|  |  |
| --- | --- |
| 1 Cron | cron, anacron et at Automatisation des tâches |

# Introduction

L'une des tâches les plus importantes d'un bon administrateur système est de **planifier les tâches qui doivent être exécutées régulièrement**.

Par exemple, un administrateur peut **créer et automatiser des travaux pour les sauvegardes**, les mises à niveau du système et pour effectuer de nombreuses autres activités répétitives.

Pour ce faire, on peut utiliser la **fonction cron**, fonction qui est **utile pour automatiser la planification périodique** **des** **travaux**.

# Planification des tâches avec cron

Sous Linux, **cron est un dragon qui s'exécute en continu** et se **réveille toutes les minutes pour vérifier un ensemble** **de** **tables afin de trouver des tâches à exécuter**.   
Ces tables sont appelées **crontabs** et **contiennent les tâches cron**.

**cron convient aux serveurs et aux systèmes constamment sous tension**, car chaque **tâche cron n'est exécutée que si le** **système fonctionne à l'heure planifiée**.   
Il peut être utilisé par les **utilisateurs ordinaires**, chacun ayant **son propre crontab**, ainsi que par **l'utilisateur root qui** **gère les crontabs du système**.

**Remarque** …  
Sous Linux, il existe également l’**utilitaire** **anacron** qui convient aux **systèmes qui peuvent être éteints** (tels que les ordinateurs de bureau ou les ordinateurs portables).   
Il **ne peut toutefois être utilisé que par root**.   
Si la machine est éteinte lorsque les **travaux anacron doivent être exécutés**, ils **s'exécuteront la prochaine fois** que la machine sera **mise sous tension**.

## crontabs utilisateur

Les crontabs utilisateur sont des **fichiers texte qui gèrent la planification des tâches cron définies** par l'utilisateur.   
Ils **portent toujours le nom du compte utilisateur** **qui les a créés**, mais **l'emplacement de ces fichiers dépend de la** **distribution utilisée** (généralement un sous-répertoire de /var/spool/cron).

Chaque ligne d'une crontab utilisateur contient **six champs séparés par un espace** …

* La **minute de l'heure** (0-59) ;
* L'**heure** **du** **jour** (0-23) ;
* Le **jour du mois** (1-31) ;
* Le **mois de l'année** (1-12) ;
* Le **jour de la semaine** (0-7 avec dimanche=0 ou dimanche=7) ;
* La **commande à exécuter**.

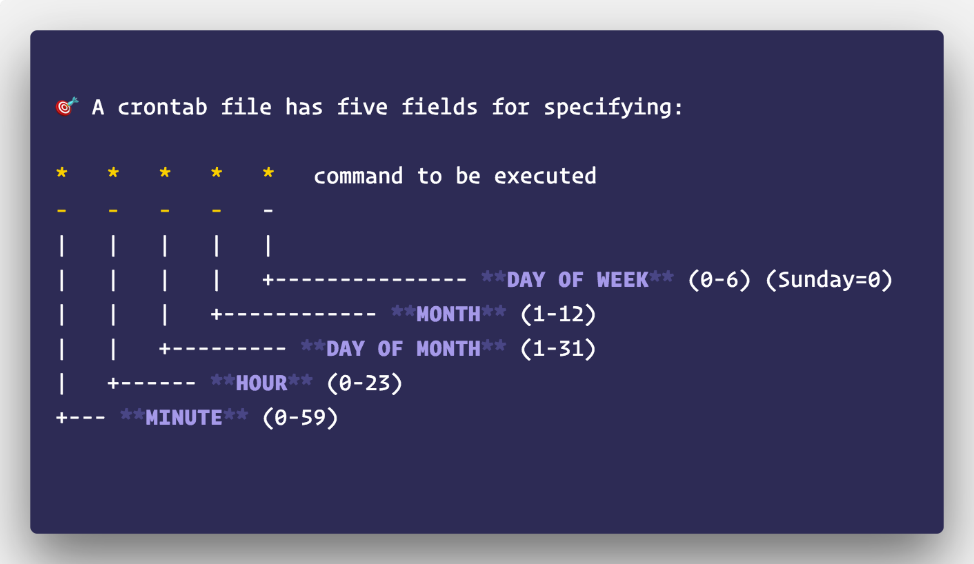
Pour le mois de l'année et le jour de la semaine, on peut utiliser les trois **premières lettres du mois au lieu du chiffre** **correspondant**.

