GNU-Linux

Administration Avancée

Version 21.12
Auteur : F. Micaux
formation@actilis.net

Table des matières

Notes personnelles

Notes personnelles

1- Installation avancée et déploiement

Notes personnelles

1.1- Installation minimale : démarrage

1.1.1- Besoins matériels

1 Go de RAM.

1 Go par processeur logique recommandé par RH.

Plusieurs processeurs (y compris pour une VM)

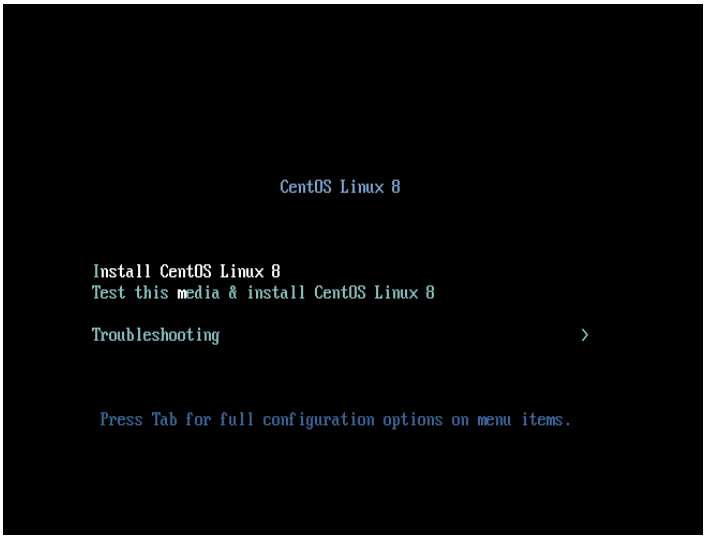
1 suffit pour CentOS 8.

Pour une installation graphique

1024x768 recommandé.

Espace disque minimum :

10 Go semblent une base courante.



Au sujet des pilotes :

Plus de support des pilotes réseau 32 bits (comme l'AMD PCNet32 émulé par de vieilles versions de VMWare.

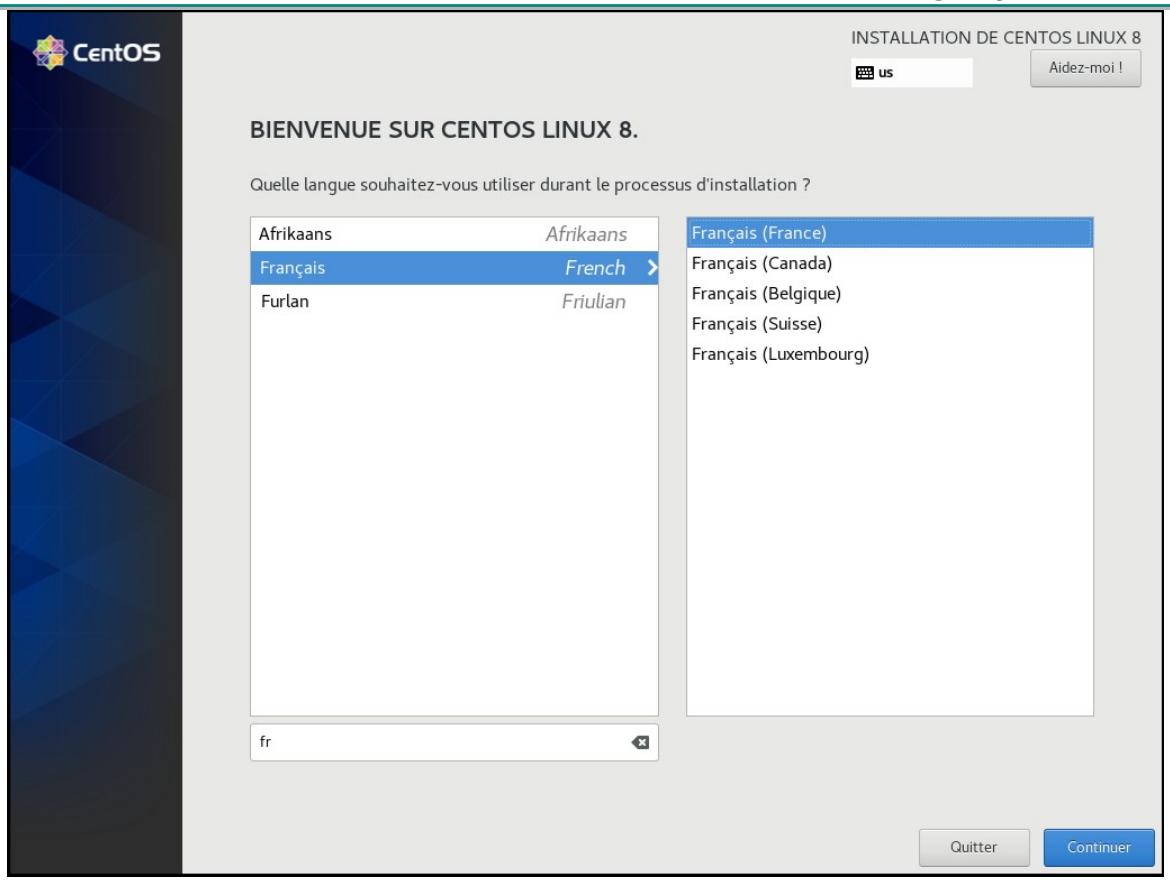
Passer à une "e1000" en ajoutant dans le fichier "vmx" décrivant la VM la ligne suivante :

```
ethernet0[1].virtualdev="e1000"
```

1 Ou ethernet1 ou ethernet2… si la VM est dotée de plus d'une interface réseau.

Notes personnelles

1.2- Écran d'accueil, choix de la langue pour l'installation

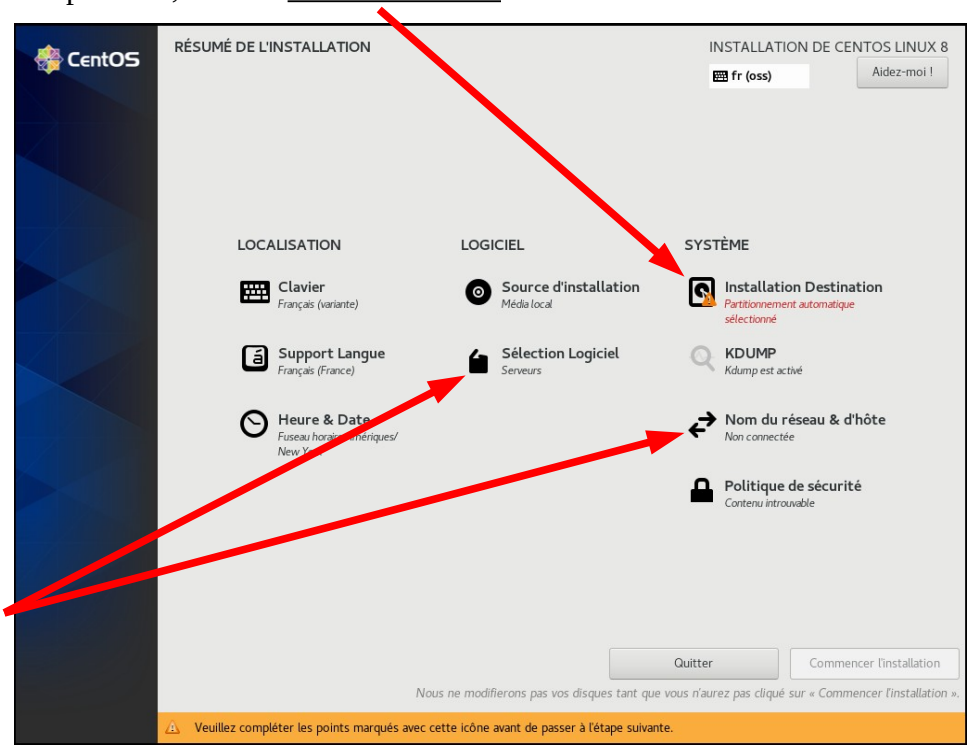


Notes personnelles

1.3- Panneau de contrôle de l'installation

Si des choix nécessaires ne sont pas faits, ils sont **mis en évidence**.

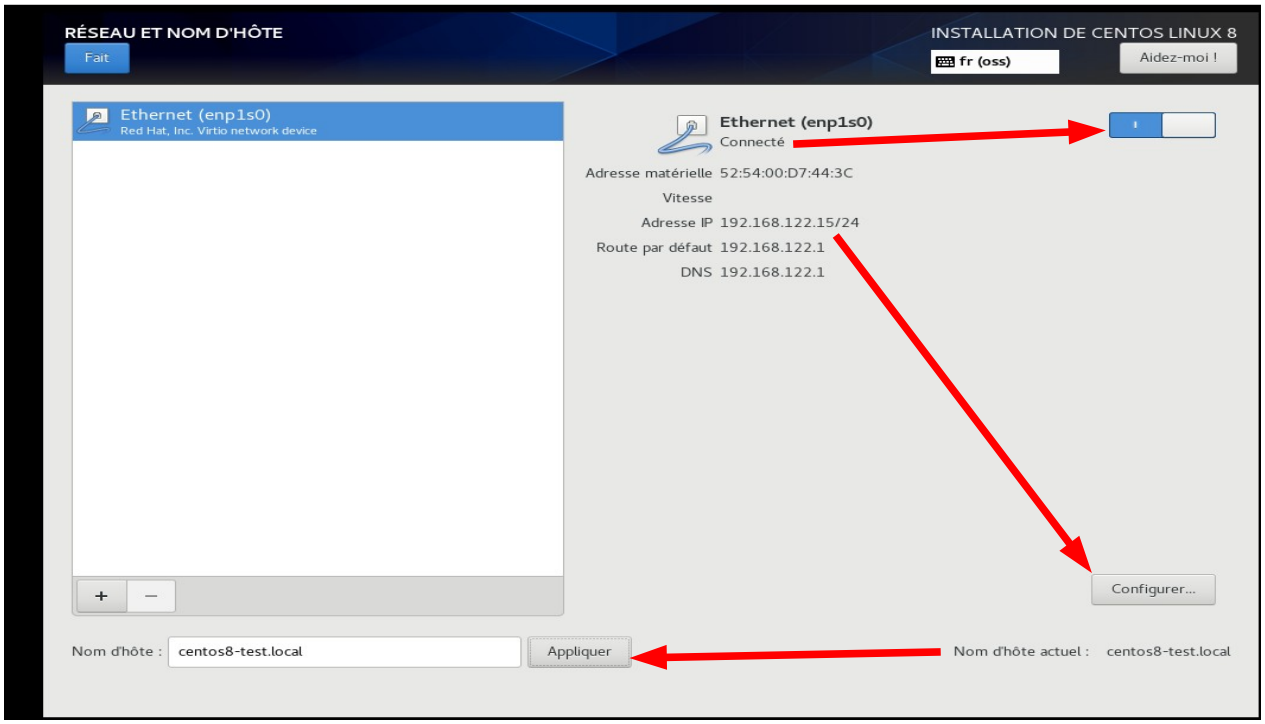
D'autres points peuvent nécessiter un peu d'attention...



Notes personnelles

1.4- Nom d'hôte et réseau

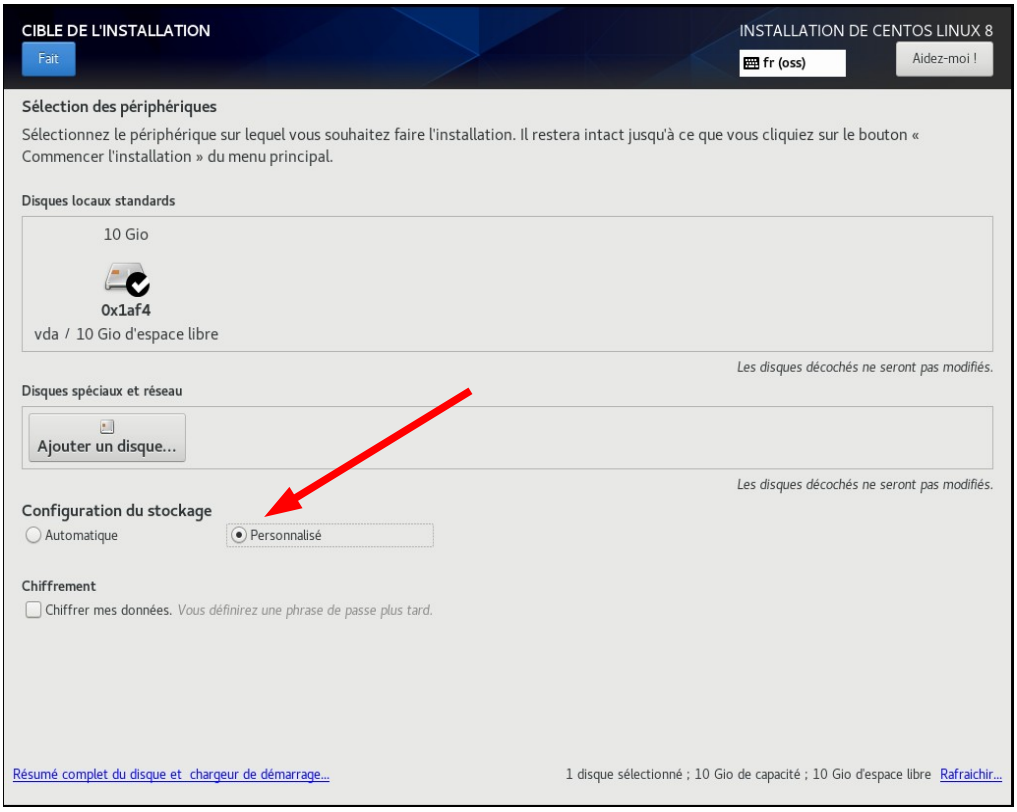
Quand c'est "Fait", cliquer sur "Fait"...



Notes personnelles

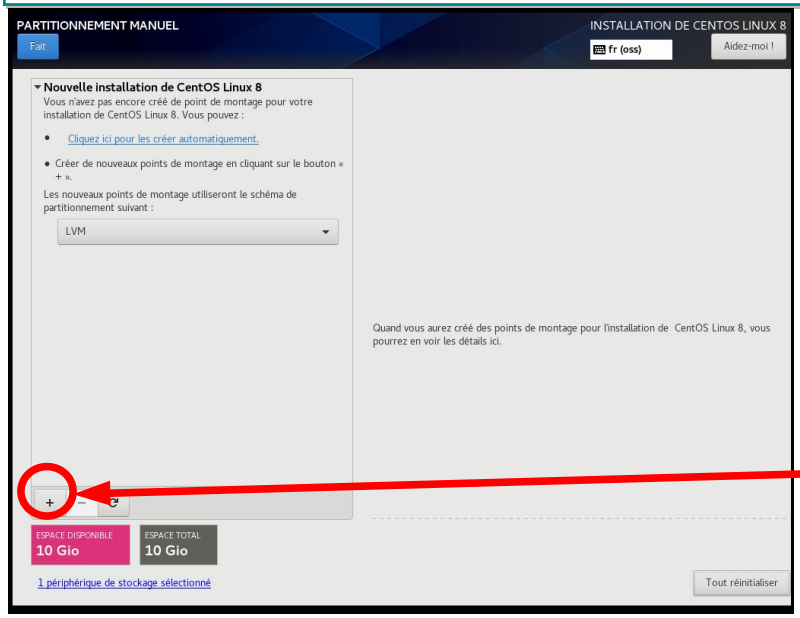
1.5- Choix du disque cible

Choisir le disque à partitionner, le mode "Personnalisé", puis cliquer sur "Fait".



Notes personnelles

1.6- Partitionnement



Choix du mode de partitionnement... le mode personnalisé (manuel) est souvent la bonne solution !

LVM[2] reste le choix conseillé pour la gestion des volumes de stockage.

Le type de filesystem par défaut dans la version 8 est **XFS sur le root-filesystem** et **ext4 sur /boot**[3].

Pour créer des partitions, c'est ici !

2 RedHat / CentOS ne supporte toujours pas une partition "/boot" reposant sur LVM, il faut donc commencer pat une partition "standard" pour /boot. 500 Mo suffiront.

3 La famille "ext" est aussi propsoée et ext4 reste un bon choix : on peut réduire la taille d'un FS ext4, ce n'est pas le cas pour XFS. BTRFS, quant à lui, n'est plus proposé (il a avait été introduit avec CentOS 7.0).

Notes personnelles

1.6.1- Quelles partitions doit-on créer ?

Au moins **2 partitions sont nécessaires** : une partition **racine**, et une pour le **swap**.

La **partition racine**, sera montée sur « / » et constituera le « **root filesystem** »

La **partition de swap** sera appelée « **swap primaire** »

Une **3ème partition** est parfois nécessaire pour **/boot**.

On a forcément une partition **/boot** pour les systèmes dont le root-filesystem est sur LVM,
L'outil d'installation n'accepte pas de poursuive sinon et indique une erreur. ("/boot on lvmlv")

Le répertoire (la partition) /boot contient :

les fichiers du **chargeur de démarrage** (**GRUB**)

le **fichier noyau** (/boot/**vmlinuz**-x.y.z..).
Il est standard et son horodatage est celui de la distribution.

le **ramdisk initial** (/boot/initramfs-x.y.z.img).
Constitué au moment de l'installation.
Intègre des modules apportant au noyau (générique) les pilotes nécessaires pour booter sur votre matériel (spécifique).

Notes personnelles

1.6.2- Définir les tailles des partitions

/boot	500 Mo recommandés. **Séparer /boot est obligatoire si « / » est sur LVM**
/ & /usr	4 Go suffisent souvent sur un serveur sans environnement graphique Sur des postes de travail, parfois, 25 Go sont à peine suffisants !

Aujourd'hui, **il n'y a plus de séparation possible de / et /usr**

Autres partitions conseillées

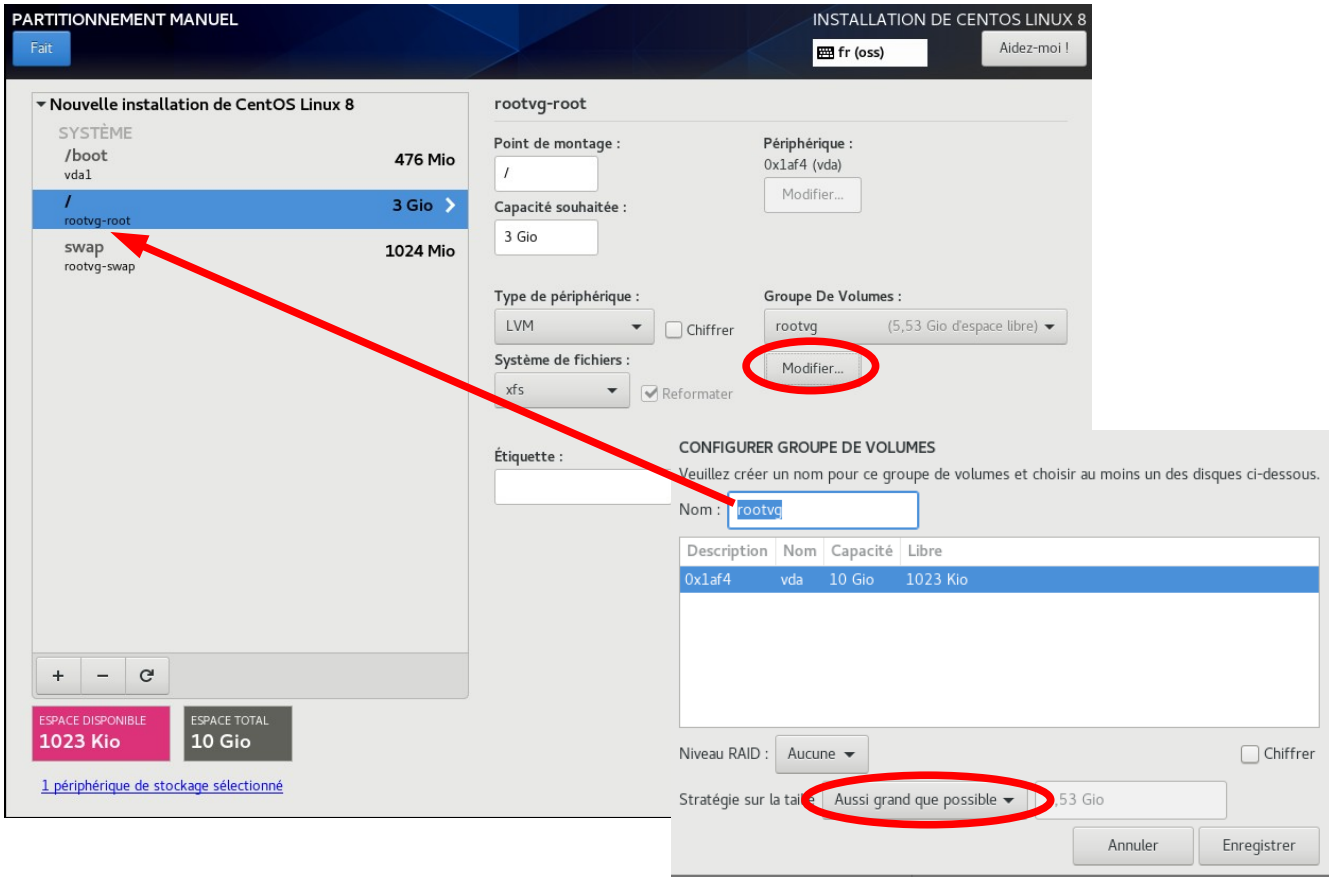
/var	1 Go est souvent un minimum
/var/log	pour éviter saturation de /var par les logs
/var/spool	pour éviter saturation de /var par un spool de "mail", par exemple.
/home & /srv	dépendent de l'utilisation qui va être faite du serveur

Et éventuellement :

/var/lib/docker	Docker y stocke les images, les volumes, …
/usr/src	pour les sources du noyau (prévoir minimum 10 Go en version 5.X)

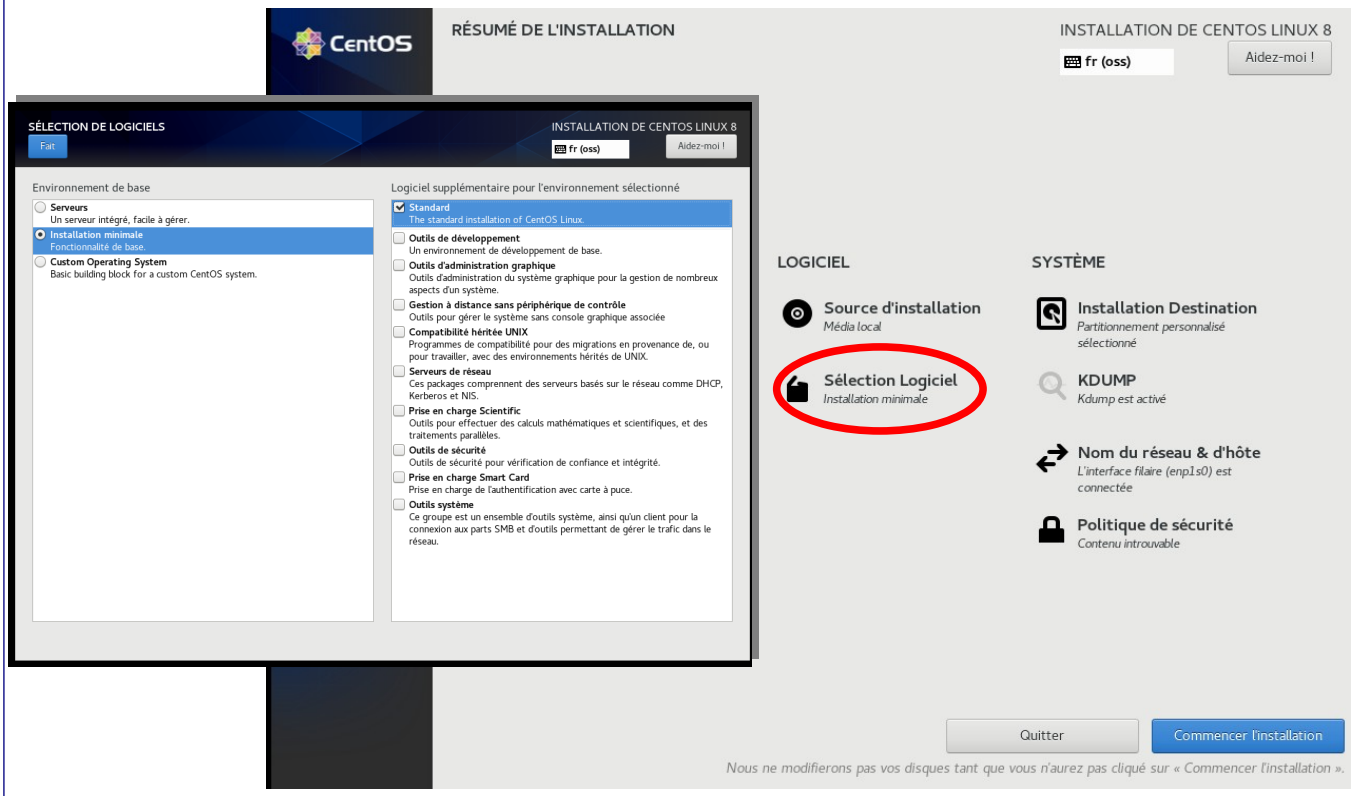
Notes personnelles

Il faut donc au minimum 3 partitions (/boot, /, et swap). Quand c'est terminé : un click sur "**Fait**" !



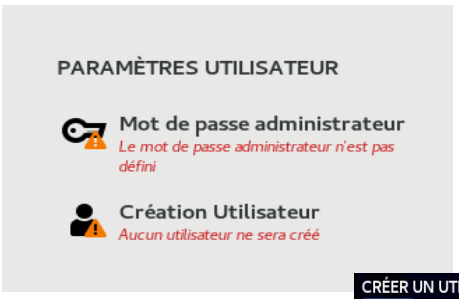
Notes personnelles

1.7- Sélectionner des packages et lancer l'installation



Notes personnelles

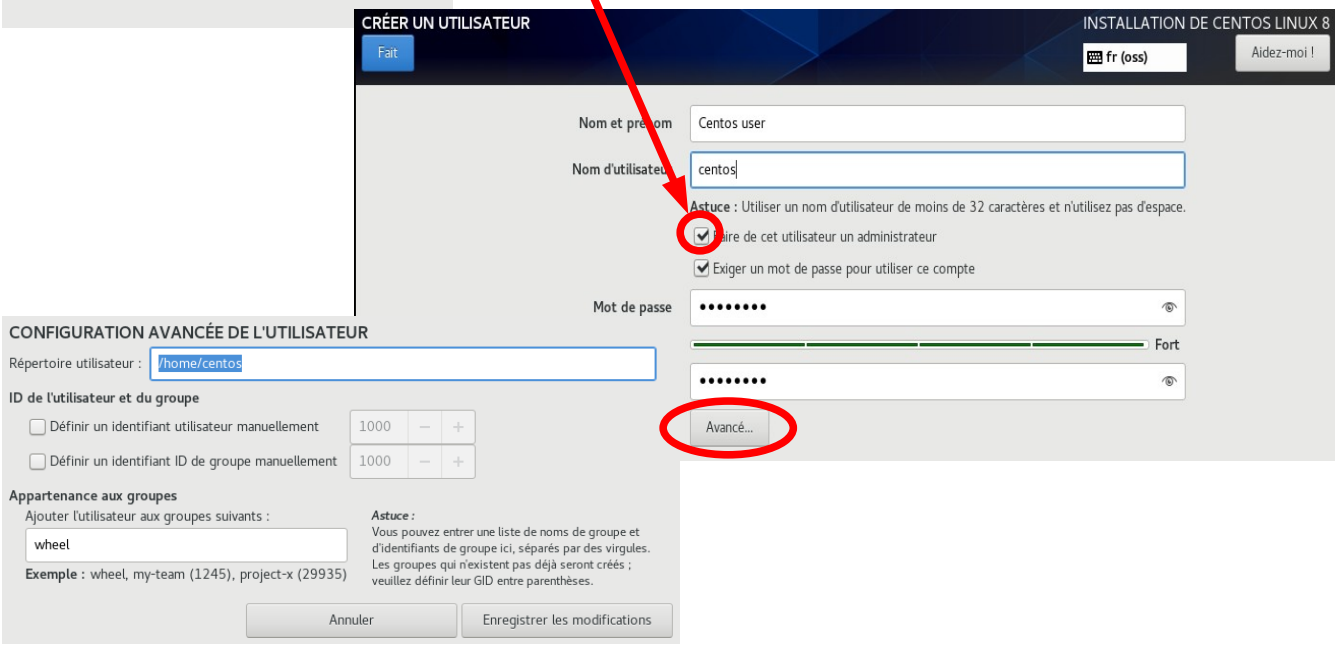
Pendant l'installation...



