   Active: active (running) since Fri 2022-02-18 09:14:46 EST; 13min ago
 Main PID: 1020 (crond)
    Tasks: 1 (limit: 1492)
   Memory: 820.0K
   CGroup: /system.slice/crond.service
           └─1020 /usr/sbin/crond -n

Feb 18 09:14:46 localhost.localdomain systemd[1]: Started Command Scheduler.
Feb 18 09:14:47 localhost.localdomain crond[1020]: (CRON) STARTUP (1.5.2)
```

Pour arrêter le service crond, utilisez la commande :

```
[root@localhost ~]# systemctl stop crond
```

Pour redémarrer le service crond, utilisez la commande :

```
[root@localhost ~]# systemctl restart crond
```

Pour activer le service crond afin qu'il démarre automatiquement :

```
[root@localhost ~]# systemctl enable crond
```

2.6 JOURNALISATION

```
[root@localhost ~]# journalctl -u crond

-- Logs begin at Fri 2022-02-18 10:07:56 EST, end at Fri 2022-02-18 10:12:49 EST. --
Feb 18 10:08:07 localhost.localdomain systemd[1]: Started Command Scheduler.
Feb 18 10:08:07 localhost.localdomain crond[1224]: (CRON) STARTUP (1.5.2)
Feb 18 10:08:07 localhost.localdomain crond[1224]: (CRON) INFO (RANDOM_DELAY will be scaled with
factor 90% if used.)
Feb 18 10:08:07 localhost.localdomain crond[1224]: (CRON) INFO (running with inotify support)
Feb 18 10:11:01 localhost.localdomain crond[1224]: (bob) RELOAD (/var/spool/cron/bob)
Feb 18 10:12:49 localhost.localdomain systemd[1]: Stopping Command Scheduler...
Feb 18 10:12:49 localhost.localdomain systemd[1]: crond.service: Succeeded.
Feb 18 10:12:49 localhost.localdomain systemd[1]: Stopped Command Scheduler.
Feb 18 10:12:49 localhost.localdomain systemd[1]: Started Command Scheduler.
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) STARTUP (1.5.2)
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) INFO (RANDOM_DELAY will be scaled with
factor 98% if used.)
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) INFO (running with inotify support)
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) INFO (@reboot jobs will be run at
computer's startup.)
```

L’option -u signifie systemd unit (service).

```
[root@localhost ~]# journalctl _COMM=crond

-- Logs begin at Fri 2022-02-18 10:07:56 EST, end at Fri 2022-02-18 10:36:02 EST. --
Feb 18 10:08:07 localhost.localdomain crond[1224]: (CRON) STARTUP (1.5.2)
Feb 18 10:08:07 localhost.localdomain crond[1224]: (CRON) INFO (RANDOM_DELAY will be
scaled with factor 90% if used.)
Feb 18 10:08:07 localhost.localdomain crond[1224]: (CRON) INFO (running with inotify
support)
Feb 18 10:09:01 localhost.localdomain CROND[1500]: (bob) CMD (touch /home/bob/$$)
Feb 18 10:10:01 localhost.localdomain CROND[1555]: (bob) CMD (touch /home/bob/$$)
Feb 18 10:11:01 localhost.localdomain crond[1224]: (bob) RELOAD (/var/spool/cron/bob)
Feb 18 10:11:01 localhost.localdomain CROND[1577]: (bob) CMD (touch /home/bob/$$ )
Feb 18 10:12:01 localhost.localdomain CROND[1600]: (bob) CMD (touch /home/bob/$$ )
Feb 18 10:12:49 localhost.localdomain crond[1224]: (CRON) INFO (Shutting down)
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) STARTUP (1.5.2)
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) INFO (RANDOM_DELAY will be
scaled with factor 98% if used.)
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) INFO (running with inotify
support)
Feb 18 10:12:49 localhost.localdomain crond[1616]: (CRON) INFO (@reboot jobs will be
run at computer's startup.)
Feb 18 10:13:01 localhost.localdomain CROND[1631]: (bob) CMD (touch /home/bob/$$ )
Feb 18 10:13:01 localhost.localdomain CROND[1632]: (bob) CMD (touch /tmp/$$ )
Feb 18 10:14:01 localhost.localdomain CROND[1655]: (bob) CMD (touch /home/bob/$$ )
Feb 18 10:14:01 localhost.localdomain CROND[1654]: (bob) CMD (touch /tmp/$$ )
Feb 18 10:15:01 localhost.localdomain CROND[1678]: (bob) CMD (touch /home/bob/$$ )
```

Le champ _COMM permet de filtrer par commande.

2.7 SÉCURITÉ

1) Si le fichier */etc/cron.allow* existe, alors vous devez être mentionnés dans celui-ci pour pouvoir utiliser le cron même si vous êtes root.

2) Si le fichier */etc/cron.allow* n'existe pas, mais que le fichier */etc/cron.deny* existe, alors vous ne devez pas être mentionnés dans celui-ci, si vous désirez utiliser le cron.

3) Si les deux fichiers n’existent pas alors vous n’êtes pas autorisé à utiliser le cron.

4) Si les deux fichiers existent, seul /etc/cron.allow sera considéré.