



# Administration Linux

Clément Weill



# 108

## Services systèmes essentiels

108.1 Gestion de l'horloge système

108.2 Journaux systèmes

108.3 Bases sur l'agent de transfert de courrier (MTA)

108.4 Gestion des imprimantes et de l'impression



# 108

## Services systèmes essentiels

108.1.a Gestion de l'horloge système



## Introduction

Au démarrage d'un ordinateur Linux, il initialise l'horloge système (*system clock*).

De plus, les ordinateurs modernes ont une horloge en temps réelle (*real time clock RTC* ou *hardware clock*) qui est sur la carte mère et qui garde l'heure indépendamment du fait que l'ordinateur soit en marche ou non.

Sur la plupart des systèmes Linux, les horloges sont synchronisées en utilisant le *network time*, qui est implémenté par *Network Time Protocol* (NTP).

Par défaut, l'horloge système utilise UTC (*Coordinated Universal Time*), et le l'heure locale est calculé par décalage à Greenwich et en prenant en compte l'heure d'été si besoin.



# date

`date` est l'utilitaire principal qui permet d'afficher l'heure locale.

Les options les plus courantes sont:

- `-u`
  - permet d'afficher l'heure UTC.
- `-I`
  - permet d'afficher l'heure au format ISO 8601. En y accolant `date` (e.g. `-Idate`) limite la sortie à la date uniquement. Les autres formats sont `hours`, `minutes`, `seconds`, et `ns`.
- `-R`
  - permet d'afficher l'heure au format RFC 5322.
- `--rfc-3339`
  - permet d'afficher l'heure au format RFC 3339



date

Le format de `date` peut être adapté par l'utilisateur avec des séquences spécifiées dans la page man.

**Par exemple**, pour obtenir le temps Unix:

```
$ date +%s
```

Le temps Unix est le nombre de secondes depuis *l'Epoch*, qui est défini comme le 1er Janvier 1970.

**Remarque:** Le temps Unix est stocké sur 32 bits, et donc s'arrête au 19 Janvier 2038.

En utilisant ces séquences, il est possible de formater la sortie à n'importe quelle exigence.



# date

De plus, `date --date` peut être utilisé pour formater une date qui n'est pas l'heure courante

**Par exemple:**

```
$ date --date='@1564013011'
```

```
Wed Jul 24 20:03:31 EDT 2019
```

L'option `--debug` peut être utile pour mieux comprendre ce qu'il se passe.

**Essayez:**

```
date --debug --date="Fri, 03 Jan 2020 14:00:17 -0500"
```



## Horloge Matérielle

Pour obtenir l'heure de l'horloge matérielle (*Hardware Clock*), un utilisateur avec des privilèges élevés peut utiliser `hwclock`.

En utilisant l'option `--verbose`, on peut observer plus d'informations.

**Essayez:**

```
$ sudo hwclock --verbose
```

Vous noterez au passage le `Calculated Hardware Clock drift`.

Cette sortie nous indique si il y a un décalage entre l'horloge système, et l'horloge matérielle.





## timedatectl

`timedatectl` est la commande pour vérifier l'état générale de la date et l'heure, ainsi que la présence ou non d'une synchronisation réseau (*Network Time Protocol*).

Par défaut, `timedatectl` retourne des informations similaires à `date`, mais avec en plus l'horloge en temps réelle (*RTC time*) et le statut du service NTP.

Si NTP est indisponible, il est recommandé d'utiliser `timedatectl` pour définir la date, plutôt que `date` ou `hwclock`.

**Exemple:** `# timedatectl set-time '2011-11-25 14:00:00'`

Il est possible de fixer uniquement l'heure indépendamment de la date, au format HH:MM:SS.



# timedatectl

`timedatectl` est aussi la commande à utiliser de préférence pour changer de fuseau horaire sur les systèmes Linux utilisant `systemd`, lorsque aucune GUI n'existe.

## Exemple:

```
$ timedatectl list-timezones | grep Paris
```

```
$ timedatectl set-timezone Europe/Paris
```

De plus, il est possible de désactiver NTP en utilisant `timedatectl` plutôt que `systemctl`:

```
# timedatectl set-ntp no
```



## Changer de Fuseau Horaire

Comme nous l'avons dit, le répertoire `/usr/share/zoneinfo` contient les informations sur les différents fuseaux horaires.

Ces fichiers permettent de calculer l'heure locale à partir de UTC.

Quand Linux veut déterminer l'heure locale, il lit le contenu de `/etc/localtime`. Pour changer manuellement le fuseau horaire, il faut créer un lien symbolique vers le fichier correspondant.

### Exemple:

```
$ ln -s /usr/share/zoneinfo/Canada/Eastern /etc/localtime
```



## Changer de Fuseau Horaire

Une fois le bon fuseau choisi, il est recommandé d'exécuter:

```
# hwclock --systohc
```

Ceci va assigner à l'horloge réelle (*hardware clock*) la valeur de l'horloge système (*system clock*).

`/etc/timezone` est similaire à `/etc/localtime`. C'est une représentation du fuseau horaire local.

**Essayez:** `cat /etc/timezone`

**Attention**, toutes les distributions Linux n'utilisent pas ce fichier.



## Changer l'Heure et la Date

`date` permet de changer l'horloge système avec l'option `--set` ou `-s`. On peut rajouter l'option `--debug` pour plus d'informations.

### Exemple:

```
# date --set="11 Nov 2011 11:11:11"
```

Il est possible de changer indépendamment la date ou l'heure, **par exemple:**

```
# date +%Y%m%d -s "20111125"
```

```
# date +%T -s "13:11:00"
```

Ici, on précise la séquence pour que la string soit interprétée correctement.



## Changer l'Heure et la Date

Une fois l'heure changé, il est conseillé de synchroniser l'horloge réelle avec celle du système, en utilisant:

```
# hwclock --systohc
```

Inversement, il est possible de fixer l'horloge réelle puis de synchroniser l'horloge système.

```
# hwclock --set --date "19/03/2024 11:12:13"
```

```
# hwclock --hctosys
```



# Exercices





# 108

## Services systèmes essentiels

108.1.b Gestion de l'horloge système





# Network Time Protocol

NTP utilise une structure hiérarchique pour disséminer le temps dans le réseau.

Les horloge de référence (reference clocks), le plus souvent des horloges atomiques, sont connecté à des serveurs au sommet de la hiérarchie.

Ces serveur sont des machines *Stratum 1*, et ne sont typiquement pas accessible au publique. En revanche, elles sont accessible aux machines *Stratum 2*, qui sont elles-mêmes accessible aux machines *Stratum 3*, et ainsi de suite.

Les serveurs *Stratum 2* sont accessible au publique, mais lors de l'implémentation de NTP dans un réseau, il est recommandé d'avoir un faible ensemble de machine connecté aux serveurs *Stratum 2*, qui dissémine NTP au reste du réseau.



# Network Time Protocol

Il existe un ensemble de termes importants pour NTP qui sont:

## ◆ Offset

- C'est la différence absolue entre l'horloge système et le temps NTP.

## ◆ Step

- Si l'*offset* est supérieur à 128ms, alors NTP opère un seul changement important à l'horloge système. C'est le *stepping*.

## ◆ Slew

- Si l'*offset* est inférieur à 128ms, alors le changement est graduel, en accélérant ou ralentissant l'horloge système. C'est le *slewing*.