Deux époques, deux approches :

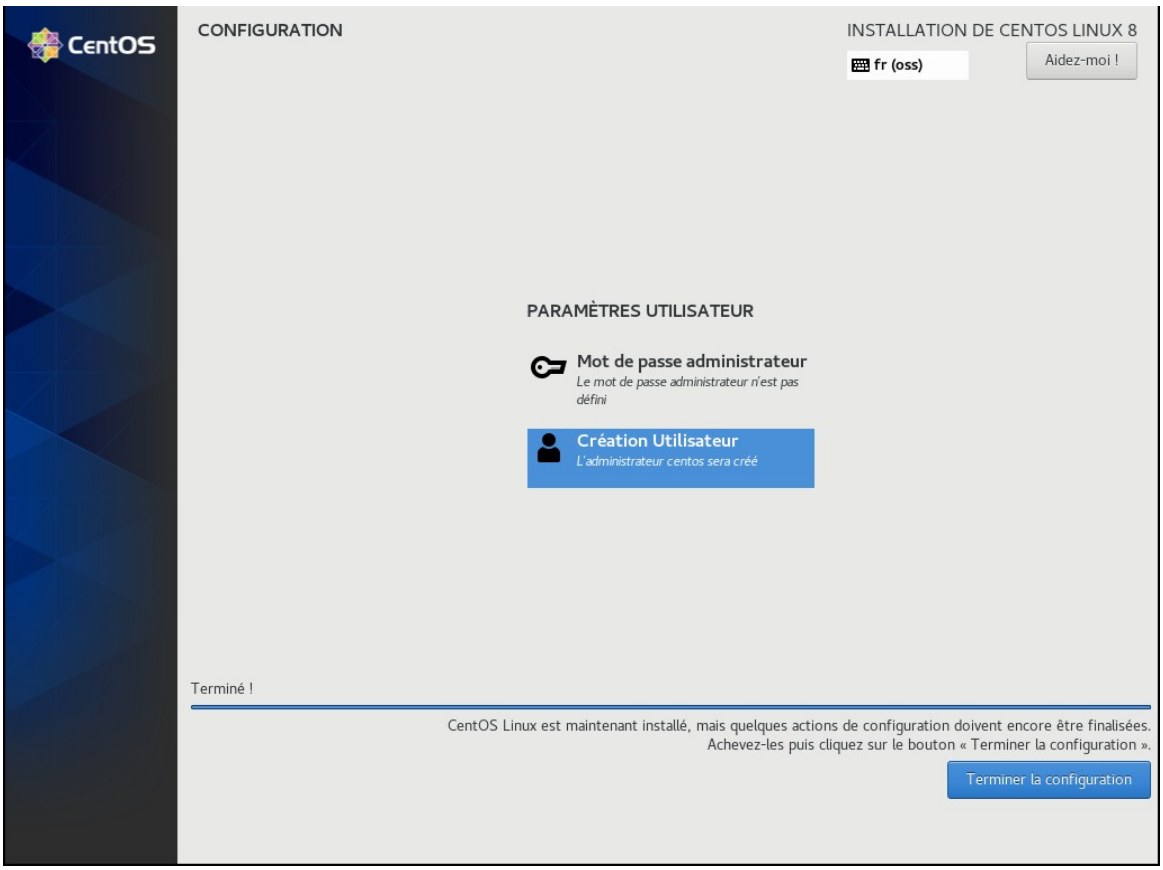
⇒ Auparavant, on devait choisir un mot de passe pour "root".

⇒ Aujourd'hui, pas de mot de passe pour root = connexion root impossible, mais on passe par un utilisateur "administratif", et il est **déclaré dans "sudo"** pour élever ses privilèges



Notes personnelles

Il n'y a plus qu'à finaliser l'installation et redémarrer...



Notes personnelles

1.8- Installation automatique avec kickstart

1.8.1- Rejouer une installation

Le principe consiste à fournir un fichier de directives d'installation.

Le fichier **/root/anaconda-ks.cfg** contient les réponses données lors de l'installation, dans le bon format... à quelques éléments près.

1.8.2- Principe du Kickstart

Anaconda, le système d'installation de RedHat, peut être automatisé grâce au système "**Kickstart**".

Le fichier texte de réponses peut être **sur le média** d'installation **ou** accessible **par une URL**.

Au démarrage de l'installation, on précise où le trouver par l'option **ks=[http://][chemin/]fichier**

Pour commencer :
Réaliser une installation manuelle proche du besoin précis (en mode graphique[4], car le mode texte ne propose pas toutes les possibilités).

Éditer ensuite une copie du fichier "**anaconda-ks.cfg**" pour l'adapter
Voir la Doc RedHat : **Kickstart Installation (RHEL 7),**
chapitre 23.3 **: "Syntax Référence"** (chap. 32 pour RHEL 6)

4 Il faut un minimum de 768 Mo de mémoire avec CentOS 7 pour une installation en mode graphique.

Notes personnelles

Le fichier anaconda-ks.cfg

```
# Kickstart file automatically generated by anaconda.
#version=DEVEL
install
url --url=http://172.17.1.29/CentOS/6/os/i386
lang en_US.UTF-8
keyboard fr-latin9
network --onboot yes --device eth0 --bootproto dhcp --noipv6
rootpw  --iscrypted
$6$D1kmbXgXB.5u8UOp$KN.9FtEcNmx8xo5YrYsElQf6UHqtj2h6Tl0.CI3ZcaxC1c6BlBOdXVYNJ4oF2UGSxb.upudh7wGmoQKY
3hDCY/
firewall --service=ssh
authconfig --enableshadow --passalgo=sha512
selinux --enforcing
timezone --utc Europe/Paris
bootloader --location=mbr --driveorder=sda,sdb --append="crashkernel=auto rhgb quiet"

clearpart --none
raid /boot --fstype=ext4 --level=1 --device=md0 raid.008001 raid.008017
raid pv.009001 --level=1 --device=md1 raid.008002 raid.008018

part raid.008001 --size=200
part raid.008002 --grow --size=200

part raid.008017 --size=200
part raid.008018 --grow --size=200

volgroup vg_root --pesize=4096 pv.009001
logvol / --fstype=ext4 --name=lv_slash --vgname=vg_root --size=3000

repo --name="CentOS"  --baseurl=http://172.17.1.29/CentOS/6/os/i386 --cost=100
```


Notes personnelles

Section "Packages à installer"

On précise les groupes de packages (**@nom**) ou packages (**nom**) à installer ou à exclure (**-nom**).

```
%packages
@core
@server-policy
openssh-clients
-sudo
```

Scripts de pré ou post installation

Des scripts "**%pre**" ou "**%post**" peuvent être prévus.

Ils sont exécutés par "sh", sauf si un interpréteur différent est spécifié.

Attention : "/" est monté dans **/mnt/sysimage** pendant l'installation. Exemple de fichier Kickstart : une installation "minimaliste"

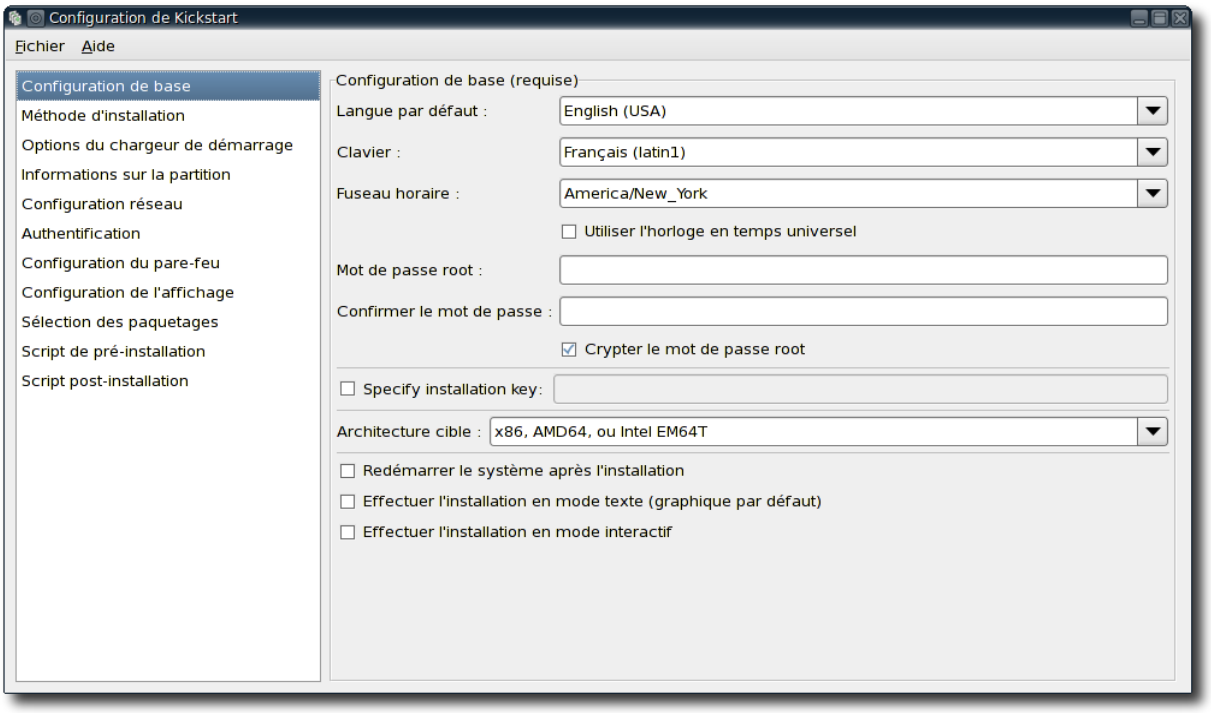
```
%post --nochroot
echo "172.17.1.29   mirrorlist.centos.org mirror.centos.org" >> /mnt/sysimage/etc/hosts
```

Pour éditer ce fichier, on peut utiliser l'outil **system-config-kickstart** (anciennement *ksconfig*).

Notes personnelles

1.8.3- L'application system-config-kickstart

Elle permet de manipuler les fichiers Kickstart. Elle est **fonctionnellement en retard** sur les possibilités offertes par Anaconda/kickstart. Elle ne gère par exemple pas LVM sur CentOS 6.2.



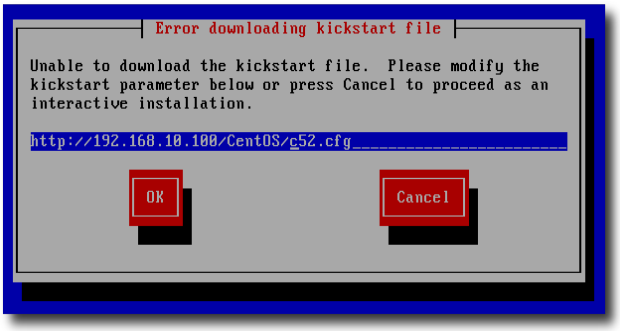
Notes personnelles

1.8.4- Utilisation du kickstart

On peut booter par le réseau (PXE, BOOTP + TFTP[5]) ou localement avec l'image ISO "Minimal" ou bien "NetInstall".

Il faut éditer le premier choix (touche "Tab" au menu), puis ajouter l'option "**ks**[6]**=...**" au démarrage d'une installation

Erreurs possibles :
Problème dans le fichier "kickstart"..



Problème d'URL ou de localisation du fichier...
Dans le cas normal, aucune question ne doit être posée pendant l'installation.

5 http://www.syslinux.org/wiki/index.php/PXELINUX
6 ks=**http**://serveur/chemin/fichier..., ks=**ftp**://serveur/chemin/fichier.... ks=**cdrom**:/fichier

Notes personnelles

1.9- CD/DVD de recovery et clé USB bootable

1.9.1- Les Recovery CD / rescue CD

Il s'agit d'un système autonome sur CD ou DVD : un live CD dont la taille est souvent réduite.

Il doit proposer un panel d'outils nécessaires lors de "disaster recovery" (disque cassé, filesystem crashé, système ne boote plus...) de manière à permettre la réparation.

1.9.1.1- Le CD ou DVD d'installation

Il permet d'amorcer le système en mode "rescue", puis de rechercher les partitions et monter les systèmes de fichiers issus d'une installation préalable.

Il offre un mécanisme "automatisé" pour amorcer un système "en bon état", et permet en principe de "chrooter" sur le système habituel pour y effectuer des opérations de maintenance.

1.9.1.2- La trousse de dépannage : systemRescueCD

systemRescueCD (230 Mo), permet la personnalisation du rescueCD (du noyau, composants).
Il offre un ensemble de logiciels : de l'outil de partitionnement au navigateur web (il propose un environnement graphique Xorg (vesa)), les outils liés aux filesystems, volumes logiques, sauvegarde / restauration / images (partimage), éditeurs, gravure, ...

La documentation est disponible en français (http://www.sysresccd.org/Online-Manual-FR).
Téléchargement : http://www.sysresccd.org/

Notes personnelles

1.9.2- Clonage de machines

On parle de clonage de machines et non pas de système de dépannage.

Ils ne nécessitent à priori pas de pré-installation d'un système pour restaurer une sauvegarde. On appelle ce type de sauvegarde une sauvegarde "**bare-metal**".

MondoRescue, **Mkcdrec, et REAR** sont des systèmes similaires permettant avant tout la sauvegarde / restauration d'images à partir de supports CD/DVD/NFS.

MondoRescue (http://www.mondorescue.org/) :
Permet d'effectuer une sauvegarde système complète sur différents médias. Il gère les CD et DVD+R/RW, bandes, images disques (locales/NFS/PXE). Il prend en charge les systèmes de fichiers suivants : ext2/3/4, reiser, xfs, jfs, vfat, ntfs, nfs, smbfs, cifs, et volumes lvm1/2.

MkcdRec : http://mkcdrec.sourceforge.net/
Permettant de sauvegarder un système GNU/Linux. Il crée un support (CD/DVD) bootable contenant un mini-système, ainsi que les backups et les scripts permettant la restauration.
En cas de problème, on peut booter sur le CD/DVD et restaurer les disques dans leur état au moment du backup, qui peut être stocké sur un autre disque local (ou distant via NFS).
Il gère les systèmes de fichiers ext 2/3/4, minix, xfs, jfs et reiserfs, ainsi que les partitionnements de type LVM et RAID logiciels. Les filesystems sont sauvegardés au format tar compressé.

MkCdRec a été abandonné par ses développeurs au profit de **Relax And Recover** (**REAR**)
http://relax-and-recover.org

Notes personnelles

1.10- Booter autrement : le projet Syslinux

Le projet "**syslinux**" (www.syslinux.org) propose des outils pour booter par d'autres voies que le disque dur, avec une syntaxe de fichier de configuration semblable à celle de LILO.

```
DEFAULT linux

LABEL linux
  KERNEL vmlinuz.img
  APPEND ro root=/dev/sda1 initrd=initrd.img [ks=http://....]
```

syslinux : permet de rendre amorçable une disquette ou un disque dur au format FAT/msdos.
Commande : **syslinux <device>**.

À la racine du support, la commande **syslinux** crée un fichier **ldlinux.sys**.
Au démarrage s'affiche un prompt ("boot: "), proposant les choix définis dans **syslinux.cfg**.

pxelinux : booter par le réseau

isolinux : pour fabriquer une image ISO bootable

extlinux (une alternative à Grub & Lilo) s'installe sur un système de fichiers ext2/3 monté.

memdisk : en conjonction avec syslinux/isolinux, ce module permet parfois de booter lorsque le BIOS ne supporte pas les images type isolinux.

Notes personnelles

1.11- Amorcer par le réseau via PXE

1.11.1- Le serveur PXE

L'amorçage **PXE** s'appuie sur un serveur **DHCP** (BOOTP) et un serveur **TFFP**.

1.11.2- Le serveur DHCP

On configure un serveur DHCP / BOOTP (package **dhcp**) capable d'indiquer au client où se trouve l'exécutable (**pxelinux.0**) à télécharger (range **dynamic-bootp**, options **next-server** et **filename**).

```
subnet 192.168.150.0 netmask 255.255.255.0 {
  range dynamic-bootp 192.168.150.201 192.168.150.250;
  option subnet-mask 255.255.255.0;
  option routers 192.168.150.200;
  option domain-name-servers 192.168.150.200;
  option domain-name "formation";

# Serveur BOOTP, la suite: c'est  tftp://next-server/filename
  next-server 192.168.150.200;
  # Ce fichier est une sorte  de GRUB ou de isolinux (cdrom), mais pour PXE
  # Le chemin est relatif au répertoire dans lequel tftpd est chrooté (/tftpboot)
  filename "/pxelinux.0";
}
```

Ici, le client demandera un fichier "/pxelinux.0" à la racine du serveut TFTP.

⇒ L'emplacement est indiqué par "*filename*".

⇒ C'est le projet "**pxelinux**" (package "syslinux") qui fournit ce fichier.

⇒ Il faut donc un serveur (*next-server*) TFTP proposant le bootstrap.

1.11.3- Le serveur TFTP

C'est un service que l'on peut démarrer grâce à Xinetd.

L'installation du package "tftp" produit la configuration suivante (disable=yes ⇒ no) :

Notes personnelles

```
# yum install tftp
# cat /etc/xinetd.d/tftp
…
service tftp
{
      socket_type           = dgram
      protocol              = udp
      wait                  = yes
      flags                 = IPv6 IPv4
      user                  = root
      server                = /usr/sbin/in.tftpd
      server_args           = -u tftp -s /srv/tftpboot
#     per_source            = 11
#     cps                   = 100 2
      disable               = yes
}
# systemctl restart xinetd
# systemctl enable xinetd
```

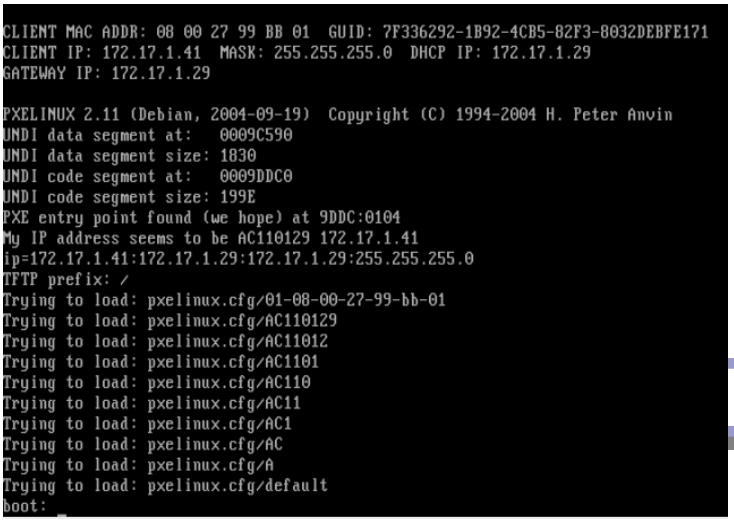
Notes personnelles

1.11.4- Configuration de PXELinux

Elle se fait dans le répertoire "**pxelinux.cfg**"

Il contient des fichiers de configuration dont la syntaxe est la même que celle de **syslinux.cfg**.

Le client (qui exécute **pxelinux.0**) recherche un fichier de configuration selon...



ARP Type 1 (selon son adresse **MAC**),
préfixée par "01-".
Ex : pour 88:99:AA:BB:CC:DD
il cherche **01-88-99-aa-bb-cc-dd**.

son **adresse IP**
(convertie en Hexadécimal par **gethostip**.
(4*2 caractères, en majuscules) :

```
# gethostip 192.168.0.1
192.168.0.1 192.168.0.1 C0A80001
```

Enfin, un fichier "**default**", si aucune des deux possibilités ci-dessus ne correspond.

Notes personnelles

1.12- Créer un CD/ROM bootable personalisé

isolinux : ISO pour ISO9660, c'est le mécanisme de boot des CD/DVD.

On rend le support bootable au moment de la gravure, en utilisant le répertoire **isolinux** et les fichiers qu'il contient dans la commande de gravure.

La commande se joue depuis l'intérieur du répertoire racine de l'arborescence de l'image.

Attention pour RHEL7 / CentOS 7, le volume doit porter le bon nom de volume (-V ...).
Repérer le bon nom sur une image d'origine avec "file".

```
# mkisofs -r -T -J -V 'CentOS 7 x86_64' -o /var/lib/libvirt/images/CentOS-7-auto.iso  -b
isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table .
```

La suite ressemble à syslinux, et le fichier menu s'appelle ici **isolinux.cfg**.
On y documente les choix possibles, dans la même syntaxe que pour syslinux.

Les chemins indiqués dans le fichier **isolinux.cfg** sont relatifs au répertoire **isolinux** qui sera présent sur le support gravé.

Attention : isolinux ne supporte pas les extensions Rock Ridge et Joliet dans les noms de fichiers, mais l'image ISO peut être produite avec ces options pour que le CD/ROM les utilise.

Notes personnelles

2- Maîtriser la configuration logicielle du système

Notes personnelles

2.1- Panorama des outils de gestion de package

2.1.1- Le monde RPM/rpm : yum ⇒ dnf, zypper, urpmi

YUM (Yellowdog Updater, Modified) est utilisée (V ≤ 7) chez **RedHat**, **CentOS & Scientific Linux**.
Depuis la version 22 de **Fedora**, YUM est remplacé par **DNF**.

Zypper est l'équivalent, mais pour **SUSE** & **OpenSUSE**
Urpmi est l'équivalent, mais pour **Mandriva** & **Mageia**

2.1.2- Le monde DEB/dpkg / APT

APT est utilisé sur **Debian** & **Ubuntu** : commandes "apt-get", "apt-cache…" , ou "**apt**"
Aptitude est un approche similaire sur Debian & Ubuntu : commande "aptitude"

2.1.3- Les autres mondes

Slackware et ses dérivées ont leurs familles d'outils (slackpkg, slapt-get, netpkg...).

Arch Linux : pacman, **Alpine** Linux : apk, **Sabayon** : equo, **Gentoo** Linux : emerge…

Compléments : voir **Package Management Cheatsheet**[7] sur **Distrowatch**[8].

7 http://distrowatch.com/dwres.php?resource=package-management
8 http://distrowatch.com/

Notes personnelles

2.2- Gestion des paquetages RPM

2.2.1- Structure d'un package RPM

RPM un système mis au point par RedHat, utilisé pour **l'installation** et la **désinstallation** de logiciels.

Il s'appuie sur un format de fichier spécifique contenant :

Un "**lead**" en-tête de 96 octets indiquant :
- le format,
- le nom et la version du package (sur 66 octets),
- le type (binaire / source), l'os cible,
- la version de RPM utilisée...

une structure de "**signatures**"
- taille du package, taille de son payload, digest SHA1 du header,
- digest md5... visibles par "**rpm --checksig --verbose**"

Une structure de "**header**"
- contient tous les **RPMTAGS**, c'est à dire les informations disponibles par "**rpm -qi**"
  - Les informations sur le paquetage (dépendances, provenance, versions, …) (--requires, --provides, ...)
  - Des scripts de pré ou post pour l'installation ou la désinstallation (--scripts)

Vient ensuite le contenu du paquetage : "**payload**"
- L'archive, au format cpio gzippé (-ql, -qc, -qd)

Notes personnelles

2.2.2- La commande rpm

Les actions se font par la commande **rpm** (installation, désinstallation, consultation de paquetages...).

La commande **rpm** est aussi une interface à d'autres commandes.
Exemple : **rpmquery** est synonyme de "**rpm -q**".

```
# rpm -qlf $(which rpm) | grep "/s*bin/"
/bin/rpm
/usr/bin/rpm2cpio
/usr/bin/rpmdb
/usr/bin/rpmquery
/usr/bin/rpmsign
/usr/bin/rpmverify
```

2.2.3- Installer, mettre à jour, désinstaller

Les 3 actions principales sont l'installation, la mise à jour, et la désinstallation de paquetages :

```
rpm -i nom_package-version.arch.rpm
rpm -U nom_package-version-N+1.arch.rpm
rpm -e nom_package
```

Un mode verbeux (**-v**) est possible ainsi qu'un affichage des barres de progression (**-h**).

Notes personnelles

2.2.4- Les requêtes : rpm -q ou rpmquery

Liste des paquetages installés

```
rpm -qa   /   rpmquery -a
```

Un paquetage est-il installé ?

```
rpm -q nom_package
```

Lister les infomations (tags) (**-i**) d'un package

À partir d'un fichier RPM

```
rpm -qpi nom_package-version.arch.rpm
```

Pour un package installé

```
rpm -qi nom-package
```

Lister le contenu (**-l**) (payload) d'un paquetage

À partir d'un fichier RPM

```
rpm -qpl nom_package-version.arch.rpm
```

Paquetage installé

```
rpm -ql nom-package
```

Variantes de "**-l**" :

"**-c**" n'affiche que les fichiers de configuration,
"**-d**" que ceux de documentation.

Option "**--dump**" affiche des détails pour chaque fichier de l'archive (nécessite -l, -c, ou -d).

Quel paquetage (rpm) a installé un fichier dans l'arborescence (**-f**)

```
rpm -qf chemin_complet_fichier
```


Notes personnelles

Lister les dépendances d'un package :

--requires : dépendances nécessaires pour le package
--provides : dépendances satisfaites (capacités) du package

Scripts de {pré,post}-{installation/désinstallation} :

--scripts

Exemple :

```
# rpm -q --scripts xinetd
postinstall scriptlet (using /bin/sh):
if [ $1 = 1 ]; then
    /sbin/chkconfig --add xinetd
fi
preuninstall scriptlet (using /bin/sh):
if [ $1 = 0 ]; then
    /sbin/service xinetd stop > /dev/null 2>&1
    /sbin/chkconfig --del xinetd
fi
postuninstall scriptlet (using /bin/sh):
if [ $1 -ge 1 ]; then
    /sbin/service xinetd condrestart >/dev/null 2>&1
fi
```


Notes personnelles

2.3- De yum à dnf

YUM (Yellowdog Updater Modified) est une surcouche à RPM qui date de 2003.

Yum est lent, gourmand en RAM, et son code (python 2) est devenu peu maintenable.

Avec RHEL 8 / CentOS 8, "**DNF**" (Dandified Yum) lui succède.

La commande "**dnf**" remplace "**yum**", et pour l'utilisation standard, l'interface reste la même.

```
[root@c8s-01 ~]# ls -l /bin/yum
lrwxrwxrwx. 1 root root 5  4 août  20:51 /bin/yum -> dnf-3
```

YUM ou DNF s'appuient sur des dépôts de packages (les mêmes) où sont catalogués :
- les descriptions de chaque package
- les dépendances de chaque package
- les fichiers fournis par chaque package

On a intérêt à utiliser "**dnf**" (ou **yum**) plutôt que "**rpm**" car il **résout** les dépendances.

Notes personnelles

2.4- Utilisation de yum / dnf

2.4.1- Lister les dépôts

dnf repolist : lister les dépôts configurés et actifs (enabled!= 0)
dnf repolist all : lister tous les dépôts configurés
dnf repoinfo : s'informer sur un dépôt

2.4.2- Chercher des packages

dnf list : recherche par nom, listage des packages disponibles, installés...
dnf search : recherche par un mot de la description du package
dnf provides : recherche par un fichier qu'on fourni par un package
dnf deplist un-package : lister les dépendances d'un package
dnf info : un package : s'informer sur un package

2.4.3- Installation et mise à jour de logiciels

dnf install package1 [package2...] : installe ou met à jour un package (ou des packages)
dnf update [package1...n] : met à jour les packages installés si des mises à jour sont disponibles
dnf upgrade : mise à jour totale avec suppression des packages rendus obsolètes par d'autres.

2.4.4- Supprimer un package

dnf remove package1 [package2..] : désinstalle un package, et ses dépendances

Notes personnelles

2.5- Rechercher des packages

2.5.1- Listage et recherche de package : dnf list

avec éventuellement :

all (par défaut),
updates : mises à jour disponibles (par rapport aux packages installés)
installed : n'affiche que les packages installés
available : tout sauf ce qui est installé
extras : (ou **dnf list --extras**) es RPMS installés mais non disponibles dans les dépôts,
obsoletes : packages installés mais rendus obsolètes par d'autres,
recent : packages ajoutés récemment aux repositories.
Par défaut, c'est 7 jours. (voir directive recent= dans yum.conf)

dnf list tient compte de

"--disablerepo" : pour exclude un repository de la recherche

et

"--showduplicates" : pour demander l'affichage de toutes les versions disponibles du package.

Notes personnelles

2.5.2- Rechercher par les descriptions des packages : *dnf* search

La recherche s'effectue dans les champs "name", "Summary", et "Description" des paquetages

Nécessite les fichiers meta-data "primary.xml.gz"

```
[root@c8s-01 cache]# dnf search curl
Dernière vérification de l'expiration des métadonnées effectuée il y a 0:57:20 le jeu. 03 déc. 2020
17:55:28 CET.
============== Nom correspond exactement à : curl ==================================================
curl.x86_64 : A utility for getting files from remote servers (FTP, HTTP, and others)
============== Nom & Résumé correspond à : curl ====================================================
collectd-curl.x86_64 : Curl plugin for collectd
collectd-curl_json.x86_64 : Curl JSON plugin for collectd
collectd-curl_xml.x86_64 : Curl XML plugin for collectd
libcurl-devel.i686 : Files needed for building applications with libcurl
libcurl-devel.x86_64 : Files needed for building applications with libcurl
libcurl-minimal.i686 : Conservatively configured build of libcurl for minimal installations
libcurl-minimal.x86_64 : Conservatively configured build of libcurl for minimal installations
nbdkit-curl-plugin.x86_64 : HTTP/FTP (cURL) plugin for nbdkit
perl-WWW-Curl.x86_64 : Perl extension interface for libcurl
python3-pycurl.x86_64 : Python interface to libcurl for Python 3
qemu-kvm-block-curl.x86_64 : QEMU CURL block driver
============== Nom correspond à : curl =============================================================
libcurl.x86_64 : A library for getting files from web servers
libcurl.i686 : A library for getting files from web servers
============== Résumé correspond à : curl ==========================================================
rubygem-curb.x86_64 : Ruby libcurl bindings
```


Notes personnelles

2.5.3- Rechercher par un un nom de fichier : *dnf* provides

(ou whatprovides)

```
[root@c8s-01 ~]# dnf --quiet  provides "*/lspci"
pciutils-3.6.4-2.el8.x86_64 : PCI bus related utilities
Dépôt               : baseos
Correspondances trouvées dans  :
Nom de fichier : /sbin/lspci
```

Permet de trouver quel paquetage peut fournir un fichier dont on connaît le nom
(mais pas forcément l'emplacement futur dans l'arborescence).

Nécessite les fichiers meta-datas "filelists.xml.gz"

S'apparente à "rpm -qf", mais avant d'installer le package !

Exemples :

```
# yum provides /usr/bin/gcc
# yum provides *man5/passwd*
```


Notes personnelles

2.5.4- S'informer sur les packages : dnf info [package]

Liste les informations concernant le package, similaire à rpm -qi.

```
[root@c8s-01 ~]# dnf --quiet info rsyslog
Paquets disponibles
Nom          : rsyslog
Version      : 8.1911.0
Publication  : 6.el8
Architecture : x86_64
Taille       : 731 k
Source       : rsyslog-8.1911.0-6.el8.src.rpm
Dépôt        : appstream
Résumé       : Enhanced system logging and kernel message trapping daemon
URL          : http://www.rsyslog.com/
Licence      : (GPLv3+ and ASL 2.0)
Description  : Rsyslog is an enhanced, multi-threaded syslog daemon. It supports MySQL,
             : syslog/TCP, RFC 3195, permitted sender lists, filtering on any message part,
             : and fine grain output format control. It is compatible with stock sysklogd
             : and can be used as a drop-in replacement. Rsyslog is simple to set up, with
             : advanced features suitable for enterprise-class, encryption-protected syslog
             : relay chains.
```

Fonctionne sur des packages installés ou pas...

```
[root@c8s-01 ~]# rpm -q rsyslog
le paquet rsyslog n'est pas installé[9]
```

9 Tiens… ***rsyslog*** n'est pas installé en standard dans CentOS 8 "minimal" ?

Notes personnelles

2.6- Mettre à jour

2.6.1- Lister les mises à jour disponibles disponibles

dnf list updates : liste toutes les mises à jour disponibles.

