## Un peu de configuration…

La « crontab » constitue un incontournable sous Linux : cet outil nous permet de programmer l'exécution régulière d'un programme.
Contrairement à `at` qui n'exécutera le programme qu'une seule fois, `crontab` permet de faire en sorte que l'exécution soit répétée : toutes les heures, toutes les minutes, tous les jours, tous les trois jours, etc.

Avant toute chose, nous devons modifier notre configuration (notre fichier `.bashrc`) pour demander à ce que Nano soit l'éditeur par défaut. En général, c'est le programme « vi » qui fait office d'éditeur par défaut. C'est un bon éditeur de texte, mais bien plus complexe que Nano et je ne vous le présenterai que plus tard.
En attendant, rajoutez la ligne suivante à la fin de votre fichier `.bashrc` :

```
export EDITOR=nano
```

Vous pouvez aussi écrire la commande suivante :

```
$ echo "export EDITOR=nano" >>  ~/.bashrc
```

Cela aura pour effet d'écrire cette ligne à la fin de votre fichier `.bashrc` situé dans votre répertoire personnel.
Fermez ensuite votre console et rouvrez-la pour que cette nouvelle configuration soit bien prise en compte.

Cette petite configuration étant faite, attaquons les choses sérieuses.

### La « crontab », qu'est-ce que c'est ?

`crontab` est en fait une commande qui permet de lire et de modifier un fichier appelé la « crontab ».
Ce fichier contient la liste des programmes que vous souhaitez exécuter régulièrement, et à quelle heure vous souhaitez qu'ils soient exécutés.

`crontab` permet donc de changer la liste des programmes régulièrement exécutés. C'est toutefois le programme `cron` qui se charge d'exécuter ces programmes aux heures demandées.
Ne confondez donc pas `crontab` et `cron` : le premier permet de modifier la liste des programmes à exécuter, le second les exécute.

Comment utilise-t-on `crontab` ?
Il y a trois paramètres différents à connaître, pas plus :

- `-e` : modifier la crontab ;
- `-l` : afficher la crontab actuelle ;
- `-r` : supprimer votre crontab. Attention, la suppression est immédiate et sans confirmation !

Commençons par afficher la crontab actuelle :

```
$ crontab -l
no crontab for mateo21
```

Normalement, vous n'avez pas encore créé de crontab. Vous noterez qu'il y a une crontab par utilisateur. Là j'édite la crontab de mateo21 car je suis loggé avec l'utilisateur mateo21, mais root a aussi sa propre crontab. La preuve :

```
$ sudo crontab -l
no crontab for root
```

Bien, intéressons-nous à la modification de la crontab. Tapez :

```
$ crontab -e
```

Si vous avez bien configuré votre `.bashrc` tout à l'heure (et que vous avez relancé votre console), cela devrait ouvrir le programme Nano que vous connaissez.
Si par hasard vous n'avez pas fait quelque chose correctement, c'est le programme « vi » qui se lancera. Comme vous ne le connaissez pas encore, tapez `:q` puis `Entrée` pour sortir. Vérifiez à nouveau votre configuration du `.bashrc` et n'oubliez pas de fermer puis de rouvrir votre console.

### Modifier la crontab

Pour le moment, si votre crontab est vide comme la mienne, vous devriez voir uniquement ceci (capture d'écran de Nano) :

```
GNU nano 2.0.7       Fichier : /tmp/crontab.4u4jHU/crontab



# m h  dom mon dow   command

                              [ Lecture de 1 ligne ]
^G Aide      ^O Écrire    ^R Lire fich.^Y Page préc.^K Couper    ^C Pos. cur.
^X Quitter   ^J Justifier ^W Chercher  ^V Page suiv.^U Coller    ^T Orthograp.
```

