Administration **Unix** (Part. 1)

Thierry Vaira

Esimed

v1.1 - 10 février 2019

esimed
École supérieure d'informatique de la Méditerranée

Sommaire

esimed
École supérieure d'informatique de la Méditerranée

Sommaire

Composants principaux

Le système d'exploitation Linux est composé :

- d'un noyau (*kernel*)
- d'un *shell* et des applications
- d'un système de fichiers
- de la mémoire virtuelle, composée de la RAM physique et de la zone de *swap*, généralement stockée sur un disque dur local.

Les tâches (services) du système d'exploitation sont assurées par des processus qui fonctionnent en permanence dans le système : les démons (*daemons*).

esimed
École supérieure d'informatique de la Méditerranée

Le noyau

Le noyau est en charge des opérations de base suivantes :

- gestion des périphériques (à travers les pilotes de périphériques, *driver*), de la mémoire, des processus et démons,
- contrôle des échanges (des flux de données, par exemple TCP/IP) entre les utilitaires du système et le matériel,
- séquençage et exécution des processus et partage du temps des processeurs (*scheduler*),
- gestion de la mémoire virtuelle.

Le shell et les applications

Le *shell* constitue l'interface entre le noyau et l'utilisateur. Les *shells* disponibles (les plus utilisés) sous Ubuntu Linux sont :

- Le *Bourne Shell* (`sh`) : développé pour l'environnement UNIX®, il constitue le *shell* le plus utilisé pour l'écriture des scripts.
- Le *C Shell* (`csh`) : version améliorée du *Bourne Shell* avec une syntaxe proche du langage C.
- Le *GNU Bourne Again Shell* (`bash`) : *shell* compatible avec le *Bourne Shell* qui incorpore les fonctionnalités telles que l'historique des commandes, les alias et l'édition de la ligne de commande. C'est le *shell* **par défaut** de Ubuntu Linux.

esimed
École supérieure d'informatique de la Méditerranée

Système de fichiers

- Un système de fichiers (*file system*) définit l'organisation d'un disque (ou plus précisément d'une partition d'un disque).
- C'est une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur ce que l'on appelle des mémoires secondaires (disque dur, disquette, CDROM, clé USB, disques SSD, etc.).
- Il existe de nombreux systèmes de fichiers différents : ext2, ext3, ext4, UFS, HFS, reiserfs, etc.
- Chaque fichier est décrit par des métadonnées (conservées dans l'**inode** sous Unix), alors que le contenu du fichier est écrit dans un ou plusieurs blocs (taille fixe) du support de stockage selon la taille du fichier. Le terme inode désigne le descripteur d'un fichier sous UNIX.
- La commande `stat` permet d'afficher l'intégralité du contenu de l'inode.

esimed
École supérieure d'informatique de la Méditerranée

L'arborescence Unix

Le système de fichiers sous Linux se compose d'une hiérarchie de répertoires, sous-répertoires et fichiers. Le répertoire le plus élevé dans l'arborescence est nommé la racine (*root*) symbolisé par `/`. L'arborescence est **unique**. Quelques répertoires importants :

- `/etc` : contient les fichiers de configuration du système et des applications
- `/dev` : contient les fichiers spéciaux de périphériques qui représente les points d'accès au matériel
- `/bin` : contient les commandes de base du système
- `/sbin` : contient les outils systèmes pour l'administration
- `/usr` : contient les commandes et applications pour les utilisateurs, dont les environnements graphiques
- `/home` : contient les répertoires personnels des utilisateurs
- `/var` : contient les fichiers dont le contenu varie en fonction de l'utilisation du système (bases de données, fichiers de logs, ...)
- `/proc` : représente le point d'accès aux informations (variables, tables, liste, ...) du noyau et des processus

Sommaire

Les paquetages (packages)

- Une majorité de logiciels sont fournis sous forme de **paquetages (*packages*)** dans l'environnement GNU/Linux.
- Un paquet (*package*) contient :
  - des fichiers décrivant le *package* (description, version, signature, dépendances, ...)
  - les fichiers à installer
  - des scripts qui s'exécutent avant ou après l'installation ou la suppression

✍ `.deb` pour les paquets DEBIAN

Les gestionnaires de paquets DEBIAN

- **dpkg** est un outil pour l'installation, la création, la suppression et la gestion des paquets Debian.
- **aptitude** est la principale interface CLI à **dpkg**.
- **APT** est un système de gestion de paquets logiciels.
- **synaptic** est la principale interface GUI à **APT**.