2.6.2- S'informer avant de mettre à jour

dnf check-update : permet de vérifier si des mises à jour sont disponibles.

Code retour de 100 le cas échéant, 0 dans le cas contraire et 1 si une erreur survient.

Retourne aussi la liste des packages candidats à la mise à jour.

2.6.3- Mettre à jour tout le système

dnf -y upgrade : attention aux packages devenus "obsolètes"

dnf -y update : la version prudente.

Notes personnelles

2.7- Traitement par groupes

Les packages sont classés par catégories ou groupes, qu'il est possible de manipuler d'un bloc.

Chaque groupe contient :
- des packages obligatoires,
- des packages "par défaut",
- des packages optionnels.

dnf grouplist : permet de lister tous les groupes existants, ceux installés, puis ceux disponibles,
un groupe est marqué installé si tous ses packages obligatoires sont présents,
ou s'il n'en a pas, qu'un des packages par défaut ou optionnels est présent ;

dnf groupinfo : liste les packages constituant un groupe ;

dnf groupinstall : installe tous les packages obligatoires d'un groupe (pas les packages optionnels) ;

dnf groupupdate : met à jour tout un groupe (et les dépendances) ;

dnf groupremove : supprime tout un groupe (y compris les packages optionnels du groupe).

Notes personnelles

2.8- Manipuler le cache

DNF travaille avec un cache local où il stocke les meta-datas des différents dépôts de manière à ne pas les télécharger systématiquement.

On peut jouer sur la durée de vie en cache de ces informations (directive **metadata_expire** du fichier de configuration), par défaut de 1h30.

2.8.1- Nettoyage du cache : dnf clean

On peut préciser :

all : nettoie (presque[10]) tout le contenu de /var/cache/dnf/xxx.
Recrée vides les répertoires correspondant aux dépôts
On peut aussi supprimer & recréer le répertoire /var/cache/dnf...

packages : supprime les packages téléchargés
(en fonction du paramètre **keepcache**, ils peuvent être conservés après installation)

metadata : supprime toutes les meta-datas

expire-cache : force une expiration du cache
dbcache : supprime le cache local au format sqlite (ou xml)
(qui est utilisé pour améliorer les temps d'accès aux meta-datas)

10 uniquement sur les dépôts activés. On peut utiliser **dnf --enablerepo='*' clean all**

Notes personnelles

2.8.2- Construire le cache : dnf makecache

Force un téléchargement de toutes les méta-datas des dépôts actifs.

Il peut (a pu) arriver ceci :

```
Not using downloaded repomd.xml because it is older than what we have:
  Current   : Mon Aug 24 21:43:18 2009
  Downloaded: Tue Aug 18 21:29:35 2009
```

Il s'agit là d'un système utilisant un miroir qui n'est pas à jour alors qu'il a déjà utilisé un miroir "à jour".

Cette histoire est arrivée avant DNF, à l'époque de Yum.

Dans ce cas :

on nettoie le cache et on le reconstruit
on s'assure que le miroir utilisé est à jour.

Si on n'utilise pas "**dnf makecache**", les méta-data sont téléchargées au besoin en fonction des requêtes.

Voir le paramètre **metadata_expire** pour régler la fréquence de vérification du cache par **dnf**.

Notes personnelles

2.9- Configuration de DNF

2.9.1- Fichiers de configuration

DNF, comme **YUM**, utilise un fichier de configuration principal… D'ailleurs, c'est le même :

```
[root@c8s-01 ~]# ls -trl /etc/yum.conf
lrwxrwxrwx. 1 root root 12  4 août  20:51 /etc/yum.conf -> dnf/dnf.conf
```

Ce fichier **/etc/dnf/dnf.conf** (ou **/etc/yum.conf**) contient des directives dans un format ".ini".

Elles sont groupées dans des sections identifiées par un nom entre crochets

```
# Commentaire
[section]
directive=valeur1
…
[section2]
directive=valeur2
…
```

Les **lignes vides sont ignorées**, comme celles commençant par le caractère #.

Les dépôts et plugins sont déclarés dans des fichiers séparés. En résumé :

la configuration principale : /etc/yum.conf ou /etc/dnf/dnf.conf
les plugins : /etc/yum/pluginconf.d
les dépôts : /etc/yum.repos.d

Notes personnelles

2.9.2- Quelques paramètres

Gestion du cache : **cachedir** et **keepcache** :

définit l'emplacement du cache

si keepcache = 1 demande à conserver les RPM téléchargés même après installation réussie.

Signatures GPG : **gpgcheck**=true : demande à valider les signatures des packages.

On peut l'inhiber avec l'option --nogpgcheck.

Utiliser les plugins : **plugins** = true

On peut désactiver chaque plugin un à un dans son propre fichier de configuration.

Passage par un proxy : Exemple avec un proxy qui nécessite une authentification :

Peut être global ou spécifique pour un dépot

Si un proxy global est activé, on peut le désactiver pour un dépôt donné (**proxy=_none_**)

http_caching (all, packages, none) indique au proxy-cache quel comportement adopter.

```
proxy=http://localhost:3128/
proxy_username=usera
proxy_password=passa
http_caching=all
```

Les paramètres de la section **[main]** sont tous documentés dans la page yum.conf (5).

Notes personnelles

2.9.3- Déclaration d'un dépôt de packages

Tout fichier correspondant à **/etc/yum.repos.d/*.repo** est pris en compte en plus de yum.conf.

Définition d'un dépôt : le nom est cité entre les crochets, les propriétés ensuite.

```
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

Name : (optionnel) : un libellé utiliser des variables (**name**=CentOS-**$releasever** - Base).

Baseurl / **Mirrorlist** : deux méthodes pour spécifier l'emplacement du dépôt :
baseurl=URL, qui pointe vers un emplacement disposant de la bonne structure ;
mirrorlist=URL, qui donne en réponse une liste d'URLS acceptables par **baseurl** ;
Yum choisit alors la première et passe à la suivante en cas de dysfonctionnement.

GPG (optionnel) : Vérifier la signature des packages (**gpgcheck**=0/1) et avec la clé **gpgkey**.
On peut demander à ne pas faire ce contrôle (**--nogpgcheck**).

On peut activer ou pas un dépôt (**enabled=0 / 1**).
À l'utilisation, on peut demander à ignorer un dépôt (**--disablerepo**=nom / *),
ou de prendre un compte un dépôt désactivé dans la configuration (**--enablerepo**=nom / *).

Notes personnelles

2.10- Compilation de logiciels par les sources

2.10.1- Failles de sécurité : où s'informer ?

Dès que possible, des alertes de sécurité sont publiées sur les sites officiels des logiciels et des listes de distributions existent, par exemple :

https://www.redhat.com/mailman/listinfo/rhsa-announce
https://lists.debian.org/debian-security-announce/

D'autres sources existent et centralisent :

http://cve.mitre.org/, http://packetstormsecurity.org/,
http://www.exploit-db.com/, http://securityvulns.com/
…

2.10.2- Maintenance des logiciels d'un serveur

Deux façon de travailler :

Compiler soi-même les sources
Installer des **paquetages binaires**

RPM (RedHat Package Manager) est utilisé pour l'installation, la mise à jour, la désinstallation, l'inventaire… par de nombreuses distributions. Il disponible sur d'autres plate-formes (AIX par exemple).

Debian & Ubuntu utilisent le système **dpkg** : mêmes objectifs et mêmes fonctions que "**rpm**".

Notes personnelles

2.10.3- Compiler soi-même les logiciels

Profiter des optimisations propres à l'architecture qui compile
Gain de performances...

Décider d'un certain nombre de paramètres
emplacement des répertoires et fichiers,
intégration de modules fonctionnels,
versions des bibliothèques à utiliser.

Plus grande réactivité dans le choix de mettre dès qu'une évolution ou un correctif est disponible,

Choisir soi-même l'emplacement des différents fichiers
Dans une arborescence se situant par défaut dans **/usr/local**,
Permet de gérer facilement la désinstallation et de la cohabitation de plusieurs versions.

Distribuer localement ses propres packages, incluant uniquement les fonctionnalités nécessaires

Notes personnelles

2.10.4- Étapes d'une compilation à partir des sources

Téléchargement de l'archive source
C'est souvent une archive **tar** compressée par **gzip**, **bzip2**, ou désormais **xz**.

Extraction de l'archive (**tar**) : **gzip** (ou **bzip2**, ou **xz**) -dc archive | **tar x**
On fait en principe une petite vérification au préalable : gzip -dc archive | tar **t** | head

Positionnement dans le répertoire des sources (**cd**)
Il contient en principe un fichier "README", souvent un fichier "INSTALL"...
et la plupart du temps un fichier "**configure**", c'est un script.

Création des Makefile : **./configure** (ou équivalent)
On commence par tester son option "--help", elle informe sur les autres
On spécifie en principe **un préfixe d'installation** : --prefix=/chemin/logiciel
éventuellement (souvent) : le répertoire de configuration : --sysconfdir=/etc/logiciel
En fin de "**configure**", un script "**config.nice**" est parfois produit pour mémoire.

Compilation (**make**)
C'est la phase de compilation, elle construit les exécutables et bibliothèques.

Tests, pas toujours proposés (**make test**, **make tests, make check**)
PHP, Zlib, OpenSSL, ...proposent avant l'installation un certain nombre de tests.

Notes personnelles

2.10.5- Étapes d'une installation par les sources

Installation (**make install**)
Cette cible du makefile principal va installer les produits dans le répertoire "préfixe"

Finalisation de l'installation… c'est manuel le plus souvent !

Référencement des pages de man,
⇒ /etc/**man_db.conf** (anciennement /etc/man.config)

PATH pour les exécutables "planqués"...
Ajout des répertoires contenant des binaires dans le PATH :
⇒ **/etc/security/pam_env.conf** (si PAM_ENV est utilisé)
⇒ **/etc/environment, /etc/profile**, "**.bash_profile**"...

Référencement des bibliothèques,
⇒ **/etc/ld.so.conf**, ldconfig...

Mise en place de scripts ou unités de démarrage
Parfois un script est fourni, sinon...

Notes personnelles

2.11- Make et les Makefile

Les Makefiles décrivent la manière de le compiler un programme ou une librairie.
Ils informent sur les dépendances entre les différents fichiers sources,
Ils énoncent les différentes étapes nécessaires à la réalisation d'une cible.

2.11.1- Intérêt des Makefile

Ne recompiler que le strict nécessaire

Grâce au Makefile, on ne recompile que ce qui est nécessaire.
L'intérêt principal est la facilité et rapidité de construction et de reconstruction des cibles.

Recompiler tout ce qui est nécessaire

Permet de forcer une compilation de certaines cibles en fonction de dépendances modifiées.

Par exemple : Si un ".h" est concerné par plusieurs ".c",
alors la modification du ".h" doit engendrer la recompilation de tous les ".c".

Mémoriser des options et commandes, de compilateurs

On peut définir des variables dans les Makefile,
Elles permettent de mémoriser des options de compilation, et d'en changer facilement pour des tests.

Générer des makefiles

L'étape "configure" a souvent pour but de produire les Makefiles, en partant d'un template, et en s'appuyant sur différents tests permettant de les adapter à la machine ou au système.

Notes personnelles

2.11.2- Make et les Makefile

Consulté par la commande "**make**" s'il s'appelle "makefile" (1), ou "Makefile" (2) et est trouvé dans le répertoire courant. Il peut être spécifié par l'option "**-f**" de "**make**" dans le cas contraire.

S'il n'est ni trouvé ni précisé : don't know how to make...

```
$ make
make: *** Pas de cibles spécifiées et aucun makefile n'a été trouvé. Arrêt.
```

Un makefile vide : $ >makefile

```
$ make
make: *** Pas de cibles. Arrêt.
```

Un makefile minimal : 2 labels (cibles) commençant en début de ligne et suivis par le caractère "**:**".

```
$ echo -e "toto:\nall:\n" > makefile
$ make all
make: Rien à faire pour « all ».
```

Si "**make**" ne précise pas la cible à atteindre, c'est la première trouvée qui est choisie.

```
$ make
make: Rien à faire pour « toto ».
```


Notes personnelles

2.11.3- Format d'un makefile

C'est un fichier texte composé d'un ou plusieurs blocs définissant des cibles (des labels), leurs dépendances et la manière d'atteindre les cible (des commandes)

```
cible: dépendance
        commandes
```

"**dépendance**" est une de pré-requis pour atteindre la cible :

```
all: $(EXEC)
```

Pour exprimer les dépendances, on peut utiliser
- des variables
- d'autres noms de cibles
- des noms de fichiers

"**commandes**" est une suite de commandes :
- Il y aura affichage de la commande avant son exécution
- On peut les faire précéder par le signe "@" (la commande n'est pas affichée, juste exécutée)

Variables :
- on les définit selon la forme : **VAR=valeur**
- on les utilises selon la forme : **$(VAR)**

Un petit tutoriel : http://gl.developpez.com/tutoriel/outil/makefile/

Notes personnelles

2.12- Référencer les pages de manuel

Habituellement stockées dans **/usr/share/man**, elles sont souvent oubliées lors de l'installation d'un produit par les sources.

2.12.1- Lire une page de manuel non "connue"

On peut utiliser un ou plusieurs "/" dans le nom des pages :

Cela permet d'indiquer le répertoire dans lequel se trouve la page.

On évite ainsi à la commande "man" de chercher la page.

Dès qu'il y a un "/" dans le nom, l'argument est alors un chemin d'accès à un fichier de man, et non plus le nom d'une commande, d'un fichier de configuration, d'un concept...

```
# man /opt/apache2/man/man8/ab.8
```

Une autre méthode consiste à utiliser l'option "-M"... mais tout ceci n'est pas très pratique.

```
# man -M /opt/apache2/man:/opt/apache/man ab
```


Notes personnelles

2.12.2- Recherche de page de manuel

Lorsqu'il n'y a pas de "/" dans le nom des pages, alors une recherche (un peu complexe) a lieu dans plusieurs répertoires dans les cas où :

l'option "-M" est fournie

la variable d'environnement MANPATH est définie

Dans les autres cas, **man** cherche en fonction de son fichier de configuration : **/etc/man_db.conf**.

2.12.3- Le fichier /etc/man_db.conf

Il contient entre-autre :

des directives **MANPATH**, ajoutant des répertoire de recherche de pages
les jokers sont supportés.

```
MANPATH        /opt/*/man
```

des directives **MANPATH_MAP**,
associant les répertoires des commandes à ceux contenant leurs pages de manuel.

```
MANPATH_MAP     /usr/local/bin          /usr/local/share/man
```


Notes personnelles

2.13- Exécutables et librairies

2.13.1- À propos des librairies, de ld... et de ELF

Tout exécutable Linux peut faire appel à des "librairies" (**bibliothèques**) :

"**en statique**" (si l'option "**-static**" a été donnée à "**ld**" lors de l'édition de liens).
Le code des librairies est alors inclus dans le fichier exécutable statique.

"**en dynamique**", dans le cas contraire, qui est le plus fréquent.
Le fichier exécutable est alors plus petit,
Il faut pouvoir retrouver les librairies lors de l'exécution du programme,
On parle de ***librairies partagées dynamiques*** : elles concernent plusieurs programmes

Comme le shell pour un exécutable, le système ne cherche pas les librairies dans le répertoire courant.

C'est le programme **ld.so** (devenu **ld-linux.so***) qui trouve et charge les librairies nécessaires pour un programme, prépare son démarrage, et le lance.

Le programme **ld.so** traite les binaires au format **a.out** (utilisés dans le passé).

Le programme **ld-linux.so*** concerne ceux au format **ELF** :
/lib/ld-linux.so.1 utilisé dans l'ancienne version de la **libc** (glibc1 = libc5)
/lib/ld-linux.so.2 (ou **ld-linux-x86-64.so.2**), dans les versions actuelles utilisant **glibc2**.

Notes personnelles

2.13.2- Recherche des librairies

Le chargeur **ld-linux-x86-64.so.2** suit l'ordre **a,b,d,e** ou **b,c,d,e** et s'arrête dès qu'il trouve :

a) DT_RPATH, ignoré si DT_RUNPATH est spécifié
b) LD_LIBRARY_PATH
c) DT_RUNPATH,
d) /etc/ld.so.cache
e) /lib (ou /lib64) puis /usr/lib (ou /usr/lib64)

2.13.2.1- Attributs DT_RPATH & DT_RUNPATH

Les exécutables ELF peuvent embarquer un "PATH librairies dynamiques", dans des attributs présentés par "**-rpath**" à l'édition de lien par la commande "**ld**" : **DT_RPATH** ou **DT_RUNPATH**[11].

2.13.2.2- Variable LD_LIBRARY_PATH

La variable d'environnement **LD_LIBRARY_PATH** est prise en compte sauf si l'exécutable est un binaire ***Set-UID*** ou ***Set-GID***, auquel cas elle est ignorée.

2.13.2.3- Le cache : ld.so.cache

Ce fichier de cache **/etc/ld.so.cache** contient une liste compilée de bibliothèques précédemment trouvées dans des chemins. Si le binaire a été lié avec l'option **-z nodeflib** de **ld**, cette étape est sautée.

2.13.2.4- Répertoires /lib(ou /lib64), puis /usr/lib (ou /usr/lib64)

Si le binaire a été lié avec l'option **-z nodeflib** de l'éditeur de lien, cette étape est sautée.

11 **DT_RPATH** est obsolète et est inhibé par la présence de **DT_RUNPATH**.

Notes personnelles

2.13.3- Les outils concernant les librairies

Les commandes **ldd**, et **ldconfig**, sont étroitement liées à ce procédé.

2.13.3.1- La commande "ldd"

La commande **ldd** permet de lister les bibliothèques qui seront utilisées par un exécutable :

```
[root@centos8-nue ~]# ldd /bin/su
        linux-vdso.so.1 (0x00007ffe321a5000)
        libpam.so.0 => /lib64/libpam.so.0 (0x00007fa582d83000)
        libpam_misc.so.0 => /lib64/libpam_misc.so.0 (0x00007fa582b7f000)
        libutil.so.1 => /lib64/libutil.so.1 (0x00007fa58297b000)
        libc.so.6 => /lib64/libc.so.6 (0x00007fa5825b9000)
        libaudit.so.1 => /lib64/libaudit.so.1 (0x00007fa58238f000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007fa58218b000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fa58319f000)
        libcap-ng.so.0 => /lib64/libcap-ng.so.0 (0x00007fa581f85000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00007fa581d65000)
```


Notes personnelles

2.13.3.2- L'utilitaire "ldconfig"

L'outil **ldconfig** sert à configurer l'éditeur de liaisons dynamiques.

Il permet d'interroger ou construire le cache "**ld.so.cache**",

Il utilise le fichier et **/etc/ld.so.conf** et le répertoire **/etc/ld.so.conf.d** pour construire un "**cache**" (que le chargeur "ld-linux.so" utilise).

Le fichier **/etc/ld.so.conf** :

Il contient une liste de répertoires où "ld.so" ou "ld-linux.so" peuvent chercher les librairies.

Le séparateur est ":", ou un espace, une tabulation, un saut de ligne ou même une virgule.

Il est possible d'inclure d'autres fichiers, grâce au mot clé "**include**".
Permet de manipuler facilement la liste de répertoires contenant les librairies.

Contenu de /etc/ld.so.conf :

```
include ld.so.conf.d/*.conf
```

À chaque modification de **/etc/ld.so.conf** ou d'un des fichiers qu'il pointe, on ré-exécute **ldconfig**.

Notes personnelles

2.13.4- Compléments sur "ld"

2.13.4.1- Forcer le préchargement d'une librairie

LD prévoit deux solutions : man ld-linux, section "ENVIRONNEMENT" et section "FILES"

Avec la variable **LD_PRELOAD**[12]
Avec le fichier **/etc/ld.so.preload**[13]

2.13.4.2- La variable "LD_TRACE_LOADED_OBJECTS"

Si elle est positionnée, alors l'exécutable liste ses dépendances dynamiques (et leur résolution) au lieu de s'exécuter normalement.

```
# export LD_TRACE_LOADED_OBJECTS=yes
# whoami
        linux-vdso.so.1 =>  (0x00007fff8efff000)
        libc.so.6 => /lib64/libc.so.6 (0x00007fa39f2f3000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fa39f686000)
# unset LD_TRACE_LOADED_OBJECTS
# whoami
root
```

"linux-vdso.so.1" (anciennement "linux-gate.so.1") est une librairie virtuelle, donc non résolue.

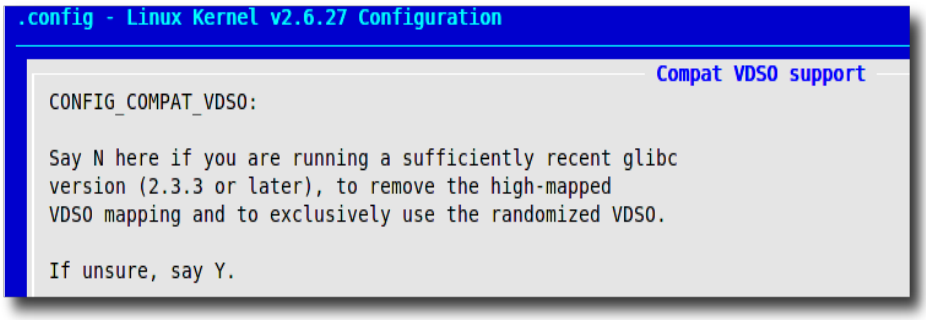
12 Liste (séparateur espace) des librairies (chemin complet) qui seront forcément (pré)chargées (même si non nécessaires)
13 L'intérêt du fichier est qu'il est non modifiable par les utilisateurs... et doit être lisible

Notes personnelles

2.13.4.3- À propos de "linux-vdso.so.1"

Il s'agit d'une passerelle entre l'espace noyau et l'espace utilisateur, qui améliore les performances des appels systèmes.

C'est une librairie virtuelle (**VDSO**), montrée par le noyau pour chaque processus, sur l'avant dernière page mémoire utilisée par le processus (si le noyau possède le flag "COMPAT_VDSO"), et à un endroit aléatoire sinon.



On peut désactiver "**VDSO**" (vdso=0 sur la ligne de commande du noyau)... et alors...

```
# ldd /bin/bash
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x00be9000)
        libdl.so.2 => /lib/libdl.so.2 (0x00110000)
        libc.so.6 => /lib/libc.so.6 (0x00c15000)
        /lib/ld-linux.so.2 (0x00f93000)
```

Voir http://lkml.org/lkml/2002/12/9/13
Voir aussi /usr/src/linux-2.6*/Documentation/kernel-parameters.txt, vers la fin (vdso).
Ce lien traite du sujet : http://www.trilithium.com/johan/2005/08/linux-gate/

Notes personnelles

2.14- Construction de paquetages RPM

2.14.1- La commande rpmbuild

D'approche un peu compliquée, "**rpmbuild**" a été écrite :
par des développeurs...,
pour des développeurs...,
donc avec le souci de rendre la vie plus facile... aux développeurs !

Elle prévoit la construction d'un paquetage **RPM** à partir d'une archive source au format **tar**. Pour créer un paquetage, il faut donc bien sûr disposer du contenu : les sources, les patches, ...

Elle gère les problématiques auxquelles on est confronté pour "installer à partir des sources" :
exigence de pré-requis (installation des librairies et includes nécessaires), vérifications...
extraction des sources : ***tar*** …..., entrée dans le répertoire des sources : ***cd*** ….
configuration des sources : ***configure***, compilation : ***make***, installation : ***make install***
opérations de pré ou post installation (création d'utilisateurs, de groupes, de fichiers...)

En complément, on peut ajouter des informations, une description... (des rpmtags).

Outillage et sources
Il faut la commande "**rpmbuild**", fournie par le paquetage **rpm-build**.

Environnement de production de paquetages
Il n'est pas forcément nécessaire d'être "root" pour construire les paquetages, bien au contraire, certains sont même prévus pour être construits par un utilisateur non privilégié.

Notes personnelles

2.14.2- Fonctionnement de la commande rpmbuild

La commande "**rpmbuild**" peut travailler à partir de deux sources différentes :

une archive tar compressée : **rpmbuild -tX** archive.tar.gz
Cela suppose que l'archive compressée contienne un fichier « .spec »

ou

un fichier "SPEC" : **rpmbuild -bX** fichier.spec
Cela suppose que l'archive tar compressée soit dans le répertoire "SOURCES".

Le niveau d'avancement (Stage) : X
L'indication "**X**" doit être la lettre de l'opération à laquelle on souhaite s'arrêter.

Notes personnelles

Les étapes de construction

Préparation : exécution des commandes et macros de la section **%prep ("p")**

Compilation : exécution des commandes et macros de la section **%build** ("**c**") (intègre p)

Installation : exécution des commandes et macros de la section **%install** ("**i**") (intègre p et c)

toute macro de la section "**%files**" est exécutée à ce moment là aussi

Listing des fichiers : contrôle du contenu de la section **%files** ("**l**")

Grâce aux options (p, l, c, i, puis b, s, ou a), l'exécution peut être stoppée à un point précis.

Lorsque ces étapes réussissent, on peut envisager de produire un package :
Package **b**inaire : "**b**" (intègre p, l, c, et i)

Package **s**ource uniquement : "**s**",

Packages "**binary** **a**nd **source**" (binaire et source) : "**a**" (intègre p, l, c, et i)

Notes personnelles

2.14.3- Le fichier SPEC

Il est au cœur de la construction d'un package RPM, et indique à **rpmbuild** :

Description du package, dépendances...	"rpm -qi", "rpm -q --requires"
Comment préparer et construire le package...	"tar ; configure ; make"
Quels fichiers sont y intégrés...	"make install" , "rpm -ql"
Scripts de pré ou post installation...	"rpm -q --scripts"

Notes personnelles

Il est donc découpé en **plusieurs sections** possibles :

Preamble : Description, dépendances... Identification des sources, des patches.

%prep : C'est généralement un "**tar**" et un "**cd**", l'application d'un **patch**...
La macro « **%setup** » simplifie ces tâches.

%build : contient les opérations de construction et de compilation : **configure** + **make**

%install : contient les opérations d'installation : **make install** et/ou **cp**, **mv**, **ln**, **install**....

Sections de Scripts : opérations de pré/post (dés)installation : **%pre**, **%post**, **%preun**, **%postun**

%files : liste (**exhaustive**) des fichiers à inclure dans le paquetage.
On peut classifier les fichiers (doc, config...)...
Sert à la création, l'installation et la désinstallation.

D'autres, moins utilisées :
%verifyscript : exécuté sur les cibles, lorsque l'on vérifie un paquetage (**rpm -V**).
%clean : pour nettoyer la machine qui a construit le paquetage après la construction.

Notes personnelles

2.14.4- Arborescence rpmbuild

Rpmbuild utilise une petite arborescence[14] : **un répertoire "rpmbuild" avec ce contenu :**

SOURCES : contient les archives de sources, les patches, les fichiers d'icônes.

SPECS : contient les fichiers SPEC, décrivant les opérations à résliser pour construire.

BUILD : répertoire de travail, là où %prep est réalisé et où %build est réalisé.

BUILDROOT : répertoire (temporaire) où %install est réalisé et %files contrôlé (nettoyé après).

RPMS : contient les paquetages binaires générés.

SRPMS : contient les paquetages sources générés.

14 Depuis RHEL 6, c'est en principe sur votre HOME que "rpmbuild" crée cette arborescence.

Notes personnelles

2.14.5- Macro personnelles et paramétrage "rpmrc"

On peut définir dans le fichier "**~/.rpmmacros**" des macros personelles :

```
$ echo '%_topdir    /home/fmicaux/rpmbuild' > ~/.rpmmacros
$ mkdir -p /home/fmicaux/rpmbuild/{SOURCES,SPECS,BUILD,RPMS,SRPMS}
```

D'autres macros, comme "**%setup**", sont "*hardcoded*".... donc absentes de ce fichier.

Des fichiers "**rpmrc**" (/usr/lib/rpm/rpmrc, /usr/lib/rpm/redhat/rpmrc, /etc/rpmrc, ~/.rpmrc) permettent de définir des informations de configuration du processus de construction.

Par défaut, sur RHEL 5, l'architecture cible est "i386" :

```
$ grep "^buildarchtranslate: $(uname -m)" /usr/lib/rpm/rpmrc
buildarchtranslate: i686: i386
```

Pour forcer la bonne architecture (ex: i686), il suffit de spécifier les préférences de cette manière :

```
$ echo "buildarchtranslate: i686: i686" > ~/.rpmrc
```

Préférences personnelles

Qu'il s'agisse des macros ou des préférences d'architectures, le fichier « personnel » (.rpmrc, .rpmmacros) prédomine sur celui du système (/usr/lib/rpm/rpmrc, /usr/lib/rpm/rpmacros).

Notes personnelles

2.15- Déclarer un dépôt de packages

2.15.1- Configuration des clients

2.15.1.1- YUM : RedHat / CentOS / Fedora

On part en principe de ce qu'attendent les clients Yum, dont la configuration indique où sont cherchés les paquetages.

On déclare une section **[nom]** par catégorie de paquetage et au moins une **URL**.

Extrait de **/etc/yum.repos.d/CentOS-Base.repo** & **/etc/yum.repos.d/epel.repo**

```
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
```

```
[epel]
name=Extra Packages for Enterprise Linux 6 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/6/$basearch
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-6&arch=$basearch
failovermethod=priority
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
```


Notes personnelles

Pour Fedora, c'est relativement similaire, extrait de /etc/yum.repos.d/fedora.repo :