#### Les champs

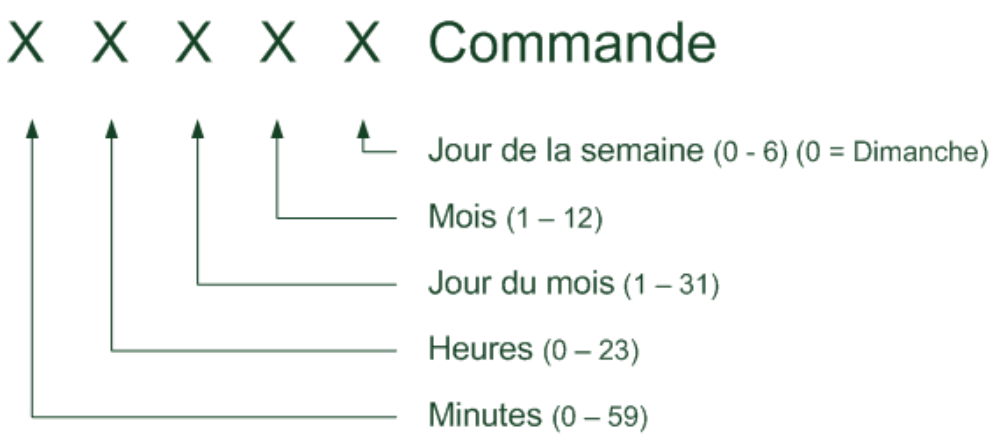
Le fichier ne contient qu'une seule ligne :

```
# m h  dom mon dow   command
```

Comme cette ligne est précédée d'un #, il s'agit d'un commentaire (qui sera donc ignoré).
Cette ligne vous donne quelques indications sur la syntaxe du fichier :

- `m` : minutes (0 - 59) ;
- `h` : heures (0 - 23) ;
- `dom` (*day of month*) : jour du mois (1 - 31) ;
- `mon` (*month*) : mois (1 - 12) ;
- `dow` (*day of week*) : jour de la semaine (0 - 6, 0 étant le dimanche) ;
- `command` : c'est la commande à exécuter.

Chaque ligne du fichier correspond à une commande que l'on veut voir exécutée régulièrement. Vous trouverez en figure suivante un schéma qui résume la syntaxe d'une ligne.




Crontab

En clair, vous devez d'abord indiquer à quel moment vous voulez que la commande soit exécutée, puis ensuite écrire à la fin la commande à exécuter.
C'est un peu comme un tableau. Chaque champ est séparé par un espace.

Chaque « X » sur mon schéma peut être remplacé soit par un nombre, soit par une étoile qui signifie « tous les nombres sont valables ».

Bien comprendre la crontab n'est pas si simple, je vous propose donc de nous baser sur quelques exemples pour voir comment ça fonctionne.

Imaginons que je veuille exécuter une commande tous les jours à 15 h 47. Je vais écrire ceci :

```
47 15 * * * touch /home/mateo21/fichier.txt
```

Seules les deux premières valeurs sont précisées : les minutes et les heures. Chaque fois qu'il est 15 h 47, la commande indiquée à la fin sera exécutée.

J'ai écrit le chemin du fichier en entier, car vous ne pouvez pas être sûrs que le `cron` s'exécutera dans le répertoire que vous voulez. Il est donc toujours préférable d'écrire le chemin du fichier en absolu comme je l'ai fait ici : `/home/mateo21/fichier.txt`.

*Au fait, pourquoi passer par la commande* ***`crontab -e`*** *pour modifier un fichier ? Il ne serait pas plus simple d'ouvrir le fichier directement avec* ***`nano .crontab`****, par exemple ?*

Oui, mais ce n'est pas comme cela que ça fonctionne. La crontab exige de passer par une commande, c'est comme ça.
Il y a quelques avantages à cela, puisque cela permet au programme de vérifier si votre fichier est correctement écrit avant de mettre à jour la crontab. S'il y a une erreur de syntaxe, on vous le dira et aucun changement ne sera apporté.

Essayez d'enregistrer et de quitter Nano. Vous verrez que la crontab vous dit qu'elle « installe » les changements (elle les prend en compte, en quelque sorte) :

```
crontab: installing new crontab
mateo21@mateo21-desktop:~$
```

Désormais, `fichier.txt` sera créé dans mon répertoire personnel tous les jours à 15 h 47 (s'il n'existe pas déjà).