## ◆ Insane Time

- Si l'*offset* est supérieur à 17 minutes, alors l'horloge système est considéré comme folle, et le daemon NTP ne change pas l'heure du système.



# Network Time Protocol

## ◆ Drift

- Le *drift* fait référence au phénomène où deux horloges se décalent au cours du temps.

## ◆ Jitter

- C'est la quantité de *drift* depuis la dernière requête NTP. Par exemple, si le dernier NTP sync a eu lieu il y a 17 minutes et que l'*offset* est de 3 millisecondes, alors le *jitter* est de 3 millisecondes.



## timedatectl

Si votre distribution Linux utilise `timedatectl`, alors par défaut elle implémente un client *SNTP* plutôt qu'une implémentation NTP complète.

C'est une implémentation moins complexe mais qui implique que la machine ne peut pas fournir NTP aux autres machines du réseau.

Dans ce cas, SNTP ne fonctionne que si le service `timesyncd` tourne sur la machine.

Comme pour l'ensemble des service systemd, on peut vérifier cela avec :

```
$ systemctl status systemd-timesyncd
```



## timedatectl

Le statut de `timedatectl` synchronisation SNTP peut être vérifié avec l'option `show-timesync`.

**Par exemple:**

```
$ timedatectl show-timesync --all
```

Cette configuration suffit souvent, mais ne permet pas de synchroniser plusieurs clients sur le réseau.

Pour cela, il faut implémenter un client NTP complet.



## NTP Daemon

L'horloge système et l'horloge réseau sont comparé à intervalle régulier, il faut donc un *daemon* qui tourne dans le fond pour exécuter cette tâche.

Pour la plupart des systèmes Linux, ce daemon s'appelle `ntpd`. Il permet à la machine d'être, en plus d'un *consommateur* (c'est à dire de synchroniser son horloge système avec une source extérieure), un *fournisseur* aux autres machines du réseau.

Sur une machine utilisant systemd, et donc qui utilise `systemctl` pour contrôler ses daemons, on peut vérifier le statut de `ntpd` avec:

```
$ systemctl status ntpd
```



## NTP Daemon

Dans certains cas, il est nécessaire de démarrer et d'activer `ntpd`, avec:

```
$ systemctl enable ntpd && systemctl start ntpd
```

Les requêtes NTP ont lieu sur le port TCP 123. Si NTP échoue, il faut vérifier que le port est ouvert et bien entrain d'écouter.



## Configuration NTP

NTP est capable de questionner plusieurs serveurs et de sélectionner le meilleur candidat à utiliser pour définir l'horloge système.

Si la connexion au réseau est perdue, NTP utilise les ajustements précédents pour prédire et estimer les ajustements futurs.

Selon la distribution Linux, la liste des serveurs NTP est situé à différents endroit, par exemple `/etc/ntp.conf`.





## Configuration NTP

Par défaut, les serveurs NTP utilisés sont spécifiés dans une section comme celle-ci:

```
# Use public servers from the pool.ntp.org project.  
  
# Please consider joining the pool  
(http://www.pool.ntp.org/join.html).  
  
server 0.centos.pool.ntp.org iburst  
server 1.centos.pool.ntp.org iburst  
server 2.centos.pool.ntp.org iburst  
server 3.centos.pool.ntp.org iburst
```



## Configuration NTP

Pour ajouter un serveur NTP, il faut utiliser la syntaxe suivante:

```
server (IP Address)
```

```
server server.url.localhost
```

Les adresses serveurs peuvent soit être des adresses IP, soit URL si le DNS a bien été configuré.

Il est aussi possible d'utiliser un *pool* de serveur NTP. Dans ce cas, quand un client fait une requête à un pool, un fournisseur est sélectionné au hasard. Cela permet de distribuer la charge sur le réseau.

En pratique, `/etc/ntp.conf` est peuplé avec un pool de serveur appelé `pool.ntp.org`.



## ntpdate

Pendant la première installation, il peut y avoir une désynchronisation importante entre l'horloge système et NTP.

Si l'*offset* est supérieur à 17 minutes, alors le daemon NTP ne change pas l'horloge système, il faut intervenir manuellement.

Tout d'abord, si il tourne, il faut désactiver `ntpd`. Pour ce faire, on peut utiliser `systemctl stop ntpd`.

Ensuite, il faut utiliser `ntpdate pool.ntp.org` pour faire une première synchronisation, où `pool.ntp.org` fait reference à une adresse IP ou l'URL d'un serveur NTP.



`ntpq`

`ntpq` est l'utilitaire qui permet de vérifier l'état de NTP.

Une fois le daemon NTP configuré et lancé, on en récupère l'état avec l'option print `-p`. Il y a aussi un mode interactif si aucune option est passée.

**Essayez:** `ntpq -p`

Les colonnes sont:

- ◆ `remote`
  - Nom d'hôte du fournisseur NTP.
- ◆ `refid`
  - Numéro de référence du fournisseur NTP.
- ◆ `st`
  - Stratum du fournisseur.



`ntpq`

- `when`
  - Nombre de secondes depuis la dernière requête.
- `poll`
  - Nombre de secondes entre les requêtes.
- `reach`
  - Numéro d'état qui indique si le serveur a été atteint. Une connection réussie incrémente ce numéro de 1.
- `delay`
  - Le temps en milliseconde entre la requête et la réponse du serveur.
- `offset`
  - Le temps en milliseconde entre l'horloge système et l'horloge NTP.
- `jitter`
  - *Offset* en milliseconde entre l'horloge système et l'horloge NTP dans la dernière requête.



## chrony

`chrony` est une autre implémentation de NTP. Il est installé par défaut sur certaines distributions Linux, mais est disponible sur l'ensemble des machines Linux.

`chronyd` est le chrony daemon, et `chronyc` est l'interface de commande. Il faut lancer et activer `chronyd` avant de pouvoir utiliser `chronyc`.

**Essayez:** `chronyc tracking`



## chrony

Cette sortie contient plus d'information que les autres implémentations, et les lignes sont:

### ◆ Reference ID

- L'identifiant de référence, et le nom, auquel la machine est actuellement synchronisé.

### ◆ Stratum

- Nombre de saut jusqu'à une machine connecté à une horloge de référence.

### ◆ Ref time

- C'est le temps UTC à laquelle la dernière mesure à la source référente a été fait.

### ◆ System time

- C'est l'écart entre l'horloge système et le serveur de synchronisation.



# chrony

- ◆ **Last offset**
  - C'est l'*offset* estimé de la dernière mis-à-jour de l'horloge.
- ◆ **RMS offset**
  - Moyenne sur un temps long de l'*offset*.
- ◆ **Frequency**
  - C'est le taux qui indique de combien l'horloge serait fausse si elle n'était pas corrigé. C'est en ppm (partie par million).
- ◆ **Residual freq**
  - Fréquence résiduelle qui indique la différence entre les mesures à la source et la fréquence utilisé.
- ◆ **Skew**
  - Estimation de la marge d'erreur de la fréquence.





## chrony

### ◆ Root delay

- Somme des délais du chemin réseau jusqu'à la machine *Stratum 1* à laquelle la machine est synchronisé.

### ◆ Leap status

- C'est l'état des sauts, qui ne peut prendre que les valeurs suivantes : *normal, insert second, delete second* ou *not synchronized*.



## chrony

Enfin, pour obtenir des informations sur l'ensemble des serveurs NTP utilisés, il faut utiliser `chronyc sources`.