```
[fedora]
name=Fedora $releasever - $basearch
failovermethod=priority
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=fedora-$releasever&arch=$basearch
#baseurl=http://fr2.rpmfind.net/linux/fedora/releases/$releasever/Fedora/i386/os/
#baseurl=http://maupiti/Fedora/releases/$releasever/os/
enabled=1
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora file:///etc/pki/rpm-gpg/RPM-GPG-KEY
```

Dans les 2 cas, si l'URL "**mirrorlist**" est accessible, elle fournit une liste de réponses du type "**baseurl**".

sinon,

on peut spécifier l'URL que l'on veut et construire un miroir spécifique

ou on peut s'adapter à la forme proposée par "baseurl" :

/centos/XX/os/i386 ou **/centos/XX/os/x86_64**..

Notes personnelles

2.15.1.2- APT : Debian, Ubuntu...

Le fichier **/etc/apt/sources.list** référence des URLs, la distribution, et les catégories de paquetages disponibles.

Extrait de /etc/apt/sources.list :

```
deb     ftp://mir1.ovh.net/debian/  stable         main contrib non-free

deb     http://security.debian.org/ stable/updates main
deb-src ftp://mir1.ovh.net/debian/  stable         main
```

Signification des champs :
Le premier mot est "**deb**" ou "**deb-src**". Il indique un type de paquetage, binaire ou source.

Le second mot est l'adresse du site à contacter, précédée du protocole utilisé.

Le troisième mot est le nom du répertoire de base côté serveur.

Ce mot est le nom d'un sous-répertoire de second niveau situé dans "**dists**" (voir plus loin) correspondant à la distribution.

Dans Yum, les 2 notions sont concaténées.

Les autres mots indiquent les sections (catégories) de paquetages disponible sur le miroir.

- Ces mots sont les noms des sous répertoires de 3ème niveau, situés dans les répertoire de second niveau se trouvant eux-même dans "dists". (voir plus loin).

Notes personnelles

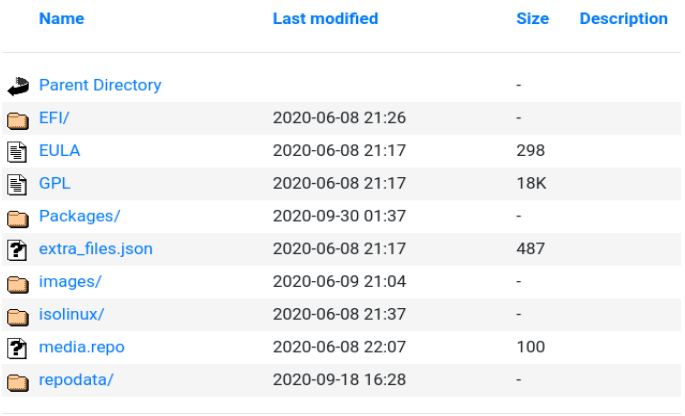
2.16- Mise en place d'un dépôt de paquets local

2.16.1- Espace de stockage et contenu d'un dépôt

Un dépôt est un répertoire d'un serveur accessible aux clients Yum, par HTTP ou FTP.

On y trouve des méta-données permettant au client d'interroger le dépôt, regroupées dans un répertoire "'**repodata**", et bien sûr les packages (dans le répertoire **Packages**).

Les méta-données sont des fichiers XML qui listent les contenus et description des paquetages disponibles.



Name	Last modified	Size	Description
Parent Directory		-	
EFI/	2020-06-08 21:26	-	
EULA	2020-06-08 21:17	298	
GPL	2020-06-08 21:17	18K	
Packages/	2020-09-30 01:37	-	
extra_files.json	2020-06-08 21:17	487	
images/	2020-06-09 21:04	-	
isolinux/	2020-06-08 21:37	-	
media.repo	2020-06-08 22:07	100	
repodata/	2020-09-18 16:28	-	

Ce sont ces fichiers que la commande "yum" (list, search, install) télécharge à chaque si la version qu'elle a en cache est expirée (d'abord **repomd.xml** (Repository Meta Data), et ensuite ceux vers lesquels il pointe, à commencer par **primary.xml.gz**).

Parent Directory		-
Packages/	12-Oct-2012 23:54	-
drpms/	13-Oct-2012 01:41	-
repodata/	13-Oct-2012 01:41	-

Depuis RHEL 6, un répertoire "**drpms**" (Delta-RPMS) peut exister, permettant une économie de bande passante lorsque le plugin "**presto**" de ***Yum*** est utilisé, ou avec ***Dnf***, lorsque le package "**deltarpm**" est présent et activé.

Notes personnelles

2.16.2- Contenu du répertoire repodata

Les métadonnées sont produites sous 2 formats : XML ou SQLite

Name	Last modified	Size	Description
Parent Directory		-	
08a9add0907af002934460d81ea0edc8bb4154db679cdc113d4c51efcbddfce4-comps-BaseOS.x86_64.xml.xz	2020-06-08 22:05	55K	
2caf33eae61c01725123a875217fe6bb0754c2916e7336286fbc392d23f42b57-other.sqlite.xz	2020-06-08 22:05	352K	
53db8eac92f79abf479e202fb013aca86d5060fd948e1ed7ea8f9493e72fd4d1-filelists.xml.gz	2020-06-08 22:05	1.2M	
190ce1d49a76eafb61b0e2738d7331a45f1efbdfb4c2c274176486f8c82f7f80-primary.xml.gz	2020-06-08 22:05	1.0M	
89376911bec34defd11535ee1f9e74237ec25e63c4dc5041ed519f1166d1cdca-other.xml.gz	2020-06-08 22:05	374K	
a6e31179a63dffb12846a2f8ad848618e7669225deb4222acadcbabb04d522f0-filelists.sqlite.xz	2020-06-08 22:05	1.2M	
a9b7530c36c9681b97c159cd98693e7fffecf2d1018d508f990934a4fa71b447-primary.sqlite.xz	2020-06-08 22:06	1.5M	
fe7d9972481aaef922e8a2fafa4065c4eb9422125da783a058a9988cd9f3eb27-comps-BaseOS.x86_64.xml	2020-06-08 22:05	290K	
repomd.xml	2020-06-08 22:06	3.9K	
repomd.xml.asc	2020-09-18 18:27	811	

Rôle de ces fichiers :
primary.xml.gz description des packages, (y compris son / chemin relatif à la base du repository).

Le nom du répertoire "Packages" est donc libre et est indiqué dans "primary.xml.gz".

filelists.xml.gz est utilisé lors de la recherche de package par "**yum provides**".

Le fichier "**comps.xml**" est une ***description multilingue des groupes de paquetages*** (ce sont les catégories disponibles à l'installation). On ne retrouve pas ce fichier dans les repository annexes.

Notes personnelles

2.16.3- Générer son propre repository

Il suffit de deux étapes :

déposer des packages RPM dans un répertoire accessible par un serveur HTTP ou FTP

créer les méta-données

C'est la commande **createrepo (**http://createrepo.baseurl.org/, package **createrepo_c**) qui crée ces méta-données.

Notes personnelles

3- Stockage, Swap, Systèmes de fichiers

Notes personnelles

3.1- Les block devices sous Linux

Les disques, partitions des disques, CD/DVD, clés USB ou cartes mémoire sont des **block-devices**[15].

Limites en taille : Avec une taille de bloc de 4ko, on peut atteindre :
Pour un filesystem : 16 TB (limite des outils « e2fsprogs »). 1 EB par conception pour ext4,
Pour un fichier : 2 TB en ext3, 16 TB en ext4, 8 EiB en XFS, 16 EiB en BtrFS

3.1.1- Les interfaces disques

SATA a succédé à IDE (PATA), qui n'est plus utilisé pour les disques
SATA = Serial ATA : SATA 1.0 (150MB/s), SATA2.0 (300 MB/s), SATA3.0 (600MB/s)
Un périphérique par connecteur, sur une nappe simple

SCSI : Reconnue par le noyau Linux dès 1993, permet de connecter disques, bandes, lecteurs cdrom...
Notion d'Id : un identifiant par périphérique + 1 pour le contrôleur
Évolution : SCSI, SCSI-2, Wide SCSI, Ultra Wide SCSI, U160, U320, ... SAS (600MB/s)
Gérée par son propre BIOS

VirtIO : Mode de gestion des disques virtuels pour les machines "KVM" entre autre.
Communication entre l'hôte et l'invité sous forme d'un FIFO
Minimaliste ... objectif performances par limitation des couches

15 C'est à dire des périphériques accédés en mode bloc (par opposition à caractère).

Notes personnelles

3.1.2- Noms des disques sous Linux

IDE (PATA) / UDMA : **/dev/hd?** (Majeur 3 pour IDE0 (hda, hdb) et 22 pour IDE1 (hdc, hdd))

SCSI, SAS & HBA Fiber Channel : **/dev/sd?** dans la plupart des cas (Majeur 8)
Si plusieurs contrôleurs, ils sont nommés scsi0 à scsin
Les noms des disques se suivent (sda, sdb, ...), classés par ID SCSI croissant.

SATA, DISQUES ET STICKS USB : vus et nommés comme des disques SCSI: **/dev/sd?**
Pour l'USB : pilote **usb-storage**

Les contrôleurs RAID mettent à disposition des disques "logiques"nommés **/dev/sd?**
Adaptec (2100s, 2010 ... : pilote **dpt-i2o**),
Intel (pilote **gdth**),
Dell Perc* (pilote **megaraid**),
MPT : à base de chipset LSI (ATTO / LSI : pilote **mptscsih**),

Contrôleurs Compaq / HP Smart Array (*CCISS*) d'ancienne génération (≤ G7)
Nommage sous la forme **C**ontrôleur / **D**isque / **P**artition : **/dev/cciss/c[0-7]d[03]p?**
jusqu'à 7 contrôleurs « cciss » (/dev/cciss/c0 à /dev/cciss/c7)
jusqu'à 4 disques par contrôleur (d0 à d3)
jusqu'à 15 partitions par disque (p1 à p15)

VIRTIO : les disques sont nommés **/dev/vd?**
pilote **virtio-blk**

Notes personnelles

3.2- Détecter les disques présents

La commande "**lsscsi**" et **/proc/scsi/scsi** permettent d'identifier les périphériques (& contrôleurs) vus :

```
# lsscsi
[0:0:0:0]    disk    VBOX     HARDDISK         1.0   /dev/sda
[0:0:1:0]    disk    VBOX     HARDDISK         1.0   /dev/sdb
[1:0:0:0]    cd/dvd  VBOX     CD-ROM           1.0   /dev/sr0
[2:0:0:0]    disk    VBOX     HARDDISK         1.0   /dev/sdc
[2:0:1:0]    disk    VBOX     HARDDISK         1.0   /dev/sdd

# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: VBOX     Model: HARDDISK         Rev: 1.0
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi0 Channel: 00 Id: 01 Lun: 00
  Vendor: VBOX     Model: HARDDISK         Rev: 1.0
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi2 Channel: 00 Id: 00 Lun: 00
  Vendor: VBOX     Model: HARDDISK         Rev: 1.0
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi2 Channel: 00 Id: 01 Lun: 00
  Vendor: VBOX     Model: HARDDISK         Rev: 1.0
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: VBOX     Model: CD-ROM           Rev: 1.0
  Type:   CD-ROM                           ANSI  SCSI revision: 05
```

On peut aussi utiliser "**lshw -C disk**", plus détaillée.

Notes personnelles

La commande **scsi-rescan** (package **sg3_utils**) : permet de détecter un disque ajouté à chaud.

Voir les options "-a" (scanner tous les hosts) et "-r" (retrait d'un disque)
Voir aussi : echo "- - -" > /sys/class/scsi_host/host**X**/scan

Dans **/proc/partitions** : on peut lister les disques physiques (sd*) ou logiques (dm-*), partitions…

```
# cat /proc/partitions
major minor  #blocks  name

   8        0    4194304 sda
   8        1     307200 sda1
   8        2     524288 sda2
   8        3    3361792 sda3
  11        0    1048575 sr0
   8       16    8388608 sdb
   8       17    4194304 sdb1
   8       18    4193280 sdb2
   8       32    8388608 sdc
   8       48    8388608 sdd
 253        0    3338240 dm-0
 253        1       4096 dm-1
```


Notes personnelles

La commande **lsblk** liste les block-devices reconnus. (Options : -t, -D, -S…)

```
# lsblk
NAME                    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                       8:0    0    8G  0 disk
├─sda1                    8:1    0  300M  0 part /boot
├─sda2                    8:2    0  512M  0 part [SWAP]
└─sda3                    8:3    0  7,2G  0 part
  └─rootvg-lvroot       253:0    0  7,2G  0 lvm  /
sdb                       8:16   0   40G  0 disk
└─sdb1                    8:17   0   40G  0 part
  ├─datavg-centos7      253:1    0   23G  0 lvm  /data/CentOS/7
  ├─datavg-pub          253:3    0    3G  0 lvm  /data/pub
  ├─datavg-exercices    253:4    0  500M  0 lvm  /data/exercices
  ├─datavg-centos       253:5    0  500M  0 lvm  /data/CentOS
  ├─datavg-debian       253:6    0  100M  0 lvm  /data/debian
  ├─datavg-proxy_cache  253:7    0  1,5G  0 lvm  /var/cache/httpd/proxy
  └─datavg-opensuse     253:8    0   27G  0 lvm  /data/opensuse
sdc                       8:32   0   20G  0 disk
└─sdc1                    8:33   0   20G  0 part
  ├─datavg-centos6      253:2    0   12G  0 lvm  /data/CentOS/6
  └─datavg-opensuse     253:8    0   27G  0 lvm  /data/opensuse
sdd                       8:48   0   20G  0 disk
└─sdd1                    8:49   0   20G  0 part
  └─datavg-opensuse     253:8    0   27G  0 lvm  /data/opensuse
```


Notes personnelles

3.3- Le MBR et les partitions

3.3.1- Boot sector ou MBR

Le **boot-sector** (ou **secteur d'amorçage**) est le premier secteur logique d'un disque dur.
Son adresse CHS est 0,0,1 (Il est au cylindre 0, tête 0, secteur 1).

Il a le nom de **Master Boot Record** (**MBR**) si le disque est utilisé dans le monde Intel.

Décomposition de ses 512 octets :

- **440** pour une routine d'amorçage (**IPL**[16])
- **4** : signature optionnelle
- **2** : laissés à 0x0000
- **64** : table des partitions primaires
- **2** : une signature = 0xAA55

Structure du Master Boot Record :

Adresse		Description		Taille en octets
Hex	Déc			
0000	0	Routine		max. **444**
01B8	440	Signature optionnelle		4
01BC	444	Habituellement nul ; 0x0000		**2**
01BE	446	**Table des partitions primaires** (Quatre entrées de 16 octets, (IBM Partition Table scheme))		**64**
01FE	510	55h	MBR signature; 0xAA55	**2**
01FF	511	AAh		
Taille totale du MBR : 444 + 2 + 64 + 2 =				**512**

16 Initial Program Loader. C'est dans les 440 premiers octets que le **stage1** de GRUB est placé par la commande "setup (hd0)" .

Notes personnelles

3.3.2- Nommage des partitions, table de partitions

Le nom d'une partition est en général composé du nom du disque et un numéro :

IDE: /dev/**hd?n**, **SATA & SCSI** : /dev/**sd?n**, **CCISS** : /dev/c?d?**p**n, **VirtIO** : /dev/**vd?n**

Sur le disque **sda (majeur 8, mineur 0)**, les partitions sont nommées **sdaX** (mineur 1 à 15):

les **partitions primaires** : **sda1** à **sda4**

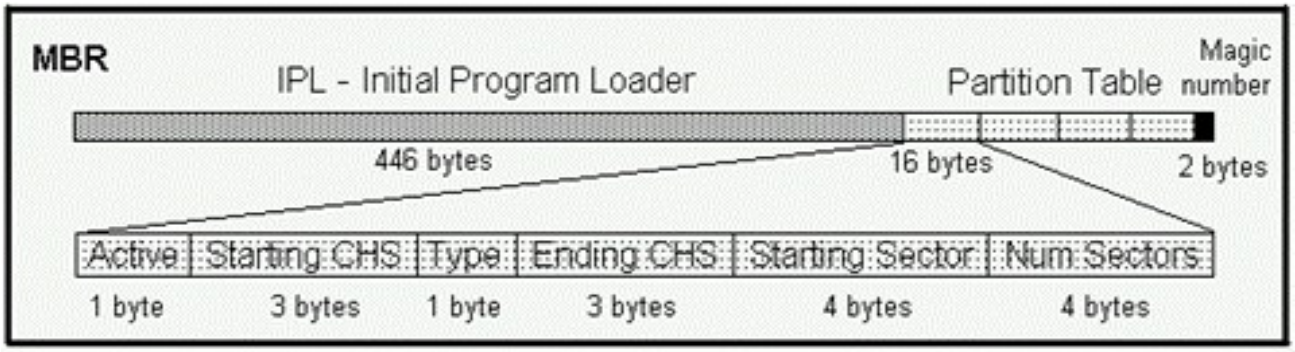
les **partitions logiques** : **sda5** à **sda15**,

Sous Linux la limite est de 15 partitions par disque SCSI/SATA

Le mineur 16 représente le disque sdb, sdb1 = mineur 17, sdb15 = mineur 31, etc.

3.3.2.1- Les partitions primaires

La déclaration des **partitions primaires** du disque réside dans les 64 octets suivant l'**IPL**.



Notes personnelles

Définition d'une partition primaire :

Adresse	Taille (o)	Description
0x00	1	**0x80** si "**active**", 0x00 si non, et invalide si autre
0x01	3	Adresse **C**(0-1023)**H**(0-254)**S**(1-63) du premier bloc - octet 1: tête - octet 2 : bits 5-0 : secteur, bits 7-6 : les bits 9-8 du n° de cylindre[17] - octet 3 : bits 7-0 du n° de cylindre
0x04	1	Type de partition (0X82, 0x83, 0x8e, 0xfd, 0x05...0xee ?)
0x05	3	Adresse CHS du dernier bloc de la partition
0x08	4 (32 bits)	LBA (Logical Block Access) du premier secteur de la partition
0x0C	4 (32 bits)	Longueur de la partition : nombre de blocs de la partition

Les adresses codées impliquent **une limite de 2 To[18] par partition et par disque**,

car impossible de coder le départ d'une partition "après la limite des 2 To".

17 Le numéro de secteur est codé sur 6 bits (0 à 63), et le numéro de cylindre est codé sur 10 bits (0 à 1023).

18 Bonne raison pour envisager le partitionnement **GPT** ("GUID Partition Table")... comme sur Mac depuis 2006. **GPT** est un autre mécanisme de gestion des disques durs et partitions, arrivant avec l'EFI, proposé par Intel en remplacement du BIOS des PC. GPT utilise un conteneur (une partition couvrant tout le disque) qui est visible comme une partition PC Bios de type "0xee".

Notes personnelles

3.3.2.2- Partition étendue et partitions logiques, EBR

Une seule des 4 partitions primaires peut avoir le type **0x05**, c'est à dire le **type Étendu**.

Elle contient des partitions logiques définies par un ou des **EBR** : **Extended Boot Record**.

Un EBR est un MBR spécifique ayant la même structure, sauf :

seules les 2 premières définitions de partitions sont censées être utilisées
- première entrée : octets 446 à 461 (0x1be à 0x1cd)
- seconde entrée : octets 462 à 477 (0x1ce à 0x1dd)

Si plusieurs partitions logiques existent,

un EBR précède toujours la partition logique qu'il décrit dans sa première entrée

la seconde entrée d'un EBR contient :
- une entrée pointant vers l'EBR de la partition suivante
- des octets à 0x00 s'il n'y a pas d'autre partition logique

Ce chaînage autorise théoriquement un nombre illimité de partitions logiques

Notes personnelles

3.4- Que faire avec une partition ?

Une partition est un **block-device**, dont les usages dépendent de son type :

3.4.1- Un volume de swap : 0x82

Une partition de type « **82** » peut être formatée par **mkswap**. Elle est ensuite activée par **swapon**.

3.4.2- Un système de fichiers : 0x83

Une partition de type « **83** » peut être formatée par **mkfs** ou dérivée, puis ensuite montée par **mount**.

3.4.3- Un volume physique LVM : 0x8e

Une partition de type « **8e** », peut être formatée en volume physique LVM par **pvcreate**.

Elle est alors admissible dans un volume group, à son tour découpé en volumes logiques. Un volume logique est avant tout un bloc device, donc utilisable en système de fichiers, ou élément de swap.

3.4.4- Un membre d'une grappe RAID : 0xfd

Une partition de type « **fd** » peut être membre d'un méta-disque manipulé par **mdadm** (soft RAID)

Une grappe RAID est avant tout un bloc device... donc utilisable en système de fichiers, élément de swap, ou ... volume physique pour LVM ou élément d'une autre grappe RAID.

Notes personnelles

3.5- Le partitionnement

3.5.1- Les outils de partitionnement

L'outil « graphique » disponible lors de l'installation varie d'un éditeur à l'autre.

Les commandes standards utilisable en exploitation sont plus pou moins scriptables :

La commande **fdisk** (le standard), scriptable (utilise son entrée-standard)

La commande **cfdisk** (crée de « belles tables de partitions »), n'est pas scriptable

La commande **sfdisk** (non interactive), adaptée pour les scripts

La commande **parted** (non interactive), adaptée pour les scripts

3.5.2- GPT

L'utilitaire **gdisk** est le pendant de "fdisk", mais pour des partitions GPT.

Notes personnelles

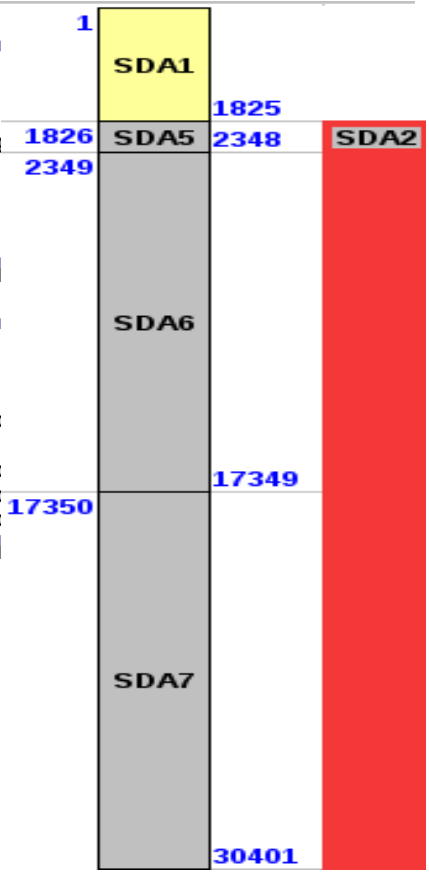
3.5.3- Lister les partitions

Avec "parted" :

```
# parted /dev/sda print
...
Numéro  Début   Fin     Taille  Type      Système de fichiers  Fanions
 1      32,3kB  15,0GB  15,0GB  primary   ext4                 démarrag
 2      15,0GB  250GB   235GB   extended
 5      15,0GB  19,3GB  4302MB  logical   linux-swap(v1)
 6      19,3GB  143GB   123GB   logical                        raid
 7      143GB   250GB   107GB   logical                        raid
```

Avec "fdisk" :

```
# fdisk -l /dev/sda
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        1825    14659281   fd  Linux raid autc
/dev/sda2            1826       30401   229536689+   5  Extended
/dev/sda5            1826        2348     4200966   82  Linux swap / So
/dev/sda6            2349       17349   120495501   fd  Linux raid autc
/dev/sda7           17350       30401   104840158+  fd  Linux raid autc
```



Cette table de partitions décrit 2 partitions primaires.

Sda1, la première, de 15 Go

Sda2, la seconde, de 235 Go, est de type étendu

elle contient 3 partitions logiques (**sda5**, **sda6**, **sda7**)

Notes personnelles

3.5.4- Créer des partitions

Commandes de "fdisk" les plus utiles :

- **u** : passage de l'affichage en secteurs / cylindres
- **c** : activer/désactiver le mode de compatibilité DOS
- **p** : print partition table
- **m** : manuel (help, aide, documentation)
- **n** : new
- **d** : delete
- **t** : changement de type
- **l** : lister les types possibles
- **w** : write
- **q** : (ou CTRL-C) quitter sans enregistrer mes bêtises.

Lorsque la table de partitions est écrite, la prise en compte par le noyau est tentée à la sortie de fdisk.

Cette prise en compte peut aussi être demandée par la commande **partprobe**, ou par **hdparm -z**.

Dans tous les cas, un "reboot" permet une prise en compte de la table de partitions.

La prise en compte immédiate par le noyau n'est possible que si le disque concerné n'est pas actuellement en cours d'utilisation (une de ses partitions est montée, ou swap activé, etc.).

Notes personnelles

3.6- Différents types de systèmes de fichiers

3.6.1- VFS

1992 : naissance de **VFS** : couche d'abstraction du type de système de fichiers.

Elle permet de monter des systèmes de fichiers de types différents de manière transparente.

3.6.2- La famille de systèmes de fichiers ext : ext2, ext3...

1993 : EXT2 : le FS « standard » du système Linux

Plusieurs copies du super bloc (informations de gestion du FS),
Taille variable du bloc, déterminée à la création : 1, 2, ou 4 Ko,
Division des blocs en groupes de blocs,
Fragmentation des blocs prévue mais pas implémentée,
Fast Symbolic Link (nom < à 64 octets) dans l'inode, ne consomme pas de bloc de données,
Comptage des montages et démontages pour exécuter « fsck » préventif automatique,
Taille maximale d'un fichier limitée 2 To avec une taille de bloc de 4 Ko.

1999 : EXT3 (proposé en 2001 sur les distributions), apporte un journal à EXT2.
Il devient rapidement le système de fichiers standard jusqu'en 2009.

2006 : EXT4 (stable depuis fin 2008) et s'est depuis imposé : c'est le système de fichiers proposé par défaut dans les distributions sorties à partir de 2009.

Notes personnelles

3.6.3- Systèmes de fichiers journalisés

ReiserFS, **Ext3/4**, **JFS**, **XFS** sont des filesystems journalisés et peuvent être agrandis en ligne.

ReiserFS : ReiserFS offre de très belles performances pour des petits fichiers (< 1Mo)

JFS (**IBM**) : vient du système d'AIX, robuste, mais est assez lent (comme ext3).

XFS (SGI) : XFS est le plus rapide sur les gros fichiers et gros filesystems.
Il est celui par défaut dans Red Hat Entreprise Linux 7.

3.6.4- Les systèmes de fichiers génériques

Issus du monde Windows : « Fat16 », « vfat », « ntfs », « iso9660 »

Systèmes de fichiers réseau : « smb/cifs », « nfs »

3.6.5- Types de systèmes de fichiers actuellement gérés par le noyau ?

```
$ grep -v nodev /proc/filesystems
        ext3
        ext2
        ext4
        xfs
```

Pour que mon noyau supporte un nouveau type de filesystem, il suffit de charger le driver !

Notes personnelles

3.7- Montage de systèmes de fichiers

3.7.1- Montage de système de fichiers

Montage : mount

Attache la racine d'un filesystem à un **répertoire existant**[19],

Le répertoire d'accueil est appelé **point de montage**

« **-o** » pour spécifier des options de montage : **ro, rw, usrquota, grpquota**, etc.

L'option particulière "**remount**" permet de modifier les options de montage.
Exemple : Passage de lecture seule à lecture/écriture.

```
mount -o remount,rw /
```

Si le point de montage n'est pas vide au moment du montage...

Les fichiers s'y trouvant sont cachés au moment du montage,
Ils seront de nouveau visibles lors du démontage

19 Qu'il faut donc au préalable avoir créé (**mkdir**)

Notes personnelles

3.7.2- Lister les montages

La commande "**mount**", sans argument, montre les informations suivantes

Les **devices** "/dev/mapper/volgroupe-vollogique" (synonymes de /dev/volgroupe/vollogique).
Le **point de montage**.
Type de système de fichiers.
La liste des **options de montage**.

```
# mount
...
/dev/vda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

Le fichier **/proc/mounts** :

le format est celui du fichier **/etc/fstab**

```
# grep boot /proc/mounts
/dev/vda1 /boot xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
```


Notes personnelles

3.7.3- Démontage de système de fichiers

Démontage : umount

Suppression du lien entre le point de montage et le filesystem.
Fermeture propre du système de fichiers.
Une des deux indications suffit :

umount /dev/périphérique

ou

umount /répertorie

Il faut qu'aucun processus ne bloque la ressource pour pouvoir la démonter...

```
# umount /boot
# umount /actilis
démontage : /actilis: périphérique occupé.
       (Dans certains cas, des infos sur les processus l'utilisant
        sont récupérables par lsof(8) ou fuser(1))
```

La commande **fuser** : identifie les processus utilisant un fichier ou répertoire... un montage...

```
# fuser /actilis
/actilis:            25401c
```

Le "**c**" montre que ce répertoire est le répertoire courant du processus 25401.

Notes personnelles

3.7.4- La commande lsof

Identifie les fichiers ouverts par les processus (ou un processus : **-p**)

```
# lsof -p $$
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME
bash    26053 root  cwd    DIR  253,0     4096 128778 /root
bash    26053 root  rtd    DIR  253,0     4096      2 /
bash    26053 root  txt    REG  253,0   903848 158487 /bin/bash
bash    26053 root  mem    REG  253,0    65928 139422 /lib64/libnss_files-2.12.so
bash    26053 root  mem    REG  253,0 99158576  20511 /usr/lib/locale/locale-archive
bash    26053 root  mem    REG  253,0  1921096 132477 /lib64/libc-2.12.so
bash    26053 root  mem    REG  253,0    19536 139408 /lib64/libdl-2.12.so
bash    26053 root  mem    REG  253,0   135896 129382 /lib64/libtinfo.so.5.7
bash    26053 root  mem    REG  253,0   154520 132951 /lib64/ld-2.12.so
bash    26053 root  mem    REG  253,0    26060  20537 /usr/lib64/gconv/gconv-modules.cache
bash    26053 root    0u   CHR  136,0      0t0      3 /dev/pts/0
bash    26053 root    1u   CHR  136,0      0t0      3 /dev/pts/0
bash    26053 root    2u   CHR  136,0      0t0      3 /dev/pts/0
bash    26053 root  255u   CHR  136,0      0t0      3 /dev/pts/0
```

Elle liste les répertoires utilisés (**cwd**, **rtd**), l'exécutable (**txt**), les librairies chargées (**mem**), les sockets, les entrées-sorties (0,1,2), etc... mentionne les types de fichiers (**DIR**, **REG**, **CHR**, **FIFO**, **unix**, **IPv4**, **IPv6**, ...), et surtout : le **PID**...

Elle peut s'intéresser aux processus d'un utilisateur (**-u**), d'un groupe (**-g**), aux sockets réseau (**-i**), ...

