MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université Paris Cité

Chapitre 6: Automatisation des tâche

Chérifa Boucetta

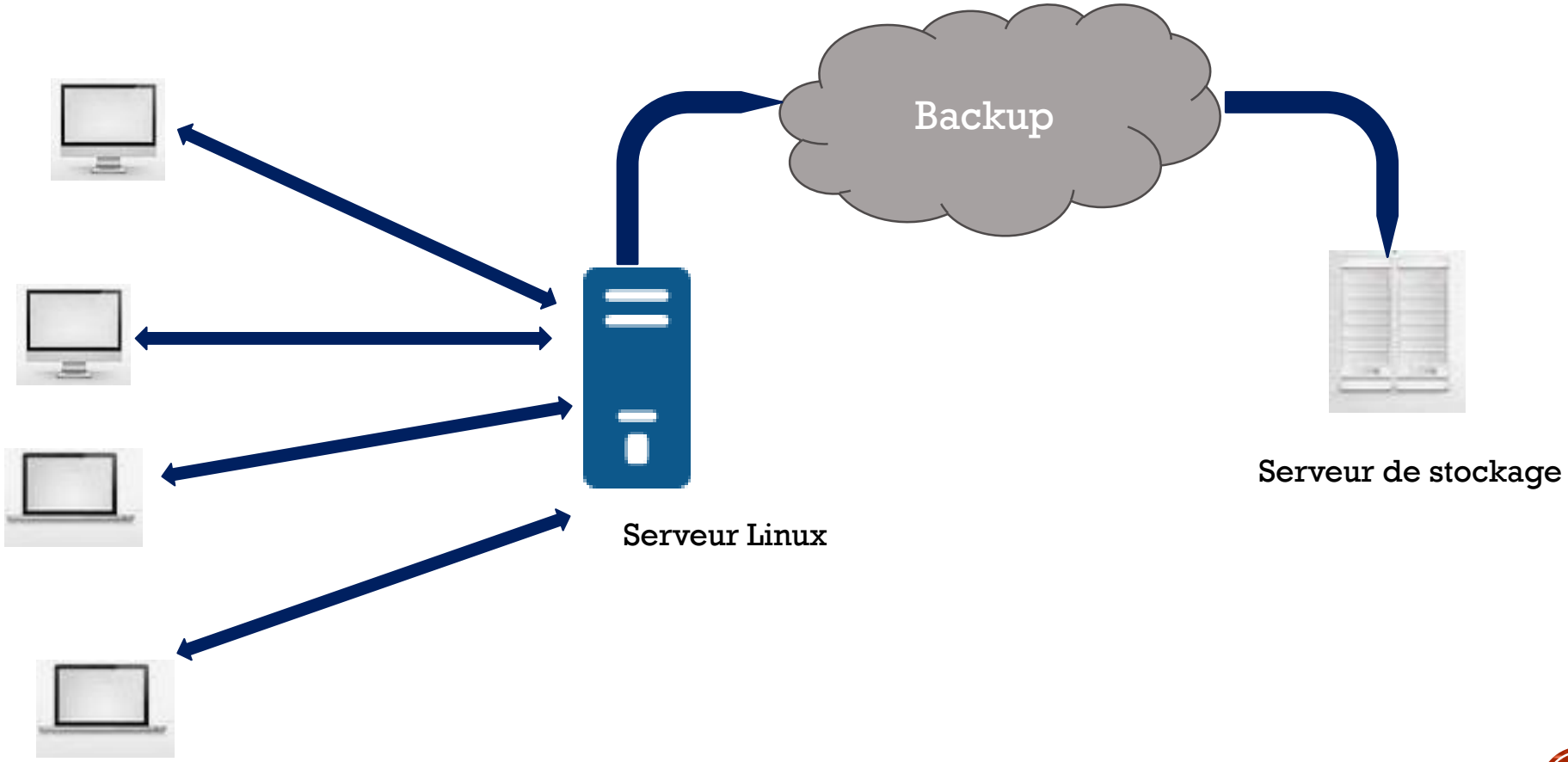
A.U: 2022-2023

Plan

- Introduction
- Cron et Crontab
- La commande at

Introduction

- Besoin: sauvegarder des données périodiquement. Comment faire??