Les cinq **premiers champs indiquent quand exécuter la commande spécifiée dans le sixième champ**, et ils peuvent contenir une ou plusieurs valeurs.

En particulier, on peut spécifier plusieurs valeurs en utilisant …

* **\* (astérisque)**Fait **référence à n'importe quelle valeur** ;
* **, (virgule)**  
  Spécifie une **liste de valeurs possibles** ;
* **- (tiret)**  
  Spécifie une **étendue de valeurs possibles** ;
* **/ (barre oblique)**  
  Spécifie des **valeurs échelonnées** ;

De nombreuses distributions incluent le **fichier /etc/crontab** qui peut être **utilisé comme référence pour** **la mise en** **page d'un fichier cron**.



Voici un exemple de fichier /etc/crontab d'une installation Debian …  
**SHELL=/bin/sh  
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin  
  
# Exemple de définition de tâche :  
# .---------------- minute (0 - 59)  
# | .------------- heure (0 - 23)  
# | | .---------- jour du mois (1 - 31)  
# | | | .------- mois (1 - 12) OR jan,feb,mar,apr ...  
# | | | | .---- jour de la semaine (0 - 6) (Dimanche=0 or 7) ou sun,mon,tue,wed,thu,fri,sat  
# | | | | |  
# \* \* \* \* \* nom d'utilisateur de la commande à exécuter.**

## crontabs système

Les crontabs système sont des **fichiers texte qui gèrent la planification des tâches cron système** et **ne peuvent être** **modifiés que par l'utilisateur root**.   
Le **fichier /etc/crontab** ainsi que tous les **fichiers du répertoire /etc/cron.d** sont des **crontabs système**.

La plupart des distributions incluent également les **répertoires /etc/cron.hourly** **et /etc/cron.daily** qui contiennent des scripts à exécuter à la **fréquence appropriée**.

Par exemple, si on souhaite **exécuter un script quotidiennement**, on peut le placer dans les **répertoires etc/cron.daily**, /**etc/cron.weekly** et **/etc/cron.monthly/**.

**Remarque** …  
Certaines distributions utilisent /etc/cron.d/hourly, /etc/cron.d/daily, /etc/cron.d/weekly et /etc/cron.d/monthly.

La **syntaxe des crontabs système** est **similaire à celle des crontabs utilisateur**, mais elle nécessite également un champ obligatoire supplémentaire qui spécifie quel utilisateur exécutera la tâche cron.

Par conséquent, chaque ligne d'une crontab système contient sept champs séparés par un espace …

* La **minute de l'heure** (0-59) ;
* L'**heure** **du** **jour** (0-23) ;
* Le **jour du m**ois (1-31) ;
* Le **mois de l'année** (1-12) ;
* Le **jour de la semaine** (0-7 avec dimanche=0 ou dimanche=7) ;
* **Nom du compte utilisateur à utiliser** lors de l'exécution de la commande ;
* La **commande ou le script à exécuter**.

Comme pour les crontabs utilisateur, on peut spécifier plusieurs valeurs pour les champs de temps à l'aide des opérateurs \*,- et /.   
On peut également indiquer le mois de l'année et le jour de la semaine avec les trois premières lettres du nom au lieu du chiffre correspondant.

## Spécifications de temps particulières

Lors de l'édition de fichiers crontab, on peut également utiliser des raccourcis spéciaux dans les cinq premières colonnes au lieu des spécifications de temps …

* **@reboot**Exécution de la tâche spécifiée une fois après le redémarrage ;
* **@hourly**  
  Exécution de la tâche spécifiée une fois par heure au début de l'heure ;
* **@daily** (ou **@midnight**)  
  Exécution de la tâche spécifiée une fois par jour à minuit ;
* **@weekly**  
  Exécution de la tâche spécifiée une fois par semaine à minuit le dimanche ;
* **@monthly**Exécution de la tâche spécifiée une fois par mois à minuit le premier jour du mois ;
* **@yearly** (ou **@annually**)  
  Exécution de la tâche spécifiée une fois par an à minuit le 1er janvier.