✍ Les sources pour les paquets sont énumérées dans le fichier `/etc/apt/sources.list`.

Informations sur les paquets

```
$ dpkg -l # liste les paquets
$ dpkg --get-selections # liste les paquets installés
$ dpkg -L <paquet> # liste les fichiers d'un paquet installé
$ dpkg -S <fichier> # recherche le <fichier> dans les paquets installés
$ dpkg -p <paquet> # affiche les informations sur un paquet
$ apt-cache show <paquet> # affiche les informations sur un paquet
$ dpkg-query -s <nom> # affiche les informations sur un paquet installé
$ apt-cache search <mot> # recherche <mot> (RE) dans les noms et descriptions

$ apt-file search|list <pattern> # utilitaire pour rechercher dans les paquets
```

Gestion des paquets

Quelques commandes :

```
# resynchroniser l'index répertoriant les paquets disponibles et sa source
$ sudo apt-get update

# installer les versions les plus récentes de tous les paquets
$ sudo apt-get upgrade|dist-upgrade

# installer un paquet
$ sudo dpkg -i <paquet>
$ sudo apt-get install <paquet>

# réconfigurer un paquet
$ sudo dpkg-reconfigure <paquet>

# réinstaller un paquet
$ sudo apt-get --reinstall <paquet>

# désinstaller un paquet
$ sudo apt-get remove|purge <paquet>
$ sudo apt-get remove --purge <paquet> # avec les fichiers de configuration
```

Sommaire

Des outils d'audit

- L'utilitaire **`who`** peut fournir des informations sur les utilisateurs connectés et la commande **`w`** affiche les utilisateurs connectés et ce qu'ils font.
- La commande **`last`** peut être utilisée pour déterminer toutes les connexions/déconnexions d'un utilisateur.
- La commande **`uptime`** nous renseigne depuis quand le système fonctionne et la charge de celui- ci.
- La commande **`vmstat`** permet de suivre en temps réel l'activité de la mémoire virtuelle, des zones de swap, des processus, ...
- La commande **`top`** permet de suivre en temps réel l'activité des processus.
- La commande **`netstat`** affiche les informations du sous-système réseau.
- La commande **`fuser`** identifie les processus qui utilisent des fichiers et la commande **`lsof`** liste les fichiers ouverts.
- Voir aussi : `ps`, `pstree`, `pgrep`, `free`, `top`, `df`, `du`, `slabtop`, ...

syslog/rsyslog

Syslog a été la solution de journalisation standard sur les systèmes Unix et Linux. Les journaux sont stockés dans `/var/log/`.

- `syslog` est à la fois un **protocole** définissant un service de journalisation et un **logiciel** responsable de la prise en charge des fichiers de journalisation du système (`/var/log/syslog`). Sa configuration est réalisée dans le fichier `/etc/syslog.conf`.
- `rsyslog` est un logiciel libre utilisé sur des systèmes d'exploitation Unix et Linux pour remplacer `syslog`. Sa configuration est réalisée dans le fichier `/etc/rsyslog.conf`.
- La journalisation dans `syslog/rsyslog` (local ou distant) se fait via la commande `logger`. Par défaut, les messages sont enregistrés dans le fichier `/var/log/messages`.

```
$ logger -p syslog.notice -t "test" -i "mon message"
```

journalctl

systemd possède son propre mécanisme de journalisation, **syslog** n'est plus requis par défaut. La commande **journalctl** permet d'accéder au *log* (journal).