Sans option, elle s'intéresse à tous les processus... pour savoir quel processus "bloque" /var... :

```
# lsof | grep /var
```


Notes personnelles

3.8- Le fichier /etc/fstab

Le fichier **/etc/fstab** liste les filesystems connus par le système**.** Il contient une ligne par filesystem.

3.8.1- Structure

Chaque ligne contient 4, 5 ou 6 champs :

Périphérique (**/dev/s**da3, **LABEL**=nom, **UUID**=72377f.....4617f6)
Point de montage (exemple : /usr)
Type de FS (exemple : ext4, xfs)
Options de montage (exemple : defaults, usrquota, grpquota, noatime,noauto)
Inclusion pour les sauvegardes incrémentales par « dump » (0 ou 1)
Vaut 0 si non précisé
Phase de contrôle automatique au montage (0, 1, ou 2)
Vaut 0 si non précisé

```
/dev/mapper/vg_kvm0201-lv_slash /                       ext4    defaults        1 1
UUID=653a55dc-8342-440c-85ac-a6ecfc4696e6 /boot         ext4    defaults        1 2
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
/dev/vg_data/lv_backup  /data/backup            ext4    defaults,noatime,noauto 0 0
/dev/vg_data/lv_www     /data/www               ext4    defaults,noatime        0 2
/dev/vg_data/lv_mysql   /data/mysql             ext4    defaults,noatime
/dev/vg_data/lv_mail    /data/mail              ext4    defaults,noatime
```


Notes personnelles

3.8.2- Montage et démontage automatique

Il est effectué lors du boot, c'est un des rôles du script **rc.sysinit** (ou **rcS.d/checkfs** sous Debian)

Il effectue un montage automatique des systèmes de fichiers définis dans **/etc/fstab**.
Seuls ceux qui n'ont pas l'option "**noauto**" sont concernés

Le système de fichiers racine (/) est déjà monté, mais en lecture seule :
Il est vérifié, et ensuite passé en lecture/écriture (l'option rw lui est ajoutée) (Phase =1)

Les systèmes de fichiers listés dans /etc/fstab sont d'abord vérifiés, puis montés (Phase =2)
Ceux ayant un "0" dans la colonne "Phase" ne sont pas vérifiés.

Montage automatique de tous les FS

```
mount -a
```

Démontage automatique de tous les FS

```
umount -a
```

Ajouter des options de montage (sans démonter et remonter) :

```
mount -o remount,ro /
mount -o remount,rw /
```


Notes personnelles

3.9- Création et maintenance de systèmes de fichiers

3.9.1- Recherche des secteurs défectueux

La commande « **badblocks** » peut être utilisée pour rechercher les blocs défectueux...

```
badblocks
```

Elle peut générer un fichier (option "**-o**" suivi du nom du fichier) listant les secteurs défectueux, ou être appelée directement par **mkfs** (option **-c**).

3.9.2- Créer un filesystem

Le seul paramètre obligatoire est le nom du block-device à formater. On peut préciser la taille du FS à créer (déduite de la taille de la partition ou du support)

```
mkfs
```

Elle peut utiliser et tenir compte d'un fichier référençant les secteurs défectueux (option "**-l**" suivie du nom du fichier), ou passer le contrôle durant le formatage (option "**-c**").

3.9.3- Vérifier un FS

Vérifie un FS, et corrige (tente de corriger) les problèmes éventuels

```
fsck
```


Notes personnelles

3.9.4- Spécifier un type de système de fichiers lors du formatage ou du contrôle

Les commandes génériques (**mkfs** ou **fsck**) permettent de gérer presque tous les FS :

```
mkfs -t ext4
mkfs -t xfs
mkfs -t jfs
...
```

Elles font appel à l'outil de formatage correspondant, qui doit être installé.

Lors du formatage, on peut spécifier des options propres à chaque type de filesystem (voir la documentation des commandes spécifiques, pas celle de "mkfs").

```
# cd /sbin
# echo mkfs.*
mkfs.btrfs mkfs.cramfs mkfs.ext2 mkfs.ext3 mkfs.ext4 mkfs.ext4dev mkfs.gfs2 mkfs.hfs mkfs.hfsplus
mkfs.jfs mkfs.msdos mkfs.nilfs2 mkfs.ntfs mkfs.reiserfs mkfs.vfat mkfs.xfs
```

Idem pour **fsck**.

```
# echo fsck.*
fsck.cramfs fsck.ext2 fsck.ext3 fsck.ext4 fsck.ext4dev fsck.gfs2 fsck.hfs fsck.hfsplus fsck.jfs
fsck.msdos fsck.ntfs fsck.reiserfs fsck.vfat fsck.xfs
```


Notes personnelles

3.9.5- Redimensionner un système de fichiers : fsadm

C'est la une commande générique compatible avec différents types de filesystem, pour deux tâches différentes : **check** ou **resize**.

```
# fsadm
fsadm: Utility to resize or check the filesystem on a device

  fsadm [options] check device
    - Check the filesystem on device using fsck

  fsadm [options] resize device [new_size[BKMGTPE]]
    - Change the size of the filesystem on device to new_size

  Options:
    -h | --help          Show this help message
    -v | --verbose       Be verbose
    -e | --ext-offline   unmount filesystem before ext2/ext3/ext4 resize
    -f | --force         Bypass sanity checks
    -n | --dry-run       Print commands without running them
    -l | --lvresize      Resize given device (if it is LVM device)
    -y | --yes           Answer "yes" at any prompts

  new_size - Absolute number of filesystem blocks to be in the filesystem,
             or an absolute size using a suffix (in powers of 1024).
             If new_size is not supplied, the whole device is used.
```

Elle remplace donc avantageusement **resize2fs**, **resize_reiserfs**, et **xfs_growfs**.
Elle ne traite pas le cas de "JFS" pour lequel le redimensionnement se fait comme ceci :

```
mount -o remount,resize /mountpoint
```


Notes personnelles

3.10- Correspondance filesystem / outils

Type	Journ.	Créer	Contrôler	Agrandir	"online"
ext2	non	mkfs mke2fs mkfs.ext2	fsck e2fsck	resize2fs	non
ext3, ext4	oui	mkfs.ext3 mkfs.ext4	fsck e2fsck	resize2fs	oui
reiserfs 3.6	oui	mkreiserfs mkfs.reiserfs	fsck.reiserfs reiserfsck	resize_reiserfs	oui
reiser4	oui	mkreiser4 mkfs.reiser4	fsck.reiser4	resizefs.reiser4	oui
xfs	oui	mkfs.xfs	fsck.xfs	xfs_growfs	oui
jfs	oui	mkfs.jfs	fsck.jfs	mount -o resize,remount	Oui
gfs	oui	gfs_mkfs mkfs.gfs	gfs_fsck fsck.gfs	gfs_grow	oui
ocfs2	oui	mkfs.ocfs2	fsck.ocfs2	tunefs.ocfs2	non

C'est la commande **mkisofs** qui permet de construire une « image iso ».

Notes personnelles

3.11- Commandes propres à ext2, ext3, et ext4

3.11.1- Création de filesystem : mke2fs ou mkfs.ext2/3/4

Concernant ext2, ext3 et ext4, les commandes sont les mêmes... :

```
# ls -li $(which mke2fs mkfs.ext{2,3,4,4dev})
655456 -rwxr-xr-x 5 root root 60456 2010-04-19 06:31 /sbin/mke2fs
655456 -rwxr-xr-x 5 root root 60456 2010-04-19 06:31 /sbin/mkfs.ext2
655456 -rwxr-xr-x 5 root root 60456 2010-04-19 06:31 /sbin/mkfs.ext3
655456 -rwxr-xr-x 5 root root 60456 2010-04-19 06:31 /sbin/mkfs.ext4
655456 -rwxr-xr-x 5 root root 60456 2010-04-19 06:31 /sbin/mkfs.ext4dev
```

Avec l'option « j » de la commande "mkfs.ext2", on construit un système de fichiers **ext3**.

Pour créer un filesystem de type ext3 : ext3 = ext2 + un journal.

```
mke2fs -j    ou   mkfs.ext2 -j
```


Notes personnelles

3.11.2- Commandes de contrôle et réparation

Vérifier un système de fichiers

```
e2fsck -y  /dev/hdXn
```

ou

```
fsck.ext2 -y /dev/hdXn
```

S'informer sur le super-bloc et les groupes de blocs : **dumpe2fs**

Effectuer des manipulations (de structure) sur un FS : **debugfs**

Sauvegarder les informations critiques concernant un système de fichiers : **e2image**

S'informer sur l'espace libre et sa fragmentation : **e2freefrag**

Notes personnelles

3.12- Tuning de système de fichiers ext* : tune2fs

La commande **tune2fs** permet de consulter ou manipuler des options sur le système de fichiers.

Les options les plus courantes :

- **-l** : lister les options en place
- **-i** : interval between checks
- **-c** : max mount counts
- **-m** : % reserved blocs (pour root, sauf si **-u** / **-g** sont utilisées)
- **-L** / **-U** : Label / UUID
- **-j** : construire le journal

Les options, **-o** et **-O** permettent d'activer ou désactiver des fonctionnalités... comme le journal (donc de passer de ext2 à ext3 et inversement) :

Migration de ext3 vers ext2 : on enlève le journal

```
# tune2fs -O ^has_journal /dev/hdX
# rm -f .journal
```

Migration de ext2 vers ext3 :

```
# tune2fs  -j  /dev/hdX
```

Attention : **-o has_journal** activerait la fonctionnalité, mais ne construirait pas le journal.

Notes personnelles

3.13- Les systèmes de fichiers et la sécurité

3.13.1- Gestion de l'effacement physique des fichiers

La commande "**rm**" ne supprime pas le contenu : seul le lien vers **l'inode** est réellement supprimé.

Pour supprimer « physiquement » les blocs de données :

des attributs ext2 (ext3), gèrent l'effacement physique (vidage des blocs de données)

La commande **shred** permet le « broyage » et éventuellement une suppression.

3.13.2- Récupération des données perdues accidentellement

Pour EXT3 :

http://www.xs4all.nl/~carlo17/howto/undelete_ext3.html : le projet **ext3grep** de Carlo Wood est packagé sur rpmforge et se trouve sur http://code.google.com/p/ext3grep/

Le projet **ext3undel** (de Andreas Itzchak Rehberg, http://www.izzysoft.de/) fournit aussi un outillage. Il est aussi packagé sur rpmforge.

XFS : c'est annoncé comme impossible... pour du texte : dd if=/dev/le_block_device | strings | grep ....
ReiserFS : un petit howto : http://antrix.net/journal/techtalk/reiserfs_data_recovery_howto.comments
JFS : pas grand chose dans le monde du Libre. Des solutions commerciales semblent exister.

Notes personnelles

3.14- RAID et LVM : performances et sécurité

RAID est une technologie permettant d'améliorer, au moyen de disques agrégés, la sécurité, et les performances. Pour des raisons de sécurité (RAID1, RAID 5, RAID 6), ou pour des raisons de répartition de charge et donc de performances (RAID 0), les différents modes RAID n'offrent pas les mêmes performances, et n'ont pas le même coût au volume.

RAID consiste à privilégier deux éléments parmi trois : sécurité, performances, coût.

3.14.1- Performances du raid logiciel

Le noyau Linux implémente un gestionnaire RAID logiciel qui sollicite donc le processeur.

Un test de débit séquentiel du block device (écritures et lectures séquentielles par dd) montre des différences importantes en fonction de plusieurs paramètres :

- niveau de RAID (0, 1, 3/4, 5, 6)
- nombre de disques (surtout en RAID 3/4, 5 et 6)
- chunk size utilisé
- ... lecture / écriture ...

3.14.2- Concernant le raid matériel

Les performances varient d'un contrôleur à l'autre, en fonction des mêmes éléments que ceux faisant varier les performances du raid logiciel.

Notes personnelles

3.14.3- Quelques règles concernant le RAID

Quelques éléments ... pour savoir comparer

En lecture, RAID1, 3/4, 5 et 6 offrent des performances similaires, légèrement meilleures en 0.
En écriture, RAID 6 (puis 5) offrent les moins bonnes performances, le meilleur étant RAID 0.
En RAID 0, les performances en écriture sont très légèrement supérieures à celles en lecture.
Les performances de RAID 3/4, 5 et 6 s'améliorent quand on ajoute des disques à la grappe.
Le temps de reconstruction est beaucoup plus important en RAID 3/4, 5 et 6 qu'en RAID 1.
RAID 0 n'apporte pas de tolérance de panne.
RAID 1, 3/4, 5, 6 apportent différents niveaux de tolérance de panne (2 disques en RAID 6)
RAID 3/4, 5 et 6 chargent plus le processeur que RAID1.
RAID 0 sur RAID 1 : meilleurs taux de transfert et temps de réponse que RAID 0.

Le meilleur (...)

Disponibilité des données : 6 (perte possible de deux disques), puis sans ordre 5, 4, et 1
Temps de réponses : 0, 5, et 6 en lecture, 0 en écriture.
Taux de transfert : 0 et 3/4, le moins bon étant RAID 1.

Applications

RAID 0 : hautes performances mais pas de données critiques
RAID 1 : disques systèmes, données critiques
RAID 3 / 4 : volumes d'I/O élevés (requêtes de gros volumes)
RAID 5 & 6 : lecture intensive

Notes personnelles

3.14.4- LVM et les performances

LVM répond à un objectif principal de l'administrateur : pouvoir agrandir les volumes de stockage.

Historiquement, RAID ne proposait pas cette possibilité. C'est toutefois possible en RAID 5.

Principe de LVM

On regroupe les disques ou partitions de disques (de type 8e), appelées **volumes physiques**, au sein de **groupes de volumes**, pouvant ainsi repousser les limites liées la taille d'un disque.

On peut partitionner ces groupes de volumes de manière indépendante des disques physiques. Ces "partitions" s'appellent alors des **volumes logiques** et sont utilisés comme des block devices classiques.

La création d'un volume logique peut être réalisée avec ou sans stripping. Avec stripping, il y a répartition de charge entre les volumes physiques, sans stripping, c'est un mode linéaire classique.

On atteint en théorie les mêmes performances avec RAID 0 qu'avec LVM, mais :

- LVM est extensible.
- LVM n'apporte pas de sécurité, comme RAID 0.
- La sécurité de LVM repose en principe sur des matrices RAID.

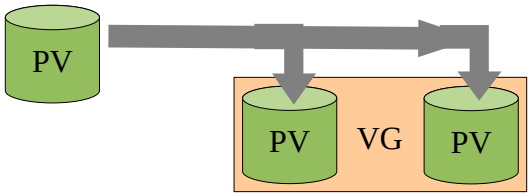
Notes personnelles

3.15- Logical Volume Manager : Concepts

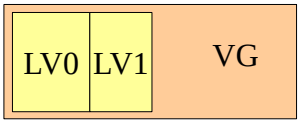
LVM2[20] consiste à faire abstraction des limites "physiques" des disques, en les agrégeant.

Des disques ou partitions deviennent des "volumes physiques"

Les "volumes physiques" rejoignent des "volume group"



Les "volume group" offrent de l'espace qu'on peut découper en "volumes logiques".



Nommage : **/dev/<nom-du-volume-group>/<nom-du-volume-logique>**

20 La version actuellement implémentée dans le noyau Linux est **LVM2**.

Notes personnelles

3.15.1- Volume physiques

Tout périphérique de type bloc peut devenir un **PV** (**Physical Volume**) :
- Disque physique ou logique, méta-disque, partition d'un disque (de préférence type **8e**)
- Volume RAID (matériel ou logiciel) ou partition d'un volume raid
- Tout périphérique accédé via iSCSI, FcoE, FC

3.15.2- Volumes groupes

Un **VG** (**Volume Group**) est constitué de PV(s)
- On l'**étend à chaud**, un en lui ajoutant des PV.
- On peut inscrire jusqu'à 255 PV par VG.
- Un PV ne peut être rattaché qu'à un seul VG.

3.15.3- Volumes logiques

Un **LV** (**Logical Volume**) est un sous-ensemble d'un VG :
- Il peut **peut être étendu** à chaud
- Un VG peut contenir plusieurs LV

Un LV est un block-device classique :
- On peut le formater (mkswap, mkfs), le monter, le partager (drbd, iscsi, fcoe...)...
- Un LV peut être "formaté" pour devenir à son tour un PV
  - Il ne pourra pas entrer dans le même VG que celui d'où il vient

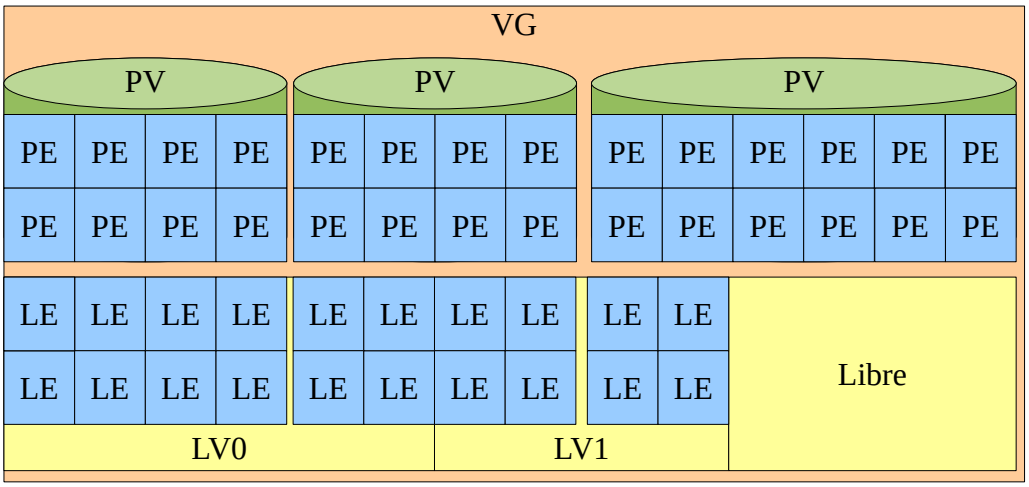
Notes personnelles

3.15.4- Unité d'allocation : les extents

Les unités d'allocation sont appelées des **Extents**.

PE (**Physical Extent**) : Division d'un **PV**, une sorte de « gros bloc », de X Mo

LE (**Logical Extent**) : Division d'un **LV**, taille identique à un **PE** (il en dépend)



Dans les commandes, on évoque les tailles :

avec "-L" ou "--size" : en Blocs, Secteurs, Ko, Mo, Go, To, Po, Eo, etc.,
ou avec "-l" ou "--extents" : en **extents**.

Notes personnelles

3.16- Pas à pas... de la partition au volume logique

Créer des partitions de type **0x8e** :

```
# fdisk /dev/sdb
```

Volume Physique : formate une partition pour utilisation avec LVM

```
# pvcreate /dev/sdb1
```

Volume Group : contient au moins 1 volume physique

```
# vgcreate mon_vg /dev/sdb1 ...
```

Volume logique : Créer un Logical Volume

```
# lvcreate -L 1500 -n lv1 mon_vg
```

Crée un LV de 1500 Mo, et crée aussi le fichier spécial /dev/mon_vg/lv1 (de type block)

Notes personnelles

3.17- Éléments de maintenance des volumes logiques

Lister les Logical Volumes : **lvscan**

```
# lvscan
  ACTIVE            '/dev/vg_raid0/lv_vbox' [50,00 GiB] inherit
  ACTIVE            '/dev/vg_raid0/lv_data' [69,00 GiB] inherit
  ACTIVE            '/dev/vg_raid1/lv_home' [20,00 GiB] inherit
  ACTIVE            '/dev/vg_raid1/lv_multimedia' [45,00 GiB] inherit
```

Les commandes **lvs** et **lvdisplay** existent aussi.

Étendre un Logical Volume : passer à 10 Go... ou ajouter 4 Go :

```
lvextend -L10G /dev/mon_vg/lv1  # (en taille "Human-readable" : -L)
lvextend -L +4G /dev/mon_vg/lv1  # (en taille "Human-readable" : -L)
lvextend -l +100%FREE /dev/mon_vg/lv1  # (ajout de 100% de l'espace libre du VG)
```

Réduire un Logical Volume : attention au système de fichiers qui repose dessus.

```
lvreduce -L -1G /dev/mon_vg/lv1
```

Le système de fichiers existant doit au préalable être réduit (**resize2fs**) sous peine de perte de données.

Supprimer un Logical Volume : Il est préférable de le démonter au préalable...

```
lvremove /dev/mon_vg/lv1
```


Notes personnelles

3.18- Éléments de maintenance des volumes groupes

Lister / afficher le détail des Volume Group :

```
vgscan
vgdisplay mon_vg
vgs
```

Activer / désactiver un Volume Group

```
vgchange -a y mon_vg
vgchange -a n mon_vg
```

Ajouter / enlever un PV à un VG :

```
vgextend mon_vg /dev/sdd1
vgreduce mon_vg /dev/sdd1
```

Il faut au préalable vérifier que le PV n'est pas encours d'utilisation.

```
pvdisplay /dev/sdd1
```

Supprimer un Volume Group : **vgremove mon_vg**

Notes personnelles

3.19- Éléments de maintenance des volumes physiques

3.19.1- Lister les volumes physiques

La commande la plus synthétique est sans doute **pvscan**.

Elle présente une information concise mais complète sur les Physical Volumes et Volume Group :

```
# pvscan
  PV /dev/md2   VG vg_raid0   lvm2 [199,96 GiB / 50,96 GiB free]
  PV /dev/md1   VG vg_raid1   lvm2 [114,91 GiB / 49,91 GiB free]
  Total: 2 [314,88 GiB] / in use: 2 [314,88 GiB] / in no VG: 0 [0   ]
```

La commande **pvs** affiche des informations similaires.

La commande **pvdisplay** donne aussi des informations

```
# pvdisplay /dev/md1
  --- Physical volume ---
  PV Name               /dev/md1
  VG Name               vg_raid1
  PV Size               114,91 GiB / not usable 3,31 MiB
  Allocatable           yes
  PE Size               4,00 MiB
  Total PE              29417
  Free PE               12777
  Allocated PE          16640
  PV UUID               D0Kk7n-XQsa-BrIu-wQU2-5DQB-frcz-ogGf1m
```


Notes personnelles

3.19.2- Supprimer un volume physique

Il faut d'abord **dégager** les blocs utilisés d'un volume physique : **pvmove**

C'est utile lorsqu'on souhaite l'affecter à un autre volume groupe

Il faut au préalable qu'il y en ait d'autres dans le même volume groupe.

La commande **pvmove** permet de libérer (déplacer) les physical extents utilisés dans un PV vers un autre PV du même VG.

Cela permet de libérer un PV, pour le retirer ensuite du VG...

Cela permet de déplacer un volume logique vers un PV plus performant...

Migrer le support physique utilisé par un volume logique

```
# pvmove -n lvdata /dev/sdc1 /dev/sdd1
```

Déplace de /dev/sdc1 vers /dev/sdd1 les données concernant le volume logique lvdata.

Déplace les données de /dev/sdc1 vers n'importe quel autre PV du même VG.

```
# pvmove -v /dev/sdc1
```

Après cette opération, /dev/sdc1 peut être **retiré du volume-group** qui le contient, puis ensuite **supprimé**.

Retirer un PV d'un VG : **vgreduce**
Supprimer un PV : **pvremove**

Notes personnelles

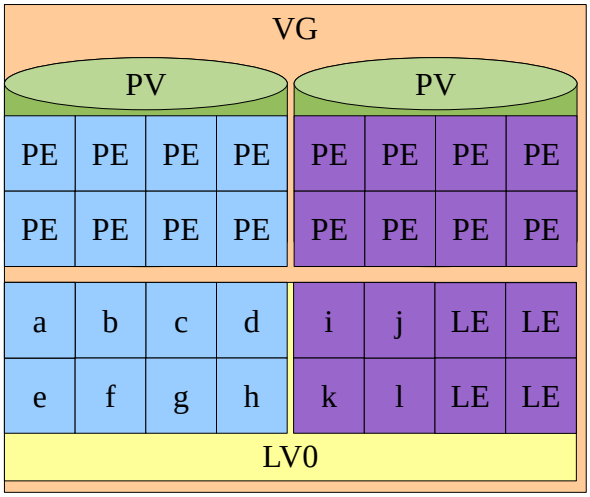
3.20- Mode d'agrégation linéaire

À la création d'un volume logique LVM propose plusieurs modes d'agrégation :

le mode linéaire est celui par défaut.

les données sont stockées sur le premier volume physique,

quand il est plein, on enchaine sur le suivant.



Notes personnelles

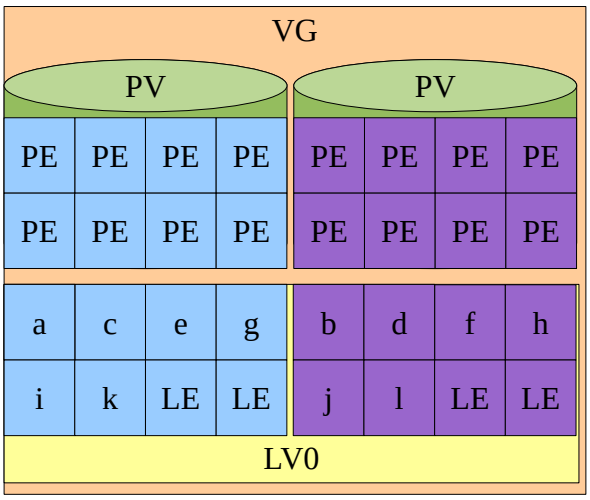
3.21- Le striping

C'est un mode d'agrégation par bandes.

les données sont réparties sur plusieurs espaces

répartition la charge de lecture et écriture,

offre une fonctionnalité du type RAID 0

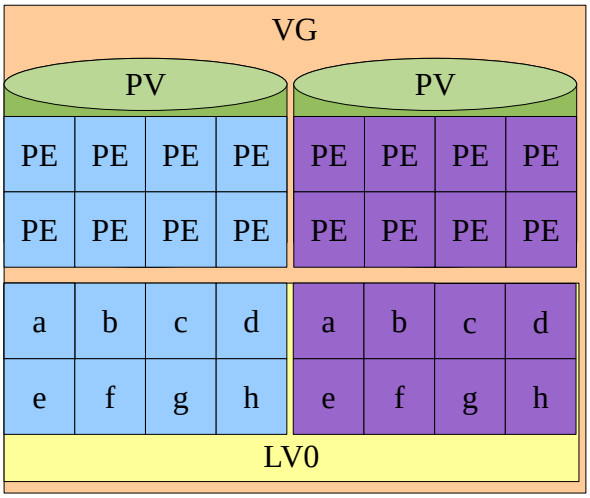


Notes personnelles

3.22- Le mirroring

les données sont recopiées à l'identique sur les supports physiques,

offre une fonctionnalité du type RAID 1



Notes personnelles

3.23- Les snapshots

Le **snapshot** est une solution pour sauvegarder des volumes à chaud.

Le snapshot (clichés) permet d'obtenir une image figée d'un volume logique à un instant T.

Le snapshot permet de minimiser les arrêts de service pour faire des sauvegardes fiables
Arrêt des bases de données
Création du snapshot "bases arrêtées"
Démarrage des bases de données
Réalisation de la sauvegarde du snapshot "bases arrêtées"

Pour des opérations à risque : tester sur un snapshot est intéressant :
Si le résultat est OK : fusion du snapshot et du volume original.
Voir **lvconvert**.

Si le résultat est KO : destruction du snapshot : retour arrière.

Le snapshot est un **volume logique** spécifique associé à un volume logique existant (LV d'origine).

Le **snapshot** constitue une photo du LV d'origine. On peut y accéder en lecture/écriture

Le LV d'origine, lui, peut continuer à vivre en lecture/écriture de son côté.

Notes personnelles

Exemple de mise en oeuvre d'un snapshot

Créer un volume logique de type snapshot pour le volume logique hébergeant "/".

```
lvcreate -s /dev/rootvg/rootlv -L 500M -n snap
```

Ajouter un fichier témoin sur "/".

```
touch /temoin
```

Première approche :

Réaliser une sauvegarde du snapshot (un tar nécessite un montage, un dd n'en aurait pas besoin).

```
mount -r /dev/rootvg/snap /mnt ; tar -C /mnt cvzf /snap-backup.tar . ; umount /mnt
```

⇒ Constater que la sauvegarde ne contient pas le fichier témoin.

Supprimer le snapshot.

```
lvremove /dev/rootvg/snap
```

Autre approche :

Plutôt que de supprimer le snapshot, on peut l'absorber vers le volume logique original,

```
lvconvert --merge /dev/rootvg/snap
```


Notes personnelles

3.24- Le cache LVM

C'est une fonctionnalité de LVM, qui est pleinement supportée lvm2 (2.02) fournie par **RHEL 7.1**.

3.24.1- Objectif

Créer un Cache-Pool reposant sur un PV rapide.

Servant de cache pour des LV plus gros reposant sur un PV plus lent.

3.24.2- Mise en œuvre

Supposons une machine dotée de...

...un disque mécanique (/dev/sda)

Le disque mécanique héberge les LV "lents".

… et un SSD (/dev/sdb).

Nous créons un cache sur un PV de /dev/sdb.