## Variables crontab

Dans un fichier crontab, il existe parfois des affectations de variables définies avant la déclaration des tâches planifiées. Les variables d'environnement couramment définies sont …

* **HOME**  
  Répertoire où cron invoque les commandes (par défaut le répertoire personnel de l'utilisateur) ;
* **MAILTO**  
  Nom de l'utilisateur ou l'adresse à laquelle la sortie standard et l'erreur sont envoyées (par défaut, le propriétaire de la crontab).   
  Plusieurs valeurs séparées par des virgules sont également autorisées et une valeur vide indique qu'aucun courrier ne doit être envoyé ;
* **PATH**  
  Chemin d'accès où les commandes peuvent être trouvées ;
* **SHELL**  
  Shell à utiliser (par défaut /bin/sh).

## Création de tâches cron utilisateur

La commande crontab est utilisée pour maintenir les fichiers crontab pour les utilisateurs individuels.

En particulier, on peut exécuter la commande crontab -e pour éditer son propre fichier crontab ou pour en créer un s'il n'existe pas …  
 **>> crontab -e  
Pas de crontab pour tux   
Select an editor. To change later, run 'select-editor'.  
 1. /bin/nano <---- easiest  
 2. /usr/bin/vim.tiny  
Choose 1-2 [1]:**

Par défaut, la commande crontab ouvre l'éditeur spécifié par les variables d'environnement VISUAL ou EDITOR afin que l’on puisse commencer à éditer son fichier crontab avec l'éditeur préféré.

Certaines distributions permettent de choisir l'éditeur dans une liste lorsqu'il crontab est exécuté pour la première fois.

Si on souhaite exécuter le script tux.sh situé dans son répertoire personnel tous les jours à 10h00 …  
**0 10 \* \* \* /home/tux/tux.sh**

On peut considérer les exemples d'entrées crontab suivants …  
**0,15,30,45 08 \* \* 2 ./home/tux/bar.sh  
30 20 1-15 1,6 1-5 ./home/tux/tuxbar.sh**

Dans la première ligne, le script bar.sh est exécuté tous les mardis à 08h00, à 08h15, à 08h30 et à 08h45.   
Sur la deuxième ligne, le script tuxbar.sh est exécuté à 20h30 du lundi au vendredi pendant les quinze premiers jours de janvier et juin.

**Remarque** …  
Bien que les fichiers **crontab puissent être modifiés manuellement**, il est **toujours recommandé d'utiliser la commande** **crontab**.   
Les **autorisations sur les fichiers crontab ne permettent généralement de les modifier** qu’à l’aide la commande crontab.

En plus de le commutateur -e, la commande crontab possède d'autres options utiles …

* **-l  
  Affichage de la crontab actuelle** sur la sortie standard ;
* **-r**  
  **Suppression de la crontab actuelle** ;
* **-u**  
  **Spécification du nom de l'utilisateur dont la crontab doit être modifiée**.   
  Cette option nécessite des privilèges root et permet à l'utilisateur root de modifier les fichiers crontab d'un utilisateur.

## Création de tâches cron système

Contrairement aux crontabs utilisateur, les **crontabs système sont mises** **à jour à l'aide d'un éditeur** : par conséquent, on n'a pas besoin d'exécuter la **commande crontab pour éditer /etc/crontab** et les **fichiers dans /etc/cron.d**.   
Lors de la modification des crontabs système, on doit spécifier le compte qui sera utilisé pour exécuter la tâche cron (généralement l'utilisateur root).

Par exemple, si on souhaite exécuter le script bartux.sh situé dans le répertoire /root tous les jours à 01h30 …  
**30 01 \* \* \* root /root/barfoo.sh >>/root/output.log 2>>/root/error.log**

Dans cet exemple, la sortie du travail est ajoutée au fichier /root/output.log, tandis que les erreurs sont ajoutées au fichier /root/error.log.

