

## Administration & Sécurité des Systèmes d'Exploitation



# Planification des tâches avec Cron

Par: L'équipe  
administration  
système

# Introduction



**Cron** est un ***daemon*** utilisé pour programmer des tâches devant être exécutées à un moment précis.

Chaque utilisateur a un fichier **crontab**, lui permettant d'indiquer **les actions** et à quelles périodes, elles devront **être exécutées**.

Il y a également un fichier **crontab** pour le système, permettant les tâches techniques, pour la mise à jour des différents programmes ou autres besoins périodiques.



# **System CRON jobs**

# System CRON jobs

Le fichier de configuration du **cron** est **/etc/crontab**

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

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

# System CRON jobs



Il est conseillé (et plus facile) de mettre les scripts à lancer dans les dossiers préconfigurer :

*/etc/cron.hourly* les scripts qui existent dans ce dossier se lancent chaque heure.

*/etc/cron.daily*

*/etc/cron.weekly*

*/etc/cron.monthly*

# System CRON jobs

Pour utiliser **cron**, ajoutez simplement les entrées à votre fichier **/etc/crontab**, Chaque ligne représente une commande à exécuter.

La table de configuration doit être remplie de la façon

```
%mm %hh %jj %MM %JJ user commande > log
```

<b>mm</b> : les minutes 0-59	<b>JJ</b> : le jour de la semaine 0-7 0 et 7 représentent le dimanche
<b>hh</b> : les heures 00-23	<b>user</b> : le nom de l'utilisateur
<b>jj</b> : le numéro de jour du mois 1-31	<b>Commande</b> : commande à lancer
<b>MM</b> : le numéro du mois 1-12	<b>Log</b> : le nom du fichier log

# Formalisme

Utilisez le format suivant pour les valeurs périodiques :

- Une valeur pour indiquer quand il faut exécuter la commande. **Ex : la valeur 15** dans le champ minute signifie la quinzième minute.
- Une liste de valeurs séparées par des virgules. **Ex :** **1,4,7,10** dans le champ mois pour janvier, avril , juillet , octobre.
- Un intervalle de valeurs. **Ex : 1-5** dans le champ jour de la semaine indique du lundi (1) au vendredi (5) . Le 0 est le dimanche et le 6 le samedi.
- Le caractère \* pour toutes les valeurs possibles. **Ex : \*** dans le champ jour du mois indique tous les jours du ou des mois



# System CRON jobs

## Exemples

```
1 2 3 4 * root /usr/bin/apt-get update
```

Cela va exécuter la commande **/usr/bin/apt-get update** tous les Avril (4), qui tombent un 3 Avril (3), à 2h01.

une étoile (toutes les valeurs) :

```
* 2 * 4 5 root /usr/bin/apt-get update
```

tous les vendredis (5), d'Avril (4), à 2h (2), et toutes les minutes (\*).



# System CRON jobs

## Une plage:

```
1 2 * 4 5-7 root /usr/bin/apt-get update
```

Tous les vendredis, samedis et dimanches (5-7), d'Avril (4), à 2h01

## un multiple :

```
*/10 2 3 4 * root /usr/bin/apt-get update
```

Tous les Avril (4), qui tombent un 3 Avril (3), à 2h et toutes les 10 minutes (\*/10).

# System CRON jobs

## Une plage:

```
1 2 3 4 3,5,7 root /usr/bin/apt-get update
```

tous les mercredis, vendredis et dimanches (4,5,7), d'Avril (4), qui tombent un 3 Avril (3), à 2h01.

```
1 2 3 4 * root run-parts /home/salah/mes_taches
```

L'option **run-parts** est utilisée pour lancer un ensemble de commandes dans un répertoire bien déterminé.

# System CRON jobs

Il existe des raccourcis intéressants :

@reboot	Run once, at startup.
@yearly	Run once a year, "0 0 1 1 *".
@annually	(same as @yearly)
@monthly	Run once a month, "0 0 1 * *".
@weekly	Run once a week, "0 0 * * 0".
@daily	Run once a day, "0 0 * * *".
@midnight	(same as @daily)
@hourly	Run once an hour, "0 * * * *".

# Exercice

Rédiger la ligne crontab pour :

1) Exécution de **df** tous les jours, toute l'année, toute les 2 minutes :

```
*/2 * * * * root df >> /tmp/libre
```

2) Exécution du script **fin\_travail.sh** qui se trouve dans le repertoire /root/script tous les jours ouvrables (de lun a vend) à 17h00 heures :

```
0 17 * * 1-5 root run-parts  
/root/script
```

# Remarque

\$ ls cron.daily

```
00-logwatch 0anacron makewhatis.cron slocate.cron  
00webalizer logrotate rpm tmpwatch
```

Parmi les programmes exécutés, remarquez **logrotate** qui permet d'effectuer des sauvegardes et de renommer des fichiers logs et des journaux du système.

Le programme **tmpwatch** est chargé de nettoyer le système des fichiers inutilisés (dans /tmp par exemple).