1/ **Créer un LV cache pour les données** reposant sur un PV rapide

```
# lvcreate -n Vbox_CacheData -L 10G rootvg /dev/sdb2
 Logical volume "Vbox_CacheData" created.
```

2/ **Créer un LV cache pour les métadonnées** sur un PV rapide. (taille = 1/1000 de celle du LV cache)

```
# lvcreate -n Vbox_CacheMeta -L 10M rootvg /dev/sdb2
 Rounding up size to full physical extent 12,00 MiB
```


Notes personnelles

3/ **Créer un "CachePool LV"**

Cela combine les données et métadonnées dans le CachePool LV

```
# lvconvert --type cache-pool --poolmetadata rootvg/Vbox_CacheMeta rootvg/Vbox_CacheData
```

Le résultat est un LV nommé Vbox_CacheData, et pour l'instant inactif.

```
# lvdisplay /dev/rootvg/Vbox_CacheData -m
 --- Logical volume ---
 LV Path                /dev/rootvg/Vbox_CacheData
 LV Name                Vbox_CacheData
 VG Name                rootvg
 LV UUID                F4Ir6L-vFYt-dPfI-FL8N-mgwp-BEyd-TwLk2R
 LV Write Access        read/write
 LV Creation host, time ge60.actilis.net, 2015-10-22 11:16:04 +0200
 LV Status              NOT available
 LV Size                10,00 GiB
 Current LE             2560
 Segments               1
 Allocation             inherit
 Read ahead sectors     auto

 --- Segments ---
 Logical extents 0 to 2559:
   Type                cache-pool
```


Notes personnelles

4/ **Lier le LV lent au CachePoolLV**

```
# lvconvert --type cache --cachepool rootvg/Vbox_CacheData rootvg/vbox
 Logical volume rootvg/vbox is now cached.
```

"rootvg/vbox" est maintenant un LV de type "cache" (voir **lvdisplay -m**, ou **lvs -a**)

```
# lvs -a rootvg/vbox
  LV   VG     Attr       LSize   Pool             Origin       Data%  Meta%  Move Log Cpy%Sync
Convert
  vbox rootvg Cwi-aoC--- 222,00g [Vbox_CacheData] [vbox_corig] 0,01   10,74           0,00
```

3.24.3- Suppression

Approche 1 : Splitter le cache Pool

```
# lvconvert VG/CachePoolLV
```

Cela conserve le LV d'origine et le CachePool (qu'on peut donc ré-associer à un autre LV).

Approche 2 : Retirer le cache Pool et conserver le LV d'origine

```
# lvremove rootvg/Vbox_CacheData
```

ou

```
# lvconvert --uncache rootvg/vbox
```

Approche 3 : Supprimer le cache LV, ce qui supprimerait aussi le CachePool...

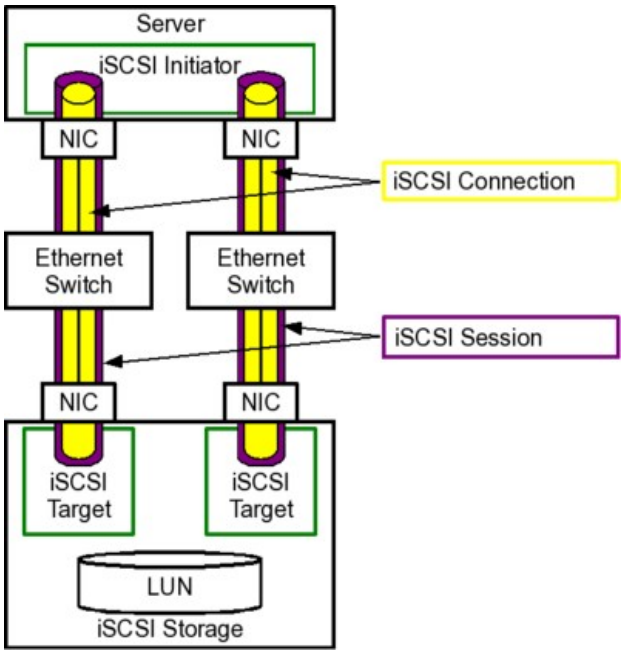
```
# lvremove rootvg/vbox … Attention, on va perdre des données…
```


Notes personnelles

3.25- ISCSI : SCSI sur le réseau Internet

3.25.1- Notions de base

iSCSI[21] est un protocole permettant d'exécuter des commandes SCSI dans des paquets IP. C'est un des protocoles d'accès à un SAN, que des logiciels offrent aussi sur les systèmes GNU/Linux.



CLIENT de stockage : On appelle "**initiator**" ISCSI le client permettant d'accéder à un stockage distant. Il peut être matériel (iSCSI HBA) ou logiciel (**iscsi-initiator**)

Package : **iscsi-initiator-utils**
(dépendance à l'installation de libvirt)

SERVEUR de stockage : On appelle "**target**" ISCSI le service de mise à disposition d'un stockage. Il est matériel dans le cas d'un SAN, et peut être logiciel.

Package(s) : **scsi-target-utils** ou **targetcli**

Un même serveur de stockage peut héberger plusieurs cibles ISCSI, et un même client peut se connecter à plusieurs cibles.

21 Internet Small Computer Systems Interface : http://www.faqs.org/rfcs/rfc3720.html

Notes personnelles

3.26- LIO : Linux-IO Target

3.26.1- Qu'est-ce que LIO ?

Auparavant, on utilisait LIO pour FCoE et "**tgtd**" (***scsi-target-utils***) pour iSCSI.

Depuis Linux 2.6.38, LIO[22] (**intégré dans le noyau**) est le standard de gestion des cibles SCSI.

Il supporte un grand nombre de fabrics : **FcoE**, **iSCSI**, **iSER**…

Il gère tous les backing-store disponibles sous Linux :

FILEIO : n'importe quel fichier présent sur un filesystem monté
Synchrone / asynchrone, Buffered / direct
Par défaut : synchrone unbuffered.

BLOCK : n'importe quel block-device apparaissant dans /sys/block
Le meilleur du point de vue performances.

PSCSI : (Linux Pass-through SCSI devices),
donc tout périphérique apparaissant dans /proc/scsi/scsi

Memory Copy RAMDISK
des ramdisks avec une émulation SCSI,
utile en production pour disposer de périphériques rapides

3.26.2- Administration de LIO

La commande d'administration est "**targetcli**" (package "**targetcli**") :

22 Linux-IO Target. Site officiel : http://linux-iscsi.org/wiki/Main_Page
Un peu de documentation : https://www.certdepot.net/rhel7-configure-iscsi-target-initiator-persistently/

Notes personnelles

```
# yum -y install targetcli
```

```
# targetcli ls
o- / ..... [...]
  o- backstores ..... [...]
  | o- block ..... [Storage Objects: 1]
  | | o- backend01 ..... [/dev/vgvm/iscsi4T (4.0TiB) write-thru activated]
  | |   o- alua ..... [ALUA Groups: 1]
  | |     o- default_tg_pt_gp ..... [ALUA state: Active/optimized]
  | o- fileio ..... [Storage Objects: 0]
  | o- pscsi ..... [Storage Objects: 0]
  | o- ramdisk ..... [Storage Objects: 0]
  o- iscsi ..... [Targets: 1]
  | o- iqn.2001-09.net.actilis:target0 ..... [TPGs: 1]
  |   o- tpg1 ..... [gen-acls, no-auth]
  |     o- acls ..... [ACLs: 1]
  |     | o- iqn.2001-09.net.actilis:client ..... [Mapped LUNs: 1]
  |     |   o- mapped_lun0 ..... [lun0 block/backend01 (rw)]
  |     o- luns ..... [LUNs: 1]
  |     | o- lun0 ..... [block/backend01 (/dev/vgvm/iscsi4T) (default_tg_pt_gp)]
  |     o- portals ..... [Portals: 1]
  |       o- 0.0.0.0:3260 ..... [OK]
  o- loopback ..... [Targets: 0]
```


Notes personnelles

La commande targetcli : interpréteur de commandes interactif (ou non)

Les commandes dépendent du chemin où l'on se trouve :
dans "iscsi", les commandes disponibles ne sont pas les mêmes que dans "backstores"...

La commande "**cd**" permet de se déplacer dans l'arborescence
Sans argument, elle devient interactive (déplacement par les flèches et validation par Entrée)

Les plus courantes étant :

ls : lister la configuration
create : créer un objet
delete : supprimer un objet
help : documentation du contexte
backstores/ help est l'aide de "backstores", iscsi/ help est celle de "iscsi"…

saveconfig : sauvegarde la configuration (sinon, elle est perdue au reboot)

Les commandes de création peuvent engendrer un passage dans le contexte créé. On inhibe cela par :

```
/> set global auto_cd_after_create=false
Parameter auto_cd_after_create is now 'false'.
```


Notes personnelles

3.26.3- Création d'une cible ISCSI

C'est l'ordre **create** du contexte "**iscsi**" :

cd iscsi, puis create… ou *iscsi/ create …*

```
/> iscsi/ create iqn.2001-09.net.actilis:target0
Created target iqn.2001-09.net.actilis:target0.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
```

On peut détruire cette ressource par la commande suivante :

```
/> iscsi/ delete iqn.2001-09.net.actilis:target0
Deleted Target iqn.2001-09.net.actilis:target0.
```

Pour la suite, supposons que la target **iqn.2001-09.net.actilis:target0** existe.

```
/iscsi> ls
o- iscsi ..................................................................... [Targets: 1]
 o- iqn.2001-09.net.actilis:target0 ..................................... [TPGs: 1]
   o- tpg1 ..................................................... [no-gen-acls, no-auth]
     o- acls ................................................................ [ACLs: 0]
     o- luns ................................................................ [LUNs: 0]
     o- portals ............................................................. [Portals: 1]
       o- 0.0.0.0:3260 ..................................................... [OK]
```


Notes personnelles

3.26.4- Enregistrer la configuration

Après chaque configuration, penser à enregistrer…par "**saveconfig**" ou par "**exit**"[23]. (après un "cd /")

```
/> saveconfig
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json
/> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json
```

3.26.5- Démarrer LIO

LIO, c'est le noyau… mais il faut prévoir de recharger la configuration à chaque reboot.

```
# systemctl enable target
ln -s '/usr/lib/systemd/system/target.service' '/etc/systemd/system/multi-user.target.
wants/target.service'
```

Après le reboot : la commande "targetctl restore" a rechargé la configuration.

```
# systemctl status target
target.service - Restore LIO kernel target configuration
  Loaded: loaded (/usr/lib/systemd/system/target.service; enabled)
  Active: active (exited) since mar. 2015-10-20 23:40:33 CEST; 1min 25s ago
 Process: 4548 ExecStart=/usr/bin/targetctl restore (code=exited, status=0/SUCCESS)
Main PID: 4548 (code=exited, status=0/SUCCESS)
```

23 C'est le paramètre global **auto_save_on_exit=true** qui provoque un enregistrement lors du exit.

Notes personnelles

3.26.6- Création d'un backing-store

Le plus simple est de se trouver dans "**/backstores**"...

```
/iscsi> cd /backstores/
/backstores> ls
o- backstores ..................................................................... [...]
 o- block ......................................................... [Storage Objects: 0]
 o- fileio ........................................................ [Storage Objects: 0]
 o- pscsi ......................................................... [Storage Objects: 0]
 o- ramdisk ....................................................... [Storage Objects: 0]
```

Ici, c'est un backstore de type "FileIO" qui va être créé.

```
/backstores> fileio/ create name=file_backend01 file_or_dev=/var/fileio01 size=500M
Created fileio file_backend01 with size 524288000
```

Pour créer un backstore "BLOCK", la commande serait par exemple la suivante :

```
/backstores> block/ create name=block_backend01 dev=/dev/sdb
Created block storage object block_backend01 using /dev/sdb.

/backstores> ls
o- backstores ..................................................................... [...]
 o- block ......................................................... [Storage Objects: 1]
 | o- block_backend01 ................... [/dev/sdb (8.0GiB) write-thru deactivated]
 o- fileio ........................................................ [Storage Objects: 1]
 | o- file_backend01 ............. [/var/fileio01 (500.0MiB) write-back deactivated]
 o- pscsi ......................................................... [Storage Objects: 0]
 o- ramdisk ....................................................... [Storage Objects: 0]
```


Notes personnelles

3.26.7- Créer un LUN

Il s'agit de l'attachement d'un backing-store à la cible.

```
/> /iscsi/iqn.2001-09.net.actilis:target0/tpg1/luns
/iscsi/iqn.20...et0/tpg1/luns> create /backstores/fileio/file_backend01 0 true
Created LUN 0.
/iscsi/iqn.20...et0/tpg1/luns> ls
o- luns ..................................................................... [LUNs: 1]
 o- lun0 ..................................... [fileio/file_backend01 (/var/fileio01)]
/iscsi/iqn.20...et0/tpg1/luns> /iscsi
/iscsi> ls
 o- iscsi ................................................................... [Targets: 1]
 | o- iqn.2001-09.net.actilis:target0 ..................................... [TPGs: 1]
 |   o- tpg1 ........................................... [no-gen-acls, no-auth]
 |     o- acls ................................................................ [ACLs: 0]
 |     o- luns ................................................................ [LUNs: 1]
 |     | o- lun0 ........................... [fileio/file_backend01 (/var/fileio01)]
 |     o- portals .............................................................. [Portals: 1]
 |       o- 0.0.0.0:3260 ....................................................... [OK]
 o- loopback ................................................................ [Targets: 0]
```

Notez au passage dans les commandes ci-dessus que la commande "**cd**" peut être omise.

3.26.8- Détruire un LUN

Ici, on appelle directement la commande "delete" dans le contexte cible, sans la précéder par "cd"...

```
/> /iscsi/iqn.2001-09.net.actilis:target0/tpg1/luns/ delete lun0
Deleted LUN 0.
```


Notes personnelles

3.26.9- Sécurité

Par défaut, l'authentification est activée.
On peut définir des "credentials" pour restreindre l'accès aux cibles.

```
/iscsi/iqn.20...:target0/tpg1> set attribute authentication=1
/iscsi/iqn.20...:target0/tpg1> acls/ create iqn.2001-09.net.actilis:client
/iscsi/iqn.20...:target0/tpg1> cd acls/iqn.2001-09.net.actilis:client/
/iscsi/iqn.20...t.actilis:client> set auth userid=iqn.2001-09.net.actilis:client
Parameter userid is now 'iqn.2001-09.net.actilis:client'.
/iscsi/iqn.20...t.actilis:client> set auth password=secret
Parameter password is now 'secret'.

/iscsi/iqn.20...t.actilis:client> / saveconfig
```

Ce client devra alors présenter ces informations pour se connecter.
Tout client qui ne dispose pas d'ACL est interdit d'accès lorsque l'attribut **authentication** vaut 1.

On peut désactiver l'authentification pour autoriser tout hôte à se connecter à une ressource

```
/> cd /iscsi/iqn.2001-09.net.actilis:target0/tpg1/
/iscsi/iqn.20...:target0/tpg1> set attribute authentication=0 generate_node_acls=1
Parameter authentication is now '0'.
Parameter generate_node_acls is now '1'.
/iscsi/iqn.20...:target0/tpg1> / saveconfig
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json
```


Notes personnelles

3.26.10- Côté client

Installer le pachage **iscsi-initiator-utils** (Debian : open-iscsi) et configurer le WWN de la machine

```
# yum -y install iscsi-initiator-utils
# echo InitiatorName=iqn.2001-09.net.actilis:client > /etc/iscsi/initiatorname.iscsi
```

Connexion au portail pour découverte des cibles :

```
# iscsiadm --mode discovery --type sendtargets --portal 172.17.1.174
172.17.1.174:3260,1 iqn.2001-09.net.actilis:target0
```

Connexion aux cibles découvertes :

```
# iscsiadm -m node -L automatic
Logging in to [iface: default, target: iqn.2001-09.net.actilis:target0, portal: 172.17.1.174,3260]
(multiple)
Login to [iface: default, target: iqn.2001-09.net.actilis:target0, portal: 172.17.1.174,3260]
successful.
```

Visualisation :

```
# cat /proc/scsi/scsi
Host: scsi6 Channel: 00 Id: 00 Lun: 00
 Vendor: LIO-ORG  Model: file_backend01   Rev: 4.0
 Type:   Direct-Access                    ANSI  SCSI revision: 05
```

Au boot, prévoir le démarrage automatique du service "iscsid' et "iscsi"

Notes personnelles

3.27- Meta-devices : le RAID logiciel sous Linux

3.27.1- Notions générales sur la technologie RAID

RAID consiste à agréger plusieurs périphériques pour n'en faire qu'un seul disposant d'une sécurité renforcée, et / ou de meilleures performances qu'un périphérique seul.

RAID 0 : Data Stripping
Les données sont réparties => pas de sécurité, hautes performances (>= 2 disques).
Un disque est perdu, tout est perdu.

RAID 1 : **Miroir**
Duplication des écritures (RAID 1) => sécurité et performances en lecture (=2 disques)
Un disque est perdu, les données sont sauvées !

RAID 2, 3, 4, 5, 6 : **Contrôle de parité**
Ces modes offrent de la sécurité à moindre coût que le mirroring (>= 3 disques)

On peut créer un meta-disque basé sur d'autres meta-disques :
RAID 10 = 0 par dessus 1, **RAID 50** = 0 par dessus 5.
RAID 100 = 0 par dessus 0 par dessus 1

(Wikipedia) tous les niveaux ici : http://en.wikipedia.org/wiki/Standard_RAID_levels
(Wikipedia) tous les niveaux composés ici : http://en.wikipedia.org/wiki/Nested_RAID_levels

Notes personnelles

3.27.2- Les méta-disques sous Linux

Les **méta-disques** sont des périphériques s'appuyant sur d'autres périphériques.
Ils peuvent reposer sur des partitions de disques IDE, SCSI, FC... sur d'autres méta-disques
Il s'agit de périphériques de type bloc (comme les partitions de disques),

Ils sont appelés **/dev/md*** (**/dev/md[0-7]** : 8 devices raid possibles (Majeur = 9))

3.27.3- Commandes et fichiers

Les méta-disques sont administrés par la commande **mdadm** (paquetage « mdadm »). On peut trouver de l'information sommaire dans le « fichier texte » **/proc/mdstat**.

3.27.4- Les méta-disques et mdadm

La commande **mdadm** permet de manipuler les méta-disques gérés par le noyau

On peut agrandir un périphérique (Raid 1, 4, 5, et 6).

mdadm offre aussi des fonctions de monitoring et d'analyse des périphériques raid.

On appuie des méta-disques sur des partitions de type "**0xfd**".

Cela n'est pas strictement nécessaire pour créer un méta-disque, mais facilite le ré assemblage automatique de celui-ci par le noyau.

À défaut, il faudra spécifier les éléments à ré-assembler dans un fichier de configuration (**mdadm.conf**)

Notes personnelles

3.28- Utilisation de mdadm

3.28.1- Éléments de mise en œuvre

La commande "**mdadm**" livre son mode d'emploi assez facilement :

```
[0:fmicaux@tupai(10:39:12)~]$ mdadm --help
mdadm is used for building, managing, and monitoring
Linux md devices (aka RAID arrays)
Usage: mdadm --create device options...
            Create a new array from unused devices.
       mdadm --assemble device options...
            Assemble a previously created array.
       mdadm --build device options...
            Create or assemble an array without metadata.
       mdadm --manage device options...
            make changes to an existing array.
…
```

Chaque mode (create, assemble, ...) dispose aussi de sa propre page d'aide

```
For detailed help on the above major modes use --help after the mode
         mdadm --assemble --help
 For general help on options use
         mdadm --help-options
```


Notes personnelles

3.28.2- Créer, agrandir, supprimer, assembler un méta-disque

Initialisation du support, si nécessaire

On commence par initialiser les partitions ayant été utilisées auparavant pour d'autres choses :

```
# mdadm --zero-superblock /dev/xxx
```

Le mode create

```
maupiti:~# mdadm --create --help
Usage:  mdadm --create device -chunk=X --level=Y --raid-devices=Z devices
...
Options that are valid with --create (-C) are:
  --chunk=      -c   : chunk size of kibibytes
  --rounding=        : rounding factor for linear array (==chunk size)
  --level=      -l   : raid level: 0,1,4,5,6,linear,multipath and synonyms
  --parity=     -p   : raid5/6 parity algorithm: {left,right}-{,a}symmetric
  --layout=          : same as --parity
  --raid-devices= -n : number of active devices in array
  --spare-devices= -x: number of spares (eXtras) devices in initial array
  --size=       -z   : Size (in K) of each drive in RAID1/4/5/6/10 - optional
  --force       -f   : Honour devices as listed on command line.  Don't
                     : insert a missing drive for RAID5.
  --run         -R   : insist of running the array even if not all
                     : devices are present or some look odd.
  --readonly    -o   : start the array readonly - not supported yet.
```


Notes personnelles

On peut créer un méta-disque RAID 1[24] auquel il manque un disque (on lui ajoutera plus tard)

```
# mdadm --create /dev/md0 -l 1 -n 2 /dev/sdb1 missing
mdadm: array /dev/md0 started.
```

Le mode assemble

Le mode automatique réalise l'assemblage identique à celui fait par le noyau au démarrage.:

```
# mdadm --assemble --scan
```

Lorsqu'un méta-disque a été arrêté et que l'on ne parvient pas à l'assembler automatiquement, on peut le ré-assembler manuellement. Spécifier le nom du méta-disque à assembler et ses composants.

```
# mdadm --assemble /dev/md2 /dev/md0 /dev/md1
```

Supprimer un méta-disque

Pour supprimer un méta-disque, il faut d'abord l'arrêter :

```
# mdadm --stop /dev/md0
```

Puis ensuite le supprimer

```
# mdadm --remove /dev/md0
```

24 Le noyau Linux implémente les niveaux 0, 1, 4, 5, 6 et 10.

Notes personnelles

3.28.3- S'informer sur les méta-disques

On peut scanner les disques pour retrouver des informations sur les périphériques :

```
[root@tupai ~]# mdadm --examine --scan # presque identique à --detail --scan
ARRAY /dev/md0 UUID=46124282:899cbc10:d422ec88:84219cf9
ARRAY /dev/md1 UUID=76ebf2f6:d64a5ac0:d4953c6c:7ce1507a
ARRAY /dev/md2 UUID=921aceac:be85e835:a6b0f58e:dfca74b6
```

On peut s'intéresser à un de ces méta-disques :

```
[root@tupai ~]# mdadm --query /dev/md0
/dev/md0: 13.98GiB raid1 2 devices, 0 spares.
```

Avec un peu plus de détails...

```
[root@tupai ~]# mdadm --detail /dev/md0
/dev/md0:
        Version : 0.90
  Creation Time : Wed Dec 24 22:18:31 2008
     Raid Level : raid1
     Array Size : 14659200 (13.98 GiB 15.01 GB)
  Used Dev Size : 14659200 (13.98 GiB 15.01 GB)
   Raid Devices : 2
  Total Devices : 2
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Mon May 10 17:49:54 2010
          State : clean
 Active Devices : 2
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 0

           UUID : 46124282:899cbc10:d422ec88:84219cf9
```


Notes personnelles

```
         Events : 0.857087

    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       1       8        1        1      active sync   /dev/sda1
```

On obtient (presque) le même résultat en partant d'un composant d'un méta-disque :

```
[root@tupai ~]# mdadm --examine /dev/sda1
```


Notes personnelles

3.28.4- Maintenance des méta-disques : le mode manage

Lorsque l'on cite un nom de méta-disque en premier argument, on entre en mode **manage**. On peut aussi utiliser "**mdadm --manage /dev/le-meta-disque**", mais "--manage" est optionnel.

Il faut forcément spécifier une action :

--add : ajouter un composant
--remove : retirer un composant (celui-ci doit être défaillant)
--fail : mettre un composant à l'état défaillant
--readonly / **--readwrite** : positionne l'état ro/rw du composant

Déclarer un élément en panne

```
# mdadm /dev/md0 --fail /dev/sdb1
```

Une panne vient d'être signalée sur "sdb1".

```
[root@tupai ~]# mdadm --detail /dev/md0
…
          State : clean, degraded
…
      1       8        1        1      active sync   /dev/sda1
      2       8       17        -      faulty spare   /dev/sdb1
```


Notes personnelles

Retrait à chaud avec --remove

```
[root@tupai ~]# mdadm /dev/md0 --remove /dev/sdb1
mdadm: hot removed /dev/sdb1
```

Il manque désormais un composant :

```
[root@tupai ~]# mdadm --detail /dev/md0
/dev/md0:
…
   Raid Devices : 2
  Total Devices : 1
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Tue Sep  8 11:25:21 2009
          State : active, degraded
 Active Devices : 1
Working Devices : 1
 Failed Devices : 0
  Spare Devices : 0
…
    Number   Major   Minor   RaidDevice State
       0       0        0        0      removed
       1       8        1        1      active sync   /dev/sda1
```


Notes personnelles

Ajout à chaud avec --add

```
[root@tupai ~]# mdadm /dev/md0 --add /dev/sdb1
mdadm: re-added /dev/sdb1
```

Dès que le remplaçant est inséré, la reconstruction commence... et on peut consulter l'avancement de la reconstruction, qui commence immédiatement :

```
[root@tupai ~]# cat /proc/mdstat
Personalities : [raid1] [raid0]
md1 : active raid1 sdb6[1] sda6[0]
      120495424 blocks [2/2] [UU]

md2 : active raid0 sda7[0] sdb7[1]
      209680128 blocks 64k chunks

md0 : active raid1 sdb1[2] sda1[1]
      14659200 blocks [2/1] [_U]
      [>....................]  recovery =  0.8% (127232/14659200) finish=7.6min speed=31808K/sec

unused devices: <none>
```


Notes personnelles

Avec le mode détail...

```
[root@tupai ~]# mdadm --detail /dev/md0
/dev/md0:
…
    Update Time : Tue Sep  8 11:29:31 2009
          State : active, degraded, recovering
 Active Devices : 1
Working Devices : 2
 Failed Devices : 0
  Spare Devices : 1

 Rebuild Status : 9% complete

           UUID : 46124282:899cbc10:d422ec88:84219cf9
         Events : 0.1201

    Number   Major   Minor   RaidDevice State
       2       8       17        0      spare rebuilding   /dev/sdb1
       1       8        1        1      active sync   /dev/sda1
```

À la fin de la reconstruction, le périphérique passe de l'état Spare Rebuilding à l'état Active Sync.

Notes personnelles

3.28.5- Redimensionner un périphérique

C'est devenu possible depuis le noyau 2.6.17.

Nous disposons d'un méta-disque RAID 5 composé de 3 volumes actifs et synchrones. Ce méta-disque supporte un système de fichiers ext4 qui est monté.

```
[root@mururoa ~]# cat /proc/mdstat
Personalities : [raid1] [raid6] [raid5] [raid4]
md0 : active raid5 dm-3[2] dm-2[1] dm-1[0]
      2097024 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
```

Nous lui ajoutons 2 volumes, le portant à 5 au total :

```
[root@mururoa ~]# mdadm --add /dev/md0 /dev/datavg/lv3
mdadm: added /dev/datavg/lv3
[root@mururoa ~]# mdadm --add /dev/md0 /dev/datavg/lv4
mdadm: added /dev/datavg/lv4
```


Notes personnelles

On peut ensuite agrandir le méta-disque

```
[root@mururoa ~]# mdadm --grow /dev/md0 –raid-devices=5
mdadm: Need to backup 256K of critical section..
mdadm: ... critical section passed.
```

Il faut alors **attendre la fin de reconstruction** souvent un peu longue en RAID5.

```
[root@mururoa ~]# cat /proc/mdstat
Personalities : [raid1] [raid6] [raid5] [raid4]
md0 : active raid5 dm-5[3] dm-4[4] dm-3[2] dm-2[1] dm-1[0]
      2097024 blocks super 0.91 level 5, 64k chunk, algorithm 2 [5/5] [UUUUU]
      [=====>.............]  reshape = 27.2% (286408/1048512) finish=5.6min speed=2240K/sec
```


Notes personnelles

On note de nouvelles informations concernant cette action "reshape" :

```
[root@mururoa ~]# mdadm --detail /dev/md0
/dev/md0:
        Version : 0.91
  Creation Time : Tue Sep  8 11:48:40 2009
     Raid Level : raid5
     Array Size : 2097024 (2048.22 MiB 2147.35 MB)
  Used Dev Size : 1048512 (1024.11 MiB 1073.68 MB)
   Raid Devices : 5
  Total Devices : 5
Preferred Minor : 0
    Persistence : Superblock is persistent

    Update Time : Tue Sep  8 12:04:00 2009
          State : clean, recovering
 Active Devices : 5
Working Devices : 5
 Failed Devices : 0
  Spare Devices : 0

         Layout : left-symmetric
     Chunk Size : 64K

 Reshape Status : 45% complete
  Delta Devices : 2, (3->5)
…
```

Lorsque c'est terminé...

```
[root@mururoa ~]# resize2fs /dev/md0
resize2fs 1.41.4 (27-Jan-2009)
Le système de fichiers de /dev/md0 est monté sur /raid ; le changement de taille doit être effectué
en ligne
old desc_blocks = 1, new_desc_blocks = 1
```


Notes personnelles

```
En train d'effectuer un changement de taille en ligne de /dev/md0 vers 1048512 (4k) blocs.
Le système de fichiers /dev/md0 a maintenant une taille de 1048512 blocs.

[root@mururoa ~]# df /raid
Sys. de fich.         1K-blocs       Occupé Disponible Capacité Monté sur
/dev/md0                4128192     36856   3881668   1% /raid
```


Notes personnelles

4- Noyau et périphériques

Notes personnelles

4.1- Lister les périphériques vus par le noyau

4.1.1- Détection manuelle du matériel : lspci et lsusb

La commande **lspci** permet de lister les périphériques connectés au système.

```
# lspci
...
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:02.0 VGA compatible controller: InnoTek Systemberatung GmbH VirtualBox Graphics Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)
...
00:0d.0 SATA controller: Intel Corporation 82801HBM/HEM (ICH8M/ICH8M-E) SATA AHCI Controller (rev
02)
```

ADDR CLASS VENDOR DEVICE

```
# lspci -n
...
00:01.1   0101:   8086:7010
00:02.0   0300:   80ee:beef
00:03.0   0200:   8086:100e (rev 02)
...
00:0d.0   0106:   8086:2829 (rev 02)
```

La signature PCI du périphérique d'adresse **00:03.0** indique : Classe **0200**, Constructeur **8086**, Périphérique n°**100e**.