Pour modifier les sources possibles, il faut modifier le fichier `/etc/chrony.conf`. Par défaut, il est peuplé par des serveurs du projet `pool.ntp.org`.

Une fois les modifications effectuées, il faut redémarrer le service `chronyd`, et utiliser `chronyc makestep` pour *step* l'horloge du système.



# Exercices





# 108

## Services systèmes essentiels

108.2.a Journaux systèmes



## Journaux Systèmes

Traditionnellement, et en laissant de côté `systemd-journald` pour l'instant, l'enregistrement des journaux a trois services dédiés: `syslog`, `syslog-ng` (*syslog new generation*), et `rsyslog` (*"the rocket-fast system for log processing"*).

`rsyslog` a apporté des améliorations conséquentes (tel que le support `RELP`) et est aujourd'hui le choix le plus populaire.

**Remarque:** `RELP` est le *Reliable Event Logging Protocol* qui étend les fonctionnalités du protocole syslog pour s'assurer de la livraison des messages.

Chacun de ces services récupère des messages d'autres services et programmes, et les stocke dans des journaux systèmes (*log files*), typiquement dans `/var/log`.



## Journaux Systèmes

Puisque `rsyslog` est aujourd'hui le *de facto* standard pour l'enregistrement des journaux système, nous allons nous concentrer sur lui.

`rsyslog` utilise un modèle client-serveur, dont le client et le serveur peuvent exister sur la même machine, ou bien être distribué dans le réseau.

Les messages sont transmis et reçus, avec un format donné, et peuvent être maintenus sur un serveur `rsyslog` centralisé.

Le daemon de `rsyslog`, `rsyslogd` travaille avec le daemon qui traite les messages kernel, `klogd`.



## Types de Journaux

Puisque les journaux sont des informations *variables*, ils sont normalement situé dans `/var/log`.

On peut en distinguer trois types principaux: les *journaux systèmes*, *journaux services*, et les *journaux programmes*.

Voici quelques journaux systèmes, ainsi que les informations qu'ils contiennent:

- `/var/log/auth.log`

- Activité lié au processus d'authentification : les utilisateurs connectés, information sudo, cron jobs, etc.



## Types de Journaux

- ◆ `/var/log/syslog`
  - Un fichier qui centralise la quasi-totalité des messages reçu par `rsyslog`. A cause de sa taille, les journaux sont distribué dans d'autres fichiers en suivant la configuration fournis dans `/etc/rsyslog.conf`.
- ◆ `/var/log/debug`
  - Information de debugging pour les programmes.
- ◆ `/var/log/kern.log`
  - Les messages du Kernel.





## Types de Journaux

- ◆ `/var/log/messages`
  - Messages informatifs qui ne sont pas lié au kernel mais à d'autres services. C'est aussi le journal destination pour les clients distants dans une implémentation réseau.
- ◆ `/var/log/daemon.log`
  - Informations liés aux daemon ou aux services qui tournent en fond.
- ◆ `/var/log/mail.log`
  - Informations liés au serveur mail, e.g. postfix.
- ◆ `/var/log/Xorg.0.log`
  - Informations liés à la carte graphique.



## Types de Journaux

- `/var/run/utmp` et `/var/log/wtmp`
  - Les connections réussies.
- `/var/log/btmp`
  - Les connections échouées, e.g. attaque brute force via ssh.
- `/var/log/faillog`
  - Les authentifications échouées.
- `/var/log/lastlog`
  - La date et l'heure de l'authentification des utilisateurs récents.



## Types de Journaux

Et quelques exemple de journaux services:

- ◆ `/var/log/cups/`
  - Répertoires pour les journaux du *Common Unix Printing System*. Il contient habituellement les journaux suivant: *error\_log*, *page\_log* et *access\_log*.
- ◆ `/var/log/apache2/` or `/var/log/httpd`
  - Répertoires pour les journaux du *Apache Web Server*. Il contient habituellement les journaux suivant: *access.log*, *error\_log*, et *other\_vhosts\_access.log*.



## Types de Journaux

- `/var/log/mysql`
  - Répertoires pour les journaux du *MySQL Relational Database Management System*. Il contient habituellement les journaux suivant: *error\_log*, *mysql.log* et *mysql-slow.log*.
- `/var/log/samba/`
  - Répertoires pour les journaux du protocole *Session Message Block (SMB)*. Il contient habituellement les journaux suivant: *log.*, *log.nmbd* et *log.smbd*.



## Lire les Journaux

Il existe de nombreux utilitaires pour lire des journaux, tel que:

- `less` ou `more`
  - qui permettent de lire une page à la fois.
- `zless` ou `zmore`
  - comme `less` et `more`, mais pour les journaux compressés avec `gzip`.
- `tail`
  - pour voir les dernières lignes d'un journal (par défaut 10). Avec l'option `-f`, permet de dynamiquement montrer les nouvelles lignes alors que celle-ci sont ajoutés.
- `head`
  - pour voir les premières lignes d'un journal (par défaut 10).
- `grep`
  - pour filtrer et chercher dans les journaux.



## Lire les Journaux

Les sorties sont au format suivant:

- ◆ Timestamp
- ◆ Nom de l'hôte d'où provient le message
- ◆ Nom du programme qui a généré le message
- ◆ PID dudit programme
- ◆ Description de l'action qui a eu lieu

**Par exemple:**

```
root@debian:~# less /var/log/auth.log
```

```
Sep 12 18:47:56 debian sshd[441]: Received SIGHUP; restarting.
```

```
Sep 12 18:47:56 debian sshd[441]: Server listening on 0.0.0.0 port 22.
```