- Tout le log : **journalctl** (l'option `-f` pour un affichage continu comme pour `tail`)
- Par service : **journalctl -u ssh**
- Par PID : **journalctl _PID=1**
- Par exécutable : **journalctl /usr/sbin/ntpd**
- Par jour : **journalctl -since="today"**
- Par niveau : **journalctl -p err**
- Gestion de l'espace : **journalctl -disk-usage**
- Suppression : **sudo journalctl -vacuum-size=10M**

✍ La configuration du journal de **systemd** est réalisée avec le fichier **/etc/systemd/journald.conf**.

Sommaire

systemd

- À l'origine les systèmes Unix/Linux utilisaient le processus **init** (**sysvinit**) pour assurer le démarrage de l'OS. Certains systèmes ont utilisé ensuite **Upstart**. La majorité utilise maintenant **systemd**.

```
$ ps -p1 | grep systemd && echo systemd || echo upstart
```

- **systemd** est un système d'initialisation du système qui active et maintient les **services** pour le noyau Linux.
- **systemd** introduit la notion d'**unité**. Une unité représente un fichier de configuration. Une unité peut être un service (`*.service`), un *target* (`*.target`), un montage (`*.mount`), un *socket* (`*.socket`), ...
- L'outil de gestion des services (et des autres unités d'ailleurs) dans **systemd** s'appelle **systemctl**.

esimed
École supérieure d'informatique de la Méditerranée

systemctl I

```
# liste les unités
$ sudo systemctl list-units

# liste les services
$ sudo systemctl list-units --type=service
$ sudo systemctl --state=running

# contrôle d'un service
$ sudo systemctl start|stop|relaod|status|enable|disable <service>

# visualise le contenu d'un service
$ sudo systemctl cat <service>

# une GUI pour systemd
$ sudo apt-get install systemd-ui
$ sudo systemadm
```

runlevel

- Dans **systemd**, la notion de *runlevel* est remplacé par le concept de *target*.

Runlevel	*Target*
0	poweroff.target
1	rescue.target
2, 3, 4	multi-user.target
5	graphical.target
6	reboot.target

- Pour afficher le *runlevel* : `$ runlevel`
- Pour changer de *runlevel* : `$ sudo systemctl isolate multi-user.target`
- Pour activer un *runlevel* : `$ sudo systemctl enable multi-user.target`
- Pour mettre un *runlevel* par défaut :

  `$ sudo systemctl set-default multi-user.target`

esimed
École supérieure d'informatique de la Méditerranée

Création et gestion d'un service systemd I

Les services gérés par **systemd** sont stockés dans `/etc/systemd/system/` (configuration locale) et `/lib/systemd/system/` (unités installées).

Un simple service

```
$ sudo vim /etc/systemd/system/hello.service
[Unit]
Description=bla bla ...
After=multi-user.target

[Service]
Type=simple
ExecStart=/bin/sh -ec "exec echo \"Bonjour Maître\" > /tmp/bonjour"

[Install]
WantedBy=default.target
```

Création et gestion d'un service systemd II

```
# Recharger systemctl
$ sudo systemctl daemon-reload

# Démarrer le service
$ sudo systemctl start hello.service

# Consulter l'état
$ sudo systemctl status hello.service

# Consulter le journal (log)
$ sudo journalctl -u hello.service

# Visualiser les propriétés assoiées à un service
$ sudo systemctl show hello.service

# Arrêter le service
$ sudo systemctl stop hello.service

# Activer le service (puis redémarrer)
$ sudo systemctl enable hello.service
```

Les fichiers d'initialisation I

- Un fichier d'initialisation contient une série de commandes qui sont exécutées lorsque le *shell* démarre. Il permet de personnaliser l'environnement de travail.
- Il existe deux sortes de fichiers d'initialisation : système et utilisateur.
- Les fichiers d'initialisation système sont gérés par l'administrateur (*root*), se trouvent dans le répertoire `/etc` et sont communs à tous les utilisateurs.
- Les fichiers d'initialisation utilisateur se trouvent à la racine du répertoire personnel (`$HOME`) de chaque utilisateur.
- On distingue deux types de fichiers d'initialisation : les fichiers exécutés à la connexion (*login*) seulement et les fichiers exécutés à chaque lancement de *shell*.