# User CRON jobs

## La sécurité dans CRON :

**/etc/cron.allow**

**/etc/cron.deny**

Si le fichier **/etc/cron.allow** existe, alors vous devez être présent dans ce fichier pour être autorisé à utiliser cette commande.

Si le fichier **/etc/cron.allow** n'existe pas mais que **/etc/cron.deny** existe, alors vous ne devez pas être mentionné dans le fichier **/etc/cron.deny** afin de pouvoir utiliser cette commande.

Si les deux fichiers sont absents, **seul root** peut utiliser cron.





# User CRON jobs



# User CRON jobs

Tous les utilisateurs peuvent planifier l'exécution de tâches. c'est pourquoi chacun dispose de sa propre **crontab**.

Un utilisateur peut éditer les commandes à planifier :  
**crontab -e**

Afficher la liste des tâches programmées

**crontab -l**

**00 \* \* \* \* /usr/bin/xclock -display :0.0**

Supprimer la table cron

**crontab -r**

# User CRON jobs

**Pour l'administrateur root :**

Afficher la liste des tâches programmées d'un utilisateur :

```
#crontab -u salah -l
```

```
00 * * * * /usr/bin/xclock -display :0.0
```

Supprimer la table cron d'un utilisateur :

```
# crontab -u salah -r
```

# KCRON

Quelques outils permettent d'éditer une crontab de manière visuelle sans passer par un éditeur de texte.

L'outil **kcron** sous KDE est très populaire et très bien adapté pour cela.

**Edit Task - KCron**

Comment: Getip DyNS IP Updater

Program: /home/cray/bin/getip Browse...

☒ Enabled

☒ Silent

**Months**

- ☒ January
- ☒ February
- ☒ March
- ☒ April
- ☒ May
- ☒ June
- ☒ July
- ☒ August
- ☒ September
- ☒ October
- ☒ November
- ☒ December

**Days of Month**

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

**Days of Week**

- ☒ Monday
- ☒ Tuesday
- ☒ Wednesday
- ☒ Thursday
- ☒ Friday
- ☒ Saturday
- ☒ Sunday

**Daily**

☒ Run every day

**Hours**

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23

AM PM

**Minutes**

0	5	10	15	20	25
30	35	40	45	50	55

OK Cancel



# **ANACRON jobs**

## **anachronistic command scheduler**

# ANACRON jobs



l'inconvénient du système **CRON** est si le système est stoppé au moment de l'exécution de la tâche, celle-ci est ignorée.

**Anacron** est donc né pour permettre l'exécution de tâches sans nécessité de laisser la machine allumée 24h/24.

Notez bien qu'anacron n'est pas du tout destiné à remplacer cron, il est complémentaire et il coexiste avec lui...

# ANACRON jobs



Anacron utilise des indications de temps relatives (« une fois par jour / par semaine / par mois ») au lieu de références temporelles absolues (« le 14 janvier 2008 à 15h 30 »).

De la sorte, même si vous « manquez » un moment ou une date particulière où l'exécution d'un « job » était prévue, celui-ci sera tout de même exécuté peu de temps après le prochain démarrage du système.



# ANACRON jobs



Anacron est un **programme** « standard » (c'est pas un démon / service), il est lancé :

- Au **démarrage du système** via un script d'init (/etc/init.d/anacron)
- Via le **crontab système**, on verra ce dernier point lors de la description des interaction cron/anacron

Le fichier de configuration d 'anacron est  
**/etc/anacrontab**



# ANACRON jobs

Le fichier de configuration d'anacron est **/etc/anacrontab**

```
#####  
#périodicité (jours),                                     #  
#|  délai (minutes),                                     #  
#|  |  nom de la tâche,                                   #  
#|  |  |  commandes                                     #  
#####  
1 5 Montre /usr/bin/xclock -display :0.0
```