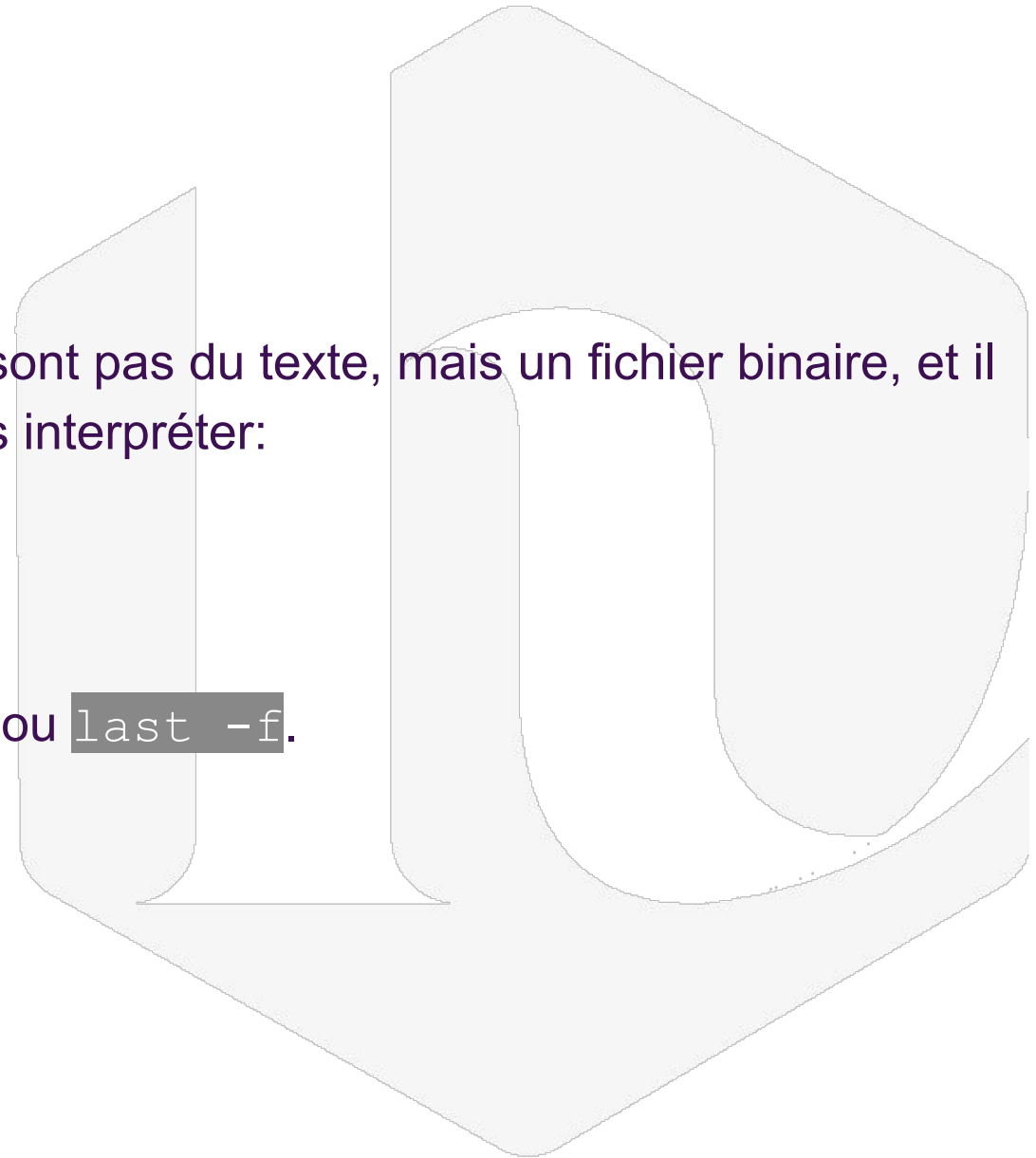
```
Sep 12 18:47:56 debian sshd[441]: Server listening on :: port 22.
```



## Lire les Journaux

**Attention**, certains journaux ne sont pas du texte, mais un fichier binaire, et il faut donc un programme pour les interpréter:

- `/var/log/wtmp`
  - Il faut utiliser `who` (ou `w`)
- `/var/log/btmp`
  - Il faut utiliser `utmpdump` ou `last -f`.
- `/var/log/faillog`
  - Il faut utiliser `faillog`.
- `/var/log/lastlog`
  - Il faut utiliser `lastlog`.





## Des Messages aux Journaux

Le processus suivant illustre comment un message est écrit dans un journal:

1. Les applications, les services et le kernel écrivent leurs messages dans un fichier spécifique (*sockets*, ou mémoires tampons), e.g. `/dev/log` ou `/dev/kmsg`.
2. `rsyslogd` récupère les informations depuis les *sockets* ou des mémoires tampons.
3. D'après les règles dans `/etc/rsyslog.conf` et/ou dans les fichiers de `/etc/rsyslog.d/`, `rsyslog` déplace les informations dans le journal approprié.





## Sous-systèmes, Priorités et Actions

Le fichier de configuration de `rsyslog` est divisé en trois parties : `MODULES`, `GLOBAL DIRECTIVES`, et `RULES`.

Pour mieux comprendre, essayons `sudo less /etc/rsyslog.conf`.

`MODULES` inclus le support des modules pour l'entretien des journaux, l'écriture et la lecture de messages, et la réception de journaux par UDP/TCP.

`GLOBAL DIRECTIVES` permet de configurer un ensemble de choses, tel que les permissions des journaux, ou du répertoire de journaux,.

`RULES` est où sous-systèmes (*facilities*), priorités et actions rentre en compte. Cela permet au daemon journalier de filtrer les messages et de les écrire ou de les envoyer là où ils sont requis.



## Sous-systèmes, Priorités et Actions

Chaque message est attribué un numéro de sous-système, qui varie de 0 à 23, et un mot-clé, le tout associé au sous-système interne de Linux qui a produit le message.

De plus, a chaque message est aussi associé un niveau de priorité, qui varie de 0 à 7.

Les tableaux suivant exposent les valeurs possibles pour les deux valeurs respectivement:



## Sous-systèmes, Priorités et Actions

Code	Mot-clé	Description
0	kern	messages du noyau
1	user	messages de l'espace utilisateur
2	mail	messages du système de messagerie
3	daemon	messages des processus d'arrière plan
4	auth	messages d'authentification
5	syslog	messages générés par syslogd lui-même
6	lpr	messages d'impressions
7	news	messages d'actualités



## Sous-systèmes, Priorités et Actions

Code	Mot-clé	Description
8	<code>uucp</code>	messages UUCP
9	<code>cron</code>	Tâches planifiées (at/cron)
10	<code>authpriv</code>	sécurité / élévation de privilèges
11	<code>ftp</code>	logiciel FTP
12	<code>ntp</code>	Synchronisation du temps NTP
13	<code>security</code>	log audit
14	<code>console</code>	log alert
15	<code>solaris-cron</code>	Tâches planifiées (at/cron)
16 - 23	<code>local0 - local7</code>	Utilisation locale libre 0 - 7



## Sous-systèmes, Priorités et Actions

Code	Gravité	Mot-clé	Description
0	Emergency	emerg (panic)	Système inutilisable.
1	Alert	alert	Une intervention immédiate est nécessaire.
2	Critical	crit	Erreur critique pour le système.
3	Error	err (error)	Erreur de fonctionnement.
4	Warning	warn (warning)	Avertissement (une erreur peut intervenir si aucune action n'est prise).
5	Notice	notice	Événement normal méritant d'être signalé.
6	Informational	info	Pour information.
7	Debugging	debug	Message de mise au point.



## Sous-systèmes, Priorités et Actions

Les règles sont au format : `<facility>.<priority> <action>.`

La partie `<facility>.<priority>` filtre les messages et applique l'`<action>` appropriée.

Les priorités sont inclusive hiérarchiquement, donc l'ensemble des messages avec la priorités spécifiées ou supérieur sont reconnu par le filtre.

L'action est souvent là où doivent aller les messages (dans quels journaux).

### Exemple:

```
auth,authpriv.* /var/log/auth.log
```

```
*.*;auth,authpriv.none -/var/log/syslog
```



## Sous-systèmes, Priorités et Actions

### Exemple:

```
auth,authpriv.* /var/log/auth.log
```

L'opérateur asterisks (\*) permet de sélectionner tous les messages qu'importe leurs priorités, ou leurs sous-systèmes.

```
*.*;auth,authpriv.none -/var/log/syslog
```

Le suffix `.none` permet de défausser les messages correspondants.

L'opérateur moins (-) devant le répertoire destination permet d'éviter les écritures excessives.

Le point virgule (;) sert de séparateur; et la virgule (,) permet de concaténer deux sous-systèmes dans la même règle.



## Sous-systèmes, Priorités et Actions

**Autre exemple:**

```
*.=debug;\
```

```
auth,authpriv.none;\
```

```
news.none;mail.none
```

```
-/var/log/debug
```

L'opérateur égal (=) permet de fixer la priorité dans la règle à une stricte égalité. On notera aussi l'usage de (;\ ) qui permet un retour à la ligne.