Notes personnelles

Un complément d'information peut être donné (option **-v**)...

```
00:03.0 0200: 8086:100e (rev 02)
        Subsystem: 8086:001e
        Flags: bus master, 66MHz, medium devsel, latency 64, IRQ 11
        Memory at f0000000 (32-bit, non-prefetchable) [size=128K]
        I/O ports at c010 [size=8]
        Capabilities: [dc] Power Management version 2
        Capabilities: [e4] PCI-X non-bridge device
```

L'option "**-t**" donne une indication arborescente, qui est intéressante mais n'apprend pas grand chose :

```
$ lspci -tvn
-[0000:00]-+-00.0  8086:3340
           +-01.0-[0000:01]----00.0  1002:4e50
           +-1d.0  8086:24c2
           +-1d.1  8086:24c4
           +-1d.2  8086:24c7
           +-1d.7  8086:24cd
           +-1e.0-[0000:02-0a]--+-00.0  14e4:169c
           |                    +-01.0  1180:0476
           |                    +-01.1  1180:0476
           |                    +-01.2  1180:0552
           |                    \-02.0  8086:4220
           +-1f.0  8086:24cc
           +-1f.1  8086:24ca
           +-1f.3  8086:24c3
           +-1f.5  8086:24c5
           \-1f.6  8086:24c6
```


Notes personnelles

La commande **lsusb** est un peu équivalente à lspci.

```
# lsusb
Bus 003 Device 003: ID 046d:c00e Logitech, Inc. M-BJ58/M-BJ69 Optical Wheel Mouse
Bus 002 Device 004: ID 12d1:1003 Huawei Technologies Co., Ltd. E220 HSDPA Modem / E270 HSDPA/HSUPA
Modem
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Avec les options "-v" (verbose mode) et "-s" (spécifier un périphérique (bus:device))

```
# lsusb -s 002:004 -v |head -n 20
Bus 002 Device 004: ID 12d1:1003 Huawei Technologies Co., Ltd. E220 HSDPA Modem / E270 HSDPA/HSUPA
Modem
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               1.10
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x12d1 Huawei Technologies Co., Ltd.
  idProduct          0x1003 E220 HSDPA Modem / E270 HSDPA/HSUPA Modem
  bcdDevice            0.00
  iManufacturer           1 HUAWEI Technologies
  iProduct                2 HUAWEI Mobile
  iSerial                 0
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2
```


Notes personnelles

4.2- Les identifiants pci et les identifiants usb

Les fichiers **pci.ids** et **usb.ids** sont dans **/usr/share/hwdata**, ou /usr/share/misc sous Debian.

Les versions "à jour" sont sur http://pciids.sourceforge.net/ et sur http://www.linux-usb.org/.

La commande **update-pciids** met à jour la version locale de **pci.ids** à partir de celle disponible sur sourceforge.

Ces fichiers ne servent que de base de données de résolution de noms pour permettre aux commandes **lspci** et **lsusb** d'afficher les noms des fabriquants et des périphériques au lieu de leurs identifiants pci.

pci.ids

Le début du fichier est consacré aux constructeurs et périphériques

```
# Syntax:
# vendor  vendor_name
#       device  device_name                             <-- single tab
#               subvendor subdevice  subsystem_name     <-- two tabs
# <pci.ids sed -n '/^8086/,$p' | egrep -i '^8086|^.*7010|^.*100e|^.*2829'
8086  Intel Corporation
        100e  82540EM Gigabit Ethernet Controller
        2829  82801HBM/HEM (ICH8M/ICH8M-E) SATA AHCI Controller
        7010  82371SB PIIX3 IDE [Natoma/Triton II
```


Notes personnelles

La fin de ce fichier est consacrée aux classes de périphériques.

```
# List of known device classes, subclasses and programming interfaces
# Syntax:
# C class       class_name
#       subclass        subclass_name           <-- single tab
#               prog-if  prog-if_name   <-- two tabs
...
C 01  Mass storage controller
        00  SCSI storage controller
        01  IDE interface
        02  Floppy disk controller
        03  IPI bus controller
        04  RAID bus controller
        05  ATA controller
                20  ADMA single stepping
                30  ADMA continuous operation
        06  SATA controller
                00  Vendor specific
                01  AHCI 1.0
        07  Serial Attached SCSI controller
        80  Mass storage controller
C 02  Network controller
        00  Ethernet controller
        01  Token ring network controller
        02  FDDI network controller
        03  ATM network controller
        04  ISDN controller
        05  WorldFip controller
        06  PICMG controller
        80  Network controller
...
```


Notes personnelles

usb.ids :
C'est la même approche, avec une sémantique ressemblante.

```
# Vendors, devices and interfaces. Please keep sorted.

# Syntax:
# vendor  vendor_name
#       device  device_name                           <-- single tab
#               interface  interface_name             <-- two tabs
...
046d  Logitech, Inc.
...
...
        c00e  M-BJ58/M-BJ69 Optical Wheel Mouse...
...
12d1  Huawei Technologies Co., Ltd.
        1001  E620 USB Modem
        1003  E220 HSDPA Modem / E270 HSDPA/HSUPA Modem
...
```

Autres fichiers présent dans "hwdata"
Dans ce répertoire, on trouve aussi une correspondance entre les noms des drivers vidéo et les types de cartes auxquelles ils se rapportent, ainsi qu'une base de données des moniteurs connus.

On trouve aussi un fichier "**upgradelist**" permet de retrouver facilement le nouveau nom d'un module lorsque l'un d'entre-eux est renommé.

Notes personnelles

4.3- Déterminer le module nécessaire à un périphérique

Les modules du noyau sont fournis avec des "maps", qui permettent de trouver facilement l'information.

Dans le répertoire **/lib/modules/$(uname -r)/** , on trouvait auparavant les fichiers "**modules.pcimap**", "**modules.usbmap**".

La première colonne de ces fichiers donne le nom d'un module à charger.
Il ne reste plus qu'à trouver les informations que nous connaissons (Vendor, Device, ...) fournies par les commandes **lspci** et **lsusb**.

modules.pcimap

```
# pci module    vendor   device   subvendor  subdevice  class    class_mask driver_data
```

modules.usbmap

```
# usb module         match_flags idVendor idProduct bcdDevice_lo bcdDevice_hi bDeviceClass
bDeviceSubClass bDeviceProtocol bInterfaceClass bInterfaceSubClass bInterfaceProtocol driver_info
```

modules.alias

Depuis les noyaux 3.X... donc avec les versions récentes de "pciutils", c'est dans le fichier **modules.alias** qu'il faut chercher.

Notes personnelles

Exemples :

```
# grep 8086.*100e modules.pcimap
e1000                0x00008086 0x0000100e 0xffffffff 0xffffffff 0x00000000 0x00000000 0x0
# grep 8086.*7010 modules.pcimap
ata_piix             0x00008086 0x00007010 0xffffffff 0xffffffff 0x00000000 0x00000000 0x0
# grep 8086.*2829 modules.pcimap
ahci                 0x00008086 0x00002829 0xffffffff 0xffffffff 0x00000000 0x00000000 0x0
```

Pour le périphérique USB (12d1 / 1003) :

```
# grep 12d1.*1003 modules.usbmap
option               0x0383      0x12d1   0x1003    0x0000       0x0000       0x00         0x00
0x00            0xff            0xff               0xff               0x0
```

Avec "**modules.alias**"... attention aux majuscules...

```
# grep -i 10ec.*8168 modules.alias
alias pci:v000010ECd00008168sv*sd*bc*sc*i* r8169
# grep -i 12d1.*1003 modules.alias
alias usb:v12D1p1003d0000dc*dsc*dp*ic*isc*ip*in* usb_storage
```


Notes personnelles

4.4- Manipuler les modules

4.4.1- Recherche / listage des modules

La commande **modinfo** donne des détails intéressants sur les modules installés, notamment une petite description qui laisse penser qu'on a trouvé le bon module, mais aussi une indication "alias" qui le confirme !

```
[root@maupiti ~]# modinfo  option
filename:       /lib/modules/2.6.27.4/kernel/drivers/usb/serial/option.ko
license:        GPL
version:        v0.7.2
description:    USB Driver for GSM modems
author:         Matthias Urlichs <smurf@smurf.noris.de>
srcversion:     F62C238B896E689B3BF7996
alias:          usb:v19D2pFFFEd*dc*dsc*dp*ic*isc*ip*
alias:          usb:v19D2p0015d*dc*dsc*dp*ic*isc*ip*
...
alias:          usb:v12D1p1403d*dc*dsc*dp*icFFiscFFipFF*
alias:          usb:v12D1p1401d*dc*dsc*dp*icFFiscFFipFF*
alias:          usb:v12D1p1004d*dc*dsc*dp*icFFiscFFipFF*
alias:          usb:v12D1p1003d*dc*dsc*dp*icFFiscFFipFF*
alias:          usb:v12D1p1001d*dc*dsc*dp*icFFiscFFipFF*
alias:          usb:v0AF0p7111d*dc*dsc*dp*ic*isc*ip*
alias:          usb:v0AF0p7100d*dc*dsc*dp*ic*isc*ip*
...
depends:        usbserial
vermagic:       2.6.27.4 mod_unload PENTIUMM
parm:           debug:Debug messages (bool)
```

4.4.2- Chargement d’un module

On peut ajouter ou retirer à la demande des pilotes du noyau sans avoir à redémarrer le système.

Il suffit pour cela de charger ou décharger un module.

Notes personnelles

On peut charger des modules de 3 manières différentes :

Au démarrage, par un script de démarrage prévu pour cela

À la demande, mais automatiquement, un thread du noyau réalise ce genre de choses.

Manuellement par **insmod** ou **modprobe**.

La commande **insmod** échoue si le module nécessite un autre module qui n'est pas chargé
Elle peut nécessiter aussi de connaître les paramètres du périphérique (IO, IRQ)

La commande **modprobe** charge automatiquement les modules nécessaires (résout les dépendances).

Elle « *probe* » automatiquement les périphériques dans la majorité des cas.

L'option « **-k** » permet le chargement avec le mode « autoclean »,
⇒ offre la possibilité de le décharger facilement en traitement par lot.

Notes personnelles

4.4.3- S'informer sur un module

La commande **modinfo** permet de visualiser différents éléments sur un module :
son emplacement, sa description succincte, son auteur,
les périphériques qu'il prend en charge,
ses options, etc.

4.4.4- Liste des modules chargés

Par la commande **lsmod**.
La liste affiche le nombre d'utilisations des modules
La liste affiche aussi les dépendances et paramètres du type « autoclean »

À voir : le "fichier" /proc/modules, une autre manière de lister les modules chargés.

4.4.5- Retrait d'un module

Par la commande **rmmod**.
La commande « **rmmod -a** » décharge les modules inutilisés chargés en « autoclean ».

4.4.6- Le répertoire /etc/modprobe.d

Contient les options de certains modules pour chargement automatique par kmod.
On peut spécifier des blacklists pour ignorer certains modules si plusieurs peuvent gérer un équipement.

Notes personnelles

4.4.7- Où sont les modules ?

Ils sont stockés dans une arborescence spécifique se trouvant dans **/lib/modules/$(uname -r)**.

```
[0:root@srv-formation 20:56:59 ~]# ls -l /lib/modules/$(uname -r)
total 3528
lrwxrwxrwx.  1 root root     50  3 avril  2014 build -> ../../../usr
drwxr-xr-x.  2 root root   4096 12 févr.  2014 extra
drwxr-xr-x. 11 root root   4096  3 avril  2014 kernel
-rw-r--r--.  1 root root 589869  3 avril  2014 modules.alias
-rw-r--r--.  1 root root 565564  3 avril  2014 modules.alias.bin
-rw-r--r--.  1 root root   1413 12 févr.  2014 modules.block
-rw-r--r--.  1 root root     69  3 avril  2014 modules.ccwmap
-rw-r--r--.  1 root root 200373  3 avril  2014 modules.dep
-rw-r--r--.  1 root root 291812  3 avril  2014 modules.dep.bin
-rw-r--r--.  1 root root     68 12 févr.  2014 modules.drm
-rw-r--r--.  1 root root    665  3 avril  2014 modules.ieee1394map
-rw-r--r--.  1 root root    141  3 avril  2014 modules.inputmap
-rw-r--r--.  1 root root   1236  3 avril  2014 modules.isapnpmap
-rw-r--r--.  1 root root     29 12 févr.  2014 modules.modesetting
-rw-r--r--.  1 root root   1956 12 févr.  2014 modules.networking
-rw-r--r--.  1 root root     74  3 avril  2014 modules.ofmap
-rw-r--r--.  1 root root  75698 12 févr.  2014 modules.order
-rw-r--r--.  1 root root 436535  3 avril  2014 modules.pcimap
-rw-r--r--.  1 root root   6259  3 avril  2014 modules.seriomap
-rw-r--r--.  1 root root 228286  3 avril  2014 modules.symbols
-rw-r--r--.  1 root root 289146  3 avril  2014 modules.symbols.bin
-rw-r--r--.  1 root root 851070  3 avril  2014 modules.usbmap
lrwxrwxrwx.  1 root root      5  3 avril  2014 source -> build
drwxr-xr-x.  2 root root   4096 12 févr.  2014 updates
drwxr-xr-x.  2 root root   4096  3 avril  2014 vdso
drwxr-xr-x.  2 root root   4096 12 févr.  2014 weak-updates
```

Les modules sont nommés **nom_module.ko** (ou nom_module.ko.gz).

Ils sont presque tous dans le répertoire "kernel".

D'une manière générale, un module est prévu pour une version N du noyau, et il ne fonctionnera pas dans une version N+1, sauf à être recompilé.

Les fichiers ".pcimap", ".usbmap", ".alias" sont intéressants pour déterminer quel module pilote quel périphérique.

Notes personnelles

4.5- Manipuler le contenu de /dev

Les périphériques sont en principe accédés au travers de fichiers spéciaux présents dans le répertoire "*/dev*".

Historiquement, */dev* contenait un grand nombre d'entrées, que les périphériques soient présents ou pas dans le système.

4.5.1- Fichiers spéciaux, majeur, mineur

```
# ls -l /dev/[sh]d[a-d]?
brw-r----- 1 root disk 3,  1 Nov 16 16:01 /dev/hda1
brw-r----- 1 root disk 3,  2 Nov 16 16:01 /dev/hda2
brw-r----- 1 root disk 3, 65 Nov 16 16:01 /dev/hdb1
brw-r----- 1 root disk 3, 66 Nov 16 16:01 /dev/hdb2
brw-r----- 1 root disk 8,  1 Nov 16 16:02 /dev/sda1
# ls -l /dev |grep ^c |head
crw------- 1 root root     5,   1 Nov 16 16:37 console
crw-rw-rw- 1 root root     1,   7 Nov 16 16:01 full
crw------- 1 root root    10, 228 Nov 16 16:01 hpet
crw------- 1 root root     1,  11 Nov 16 16:01 kmsg
crw-r----- 1 root kmem     1,   1 Nov 16 16:01 mem
crw-rw-rw- 1 root root     1,   3 Nov 16 16:01 null
crw-rw---- 1 root root    10, 144 Nov 16 16:01 nvram
crw------- 1 root root     1,  12 Nov 16 16:01 oldmem
crw-rw---- 1 root lp      99,   0 Nov 16 16:01 parport0
crw-rw---- 1 root lp      99,   1 Nov 16 16:01 parport1
```


Notes personnelles

Les fichiers spéciaux

Chaque entrée du répertoire /dev est de type "**b**" (block device) ou "**c**" (character device).

On y trouve éventuellement des liens symboliques (floppy, dvd, cd ...) ou sous-répertoires (/dev/pts, /dev/mon_volume_group, ...) pointant à leur tour des fichiers spéciaux de type "b" ou "c".

```
# ls -l /dev/vg_rootfs/
total 0
lrwxrwxrwx 1 root root 31 Nov 16 16:01 lv_rootfs -> /dev/mapper/vg_rootfs-lv_rootfs
# ls -lL /dev/vg_rootfs/
total 0
brw-rw---- 1 root disk 253, 0 Nov 16 16:02 lv_rootfs
```

Majeur, Mineur

À la place de l'indication de taille de ces fichiers sont donnés deux éléments séparés par une virgule : **le majeur et le mineur.** Ce sont des "adresses" faisant l'interface entre le noyau et l'administrateur.

Les numéros de majeur et mineur sont normalisés (http://www.lanana.org/docs/device-list/ ou ftp://ftp.kernel.org/pub/linux/docs/device-list/).

Le fichier /proc/devices

On trouve dans **/proc** un fichier "devices", dont le contenu liste uniquement les majeurs des périphériques que PEUT gérer l'ensemble noyau actuel + modules chargés.

Notes personnelles

4.5.2- Création des fichiers spéciaux

La commande **mknod** permet de créer des fichiers spéciaux :

```
mknod nom Mode Majeur Mineur
```

La commande /sbin/MAKEDEV

Cette commande crée des groupes d'entrée dans */dev*, et elle n'est qu'une interface à **mknod**.

Création d'un fichier spécial

Il faut connaître majeur et le mineur associés au périphérique afin d'être en phase avec le driver !

Exemple de ttyUSB0 et ttyUSB1 : rechercher "ttyUSB" dans "devices.txt"

```
188 char        USB serial converters
                  0 = /dev/ttyUSB0      First USB serial converter
                  1 = /dev/ttyUSB1      Second USB serial converter
                    ...

189 char        USB serial converters - alternate devices
                  0 = /dev/cuusb0       Callout device for ttyUSB0
                  1 = /dev/cuusb1       Callout device for ttyUSB1
                    ...
```

Ces périphériques semblent être connus :

- fichier spécial de type Character,
- majeur 188, (et si besoin 189)
- mineur 0 et mineur 1.

```
# mknod -m 770 ttyUSB0 c 188 0
# mknod -m 770 ttyUSB1 c 188 1
```


Notes personnelles

4.6- Gestion dynamique des périphériques : Udev

Udev est un système de gestion dynamique de /dev.

Il permet de limiter le contenu de ***/dev*** aux noms des périphériques physiquement présents.

Le système **udev** peut, à priori, être ignoré des utilisateurs puisqu'il est destiné à leur rendre la vie plus facile et à fonctionner de manière tout à fait transparente.

URL : http://kernel.org/pub/linux/utils/kernel/hotplug/

4.6.1- Le daemon udevd

Averti par le noyau de l'ajout ou du retrait d'un périphérique, le démon **udevd** agit en conséquence pour maintenir à jour le répertoire ***/dev***.

Il est normalement démarré par le script **/etc/rc.sysinit** (ou /etc/init.d/udev**)**.

Le démon **udevd** utilise des règles (configurées dans le répertoire **/etc/udev/rules.d**) pour décider des actions à réaliser pour la création effective du fichier */dev/xxx*.

Udev trouve les autres informations dont il a besoin dans l'arborescence **/sys**.

Notes personnelles

4.6.1.1- Fichier de configuration principal : /etc/udev/udev.conf

Le fichier **udev.conf** contient la configuration de base du système udev.

Il contient des définitions de variables : variable=valeur.

Une variable peut ne pas être définie s'il existe une valeur par défaut.

```
# cat /etc/udev/udev.conf
# The initial syslog(3) priority: "err", "info", "debug" or its
# numerical equivalent. For runtime debugging, the daemons internal
# state can be changed with: "udevcontrol log_priority=<value>".
udev_log="err"
```

Variable	Description	Valeur par défaut
udev_root	Le répertoire qui contient les noms des périphériques.	/dev
udev_rules	Le répertoire qui contient les fichiers de règles.	/etc/udev/rules.d
udev_log	Le niveau de log pour syslog : err, info ou debug.	

Notes personnelles

4.6.1.2- Les fichiers de règles

Ils définissent des règles à appliquer à la création de certains fichiers périphériques. Chaque ligne contient au moins un couple "clé/valeur". L'ensemble des clés d'une ligne définit une règle.

Il existe deux types de clés : les **clés comparatives** et les **clés de définition**.

Si toutes les clés de comparaison d'une règle sont satisfaites alors les actions des clés de définition sont exécutées pour le périphérique.

Les clés comparatives utilisent les opérateurs pour exprimer des conditions :

- **==** : pour tester l'égalité.
- **!=** : pour tester la différence.

Les clés de définition utilisent les opérateurs pour définir des actions :

- **=** : pour affecter une valeur à une clé.
  - La clé est réinitialisée et la(es) ancienne(s) valeur(s) de la clé est(sont) remplacée(s).
- **:=** : pour affecter une valeur à une clé et en interdire la modification par d'autres règles.
- **+=** : pour ajouter une valeur à une clé.
  - Une clé peut contenir une liste de valeurs.

Notes personnelles

Quelques clés Comparatives

ACTION	Définit l'évènement déclencheur.
DEVPATH	Définit le chemin d'accès au périphérique dans sysfs..
ENV{Variable}	La variable d'environnement Variable.
ATTR{V}	L'attribut V lu dans les attributs du périphérique.
KERNEL	Définit le nom du périphérique attribué par le noyau.
SUBSYSTEM	M le type de périphérique (block, character, usb...)
DRIVER	Définit le nom du pilote (« driver ») du périphérique.

Quelques clés de définition

PROGRAM	Nom d'un programme externe à exécuter.
NAME	Définit le nom du fichier périphérique à créer dans /dev.
RUN	Nom d'un programme supplémentaire à exécuter.
SYMLINK	Nom(s) des liens symboliques à créer pour ce périphérique.
OWNER, **GROUP**, **MODE**	Le propriétaire, le groupe et les droits du périphérique.

Notes personnelles

Exemples de règle

Le fichier qui suit définit une règle pour un disque dur.

L'utilisation des jokers dans les valeurs.

On peut utiliser les jokers connus du shell : *, ? et [] pour donner un modèle de chaîne de caractères :

- [sh]d[a-z] signifie sda à sdz ou hda à hdz.
- ?* signifie au moins un caractère.

Si on détecte l'ajout (« add »), d'un périphérique est de type bloc (« block ») dont le nom est hdx ou sdx avec x de a à z, il faut exécuter le programme **/lib/udev/hdparm**.

```
ACTION=="add", SUBSYSTEM=="block", KERNEL=="[sh]d[a-z]", RUN+="/lib/udev/hdparm"
```

Ici, c'est en fonction de l'adresse MAC qu'on décide du nom de l'interface réseau

```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="08:00:27:c2:82:10", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth0"
# PCI device 0x1af4:0x1000 (virtio-pci) (custom name provided by external tool)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="52:54:00:00:05:fe", ATTR{type}=="1",
KERNEL=="eth*", NAME="eth1"
```


Notes personnelles

Quelques variables prédéfinies utiles dans les règles.

On peut utiliser dans une valeur de clé une information détenue par udev grâce à un ensemble de variables exprimées sous la forme **%v** ou **$variable**.

%k, $kernel	Le nom noyau du périphérique.
%n, $number	Le numéro du périphérique, ainsi %n vaut 3 pour sda3.
$driver	Le nom du pilote (« driver ») du périphérique.
%M	Le majeur du périphérique.
%m	Le mineur du périphérique.
%E{variable},	La valeur de la variable d'environnement « variable ».
$e{variable}	

Notes personnelles

4.6.2- Les commandes et outils liés à udev

La commande **udevadm** regroupe un certain nombre d'actions.

Syntaxe :

udevadm commande [options] [arguments]
ou (**udevcommande**) dans les anciennes versions.

La commande indique quelle action **udevadm** doit exécuter.

Les actions sont :

udevadm **info** [options]
udevadm **trigger** [options]
udevadm **settle** [options]
udevadm **control** [options] instruction
udevadm **monitor** [options]
udevadm **test** [options] devpath
udevadm version
udevadm help

Les options des commandes sont de la forme **--option=valeur**

Notes personnelles

La commande udevadm info (ou udevinfo)

L'exemple qui suit affiche **toutes** les information du disque sda .

```
# udevadm info     --query=all     --name=sda
P: /block/sda
N: sda
S: disk/by-id/scsi-SATA_VBOX_HARDDISK_VB695a397c-96044c93
S: disk/by-path/pci-0000:00:0d.0-scsi-0:0:0:0
E: ID_VENDOR=ATA
E: ID_MODEL=VBOX_HARDDISK
E: ID_REVISION=1.0
E: ID_SERIAL=SATA_VBOX_HARDDISK_VB695a397c-96044c93
E: ID_TYPE=disk
E: ID_BUS=scsi
E: ID_PATH=pci-0000:00:0d.0-scsi-0:0:0:0
E: ID_FS_USAGE=raid
E: ID_FS_TYPE=LVM2_member
E: ID_FS_VERSION=LVM2
E: ID_FS_UUID=
E: ID_FS_LABEL=
E: ID_FS_LABEL_SAFE=
...
```

À voir : l'option **--atribute-walk** (l'option "-a") : listage de tous les attributs d'une entrée

```
# udevadm info -a -p /sys/block/sda
```


Notes personnelles

La commande udevadm monitor

Exemple d'ajout d'un disque USB (contenant 2 partitions) :

```
# udevadm monitor
udevmonitor prints the received event from the kernel [UEVENT]
and the event which udev sends out after rule processing [UDEV]

UEVENT[1226873208.648444] add@/devices/pci0000:00/0000:00:1d.7/usb1/1-2
UDEV  [1226873208.650135] add@/devices/pci0000:00/0000:00:1d.7/usb1/1-2
…
UEVENT[1226873213.706073] add@/block/sda
UEVENT[1226873213.706258] add@/block/sda/sda1
UEVENT[1226873213.706345] add@/block/sda/sda2
…
UDEV  [1226873214.266377] add@/block/sda
UDEV  [1226873214.712354] add@/block/sda/sda2
UDEV  [1226873215.667436] add@/block/sda/sda1
```

La dernière partie est intéressante : c'est bien udev qui a lancé la création de /dev/sda*

```
# ls -l /dev/sd*
brw-r----- 1 root disk 8, 0 nov 16 23:06 /dev/sda
brw-r----- 1 root disk 8, 1 nov 16 23:06 /dev/sda1
brw-r----- 1 root disk 8, 2 nov 16 23:06 /dev/sda2

# stat -c %n:%Z /dev/sda*
/dev/sda:1226873214
/dev/sda1:1226873215
/dev/sda2:1226873214
```

%Z : ctime

Notes personnelles

4.6.3- Écrire ses propres règles

Les noms des fichiers de règle sont de la forme ***nn-description.rules***. ils sont parcourus et exécutés par **udev** dans l'ordre des préfixes numériques **nn**.

Exemple : reconnaître une clé USB grâce à son LABEL quel que soit le nombre de périphériques usb déjà montés (donc le nom du périphérique... sda, sdb, sdc...).

```
# udevadm monitor
...
UDEV  [1226875020.200708] add@/block/sda
UDEV  [1226875020.245582] add@/class/scsi_device/26:0:0:0
UDEV  [1226875020.426675] add@/block/sda/sda1
^C
# ls -l /dev/sda*
brw-r----- 1 root disk 8, 0 nov 16 23:37 /dev/sda
brw-r----- 1 root disk 8, 1 nov 16 23:37 /dev/sda1
```

Pour construire la règle, observons les attributs d'une des cartes mémoire, une fois montée :

```
# udevadm info -q all -n sda1
P: /block/sda/sda1
N: sda1
S: disk/by-id/usb-Multi_Flash_Reader_058F0O1111B-part1
S: disk/by-path/pci-0000:00:1d.7-usb-0:2:1.0-scsi-0:0:0:0-part1
S: disk/by-label/EOS_DIGITAL
E: ID_VENDOR=Multi
E: ID_MODEL=Flash_Reader
E: ID_REVISION=1.00
E: ID_SERIAL=Multi_Flash_Reader_058F0O1111B
E: ID_TYPE=disk
E: ID_BUS=usb
E: ID_PATH=pci-0000:00:1d.7-usb-0:2:1.0-scsi-0:0:0:0
E: ID_FS_USAGE=filesystem
E: ID_FS_TYPE=vfat
```


Notes personnelles

```
E: ID_FS_VERSION=FAT32
E: ID_FS_UUID=
E: ID_FS_LABEL=EOS_DIGITAL
E: ID_FS_LABEL_SAFE=EOS_DIGITAL
```

La fois prochaine, ce périphérique s'appellera peut-être sdb1, et non plus sda1, mais son label restera EOS_DIGITAL, car c'est le label attribué au formatage de toutes ses cartes par l'appareil photo.

La règle :

```
# cat 99-montages.rules
ENV{ID_FS_LABEL}=="EOS_DIGITAL",SYMLINK+="camera",ENV{GENERATED}="1"
```


Notes personnelles

4.7- Qu'est-ce que le noyau standard

Le noyau standard a été installé par l'outil d'installation de la distribution.

Il est amorcé par le chargeur de démarrage, et est en principe stocké dans le répertoire /boot.

Le fichier noyau est « **vmlinuz-X.Y.Z-extensions** ». Il est compressé.

Il est fourni avec d'autres fichiers, dont le nom porte aussi le numéro de version.

```
# ls -l /boot/*$(uname -r)*
-rw-r--r--  1 root root   48299 jun  9 00:39 /boot/config-2.6.9-11.EL
-rw-r--r--  1 root root  981301 oct 14 18:59 /boot/initrd-2.6.9-11.EL.img
-rw-r--r--  1 root root  715755 jun  9 00:39 /boot/System.map-2.6.9-11.EL
-rw-r--r--  1 root root 1435513 jun  9 00:39 /boot/vmlinuz-2.6.9-11.EL
```

Premier constat : sur ce système installé le 14 octobre en fin d'après-midi, le fichier **initrd.img** date de l'installation et a été construit à la fin de celle-ci, alors que les autres sont issus du paquetage standard fournissant le noyau.

Le fichier **initrd.img**, dépendant de l'architecture de la machine, est construit sur place.

Notes personnelles

On limite habituellement le nombre de pilotes "built-in" (taille du noyau, utilisation mémoire), et on s'appuie sur des modules utilisés à la demande.

Pour que le noyau puisse « voir » la partition racine (/), il lui faut les pilotes pour :

le type de contrôleur disque (SCSI, SATA, UDMA, IDE, ..)

le type de table de partition (PC-bios...)

le type de système de fichiers utilisé par « / » (ext2/3, reiserfs, ...)

System.map : fait le lien entre les adresses mémoire des symboles du noyau et leurs noms.

On n'est pas obligé de le tenir à jour, sauf pour des raisons de performances. En effet, **klogd**, qui l'utilise, le cherchera, dans l'ordre, sous un des noms suivants :

/boot/System.map
/System.map
/usr/src/linux/System.map
il peut aussi chercher dans **/usr/src/linux-version/System.map**.

C'est forcément moins performant que s'il trouvait directement dans /boot/System.map.

Notes personnelles

4.8- Installation des sources du noyau

Installer le source du nouveau noyau par le paquetage
Utilisation d'un paquetage fourni par la distribution.

RH / CentOS : Il n'est plus fourni en standard, mais on peut le trouver sous forme d'un "SRPM".
Voir http://wiki.centos.org/HowTos/I_need_the_Kernel_Source

Debian : Le package est "**linux-source**" (Debian), et son contenu est installé dans le répertoire /usr/src/linuxsource-version.

Site officiel : kernel.org
Récupérer l'archive tar compressé.

On peut la stocker dans /tmp puis l'installer dans /usr/src

```
cd /usr/src
tar xjf /tmp/linux-x.y.tar.xz
```

Dans la suite de ce module, nous parlerons de **/usr/src/linux-version** comme chemin racine des sources du noyau.

Notes personnelles

4.9- Construction d'un nouveau noyau en 3 étapes

0 - Avant de commencer

La commande **make help** donne la liste des actions possibles

1 - Nettoyer les sources

Dans le répertoire **/usr/src/linux-version**, exécuter la commande suivante

```
make mrproper
```

2 - Configurer la compilation

On décide de ce que l'on va compiler ou pas (en statique, en module),
Le but est la génération du fichier « **.config** » à la racine des sources.

Plusieurs outils de configuration sont possibles :

```
make config   : mode texte ligne à ligne
make oldconfig       : mode automatique (MAJ .config pour nouvelles options)
make menuconfig      : mode menu texte (utilisant "ncurses")
make gconfig  : mode graphique X11 (nécessitte Tcl/Tk)
make xconfig  : mode graphique X11/QT (tree-list façon KDE...)
...
```

Cette étape nécessite une bonne connaissance du matériel qui compose le serveur.

Notes personnelles

Nous devons définir un noyau adapté au matériel dont nous disposons.

Il n'y a donc pas « *un .config qui marche pour tout le monde* »

Pour ne pas partir de zéro : après avoir effectué le « make mrproper », on peut partir du fichier /boot/config-version fourni avec le noyau standard.

Il doit être copié sous le nom « .config » dans le répertoire des sources du noyau.

Il existe d'autres manières de générer le .config ...