esimed
École supérieure d'informatique de la Méditerranée

Les fichiers d'initialisation II

File	Description
/etc/profile	The system-wide initialization file; executed when you log in.
/etc/bashrc	Another system-wide initialization file; may be executed by a user's *.bashrc* for each **bash** shell launched.
~/.bash_profile	If this file exists, it is executed automatically after */etc/profile* when you log in.
~/.bash_login	If *.bash_profile* doesn't exist, this file is executed automatically when you log in.
~/.profile	If neither *.bash_profile* nor *.bash_login* exists, this file is executed automatically when you log in.
~/.bashrc	This file is executed automatically when **bash** starts.
~/.bash_logout	This file is executed automatically when you log out.
~/.inputrc	This file contains optional key bindings and variables that affect how **bash** responds to your keystrokes.

- Le répertoire `/etc/skel` contient des modèles de fichiers d'initialisation pour les utilisateurs. Le contenu de ce répertoire est copié dans le répertoire de l'utilisateur lors de sa création à l'aide de la commande **`useradd`**, si l'option `-m` est utilisée.

Sommaire

Les utilisateurs

- Sur UNIX, chaque utilisateur est identifié par un nom, par un UID (*User IDentification*) et par un GID (*Group IDentification*).
- Il peut appartenir à plusieurs groupes, eux-mêmes identifiés par un nom et par un GID.
- Comptes utilisateurs locaux définis dans les fichiers : `/etc/passwd`, `/etc/shadow` et `/etc/group` ...
- Commandes de base : `id`, `whoami`, `who`, `who am i`, `w`, `last`, `users`, ...
- Commandes d'administration : `useradd`, `userdel`, `usermod`, `passwd`, `chfn`, `chage`, `chsh`, `groupadd`, `groupdel`, `groupmod`, `adduser`, `addgroup`, `deluser`, ...

esimed
École supérieure d'informatique de la Méditerranée

Sommaire

Contrôler l'accès aux fichiers

Sous UNIX, il existe deux types de sécurité pour les fichiers et répertoires : les droits et permissions UNIX et les ACL (*Access Control List*).
Il y a trois types de permissions :

- `r` : accès en lecture
- `w` : accès en écriture
- `x` : exécution (fichier), traversée (répertoire)

Chacune de ces permissions peuvent être attribuée à :

- `u` : l'utilisateur (propriétaire)
- `g` : le groupe (propriétaire)
- `o` : les autres
- `a` : tout le monde

Droits et permissions supplémentaires

- SETUID : exécution avec l'UID du propriétaire du fichier, pas d'effet sur les répertoires
- SETGID : exécution avec l'GID propriétaire du fichier, création d'un répertoire avec le GID propriétaire du répertoire parent
- BIT STICKY : suppression restreinte pour le répertoire, conserver l'image d'un programme en mémoire

Les commandes :

- `chmod` : modifier les droits et permissions UNIX
- `umask` : fixer le masque de création de fichiers

esimed
École supérieure d'informatique de la Méditerranée

ACL (*Access Control List*)

Les ACL permettent d'ajouter une gestion avancée des droits :

- Ainsi, il devient possible d'autoriser un utilisateur à effectuer des opérations sur un fichier (ou répertoire) sans autoriser tout un groupe ou tout le reste du monde.
- Au moyen des ACL, on peut donc étendre le nombre d'utilisateurs et de groupes ayant des droits sur un même fichier (ou répertoire).
- Un fichier dont les ACL auront été spécifiés verra s'ajouter un + à la fin de la liste des droits (`ls -l`)
- Il faut installer le *package* `acl` : `[sudo] apt-get install acl`

Les commandes :

- l'attribution des droits : commande `setfacl`
- la lecture des droits : commande `getfacl`

ACL (*Access Control List*)

Remarques :

- droits par défaut : ajout d'un attribut *default* (`d:`) aux répertoires seulement et qui se transmet à tous les fichiers créés dans le répertoire.
- masque : son intérêt est de pouvoir limiter toutes les permissions d'un fichier sauf celles du propriétaire

Si le système de fichiers de destination le permet :

- La commande `mv` préserve toujours les droits.
- La commande `cp` peut préserver les droits avec l'option `-a`.