Introduction

- Il est intéressant que les tâches habituelles/périodique soient réalisées automatiquement par le système plutôt que d'avoir à les lancer manuellement en tant qu'utilisateur.
- Un administrateur systèmes peut utiliser des tâches automatisées pour:
  - effectuer des copies de sauvegarde périodiques
  - nettoyer les anciens journaux
  - mettre à jour des paquets
  - exécuter des scripts d'entretien (comme le nettoyage des fichiers temporaires)
- **at** et **cron** sont les programme Linux natifs qui permettent de planifier des tâches.

Présentation du Cron

- **Cron** est le diminutif de **crontab** qui est le diminutif de **ch**r**on**o **tab**le qui signifie table de planification.
- **Cron** est un ***daemon*** utilisé pour planifier/programmer des tâches avant d'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 une **crontab** pour le système, permettant les tâches techniques, pour la mise à jour des différents programmes ou autres besoins périodiques.

La planification des tâches via “cron”

- Les tâches planifiées cron sont définies au niveau du système dans le fichier /etc/crontab et dans le dossier **/etc/cron.d/**
- Il est déconseillé de modifier directement le fichier /etc/crontab. Les tâches qui y sont définies sont implicitement exécutées par *root*

```
user@ubuntu-srv:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# 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
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
user@ubuntu-srv:~$
```

La planification des tâches via “cron”

- Le paquet ***cron*** propose par défaut des commandes planifiées qui exécutent :
  - Une fois par heure les programmes du répertoire **/etc/cron.hourly/** ;
  - Une fois par jour les programmes du répertoire **/etc/cron.daily/** ;
  - Une fois par semaine les programmes du répertoire **/etc/cron.weekly/** ;
  - Une fois par mois les programmes du répertoire **/etc/cron.monthly/**.

```
user@ubuntu-srv:~$ ls /etc/cron
cron.d/       cron.daily/   cron.hourly/  cron.monthly/ crontab       cron.weekly/
user@ubuntu-srv:~$ ls /etc/cron
```

- Les tâches Cron peuvent être programmées par l’installation de logiciels ou par les utilisateurs.
  - De nombreux paquets profitent de ce service: en déposant dans ces répertoires des scripts de maintenance, ils assurent le fonctionnement optimal de leur service.

Syntaxe d'un fichier Cron

- Une tâche planifiée dans un fichier de Cron est composée de 3 données différentes :
  - Sa période de répétition;
  - L'utilisateur système sous lequel la tâche sera réalisée ;
  - La commande à réaliser ;
- La syntaxe à respecter d'un fichier **crontab** :

```
#|----------------- 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)
#|   |   |   |   |
 *   *   *   *   *      user    command à exécuter
```

Exemples

- $ crontab -e
- Une commande complète ressemble à:

```
59 23 * * * root /home/backup/backup.cmd &> /dev/null
```

- # Télécharge les données tous les soirs à 19:25

```
25          19          *          *          *          $HOME/bin/getdata.py

# Le matin à 8:00, en semaine (lundi à vendredi)

00          08          *          *          1-5        $HOME/bin/script.sh
```

- **Remarque:** Dans un fichier de Cron pour un utilisateur en particulier, le nom d'utilisateur ne doit pas figurer puisque les actions seront réalisées sous l'utilisateur auquel appartient ce fichier.