**Avertissement** …  
Sauf si la sortie est redirigée vers un fichier comme dans l'exemple précédent (ou si la variable MAILTO est définie sur une valeur vide), toute la sortie d'une tâche cron sera envoyée à l'utilisateur par courriel.   
Une **pratique courante consiste à rediriger la sortie standard vers /dev/null** (ou vers un fichier pour une révision ultérieure si nécessaire) et à **ne pas rediriger l'erreur standard**.   
De cette façon, l'utilisateur sera immédiatement **informé par courriel de toute erreur**.

## Configuration de l'accès à la planification des tâches

Sous Linux, les **fichiers /etc/cron.allow** et sont utilisés pour **définir des restrictions**.   
En particulier, ils sont utilisés pour **autoriser ou interdire la planification de tâches cron pour différents utilisateurs** …

* Si le **fichier /etc/cron.allow existe**, seuls les **utilisateurs non root répertoriés peuvent planifier** des tâches cron à l'aide de la commande ;
* Si le **fichier /etc/cron.deny** **existe**, seuls les **utilisateurs non-root répertoriés dans ce fichier ne peuvent pas planifier de tâches cron** à l'aide de la commande ;  
  (dans ce cas, un vide signifie que chaque utilisateur est autorisé à planifier des tâches cron) ;
* Si **aucun des deux fichiers existe**, l'accès de l'utilisateur à la **planification des tâches cron dépend de la** **distribution utilisée**.

**Remarque** …  
Les **fichiers /etc/cron.allow** et **/etc/cron.deny** contiennent une **liste de noms d'utilisateur**, **chacun sur une ligne** **distincte**.

## Alternative à cron

En utilisant systemd en tant que gestionnaire de système et de service, il est possible de **définir des minuteries comme alternative à cron pour la planification des tâches**.

Les **temporisateurs** (**timers**) sont des **fichiers d'unité systemd identifiés par le suffixe .timer**.  
Pour chacun d'entre eux, il doit y avoir un **fichier d'unité correspondant qui décrit l'unité à activer** lorsque le temporisateur s'est écoulé.   
Par défaut, un **temporisateur active un service portant le même nom**, **à l'exception du suffixe**.

Un temporisateur comprend une **section [Timer]** qui **spécifie quand les tâches planifiées doivent s'exécuter**.   
Plus précisément, on peut utiliser la **directive OnCalendar=** afin de définir des **minuteries en temps réel** qui fonctionnent de la même manière que les tâches cron (elles sont basées sur des expressions d'événements de calendrier).

La directive OnCalendar= requiert la syntaxe suivante …  
**<Jour de la semaine> <Année-Mois-Jour> <Heure:Minute:Seconde>**La directive <Jour de la semaine> étant facultatif.

Les **opérateurs \*,-/** et ont la **même signification que ceux utilisés pour les tâches cron**, tandis que l’on peut **utiliser ..** **entre deux valeurs pour indiquer une étendue contiguë**.   
Pour la **spécification <Jour de la semaine>**, il est possible d’utiliser les **trois premières lettres du nom ou** **le nom** **complet**.

**Remarque** …  
On peut également définir des **minuteries monotones qui s'activent après un certain temps** **à partir d'un point de** **départ spécifique** (par exemple, lorsque l’hôte a été démarrée ou lorsque la minuterie elle-même est activée).

Par exemple, si on souhaite exécuter le service nommé /etc/systemd/system/tuxbar.service à 05h30 le premier lundi de chaque mois, on peut ajouter les lignes suivantes dans le fichier d’unité /etc/systemd/system/tuxbar.timer ...  
**[Unit]  
Description=Démarrage du service tuxbar**

**[Timer]  
OnCalendar=Mon \*-\*-1..7 05:30:00  
Persistent=true**

**[Install]  
WantedBy=timers.target**

Une fois que l’on a créé le nouveau minuteur, on peut l'**activer** **et le démarrer** en exécutant les commandes suivantes en tant que root …  
**>> sudo systemctl enable tuxbar.timer  
>> sudo systemctl start tuxbar.timer**

On peut **modifier la fréquence de la tâche planifiée**, en modifiant la **valeur OnCalendar**, puis en entrant la **commande** **systemctl daemon-reload**.

Enfin, si on souhaite **afficher la liste des minuteries actives triées par le temps** qu'elles s'écoulent ensuite, on peut utiliser la **commande systemctl list-timers**.   
On peut également ajouter le commutateur --all pour afficher les unités de minuterie inactives.