Il est possible de manuellement entrer un message avec la commande `logger` qui ajoute le message à `/var/log/syslog` (ou `/var/log/messages` si la requête vient du réseau)





## Serveur Central de Journaux

Pour mieux comprendre ce principe, nous utiliserons la configuration suivante:

Rôle	Hostname	OS	IP Address
Central Log Server	suse-server	openSUSE Leap 15.1	192.168.1.6
Client	debian	Debian GNU/Linux 10 (buster)	192.168.1.4



## Côté Serveur

openSUSE a un fichier de configuration dédié pour la journalisation à distance (`/etc/rsyslog.d/remote.conf`).

Il faut activer la réception de messages via TCP sur le port 514, en décommentant les lignes associés.

```
$InputTCPServerRun 514 # Starts a TCP server on selected port
```

Puis en redémarrant le service `rsyslog`, avec:

```
systemctl restart rsyslog
```



## Templates et Filtrage

Par défaut, les journaux client seront enregistré dans le fichier `/var/log/messages` du serveur, mélangé avec ceux du serveur aussi.

Cependant, nous allons créer un *template* et un *filtre* pour séparer les journaux du client et ceux du serveur.

Pour ceci, il faut ajouter ce qui suit au fichier `/etc/rsyslog.conf` (ou `/etc/rsyslog.d/remote.conf`):

```
$template RemoteLogs,  
"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"  
  
if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs & stop
```



# Templates

Le template correspond à la première ligne, et permet spécifier le nom du journal en utilisant une génération dynamique de nom .

Un template est composé de :

- ◆ Directive de template (`$template`)
- ◆ Nom du template (`RemoteLogs`)
- ◆ Texte du template  
(`"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"`)
- ◆ Options

Notre exemple s'appelle `RemoteLogs`, et son texte consiste en un chemin absolue dans `/var/log`.



## Templates

Tous les messages journaux de notre client vont aller dans le répertoire `remotehosts` du serveur, où un sous-répertoire basé sur le nom de la machine (`%HOSTNAME%`) est créé. Chaque fichier dans ce répertoire sera composé par la date (`%NOW%`), la sévérité (e.g. *priority*) du messages en format texte (`%syslogseverity-text%`) et le suffix `.log`.

Les mots entre les signes pourcentages sont des *propriétés* et permettent d'accéder au contenu du messages (date, priorité, etc).

Un message `syslog` à un ensemble de *propriétés* bien défini, qui peuvent être utilisé dans les templates. Ces *propriétés* sont accessible, et modifiable, par le remplaceur de propriétés (*property replacer*), qui implique de les écrire entre pourcentages.



## Filtrage

Le filtre est un ensemble de condition, et des actions associées.

Un filtre est composé de :

- ◆ Expression de filtrage (`if $FROMHOST-IP== '192.168.1.4'`)
- ◆ Actions (`?RemoteLogs & stop`)

Notre filtre vérifie l'adresse IP de la machine qui transmet le message , et si il est égale à notre client, appelle le template `RemoteLogs`.

La dernière partie (`& stop`) permet de s'assurer que le messages n'est pas envoyé ailleurs simultanément, en particulier `/var/log/messages`.

**Remarque:** `man rsyslog.conf` pour plus d'informations sur les filtres et les templates.



## Côté Client

On ajoute l'adresse du serveur dans le fichier de résolution d'adresse, `/etc/hosts`.

C'est à dire la ligne suivante: `192.168.1.6 suse-server`.

Notre client Debian doit alors rediriger l'ensemble des messages vers notre serveur, il faut rajouter la lignes suivante dans `/etc/rsyslog.conf`:

```
*.* @@suse-server:514
```

C'est l'opérateur (`@@`) qui indique une adresse à distance.

Puis pour activer la journalisation à distance, il faut redémarrer le service.



## Rotation des Journaux

La rotation régulière des journaux permet deux choses:

- ◆ Empêche les journaux de prendre trop de place dans la mémoire.
- ◆ Permet d'avoir des journaux de taille raisonnable, plus lisible.

C'est le service `logrotate` qui s'en occupe, qui déplace et renomme les journaux, et qui les compresse ou les supprime selon leur âge.

**Essayez:** `ls /var/log/messages*`

```
/var/log/messages          /var/log/messages.1
/var/log/messages.2.gz     /var/log/messages.3.gz
/var/log/messages.4.gz
```





## Rotation des Journaux

A la prochaine rotation :

- `messages.4.gz` sera supprimé.
- le contenu de `messages.3.gz` sera déplacé vers `messages.4.gz`.
- le contenu de `messages.2.gz` sera déplacé vers `messages.3.gz`.
- le contenu de `messages.1` sera compressé et déplacé vers `messages.2.gz`.
- le contenu de `messages` sera déplacé vers `messages.1`.
- `messages` est vidé et prêt à recevoir de nouveaux messages.

On remarquera que les trois fichiers les plus vieux sont compressés, et les deux plus récents non.



## Rotation des Journaux

`logrotate` est un processus automatisé, parfois par un cron job journalier à travers le script `/etc/cron.daily/logrotate`.

C'est le fichier `/etc/logrotate.conf` qui en assure la configuration.

**Essayez:** `sudo less /etc/logrotate.conf`

Il y a aussi les fichiers de configuration dans `/etc/logrotate.d` pour les paquets spécifiques à charger. Il s'agit, entre autres, de définitions locales et précise les journaux à renouveler.

**Par exemple,** un extrait de `/etc/logrotate.d/rsyslog`



# Rotation des Journaux

```
/var/log/messages
```

```
{
```

```
    rotate 4
```

```
    weekly
```

```
    missingok
```

```
    notifempty
```

```
    compress
```

```
    delaycompress
```

```
    sharedscripts
```

```
    postrotate
```

```
        invoke-rd rsyslog rotate > /dev/null
```

```
    endscript
```

```
}
```





## Rotation des Journaux

- `rotate 4`
  - Retient jusqu'à 4 semaines
- `weekly`
  - La rotation est hebdomadaire
- `missingok`
  - Ne pas lancer d'erreur quand le fichier est absent
- `notifempty`
  - Ne pas appliquer si le journal est vide
- `compress`
  - Compressé les journaux avec `gzip`
- `delaycompress`
  - Décale la compression au prochain cycle de rotation



## Rotation des Journaux



`sharedscripts`

- Lié aux scripts *prerotate* et *postrotate*. Les scripts ne sont lancés qu'une fois pour tous les logs



`postrotate`

- Indique le début d'un script *postrotate*.



`invoke-rs.d rsyslog rotate > /dev/null`

- Invoque cette commande après une rotation.



`endscript`

- Indique la fin d'un script, ici *postrotate*.

**Remarque:** Pour plus d'informations, consulter la page `man logrotate.conf`



## Kernel Ring Buffer

Puisque le kernel génère des messages avant que `rsyslogd` soit disponible, il faut un système pour les conserver.

C'est le rôle du *Kernel Ring Buffer*. C'est une structure de taille donnée, et donc les messages les plus récents poussent les plus vieux qui disparaissent.

La commande `dmesg` permet d'en afficher le contenu.

**Essayez:** `dmesg | grep "usb"`



# Exercices





# 108

## Services systèmes essentiels

108.2.b Journaux systèmes





## systemd

Avec l'avènement de `systemd`, qui remplace `SysV init`, comme manager du système et de service, apporte aussi un nouveau daemon pour la journalisation : `systemd-journald`.