Sommaire

Gérer les processus

- `ps`, `top`, `jobs` : lister les processus
- `pidof` : afficher le PID d'un processus
- `kill`, `killall`, `pkill` : envoyer un signal (`kill -l`) à un processus ou plusieurs (dont : interrompre, stopper, terminer ou tuer)
- `&` à la fin de la ligne de commande : lancer une commande en arrière plan
- `nohup` : détacher le processus de la console
- Ctrl + Z , `fg`, `bg` : mettre en pause, déplacer une tâche au premier plan, en arrière plan
- `at` : lancer des commandes à une heure précise (différé)
- `batch` : exécuter des commandes lorsque la charge système le permet
- `cron` : planifier l'exécution de commandes (périodiques)

esimed
École supérieure d'informatique de la Méditerranée

Sommaire

- La création d'une partition sur un disque existant ou un nouveau disque peut se faire avec `parted`, `fdisk`, `sfdisk` ou l'outil **GParted**.
- La création d'un système de fichier (formatage) sur une partition peut se faire avec la commande `mkfs`.
- On peut réaliser des copies physiques avec la commande `dd`.
- Un système de fichiers doit être monter sur l'arborescence racine avec la commande `mount`.
- Les systèmes de fichiers montés automatiquement au démarrage du système sont listés dans `/etc/fstab`. Le fichier `/etc/mtab` contient la liste des systèmes de fichiers actuellement montés sur le système.

```
$ lsblk
$ sudo fdisk -l
$ df -hT
$ cat /proc/partitions
$ cat /proc/mounts
$ mount
$ sudo du -sh /
```

Sommaire

Zone de swap

La zone de *swap* est utilisée lorsque la mémoire physique (RAM) est remplie. Si le système a besoin de plus de ressources mémoires et que la mémoire physique est remplie, les pages de mémoire (bloc de taille fixe) inactives (non utilisées depuis un certain temps) sont déplacées dans la zone de *swap*.
Quelques règles de base pour la zone de *swap* :

- Elle peut être stockée dans une partition dédiée ou un fichier
- Le paramètre `swappiness` (une valeur entre 0 et 100) contrôle la tendance du noyau à déplacer les processus de la mémoire physique vers le *swap*. La valeur par défaut est 60. Une valeur faible fait que le noyau évitera de permuter (0 désactivant le *swap*). Son réglage se fait dans le fichier `/etc/sysctl.conf` : `vm.swappiness = 10`
- Avec une Raspbian, un fichier de *swap* `/var/swap` de 100M est utilisé par défaut.

Configuration

- Les zones de *swap* en cours d'utilisation par le système peuvent être listées avec la commande **`swapon -s`** (partition ou fichier)
- La création d'une partition sur un disque existant ou un nouveau disque peut se faire avec les outils **`parted`**, **`fdisk`** ou autre. Pour un fichier, on utilisera la commande **`dd`**.
- Il faut ensuite créer les structures de la zone de *swap* avec la commande **`mkswap`** et l'activer avec **`swapon`**
- L'arrêt de l'utilisation de la zone de *swap* se fera avec la commande **`swapoff`** (partition ou fichier).
- Pour les partitions de *swap* (seulement), il faut renseigner leur montage dans le fichier **`/etc/fstab`**.

esimed
École supérieure d'informatique de la Méditerranée

Fichier de swap

- Les fichiers de *swap* sont contrôlés par la commande `dphys-swapfile setup|install|swapon|swapoff|uninstall` ou `swapon/swapoff` (partition ou fichier)
- La configuration des fichiers de *swap* est réalisée dans le fichier `/etc/dphys-swapfile` :
  ```
  $ cat /etc/dphys-swapfile | sed '/^$/d;/^#/d'
  CONF_SWAPSIZE=100
  ```
- Contrôle du service de fichiers de *swap* :
  ```
  $ sudo systemctl start|stop|status|enable|disable
  dphys-swapfile.service
  ```
- Statistiques et informations : `free`, `vmstat`, `top`, ...