**Remarque** …  
Les **minuteries sont enregistrées dans le journal systemd** et que on peut **consulter les journaux des différentes unités** à l'aide de la **commande journalctl**.   
Si on agit en tant qu'utilisateur ordinaire, on doit utiliser le **commutateur --user des commandes systemctl et journalctl**.

Au lieu de la forme normalisée plus longue mentionnée ci-dessus, il est possible d’**utiliser** **des expressions spéciales** qui décrivent des **fréquences particulières** pour l'exécution des tâches …

* **hourly**  
  Exécution de la tâche spécifiée **une fois par heure au début de l'heure** ;
* **daily**  
  Exécution de la tâche spécifiée **une fois par jour à minuit** ;
* **weekly**  
  Exécution de la tâche spécifiée **une fois par semaine à minuit le lundi** ;
* **monthly**  
  Exécution de la tâche spécifiée **une fois par mois à minuit le premier jour du mois** ;
* **yearly**  
  Exécution de la tâche spécifiée **une fois par an à minuit le premier janvier**.

# Planification des travaux avec at

On doit parfois exécuter une **tâche à un moment précis dans le futur**, **une seule fois**.  
Pour ce faire, on peut utiliser un utilitaire puissant : la **commande at**.

La commande at est utilisée pour la **planification de tâches ponctuelles** et nécessite uniquement que l’on **spécifie quand le travail doit être exécuté à l'avenir**.

Après avoir entré at en **ligne de commande suivie de la spécification de l'heure**, on entre dans l'invite at où on peut définir les commandes à exécuter.   
On peut quitter l'invite at avec la séquence de **touches CTRL+D** …  
**>> at now +5 minutes  
warning: commands will be executed using /bin/sh  
at> date  
at> Ctrl+D  
job 12 at Sat Sep 14 09:15:00 2019**

Le travail at dans l'exemple précédent exécute simplement la commande date après cinq minutes.   
Comme pour cron, la **sortie standard et l'erreur sont envoyées par courriel**.   
Le **démon atd devra être en cours d'exécution** sur le système pour que l’on puisse utiliser la planification des tâches at.

**Remarque** …  
Sous Linux, la **commande batch est similaire à at**.  
Toutefois, avec **batch les travaux ne sont exécutés que lorsque la charge du système est suffisamment faible** pour le permettre.

Les options les plus importantes qui s'appliquent à la commande at sont …

* **-c**  
  **Affichage des commandes d'un ID de tâche spécifique** sur la sortie standard ;
* **-d**  
  **Suppression des travaux en fonction de leur ID** de travail.   
  Cette option est un alias pour atrm ;
* **-f**  
  **Lecture du travail à partir d'un fichier** au lieu de l'entrée standard ;
* **-l**  
  **Affichage des tâches en attente** de l'utilisateur.   
  Si l'utilisateur est root, tous les travaux de tous les utilisateurs sont répertoriés.   
  Cette option est un alias pour atq.
* **-m**  
  **Envoie d’un courriel à l'utilisateur** à la fin du travail même s'il n'y a pas eu de sortie ;
* **-q**  
  **Spécification d’une file d'attente sous la forme d'une seule lettre** de a à z et de A à Z   
  (par défaut a pour at et b pour batch).   
  Les travaux dans les files d'attente avec les lettres les plus hautes sont exécutés avec une précision accrue.   
  Les travaux soumis à une file d'attente avec une majuscule sont traités comme des travaux batch ;
* **-v**  
  **Affichage de l'heure à laquelle le travail s'exécutera** avant la lecture du travail.

## Répertorier les tâches planifiées avec atq

On peut planifier deux at tâches supplémentaires …

* la première exécute le script tu.sh à 09h30 ;
* tandis que la seconde exécute le script bar.sh après une heure.