Les forces de `systemd` sont:

- Facilité de configuration
  - fichier d'unité par opposition aux scripts SysV init.
- Versatilité de management
  - en plus des daemon et des processus, s'occupe aussi des devices, sockets et des points de montage.
- Rétro-compatibilité
  - fonctionne avec SysV et Upstart.
- Chargement parallèle au démarrage
  - plutôt que séquentiellement.



# systemd

`systemd` opère sur des *unités*, qui sont défini dans des fichiers d'unités. Ce sont des fichiers textes situé dans `/lib/systemd/system` qui sont divisé en *sections* et *directives*.

Il existe de nombreux types d'unités: `service`, `mount`, `automount`, `swap`, `timer`, `device`, `socket`, `path`, `timer`, `snapshot`, `slice`, `scope`, et `target`.

Chaque fichier d'unité à un nom du type: `<resource name>.<unit type>` (e.g. `reboot.service`).

Une *cible* (*target*) est un type d'unité spéciale qui ressemble aux *runlevels* de `SysV init` (e.g. `graphical.target` est similaire à `runlevel 5`), mais sont mutuellement inclusif.



## systemd-journald

Le fichier de configuration est `/etc/systemd/journald.conf`, et comme avec d'autres services, on interagit avec en utilisant `systemctl`.

Le journal généré peut être stocké sur le disque dur, ou de manière volatile sur un système de fichier temporaire (*tmpFS*).

C'est un est un binaire, il faut donc utiliser `journalctl` pour le lire.

**Essayez:** `sudo journalctl`

Cela affiche l'ensemble du journal par ordre chronologique, avec les plus vieilles entrées en premier.

Il existe des options pour faire des recherches plus affinées:



## systemd-journal

• `-r`

- Inverse l'ordre d'impression, avec les messages les plus récents en premier.

• `-f`

- Imprime que les messages les plus récents, et de manière dynamique, similairement à `tail -f`.

• `-e`

- Saute à la fin du journal, de manière à afficher le dernier message.

• `-n <valeur>, --lines=<valeur>`

- Imprime les *valeurs* dernières lignes (10 par défaut).



• `-k, --dmesg`

- équivalent à la commande `dmesg`.






## Naviguer et Chercher

Pour naviguer dans le journal, on peut utiliser:

- ◆ Les touches *PageUp*, *PageDown* et les flèches du clavier.
- ◆  pour aller à la fin de la sortie.
- ◆  pour aller au début de la sortie.

Pour chercher une string à partir de la position actuelle:

- ◆ Recherche vers l'avant:  puis la string à chercher et Enter.
- ◆ Recherche vers l'arrière:  puis la string à chercher et Enter.

Pour naviguer entre les résultats de la recherche,  permet d'aller au résultat suivant, et  au résultat précédent.



## Filtrer le Journal

### Boot number:

#### • `-list-boots`

- Liste l'ensemble des démarrages (*boots*) disponibles, avec `0` le démarrage actuel, `-1` le précédent, `-2` celui avant ça, etc.), puis un boot ID, et enfin un timestamp.

#### • `-b, --boot`

- Montre l'ensemble des messages pour le démarrage actuel. Peut être décalé comme montré juste avant, mais il faut pour cela que les journaux soit persistents.

### Priorité:

#### • `-p`

- Permet de filtrer selon la priorité/sévérité du message (c.f. `syslog`).



## Filtrer le Journal

### Interval de Temps:

- ◆ `--since` et `--until`
  - Permet de filtrer pour n'afficher que les messages avec un temps donnée. Le format devrait suivre `YYYY-MM-DD HH:MM:SS`.
  - Il est possible d'utiliser la syntaxe `"integer time-unit ago"`, ou même les signes `+` et `-`.
  - **Ex:** `--since "-2 minutes"` et `--since "2 minutes ago"` sont équivalents.
  - Il existe aussi quelques mots spécifiques: `yesterday`, `today`, `tomorrow`, et `now`.

### Programme:

- ◆ Pour voir les messages liées à un exécutable, la syntaxe est `journalctl /path/to/executable`.



## Filtrer le Journal

### Unité:



`-u`

- Permet de filtrer pour n'afficher que les messages d'une unité donnée.
- **Ex:** `journalctl -u ssh.service`

### Champs:

Le journal peut aussi être filtrer avec des champs spécifiques en utilisant la syntaxe suivante:



`<field-name>=<value>`



`<field-name>=<value>`



`<field-name>=<value>`





## Filtrer le Journal

- `PRIORITY=`
  - L'un des 8 niveaux de priorité/sévérité de `syslog`.
- `SYSLOG FACILITY=`
  - L'un des 23 niveaux de sous-système (*facility*).
- `PID=`
  - N'affiche que les messages liées à l'ID processus donné.
- `BOOT ID=`
  - N'affiche que les messages liées à l'ID de démarrage donné.
- `TRANSPORT=`
  - N'affiche que les messages d'un transport donné. Les valeurs possibles sont:
    - `audit` (sous-système d'audit du kernel)
    - `driver` (généré de manière interne)
    - `syslog` (socket syslog)
    - `journal` (journal protocol natif)
    - `stdout` (sortie standard ou sortie erreur des services)
    - `kernel` (kernel ring buffer)



## Filtrer le Journal

Les champs ne sont pas mutuellement exclusifs, il est possible de combiner les champs dans une même requête.

**Exemple:** `journalctl PRIORITY=4 SYSLOG FACILITY=0`

Par défaut, la requête utilise la combinaison *ET* des deux champs, pour utiliser l'un *OU* l'autre champs, il faut les séparer d'un plus (+).

**Exemple:** `journalctl PRIORITY=3 + SYSLOG FACILITY=0`

De plus, si deux valeurs d'un même champs sont dans la requête, alors par défaut il s'agit d'un *OU* logique entre les deux.

**Exemple:** `journalctl PRIORITY=3 PRIORITY=4`

**Remarque:** Pour plus d'informations, voir `man systemd.journal-fields`.



## Entrée Manuelle

Tout comme `logger`, il existe une interface de commande pour envoyer des messages au journal.

Il s'agit de : `systemd-cat`.

Il lit *stdin* et l'envoie au journal, mais peut être utilisé avec un pipe pour passer le *stdout* d'une commande.

De plus, `systemd-cat` suit d'une commande passe à la fois *stdout* et *stderr* au journal.

L'option `-p` permet de spécifier le niveau de priorité.

**Exemple:** `systemd-cat -p emerg echo "Ceci n'est pas grave"`

**Remarque:** pour plus d'information voir `man systemd-cat`.



## Stockage des Journaux

Il y a trois options pour le stockage du journal:

- ◆ La journalisation peut être désactivé.
- ◆ Sauvé sur une mémoire volatile, par exemple `/run/log/journal`, qui est perdu à chaque redémarrage.
- ◆ La rendre persistante, et donc sauvé sur le disque, par exemple `/var/log/journal`.

Le comportement par défaut est: si n'existe pas `/var/log/journal`, alors les journaux sont sauvé dans répertoire dans `/run/log/journal`, et donc perdu au redémarrage.

Le nom répertoire est construit en utilisant `/etc/machine-id`.



## Stockage des Journaux

Il est possible de modifier le comportement directement dans le fichier de configuration `/etc/systemd/journald.conf`.

Il faut changer la valeur de `Storage=`, qui peut prendre les valeurs `volatile`, `persistent`, `auto`, et `none`.

La différence entre `persistent` et `auto`, est que `auto` ne crée pas les répertoires si ceux-ci n'existent pas. C'est le comportement par défaut.