Revenez dans la crontab, nous allons voir d'autres exemples (tableau suivante).

| Crontab | Signification |
|---|---|
| `47 * * * * commande` | Toutes les heures à 47 minutes exactement.> & Donc à 00 h 47, 01 h 47, 02 h 47, etc. |
| `0 0 * * 1 commande` | Tous les lundis à minuit (dans la nuit de dimanche à lundi). |
| `0 4 1 * * commande` | Tous les premiers du mois à 4 h du matin. |
| `0 4 * 12 * commande` | Tous les jours du mois de décembre à 4 h du matin. |
| `0 * 4 12 * commande` | Toutes les heures les 4 décembre. |
| `* * * * * commande` | Toutes les minutes ! |

*Est-il possible d'exécuter une commande plus fréquemment que toutes les minutes ?*

Non, c'est impossible avec `cron`. La fréquence minimale, c'est toutes les minutes.

### Les différentes notations possibles

Pour chaque champ, on a le droit à différentes notations :

- `5` (un nombre) : exécuté lorsque le champ prend la valeur 5 ;
- `*` : exécuté tout le temps (toutes les valeurs sont bonnes) ;
- `3,5,10` : exécuté lorsque le champ prend la valeur 3, 5 ou 10. Ne pas mettre d'espace après la virgule ;
- `3-7` : exécuté pour les valeurs 3 à 7 ;
- `*/3` : exécuté tous les multiples de 3 (par exemple à 0 h, 3 h, 6 h, 9 h…).

Vous connaissiez déjà les deux premières notations. Celles que nous venons de découvrir nous permettent de démultiplier les possibilités offertes par la crontab.
Voici, sur le tableau suivante, quelques exemples d'utilisation.

| Crontab | Signification |
|---|---|
| `30 5 1-15 * * commande` | À 5 h 30 du matin du 1er au 15 de chaque mois. |
| `0 0 * * 1,3,4 commande` | À minuit le lundi, le mercredi et le jeudi. |
| `0 */2 * * * commande` | Toutes les 2 heures (00 h 00, 02 h 00, 04 h 00…) |
| `*/10 * * * 1-5 commande` | Toutes les 10 minutes du lundi au vendredi. |

Comme vous le voyez, la crontab offre de très larges possibilités (pour peu que l'on ait compris comment elle fonctionne).

### Rediriger la sortie

Pour le moment, nous avons exécuté notre commande très simplement dans la crontab :

```
47 15 * * * touch /home/mateo21/fichier.txt
```

Toutefois, il faut savoir que si la commande renvoie une information ou une erreur, vous ne la verrez pas apparaître dans la console. Normal : ce n'est pas vous qui exécutez la commande, mais le

programme `cron`.

Que se passe-t-il alors si la commande renvoie un message ? En fait, le résultat de la commande vous est envoyé par e-mail. Chaque utilisateur possède sa propre boîte e-mail sur les machines de type Unix, mais je ne vais pas m'attarder là-dessus. Nous allons plutôt voir comment rediriger le résultat.

Tenez : rediriger une sortie, vous savez faire ça, non ?

```
47 15 * * * touch /home/mateo21/fichier.txt >> /home/mateo21/cron.log
```

Tous les messages seront désormais ajoutés à la fin de `cron.log`. Tous ? Non, on oublie d'y rediriger aussi les erreurs !

```
47 15 * * * touch /home/mateo21/fichier.txt >> /home/mateo21/cron.log 2>&1
```

Voilà, c'est mieux.
Cette fois, tout sera envoyé dans `cron.log` : les messages et les erreurs.

*Et si je ne veux pas du tout récupérer ce qui est affiché ?*

Nous avons déjà appris à le faire ! Il suffit de rediriger dans `/dev/null` (le fameux « trou noir » du système). Tout ce qui est envoyé là-dedans est immédiatement supprimé : hop, plus de trace, le crime parfait.

```
47 15 * * * touch /home/mateo21/fichier.txt > /dev/null 2>&1
```

### En résumé

- `date` permet d'obtenir la date et l'heure mais aussi de modifier celles-ci.
- `at` retarde l'exécution d'une commande à une heure ultérieure.
- On peut exécuter plusieurs commandes d'affilée en les séparant par des points-virgules : `touch fichier.txt; rm fichier.txt`.
- La commande `sleep` permet de faire une pause entre deux commandes exécutées d'affilée.
- `crontab` permet de programmer des commandes pour une exécution régulière. Par exemple : tous les jours à 18 h 30, tous les lundis et mardis à 12 h, tous les 5 du mois, etc. On modifie la programmation avec `crontab -e`.