**>> at 09h30  
warning: commands will be executed using /bin/sh  
at>** **./foo.sh**  
**at>** **CTRL+D**  
**job 13 at Sat Sep 14 09:30:00 2019**

**>>** **at now +2 hours  
warning: commands will be executed using /bin/sh  
at>** **./bar.sh  
at>** **Ctrl+D  
job 14 at Sat Sep 14 11:10:00 2019**

Pour **répertorier les tâches en attente**, on peut utiliser la **commande atq** qui **affiche les information**s suivantes pour chaque tâche …

* **ID de la tâche** ;
* **date d'exécution de la tâche** ;
* **heure d'exécution de la tâche** ;
* **file d'attente**   
  et
* **nom d'utilisateur**.

**Remarques** …  
La **commande at -l est un alias pour atq**.  
Si on exécute **atq en tant que root**, il **affichera les travaux en file d'attente pour tous les utilisateurs**.

## Suppression des travaux avec atrm

Si on souhaite **supprimer une tâche at**, on peut utiliser la **commande atrm suivie de l'ID de la tâche**.

Par exemple, pour supprimer la tâche avec l'ID 14 …  
**>>** **atrm 14**

Il est possible de supprimer plusieurs travaux avec **atrm en spécifiant plusieurs ID séparés par des espaces**.

**Remarques** …  
La **commande at -d est un alias pour atrm**.  
Si on exécute **atrm en tant que root**, on peut **supprimer les travaux de tous les utilisateurs**.

# Configuration de l'accès à la planification des tâches

L'**autorisation** **pour les utilisateurs ordinaires de planifier des travaux at** est **déterminée par les fichiers /etc/at.allow** et **/etc/at.deny**.

* Si le **fichier /etc/at.allow existe**, seuls les **utilisateurs non root répertoriés peuvent planifier des tâches** à l'aide de la commande at ;
* Si le **fichier /etc/at.deny existe**, seuls les **utilisateurs non-root répertoriés dans ce fichier ne peuvent pas** **planifier de tâches** à l'aide de la commande at ;  
  (dans ce cas, un vide signifie que chaque utilisateur est autorisé à planifier des tâches at) ;
* Si **aucun de ces deux fichiers existe**, l'**accès** de l'utilisateur à la planification des tâches **dépend de la distribution utilisée**.

# Spécifications de temps

On peut **spécifier le moment exécuter une tâche** at particulière à l'aide de la **forme HH:MM**, éventuellement suivi de **AM ou PM en cas de format 12 heures**.   
Si l'**heure spécifiée** **s'est déjà écoulée**, le **jour suivant est pris en compte**.

Si on souhaite **planifier une date particulière** à laquelle le travail s'exécutera, on doit **ajouter les informations de date** **après l'heure** à l'aide de l'une des formes suivantes …

* **<Mois> <Jour>** ;
* **<Mois> <JOUR> <Année>** ;
* **MMJJAA** ;
* **MM/JJ/AA** ;
* **JJ.MM.AA**   
  et
* **AAAA-MM-JJ**.

Les mots-clés suivants sont également acceptés …

* **midnight** ;
* **noon** ;
* **teatime** (16 h)   
  et
* **now suivis d'un signe plus** (+) et d'une **période** (minutes, heures, jours et semaines).

Enfin, on peut indiquer à at d'exécuter le travail aujourd'hui ou demain en suffixant l'heure avec les **mots today ou tomorrow**.

# Une alternative à at

En utilisant systemd en tant que gestionnaire de système et de service, il est également possible de **planifier des tâches** **ponctuelles avec la commande systemd-run**.   
Il est généralement utilisé pour **créer une unité de temporisation transitoire** afin qu'une commande soit exécutée à un moment précis **sans qu'il soit nécessaire de créer un fichier de service**.

Par exemple, en tant qu'utilisateur root, on peut exécuter la commande date à 11h30 le 06/10/2022 …  
**>>** **sudo systemd-run --on-calendar='2022-10-06 11:30' date**

Si on souhaite exécuter le script tux.sh, situé dans le répertoire de travail actuel, après deux minutes, …  
**>>** **sudo systemd-run --on-active="2m" ./tux.sh**

**Remarque** …  
Les **minuteries sont enregistrées dans le journal systemd** et qu’il est possible de consulter les journaux des différentes unités à l'aide de la **commande journalctl**.   
Si on agit en tant qu'**utilisateur ordinaire**, il faut utiliserl e **commutateur --user avec les commandes systemd-run et journalctl**.