**Remarque:** Si les journaux dans `/var/log/journal/<machine-id>/` ou `/run/log/journal/<machine-id>` sont corrompus, ou que le daemon n'a pas été arrêté de façon satisfaisante, alors un `~` est accolé à la fin.



## Nettoyer le Journal

Les journaux sont stockés dans des fichiers journaux (*journal files*) qui se terminent en `.journal` ou `.journal~`.

Pour savoir combien de place est occupé par les journaux, on peut utiliser l'option suivante : `journalctl --disk-usage`

`systemd` fixe par défaut la limite des journaux à 10% de la taille du système de fichiers. Une fois la limite atteinte, les journaux les plus vieux sont supprimés pour rester le plus proche possible de cette valeur.

Sinon, les limites de taille sont fixées dans le fichier de configuration `/etc/systemd/journal.conf`. Elles s'appliquent soit à la mémoire persistante, et sont préfixées de `System`, ou à la mémoire volatile, et sont préfixées de `Runtime`.



## Nettoyer le Journal



`SystemMaxUse=`, `RuntimeMaxUse=`

- Contrôle la quantité de place que les journaux occupe, par défaut 10% de la taille du disque, sans dépasser 4GB.



`SystemKeepFree=`, `RuntimeKeepFree=`

- Contrôle la quantité de place libre qui doit être laisser pour les autres utilisateurs. Par défaut 15%, et sans dépasser 4GB.

`systemd` utilise la plus petite des deux valeurs précédentes, pour s'assurer de la validité des deux paramètres.



## Nettoyer le Journal

- `SystemMaxFileSize=`, `RuntimeMaxFileSize=`
  - Contrôle la taille maximale que chacun des fichiers journaux peut occuper. Par défaut,  $\frac{1}{8}$  de `*MaxUse`.
- `SystemMaxFiles=`, `RuntimeMaxFiles=`
  - Contrôle le nombre maximum de fichiers possible, par défaut 100.

Autre que les critères sur la taille des fichiers, il existe aussi des critères basées sur le temps, en utilisant `MaxRetentionSec=` et `MaxFileSec=`.

Pour plus d'informations, on peut utiliser `man journald.conf`.





## Nettoyer le Journal

Il est aussi possible de nettoyer le journal directement avec les options suivantes:



`--vacuum-time=`

- Supprime toutes les entrées plus vieille que le temps indiqué. La valeur doit être suivi d'un suffixe suivant: `s`, `m`, `h`, `days` (or `d`), `months`, `weeks` (or `w`), `years` (or `y`).



`--vacuum-size=`

- Supprime les fichiers journaux archivés jusqu'à ce qu'ils occupent moins de place qu'indiqué par la valeur.



`--vacuum-files=`

- Supprime les fichiers journaux archivés jusqu'à ce qu'ils y en ai plus que la valeur indiquée.



## Nettoyer le Journal

Pour se débarrasser de tous les fichiers journaux, y compris ceux actifs, il faut utiliser le signal `SIGUSR2`, qui applique une rotation immédiate des journaux avec l'option `--rotate`.

Le signal `SIGUSR1`, ou l'option `--flush`, permet d'écrire les journaux de `/run/` vers `/var/` pour les rendre persistents.

Le signal `SIGRTMIN+1`, ou l'option `--sync`, permet d'écrire sur le disque tous les messages journaux non sauvés.

Pour vérifier l'état et la cohérence des fichiers journaux, on peut utiliser l'option `--verify`.



## Sauvetages des Journaux

Il peut être nécessaire de récupérer les journaux d'une machine défaillante, à partir par exemple d'une clé bootable.

`journalctl` cherche pour les fichiers dans `/var/log/journal/<machine-id>` hors le système de sauvetage et le système défaillant n'auront pas le même identifiant.

Il faut donc utiliser l'option:

`-D </path/to/dir>`, `--directory=</path/to/dir>`

Il faut donc monter le `rootfs` du système défaillant (`/dev/sda1`) sur le système de sauvetage.



## Sauvetages des Journaux

D'autres options qui peuvent être utiles dans ce cas de figures sont:

- `-m, --merge`
  - Fusionne l'ensemble des journaux dans un unique journal `/var/log/journal.`
- `--file`
  - Montre l'ensemble des entrées dans un fichier spécifique.
- `--root`
  - Recherche en arborescence depuis le fichier donné. (e.g. `journalctl --root /faulty.system/`).

**Remarque:** pour plus d'information, `man journalctl`.



## Rétrocompatibilité

Les entrées des journaux peuvent être passé à un traditionnel daemon `syslog` en :

- En passant les message à la socket `/run/systemd/journal/syslog` pour que `syslogd` les lise. Il faut que `ForwardToSyslog=yes` soit défini dans le fichier de configuration.
- Que `syslogd` se comporte comme `journalctl`, en lisant directement les fichiers journaux. Pour cela, il faut que `Storage` ne soit pas `none`.



# Exercices





# 108

## Services systèmes essentiels

108.3 Bases sur l'agent de transfert de courrier



## MTA Local et Distant

Dans les systèmes Unix, chaque utilisateur a sa propre *inbox*, c'est à dire où sont stocké les mails de l'utilisateur.

C'est le *Mail Transfer Agent* (MTA) qui s'occupe de cela. Il s'agit d'un service du système. C'est lui aussi qui implémente *Simple Mail Transfer Protocol* (SMTP).

Il y a une *outbox* email pour l'ensemble des utilisateurs, a chaque fois qu'un message est placé dedans, MTA va essayer de l'envoyer au noeud réseau à partir du nom de domaine dans l'adresse (e.g. la partie après @).





## MTA Local et Distant

En pratique, la plupart des utilisateurs ont un compte mail distant, et n'ont pas de service MTA locale.

Contrairement à un compte local, un *remote mailbox*, nécessite l'authentification de l'utilisateur, à travers l'utilisation de IMAP ou POP3.

Si un daemon MTA tourne localement, un utilisateur peut envoyer des mails à d'autres utilisateurs locaux; ou à d'autres utilisateurs sur le réseau tant que leurs systèmes ont un service MTA qui tourne, et qui accepte les connections.

Traditionnellement, c'est le port 25 qui sert pour les communication SMTP.



## MTA Local et Distant

Pour s'assurer du bon fonctionnement du réseau d'échange de mails, il est nécessaire que tous les noeuds du réseau aient un MTA actif qui soit capable de :

- ◆ Maintenir la queue de message sortant (*outbox*). Pour chaque message dans la queue, le MTA local va évaluer le MTA destination à partir de l'adresse.
- ◆ Communiquer avec les daemons MTA distants en utilisant SMTP. Le MTA local doit être capable d'utiliser SMTP sur un pile TCP/IP pour recevoir, envoyer et rediriger les messages des autres MTA distant.
- ◆ Maintenir une boîte de réception (*inbox*) pour chaque utilisateur. Le MTA utilise le format *mbox*: un fichier texte qui contient l'ensemble des mails.



## MTA Local et Distant

Normalement, une adresse mail inclut un nom de domaine, e.g. `lpi.org` dans `info@lpi.org`. C'est le MTA du émetteur qui questionne le service DNS pour le MX (*mail exchange*) correspondant.

Le MX DNS contient l'adresse IP du MTA qui s'occupe des mails pour ce domaine. Si le même domaine à plusieurs MX, le MTA devrait les contacter d'après leurs priorités.