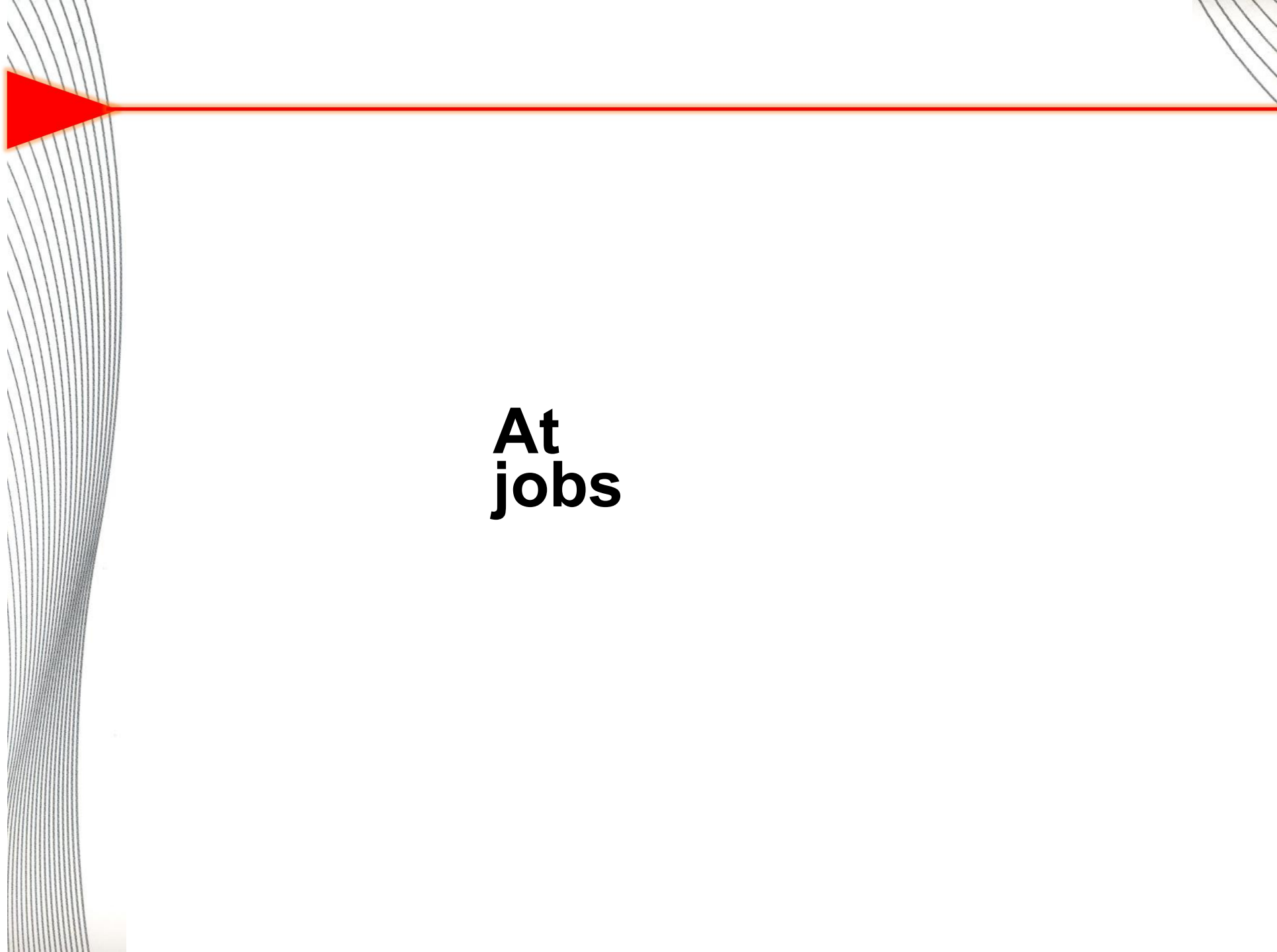
Dans l'exemple ci-dessus la commande xclock sera exécutée tous les jours 5 minutes après le démarrage d'anacron.

# ANACRON jobs

Si la machine reboot plusieurs fois dans la journée, anacron est exécuté plusieurs fois.


Afin de ne pas lancer une tâche quotidienne à chaque reboot anacron stock la date (format AAAAMMJJ) de sa dernière exécution dans un **fichier de « log »** portant le nom de la tâche et présent dans **/var/spool/anacron**.

Lorsqu'il parcourt les tâches qui lui sont affecté, anacron vient donc lire le fichier de « log » correspondant, compare la date du jour, la date de dernière exécution et l'intervalle de la tâche et la lance le cas échéant.



**At  
jobs**

# at et le demon atd



Atd est un **démon** qui s'occupe des commandes à exécuter une seule fois à un instant précis et futur

De nombreuses tâches sont régulièrement planifiées :

- la rotation des logs;
- mise à jour de la base de données du programme locate;
- les sauvegardes;
- des scripts d'entretien (comme le nettoyage des fichiers temporaires).

# Syntaxe

## dans 2 minutes :

# at now + 2 minutes

*Entrée*

**at>** ls /bin > /fich.log *Entrée*

*"Ctrl+D"*

## à 13h37 :

# at 13:37

*Entrée*

**at>** ls /bin > /fich.log *Entrée*

*"Ctrl+D"*

# at et le demon atd

La commande :

`atq` pour voir la liste de taches

`atrm num_tache` : pour supprimer une tache

**Vous voulez obtenir des informations sur l'état du système de fichiers toutes les **10 minutes aujourd'hui** pour vous aider à étudier certaines questions de performance.**

**Vous soupçonnez que cela peut venir de la mémoire ou des E/S et vous voulez garder un œil sur ces ressources.**

**En tant que super-utilisateur, utilisez la commande crontab -e pour éditer votre fichier cron.**

**1) Saisissez la ligne suivante dans votre fichier crontab :**

```
*/10 8-17 * * * root /usr/bin/free>>/f1 ;  
/usr/bin/iostat>>/f1
```

**2) Créez une entrée cron qui démarre xclock chaque 2 minutes.**

**3) Utilisez at pour démarrer xclock dans les 5 minutes prochaines**