Syntaxe d’un fichier Cron

- Champs date et heure:

Champs	valeurs autorisées
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (ou les noms en anglais)
day of week	0-7 (1 correspondant au lundi — le dimanche est représenté à la fois par 0 et par 7 ; il est également possible d'employer les trois premières lettres du nom du jour en anglais comme Sun, Mon, etc.) ;

Syntaxe d’un fichier Cron

- Chaque condition peut s'exprimer sous la forme d'une énumération de valeurs possibles (séparées par des virgules).
- La syntaxe **a-b** décrit l'intervalle de toutes les valeurs comprises entre a et b.
- **Exemple:**
  - plages de valeurs :

```
5-10
```

  - Les listes de plages :

```
1-4,8-12
```

Syntaxe d'un fichier Cron

- La syntaxe **a-b/<nombre>** décrit un intervalle avec un incrément de nombre (cadence)
- Exemple:

```
0-10/2
```

correspond à:

```
0,2,4,6,8,10
```

- Le joker * représente toutes les valeurs possibles c-à-d chaque unité de temps ;

Exemples

min	heure	jour/mois	mois	jour/semaine	Périodicité
*	*	*	*	*	Toutes les minutes
30	0	1	1,6,12	*	à 00:30 le premier janvier, juin et décembre
0	20	*	10	1-5	à 20:00 chaque jour de la semaine (du lundi au vendredi) d'octobre
0	0	1,10,15	*	*	à minuit les premiers, dixièmes, et quinzième jours de chaque mois
5,10	0	10	*	1	à 00:05 et 00:10 chaque lundi et le 10 de chaque mois

Raccourcis Cron

- Il existe aussi des raccourcis qui remplacent les cinq premiers champs d'une entrée de crontab, décrivent les planifications les plus classiques.

Raccourcis	Description	Équivalent
@reboot	Au démarrage du système	Aucun
@yearly	Tous les ans	0 0 1 1 *
@annually	Tous les ans	0 0 1 1 *
@monthly	Tous les mois	0 0 1 * *
@weekly	Toutes les semaines	0 0 * * 0
@daily	Tous les jours	0 0 * * *
@midnight	Tous les jours	0 0 * * *
@hourly	Toutes les heures	0 * * * *

Fichier Cron

- Un fichier cron peut commencer par une section de définition de variable.
  - Définir le chemin des exécutable (**PATH**), le shell à utiliser par défaut (**SHELL**), le dossier de démarrage (**HOME**) et l'adresse courriel à utiliser pour envoyer à un tiers le résultat de la commande (**MAILTO**).
- Exemple :

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=admin@domain.com
HOME=/root
# Une tâche du planning
   */1   *  *  *  *  root  ls -la
#Ceci permet de recevoir par mail à l'adresse admin@domain.com le
contenu du dossier /root. Et ce, toutes les minutes...
```

La commande contab

- **Affichage de la liste des crons planifiés**
  - Pour afficher la liste des tâches planifiées par cron l'utilisateur peut utiliser la commande :

```
$crontab   -l
```

- **Editer des crons**

```
$crontab  -e
```

- **Suppression des crons**
  - Pour supprimer une tâche particulière l'utilisateur peut éditer son fichier cron avec la commande **crontab -e** puis supprimer la ligne relative à cette planification.
  - Sinon, pour supprimer tous ses crons l'utilisateur peut utiliser la commande :

```
$crontab  -r
```

Restriction d'accès au cron

- Il est possible de limiter les tâches Cron à certains utilisateurs particuliers ou d'interdire certains utilisateurs à en avoir.
- Pour ce faire, en fonction du besoin, on peut créer un des deux fichiers suivants :
  - **/etc/cron.allow:** limitera la possibilité d'avoir des tâches cron aux utilisateurs listés ; Tous les autres seront automatiquement dépourvus de cette fonctionnalité.
  - **/etc/cron.deny**: interdira la possibilité d'avoir des tâches cron aux utilisateurs listés ;

Plan

- Introduction
- Cron et Crontab
- La commande at

Planification avec la commande at

- Le démon **atd** est celui qui s'occupe des commandes à **exécuter une seule fois**, à un instant précis et futur.
- La commande **at** permet de programmer des commandes à n'exécuter qu'une fois – contrairement à cron — à un moment donné.
  - Elle prend l'horaire et la date prévus en paramètres sur sa ligne de commande, et la commande à exécuter sur son entrée standard.
  - La commande enregistrée hérite de l'environnement courant utilisé au moment de sa définition
- Exemple:

```
user@ubuntu-srv:~$ ls
82013.data  A.ver6  axty4  Bureau  concat  concat.sh  Documents  Images  Modèles  Musique  Public  reptest  scipt1.sh  Téléchargements  test  testif.sh  touch  Vidéos  X2013.data
user@ubuntu-srv:~$ date
mer. 13 oct. 2021 12:43:33 CEST
user@ubuntu-srv:~$ echo "touch file.txt" |at 1245
warning: commands will be executed using /bin/sh
job 4 at Wed Oct 13 12:45:00 2021
user@ubuntu-srv:~$ ls
82013.data  A.ver6  axty4  Bureau  concat  concat.sh  Documents  file.txt  Images  Modèles  Musique  Public  reptest  scipt1.sh  Téléchargements  test  testif.sh  touch  Vidéos  X2013.data
user@ubuntu-srv:~$
```

L'horaire est indiqué en suivant les conventions habituelles : 12:45 représente 12 h 45

Créer une tâche planifiée

- La date peut être précisée au format **JJ.MM.AA** (27.10.2021 représentant ainsi 27 Octobre 2021) ou **AAAA-MM-JJ** (cette même date étant alors représentée par 2021-10-27).
  - En son absence, la commande sera exécutée dès que l'horloge atteindra l'heure signalée (le jour même où le lendemain).
  - On peut encore écrire explicitement *today* (aujourd'hui) ou *tomorrow* (demain).
  - Une autre syntaxe permet d'exprimer une durée d'attente : **at now +** *nombre période* commande.
  - La *période* peut valoir **minutes**, **hours** (heures), days (jours) ou **weeks** (semaines). Le *nombre* indique simplement le nombre de ces unités qui doivent s'écouler avant exécution de la commande

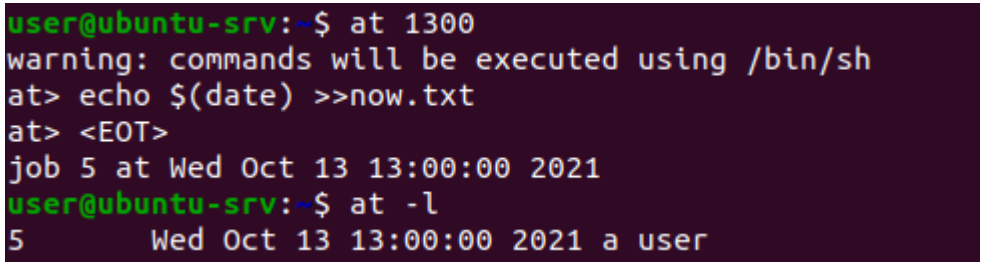
Exemple

- **Exemple:**

```
$at     1300
at> echo  $date >>now.txt
at>CTRL-d
```

```
$at     3pm     tomorrow
at> ps aux
at> CTRL-d
```

```
$at     now     +50     minutes
>cat  /var/log/messages  | mail    -s    “Logs”  admin@rtlab.net
>CTRL-d
```



Affichage des tâches planifiées par at

- at -l ou atq : affiche la liste des jobs introduits par la commande “at”.

```
$atq
```

- Exemple :

```
$atq
2   Tue Feb   9 10:34:00 2016 a rsi22
3   Tue Feb   10 12:00:00 2016 a
rsi22
```

- Le premier champ correspond au numéro du job planifié.

Planification avec la commande at

- **Supprimer une tâche planifiée**
- Pour annuler une tâche planifiée, il suffit d'exécuter la commande **atrm** *numéro-de-tâche*.

```
atrm  numéro-de-tâche
```

  - Le numéro de tâche est indiqué par la commande **at** lors de la planification mais on pourra le retrouver grâce à la commande **atq**, qui donne la liste des commandes actuellement planifiées.

- **Sécurisation de l'accès à la commande at**
  - Le même mécanisme de cron encadre atd, avec les fichiers /etc/at.allow et /etc/at.deny.

Fin du Chapitre 6