✍ Suivant l'usage de son système, on peut envisager de désactiver ou déplacer sur un disque USB externe le *swap* sur une Raspberry Pi.

Systèmes de fichiers en mémoire : *ramfs* et *tmpfs*

Les systèmes de fichiers en mémoire sont parfois nommés pseudo systèmes de fichiers. Ils ne résident pas sur un disque physique mais en mémoire. Il en existe deux sortes :

- ceux qui permettent de créer un disque en mémoire : **ramfs**.
- ceux qui permettent d'accéder à des informations du système par l'intermédiaire de fichiers : **tmpfs** (il utilise la mémoire partagée du système).

✍ Les répertoires `/tmp`, `/var/log`, `/var/tmp` et `/var/run` sont de « bons candidats » pour `tmpfs`. C'est déjà le cas pour `/var/run` (cf. `/run`). Il suffit d'ajouter une ligne dans `/etc/fstab` :

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,size=100m 0 0
```

ramdisk

- Un *ramdisk* est une zone de mémoire utilisée comme partition.
- Une fois montée, cette partition est utilisable comme n'importe quelle partie du système de fichiers.
- Pour créer un *ramdisk*, rien de plus simple : il suffit de le monter avec la commande `mount` et l'option `-t ramfs` :

```
$ sudo mkdir /ramdisque
$ sudo mount -t ramfs -o maxsize=10M none /ramdisque
```

- On peut aussi faire en sorte qu'il soit créé au démarrage en rajoutant une ligne au fichier `/etc/fstab`.

esimed
École supérieure d'informatique de la Méditerranée

Sommaire

Qu'est ce que le RAID ?

- Le RAID (*Redudant Array Of Independent Disks*) permet de combiner plusieurs disques en une seule partition dans le but d'augmenter soit la performance, soit la redondance des informations, soit les deux.
- Les différentes méthodes RAID les plus courantes sont nommées par **niveau** (*level*) :
  - le *disk striping* (volume segmenté) ou RAID level 0
  - le *mirroring* (volume en miroir) ou RAID level 1
  - le *disk striping with parity* (volume segmenté avec parité) ou RAID 5
- Il existe deux sortes de RAID :
  - Le RAID matériel (les contrôleurs de disques gèrent eux-mêmes le RAID)
  - Le RAID logiciel (il existe plusieurs solutions pour Linux, dont le pilote **MD**)

esimed
École supérieure d'informatique de la Méditerranée

Principe

Le principe de base est la distribution des données sur plusieurs disques d'un même ensemble (*disk array*), les disques étant regroupés en un seul volume logique. Les données sont découpées en blocs de taille fixe (*chunks*), puis ces blocs sont distribués sur les différents disques du volume logique suivant un algorithme déterminé par le niveau RAID.

- Level 0 : est utilisé pour améliorer la **performance** en distribuant la charge de lecture/écriture de façon équitable sur tous les disques de l'ensemble.
- Level 1 : est utilisé pour améliorer la **sécurité** en augmentant la redondance, les données étant écrites en plusieurs exemplaires sur plusieurs disques en même temps.
- Level 4 : est destiné à améliorer la sécurité en utilisant un disque pour le stockage de la parité. Ce niveau est très peu utilisé.
- Level 5 : offre un maximum de sécurité avec une bonne performance. Il utilise le principe du striping du Level 0 et la technique de la parité du Level 4, sauf que la parité est répartie sur l'ensemble des disques, éliminant le problème de performance du Level 4. C'est le niveau le plus utilisé.

Configuration

- La configuration d'un ensemble RAID (pilote **MD**) se fait à l'aide de la commande `mdadm`
- Les informations concernant les ensembles RAID en cours de fonctionnement sont accessibles avec le fichier `/proc/mdstat`
- Pour que les ensembles soient activés au démarrage il faut créer un fichier de configuration `/etc/mdadm/mdadm.conf`
- Pour que l'ensemble soit monté automatiquement au démarrage, il faut ajouter une ligne au fichier `/etc/fstab`

esimed
École supérieure d'informatique de la Méditerranée