Si l'adresse destinataire n'a pas de domaine, ou que le domaine n'a pas de MX associé, alors la partie après le `@` est considéré comme l'hôte du MTA destinataire.

La sécurité est aussi importante puisque les MTA sont visible sur Internet. Un MTA qui transmet les messages sans verifier est un *open relay*.



## Linux MTA

Traditionnellement, le MTA pour les systèmes Linux est *Sendmail*, mais il existe aussi *Postfix*, *qmail*, et *Exim*, qui sont plutôt utilisés quand il s'agit d'implémenter des fonctionnalités avancées.

Si le MTA local n'accepte pas les connexions réseaux, alors il ne peut qu'envoyer des mails localement. Pour le MTA `sendmail`, le fichier `/etc/mail/sendmail.mc` doit être modifié pour accepter les connexions non-locales, en y modifiant:

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

avec la bonne adresse réseau, et puis redémarrer le service.

**Remarque:** La plupart des distributions Linux n'ont pas de MTA installé par défaut pour des raisons de sécurité.



## Linux MTA

Une fois qu'un MTA tourne, les nouveaux mails lui sont passés avec des commandes SMTP envoyées à travers une connexion TCP.

La commande `nc` peut envoyer directement les commandes SMTP au MTA, pour mieux comprendre le fonctionnement du protocole.

**Par exemple**, soit l'utilisateur `emma` sur le post `lab1.campus` qui veut envoyer un message à l'utilisateur `dave` sur le post `lab2.campus`.

```
$ nc lab2.campus 25
```

```
220 lab2.campus ESMTP Sendmail 8.15.2/8.15.2; Sat, 16 Nov  
2019 00:16:07 GMT
```



## Linux MTA

```
HELO lab1.campus
```

```
250 lab2.campus Hello lab1.campus [10.0.3.134], pleased to  
meet you
```

```
MAIL FROM: emma@lab1.campus
```

```
250 2.1.0 emma@lab1.campus... Sender ok
```

```
RCPT TO: dave@lab2.campus
```

```
250 2.1.5 dave@lab2.campus... Recipient ok
```



## Linux MTA

**DATA**

354 Enter mail, end with "." on a line by itself

**Subject: Recipient MTA Test**

**Hi Dave, this is a test for your MTA.**

**.**

250 2.0.0 xAG0G7Y0000595 Message accepted for delivery

**QUIT**

221 2.0.0 lab2.campus closing connection



## Linux MTA

La commande `sendmail` permet d'envoyer directement un mail, et s'occupe des commandes SMTP a envoyer.

**Par exemple:**

```
$ sendmail dave@lab2.campus
```

```
From: emma@lab1.campus
```

```
To: dave@lab2.campus
```

```
Subject: Sender MTA Test
```

```
Hi Dave, this is a test for my MTA.
```

```
.
```





## Linux MTA

La commande `mailq`, ou `sendmail -bp`, permet de lister les messages non-délivrés, et la cause de leurs échecs.

Par défaut, la queue de l'*outbox* est dans `/var/spool/mqueue/`, mais différents MTA utilisent différent répertoire.

Si un hôte de destination n'est pas accessible, le MTA garde le messages et cherche à contacter les autres entrées MX si il y en a. Si aucune n'est accessible, le message reste dans la queue *outbox* et le MTA essaiera de l'envoyer périodiquement.

La commande `sendmail -q` permet de re-essayer l'envoi instantanément.



## Linux MTA

`sendmail` place les messages arrivant dans un fichier nommé après l'utilisateur dont c'est la boîte, e.g. `/var/spool/mail/dave`.

Le MTA a fini son travail une fois que le message est sauvé dans la boîte mail de l'utilisateur.

Il peut y avoir d'autres filtrages qui sont effectué après, pour éviter les spams par exemple en appelant *SpamAssassin*.

Il est préférable d'utiliser un programme client mail (tel que Thunderbird, Evolution ou KMail), pour lire ses mails plutôt que le fichier texte directement.



## La Commande `mail`

Il est bien plus pratique d'utiliser une application client pour lire et écrire ses mails, ce sont des *Mail User Agent* (MUA).

Par exemple *Mozilla Thunderbird*, *Gnome's Évolution*, même les *Webmail* peuvent être vu comme des MUA.

A l'origine, la commande `mail` était prévu pour l'échange de mail entre utilisateurs locales d'un même système. Aujourd'hui, dans la plupart des distributions Linux, la commande `mail` est un lien symbolique vers `mailx`.

Qu'importe son implémentation, `mail` à deux modes: *normal mode* et *send mode*. Si une adresse est passé en argument, alors la commande rentre en *send mode*, et sinon elle est en *normal mode* (e.g. en mode lecture).



## La Commande `mail`

En *normal mode* (e.g. en mode lecture), les messages reçus sont listés avec un index numérique pour que l'utilisateur puisse y faire référence lors de ces commandes.

**Exemple:** `print 1`

Les commandes classiques `print`, `delete`, et `reply`, peuvent s'abréger en `p`, `d`, et `r`. De même avec `quit` en `q` pour quitter le programme.

En *send mode*, c'est le *stdin* qui sert de corps au message, ce qui peut être pratique pour envoyer des mails lors d'erreur ou d'échec d'un script programmé.

**Exemple:** `$ mail -s "Maintenance fail" henry@lab3.campus  
<<<"The maintenance script failed at `date`"`



## Livraison Personnalisée

Par défaut, un utilisateur `carol` sur l'hôte `lab2.campus` à une adresse mail du type `carol@lab2.campus`. On peut étendre cela en utilisant le mécanisme de routage fourni dans le fichier `/etc/aliases`.

Une adresse aliases est une adresse destination "virtuelle" qui redirige les messages vers une boîte existante.

**Par exemple:** Si on rajoute la ligne `postmaster: carol` au fichier `/etc/aliases` sur `lab2.campus`, alors les mails envoyé à `postmaster@lab2.campus` iront dans la boîte de réception de Carol.

Après avoir modifier les aliases, il faut mettre à jour le MTA, soit avec `newaliases`, soit avec `sendmail -bi` ou `sendmail -I`.



## Livraison Personnalisée

Les aliases sont définies une par ligne, au format `<alias>: <destination>`.

En plus des boîtes de réception ordinaire indiquées par le nom d'utilisateur, les autres destinations possibles sont:

- Un chemin absolu vers un fichier. Les messages seront ajoutés au fichier.
- Une commande à exécuter sur le message. Il faut alors que la destination commence avec le symbole *pipe* (`|`).
- un fichier *include*. Un alias peut avoir plusieurs destinations (séparées par une virgule), cela peut être pratique de les garder dans un fichier externe. Il faut indiquer le chemin avec le mot clé `:include:`.
- Une adresse externe, c'est une forme de forwarding.
- Un autre alias



## Livraison Personnalisée

Finalement, un utilisateur peut créer un fichier `.forward` dans son répertoire maison, qui permet de créer des aliases que pour lui-même.

Dans ce cas, que la partie `<destination>` est nécessaire, un par ligne.

**Exemple**, chez `dave` sur la machine `lab2.campus`:

```
$ cat ~/.forward
```

```
emma@lab1.campus
```

L'ensemble des mails envoyé à `dave@lab2.campus` seront forwarder à `emma@lab1.campus`.



# Exercices







# 108

## Services systèmes essentiels

108.4 Gestion des imprimantes et de l'impression



## Le Service CUPS

Le service *Common Unix Printing System* (CUPS) s'occupe de la gestion de l'impression et des imprimantes.