```
randconfig      - New config with random answer to all options
defconfig       - New config with default answer to all options
allmodconfig    - New config selecting modules when possible
allyesconfig    - New config where all options are accepted with yes
allnoconfig     - New minimal config
```


Notes personnelles

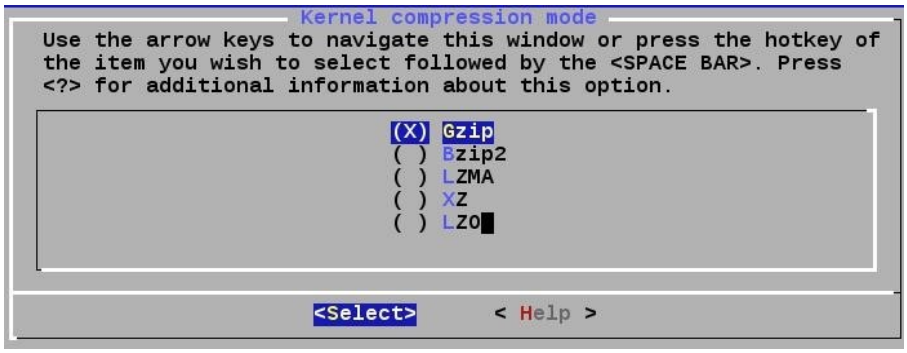
3 - Compiler le noyau

Le but est de générer le fichier « **vmlinux** », qui est l'image du noyau.

Le noyau devenant trop gros pour booter, on utilise depuis Linux 2.0 une image compressée (zImage).

Historiquement compressée par **gzip**, elle est devenue encore trop grosse pour booter (Linux 2.2).

On est donc passé à une "**bzImage**" (compressée initialement par **bzip2**... puis désormais **lzma**, **xz** ou **lzo)**.



Pour compiler :

```
make all              : pour construire le noyau et les modules
```

Il effectue make bzImage + make modules.

Le résultat

Le **noyau bootable** est **/usr/src/linux/arch/x86/boot/bzImage**

Les **modules** sont compilés, mais pas installés.

Notes personnelles

4.10- Installer le noyau et les modules

Installer les modules

Les modules seront installés dans **/lib/modules/VERSION**

```
make   modules_install
```

ou plutôt… (voir Makefile)

```
make   modules_install  INSTALL_MOD_STRIP=1
```

⇒ La taille des modules doit alors drastiquement s'abaisser.

Installer le noyau

C'est le fichier /usr/src/linux-version**/arch/x86/boot/bzImage.**

Il faut le copier dans **/boot** sous un nom comme **vmlinuz-x.y.z**...

La commande **make install** fait ce travail :

- copie le noyau,
- génère le ramdisk initial.
- ajoute une section dans Grub,

La cible "install" du Makefile suppose que les modules soient d'abord installés.

Notes personnelles

Forcer une compilation complète :

On peut forcer une recompilation « propre » en lançant, après génération du fichier « .config » et avant le « make all » la commande suivante :

```
make clean    : Nettoie les résidus d'anciennes compilations (garde le .config)
```

Pour faire « pro » :
On prendra la peine de supprimer le fichier « **.version** », c'est un compteur de compilations (...)

```
borabora:~# uname -rv
2.6.13.4 #9 Wed Oct 26 16:05:05 CEST 2005
```

On relancera ensuite le « make all ».

Notes personnelles

4.11- Construire le ramdisk initial (initrd ou initramfs)

Même si l'on peut sans doute s'en passer, toutes les distributions utilisent aujourd'hui un ramdisk initial.

Pour s'en passer

Aucun travail spécifique si vous utilisez un noyau sans module.

Vérifiez dans le cas contraire que les pilotes nécessaire au montage de « / » par le noyau sont intégrés en « **built-in** » dans celui-ci :

- Pilote du contrôleur disque (au moins pour celui du root-filesystem)
- Pilote de la table de partitions (PC-Bios Partition table)
- Pilote du système de fichiers racine (ext2, ext3, ext4 ...)
- ...

Pour le construire

Sous RH EL / CentOS :

```
mkinitrd /boot/initrd-version.img version-du-noyau
```

Sous Debian / Ubuntu :

```
update-initramfs -c -k version-du-noyau      # pour le créer
update-initramfs -u -k version-du-noyau      # pour le mettre à jour
```


Notes personnelles

4.12- Compléments sur le noyau

Pour créer un noyau monolithique (sans module)

Inhiber dans la section « Loadable Module Support » :

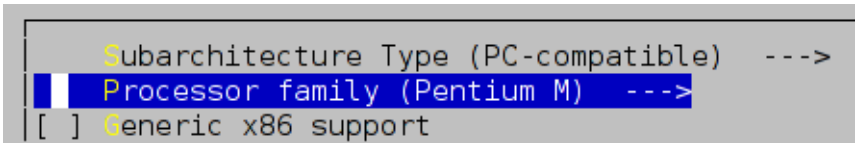
```
CONFIG_MODULES : « Enable Loadable Module Support »
```

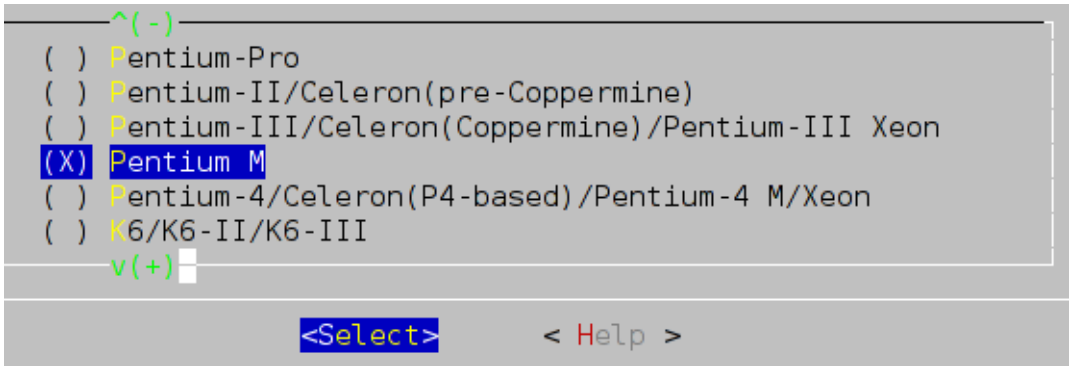
Pour choisir les autres options : seuls **[y]** et **[n]** sont possibles. Il n'y a plus aucun < >, **<y>**, ou **<m>**

Autres choix que y/n/m

Dans « make menuconfig », d'autres types de choix existent :

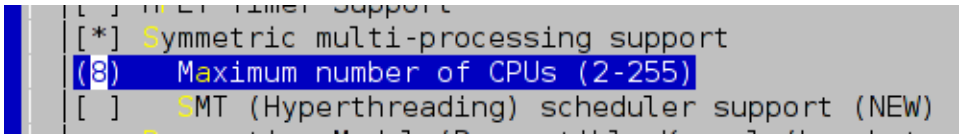
Les « radio-button », entre parenthèses : (Processor Type & Features / Processor Family)

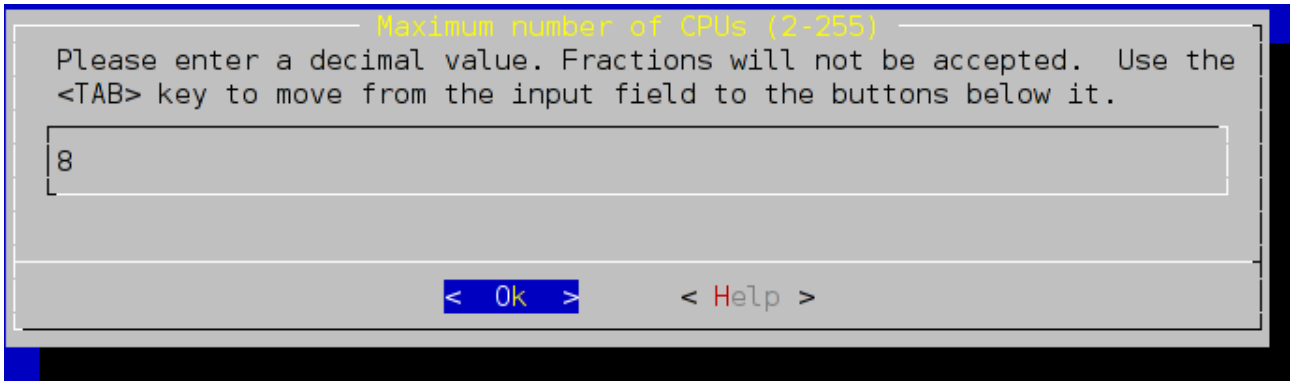




Notes personnelles

...des zones de saisie présentées entre parenthèses :





Notes personnelles

4.13- Informations complémentaires sur le noyau

4.13.1- Commandes d'information

uname -a : nom, version, date de compilation kernel.

dmesg : informations "Kernel Ring Buffer"

Ajouter les timestamps (à chaud) :

```
# echo Y > /sys/module/printk/parameters/time    (ou N pour désactiver)
```

au boot : paramètre de boot **printk.time** (=1/Y/y ou =0/N/n)

Écrire dans le Kernel Ring Buffer :

```
# echo Salut > /dev/kmsg
```

4.13.2- Fichier de trace

/var/log/dmesg : informations de démarrage du noyau

4.13.3- Informations sur le noyau

Les pseudo-fichiers de /proc donnent des informations intéressantes :

/proc/cmdline	/proc/devices	/proc/interrupts
/proc/version	/proc/partitions	/proc/meminfo
/proc/cpuinfo	/proc/ioports	/proc/stat

Notes personnelles

4.14- Les paramètres dynamiques de Linux

Le pseudo Filesystem /proc

Il donne des informations système : un répertoire par processus en cours d'exécution

Les répertoires /proc/XXX (avec XXX = un PID) : informations sur les processus.

```
ls -l /proc/$$/cwd
```

Il contient un ensemble de pseudo fichiers de taille nulle gérés par le noyau Linux

/proc/cpuinfo, /proc/version, /proc/partitions...

Ces paramètres du noyau sont consultables comme des fichiers texte

/proc/sys/...

Lire la valeur d'un paramètre :

```
cat PARAM
```

Modifier la valeur d'un paramètre :

```
echo VALEUR > PARAM
```

Exemples de lecture / modification de paramètre

```
echo 1 >/proc/sys/net/ipv4/ip_forward
```

Il contient les paramètres du noyau Linux, décrits dans la documentation du noyau :

Voir /usr/src/linux/Documentation/filesystems/**proc.txt**

Depuis 2.6.30, le détail des paramètres est dans **/usr/src/linux/Documentation/sysctl.**

Notes personnelles

4.15- Tuning du noyau

4.15.1- La commande sysctl

La commande **sysctl** permet de lister et modifier les paramètres dynamiques du noyau.

Elle est appelée au démarrage du système de manière à appliquer de manière "propre" les paramètres du noyau, en se basant sur un fichier de configuration.

Elle utilise le fichier de configuration **/etc/sysctl.conf**, installé par le paquetage "**initscripts**".

Extrait de /etc/rc.d/rc.sysinit :

```
sysctl -e -p /etc/sysctl.conf >/dev/null 2>&1
```

Cette commande est forcément disponible sur toutes les distributions modernes de GNU/Linux car livrée avec le paquetage "**procps**" (ou **procps-ng**) ... comme ps, free, vmstat, uptime, w....

Utilisation de la commande sysctl

```
sysctl classe ou parametre : afficher la valeur d'un paramètre
sysctl -a : lister tous les paramètres (et leurs valeurs)
sysctl -N classe ou paramètre : n'afficher que les noms des paramètres
sysctl -n parametre : n'afficher que la valeur du paramètre
sysctl -w parametre=valeur : modifier la valeur d'un paramètre
```


Notes personnelles

4.15.2- Les paramètres dynamiques du noyau

Depuis 2.6.30, le détail des paramètres est dans **/usr/src/linux/Documentation/sysctl.**

On peut les lister par à la commande "**sysctl -a**" ou via le répertoire "**/proc/sys**".

Leur nombre est très variable d'un système à l'autre, car il dépend des fonctionnalités chargées dans le noyau, ainsi que de la version du noyau.

Il y a entre 600 et 700 paramètres sur des noyaux "classiques".

Ils sont rangés en catégories et sous-catégories, et les catégories "racine" sont suivantes :

```
# sysctl -a | cut -d'.' -f1 | sort -u
dev
fs
kernel
net
sunrpc
vm
xen
```


Notes personnelles

Les sous-catégories des paramètres **réseau** sont alors :

```
# sysctl -a |grep ^net | cut -d'.' -f1,2 | sort -u
net.core
net.ipv4
net.ipv6
net.token-ring
net.unix
```

ou encore

```
# sysctl net | cut -d'.' -f1,2 | sort -u
net.core
net.ipv4
net.ipv6
net.token-ring
net.unix
```


Notes personnelles

4.15.3- Paramètres noyau "fs" : tuning sur les systèmes de fichiers

La classe "**fs**" contient des paramètres (souvent des informations) concernant les systèmes de fichiers et tout ce qui en découle (gestion des fichiers, inodes, répertoires, quotas...)

Certains de ces paramètres nécessitent parfois d'être modifiés, mais il s'agit toujours de besoins nécessaires pour une application précise.

Ils n'ont **pas réellement d'influence sur les performances**, il s'agit plus de **modification de limites**.

Le nombre de paramètres dépend des fonctionnalités chargées dans le noyau :

```
# sysctl  -N fs  | sort -u | wc -l
29
# modprobe nfs
# sysctl  -N fs  | sort -u | wc -l
41
# rmmod nfs lockd fscache nfs_acl
# lsmod | grep nfs
# sysctl  -N fs  | sort -u | wc -l
29
```


Notes personnelles

Paramètres "importants" de la classe "fs"

Posix Message Queues :
fs.mqueue.msgsize_max, **fs.mqueue.msg_max**, **fs.mqueue.queues_max**

Contextes d'entrée/sorties :
fs.aio-nr et **fs.aio-max-nr** : nombre actuel de requêtes i/o en cours et limite.

Uid & Gid de débordement (FS gérant les uid et gid sur 16 bits) :
fs.overflowgid, **fs.overflowuid** (65534)

Handles de fichiers :
fs.file-max : nombre maximum "file handles" allouables. Par défaut : 10% de la RAM (en ko) ,
fs.file-nr : X=handles alloués, Y=handles utilisés, Z=fs.file-max

Handles liés aux inodes :
fs.inode-state : 2 éléments (fs.inode-nr) et 5 "0"...
fs.inode-nr : nr_inodes et nr_free_inodes. Inodes utilisées = nr_inodes - nr_free_inodes.

D'autres paramètres, notamment fs.quota.*, fournissent des informations plutôt que des réglages.

Notes personnelles

4.15.4- Paramètres noyau "kernel" : kernel internals

La classe "kernel" contient des paramètres que l'on doit modifier pour les adapter à des besoins liés aux applications. Par exemple, c'est ici que l'on paramètre les IPC (SHM, MSG, SEM...)

4.15.4.1- Tuning des IPCS

Sémaphores : "sem"

Un seul paramètre ("**sem**"), mais 4 valeurs : **SEMMSL**, **SEMMNS**, **SEMOPS**, **SEMMNI**

```
kernel.sem = 250        32000   32      128
```

Files d'attente de messages : "msgmnb", "msgmni", "msgmax"

```
kernel.msgmnb = 65536
kernel.msgmni = 16
kernel.msgmax = 65536
```

Mémoire partagée : "shmmni", "shmmax", "shmall"

```
kernel.shmmni = 4096
kernel.shmall = 268435456
kernel.shmmax = 4294967295
```


Notes personnelles

Valeurs par défaut des IPCS

Sémaphores : (**include/linux/sem.h**) :

```
#define SEMMNI  128             /* <= IPCMNI  max # of semaphore identifiers */
#define SEMMSL  250             /* <= 8 000 max num of semaphores per id */
#define SEMMNS  (SEMMNI*SEMMSL) /* <= INT_MAX max # of semaphores in system */
#define SEMOPM  32              /* <= 1 000 max num of ops per semop call */
```

Message Queues : **include/linux/msg.h**

```
#define MSGMNI    16   /* <= IPCMNI */  /* max # of msg queue identifiers */
#define MSGMAX  8192   /* <= INT_MAX */ /* max size of message (bytes) */
#define MSGMNB 16384   /* <= INT_MAX */ /* default max size of a message queue */
```

Shared Memory : **include/linux/shm.h**

```
#define SHMMAX 0x2000000                 /* max shared seg size (bytes) */
#define SHMMIN 1                         /* min shared seg size (bytes) */
#define SHMMNI 4096                      /* max num of segs system wide */
#define SHMALL (SHMMAX/PAGE_SIZE*(SHMMNI/16)) /* max shm system wide (pages) */
#define SHMSEG SHMMNI                    /* max shared segs per process */
```


Notes personnelles

4.15.4.2- Autres paramètres de la classe "kernel"

On trouve aussi dans cette classe des informations que l'on ne modifie pas forcément par sysctl :

Changement du nom d'hôte : "kernel.hostname", "kernel.domainname"

```
# sysctl kernel.hostname
kernel.hostname = formation5.actilis
# hostname toto
# sysctl kernel.hostname
kernel.hostname = toto
# sysctl -w kernel.hostname=formation5.actilis
kernel.hostname = formation5.actilis
# hostname
formation5.actilis
```

Certains sont non modifiables : ..."ostype", "osrelease", "version"...

```
# sysctl -w kernel.osrelease=2.8.0
error: "Operation not permitted" setting key "kernel.osrelease"
```


Notes personnelles

4.15.4.3- Autres paramètres

kernel.ctrl-alt-del :
Si =0 (par défaut) alors le noyau intercepte l'événement et le transmet à init..
Si > 0, le noyau "reboote à chaud" en cas de "*Three Finger Salute*"

kernel.panic : 0 par défaut,
Permet de paramétrer un **délai en secondes** au bout duquel le noyau **reboote** suite à un kernel-panic.

kernel.pid_max :
1 + plus grand PID souhaité.
Max/défaut=2^15 sur arch 32bits, 2^22 sur 64bits.

kernel.random : gestion du random générator (man 4 random)

kernel.modprobe : chemin d'accès à la commande "modprobe".

Notes personnelles

4.15.5- Paramètres noyau "net" : le réseau

C'est la classe la plus peuplée.

Entre 500 à 800 paramètres...

On retrouve les sous-catégories **net.core**, **net.ipv4**, **net.ipv6**, **net.token-ring**, **net.unix** à l'intérieur desquelles on trouve un grand nombre de paramètres.

Il y a environ 75 sous catégories de paramètres dans la catégorie **net.ipv4**, concernant les différents protocoles (icmp, igmp, ip, tcp, ...),.

Ces paramètres ainsi que ceux propres aux interfaces sont documentés dans le fichier **Documentation/networking/ip-sysctl.txt** fourni avec les sources du noyau.

4.15.5.1- La catérogie net.ipv4

Les paramètres concernant tcp sont documentés dans la page de manuel **tcp(7)** et ceux concernant ip sont décrits dans la page de manuel **ip(7)**.

Le plus connu de ses paramètres est sans doute **net.ipv4.ip_forward**, qui permet d'activer (valeur 1) ou de désactiver (valeur 0) le routage des paquets IPV4 par le noyau.

Notes personnelles

Dans cette catégorie, on retrouve des paramètres icmp : réponse au ping ?

```
net.ipv4.icmp_echo_ignore_all = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_errors_use_inbound_ifaddr = 0
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.icmp_ratelimit = 1000
net.ipv4.icmp_ratemask = 6168
```

ou encore la plage de numéros ports sortants :

```
net.ipv4.ip_local_port_range = 32768    61000
```

Outre le paramétrage fin des protocoles **icmp**, **ip** et **tcp** ou encore les notions de routage, on trouve une sous-catégorie "**conf**", dans laquelle chaque interface fait l'objet de plusieurs paramètres :

forwarding, **mc_forwarding**, **accept_redirects**, **secure_redirects**, **shared_media**, **rp_filter**, **send_redirects**, **accept_source_route**, **proxy_arp**, **medium_id**, **bootp_relay**, **log_martians**, **tag**, **arp_filter**, **arp_announce**, **arp_ignore**, **arp_accept**, **disable_xfrm**, **disable_policy**, **force_igmp_version**, **promote_secondaries**.

Ils peuvent être définis pour chaque interface ou pour toutes les interfaces (sous-catégorie "all").

Des valeurs par défaut peuvent être affectées (sous-catégorie "default") de manière à être prises en compte au démarrage de chaque interface.

Notes personnelles

4.15.6- Paramètres noyau "sunrpc" : paramétrage de NFS

Activation et désactivation des flags :

```
sunrpc.rpc_debug = 0
sunrpc.nfs_debug = 0
sunrpc.nfsd_debug = 0
sunrpc.nlm_debug = 0
```

Nombre de slots tcp & udp

Nombre de requêtes qui peuvent être gérées simultanément. On peut augmenter ces valeurs, mais le nombre de threads du serveur NFS augmentera en conséquence.

```
sunrpc.udp_slot_table_entries = 16
sunrpc.tcp_slot_table_entries = 16
```

Numéros de ports mini/maxi pour xprt_bindresvport()

Cette fonction concerne l'interface xprt (c'est l'interface d'appel RPC asynchrone, voir net/sunrpc/xprc.c) qui doit se mettre en écoute (bind) sur un port privilégié (resvport), par appel à la fonction bindresvport(3).

Pour éviter des ports déjà utilisés, on peut restreindre la plage choisie entre un minimum et un maximum.

```
sunrpc.min_resvport = 665
sunrpc.max_resvport = 1023
```


Notes personnelles

4.15.7- Paramètres noyau "vm" : Gestion de la mémoire

VM : Le Virtual Memory Manager

Linux dispose d'un gestionnaire de mémoire virtuelle paramétrable.

Il est responsable de la gestion des pages mémoire et se paramètres portent sur la gestion de la mémoire (y compris du swap), des buffers et du cache.

Debug/verbose

block_dump : Active la trace des accès disque.

Exemple : (visible par dmesg ou ***syslog(attention)***)

```
rm(1262): dirtied inode 375412 (?) on dm-0
ps(1263): dirtied inode 69566466 (2) on proc
```

Attention : Stopper klogd avant de passer block_dump à 1, car très verbeux sur un système actif.

Notes personnelles

Contrôle du cache et des écritures disque ([pdflush])

drop_caches (> 2.6.16) : On utilise ce paramètre comme un déclencheur pour le vidage du cache :
- **sysctl -w vm.drop_caches=1** : pagecache
- **sysctl -w vm.drop_caches=2** : dentries + inodes
- **sysctl -w vm.drop_caches=3** : dentries + inodes + pagecache

dirty_background_ratio (10%): démarre une écriture en arrière plan des "dirty buffers" lorsqu'elles occupent plus de 10% de la mémoire.

dirty_ratio (40%) : démarre une écriture "active" (forçage) des "dirty buffers" lorsqu'elles occupent 40% de la mémoire.

dirty_expire_centisecs (3000) : age des "dirty buffers" en centièmes de secondes pour que pdflush commence à les écrire sur disque.

dirty_writeback_centisecs (500) : défini le délai entre deux réveils de **pdflush**.

laptop_mode : concentre les écritures de manière à en réduire la fréquence, et de ce fait permet d'économiser la batterie.

Notes personnelles

4.15.8- Gestion de la surcharge mémoire : OOM Killer et Overcommit_memory

4.15.8.1- OOM et OOM killer

Sous Unix, les programmes qui demandent de la mémoire le font via l'instruction "malloc()". Si la fonction retourne NULL, ils s'arrêtent, ou s'adaptent, éventuellement.

Linux, peut être paramétré pour accepter toute demande de mémoire, en misant sur le fait que les programmes demandent toujours plus que ce dont ils ont besoin.

Si c'est le cas, tout va bien, mais dans le cas contraire, tout va mal... car le OOM Killer (Out Of Memory) va tuer le demandeur (vm.**oom_kill_allocating_task=1**) ou un autre processus (vm.**oom_kill_allocating_task=0**) selon des règles heuristiques.

Sa façon de choisir les victimes est parfois arbitraire, puisqu'il est arrivé que des processus exécutant "named", ou encore "sshd" soient les heureux élus... Cela s'appelle simplement un déni de service.

4.15.8.2- Paramétrer l'over-commit

En théorie, les 3 programmes fournis en annexe (*memtest1*, *memtest2*, et *memtest3*) devraient obtenir la même quantité de mémoire et devrait signaler le "malloc failure" au lieu de *crasher.*

En fait, ils ne *crashent* pas au même moment. Cela dépend du **paramétrage de l'over-commit**.

Notes personnelles

overcommit_memory (0, 1 ou 2) : conditions d'acceptation/de refus de demandes de mémoire.

0 (valeur par défaut) : gestion heuristique (donc peu précise) de la mémoire disponible.
Peut autoriser des surcharges de la mémoire... refusera, au bout d'un moment.

1 : pas de gestion des risque de surcharge mémoire.
Le risque est donc accru, mais les performances des applications gourmandes aussi.
Ne refusera pas de demande...

2 : gestion précise de la quantité de mémoire disponible.
Refus d'allocation si l'on demande plus que la quantité "swap + overcommit_ratio".

overcommit_ratio (50 par défaut) :
fraction (pourcentage) de la mémoire physique autorisée en plus du swap pour l'over-commit.

Tester le comportement de memtest1, 2, et 3 avec les valeurs suivantes :

```
sysctl -w vm.overcommit_memory=2
```

panic_on_oom (0) : Faut-'il crasher (le noyau) dès qu'un problème d'overcommit se produit ?
ou juste tuer le processus...
Valeur 0 : pas de panic,
Valeur 1, kernel-panic.

Notes personnelles

4.15.9- Optimiser la gestion de la mémoire et du swap

4.15.9.1- Stratégie de swap : swappiness

Ce paramètre peut avoir une valeur de 0 à 100.
Une longue discussion à ce sujet a eu lieu sur http://kerneltrap.org/node/3000 .

Plus la valeur est élevée, plus le noyau utilisera le swap. Valeur par défaut : **60**.

Taille des clusters d'I/O de swap : page-cluster

La valeur (3 par défaut) contrôle le nombre de pages swappées en même temps.
Elle est exprimée en puissances de 2. La valeur de 3 signifie 8 pages à la fois (une page pèse 4 ko).

Récupération de mémoire cache : vfs_cache_pressure

Ce paramètre (par défaut 100) définit la manière dont le noyau privilégie la conservation en mémoire d'informations sur les "**dentries**" (directory entries) ou les "**inodes**" (description des fichiers).

En dessous de 100, le noyau préfère retenir en mémoire cache les "dentries" et "inodes",

Au dela de 100, le noyau préférera rechercher à nouveau ces informations.

Notes personnelles

4.15.10- Autres paramètres liés à vm

Accélération des appels système (Virtual Dynamic Shared Object) :

vdso_enabled (1=enabled par défaut).
Mode 2 pour "compat", si compilé dans le noyau (pas nécessaire si glibc >2.3.3).

Limites de zones de mémoire "map" qu'un processus peut demander (65536 par défaut) :

max_map_count : (voir *malloc()*, *mmap()* et *mprotect()*)

lowmem_reserve_ratio : mal documenté.
Ratio "total / libres" (en pages mémoire) dans les zone mémoire (DMA, Normale, Haute)
Voir /proc/meminfo + linux:mm/page_alloc.c.

legagy_va_layout (0) : mode de mapping de la mémoire :
0 = 32bits mmap, autre : comme en 2.4 (legacy mmap).
Voir http://lwn.net/Articles/91829/

Forcer VM à conserver une quantité de mémoire libre (mal documenté) : **min_free_kilobytes**

Notes personnelles

4.15.11- Annexe : programmes memtest*

4.15.11.1- memtest1.c

Allocation et on rend la main.

```
#include <stdio.h>
#include <stdlib.h>

int main (void) {
        int n = 0;

        while (1) {
                if (malloc(1<<20) == NULL) {
                        printf("malloc failure after %d MiB\n", n);
                        return 0;
                }
                printf ("got %d MiB\n", ++n);
        }
}
```


Notes personnelles

4.15.11.2- memtest2.c

Allocation, occupation, et on rend la main

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main (void) {
        int n = 0;
        char *p;

        while (1) {
                if ((p = malloc(1<<20)) == NULL) {
                        printf("malloc failure after %d MiB\n", n);
                        return 0;
                }
                memset (p, 0, (1<<20));
                printf ("got %d MiB\n", ++n);
        }
}
```


Notes personnelles

4.15.11.3- memtest3.c

Allocation d'un tableau forçant l'utilisation des zones allouées, et on rend la main

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define N       10000

int main (void) {
        int i, n = 0;
        char *pp[N];

        for (n = 0; n < N; n++) {
                pp[n] = malloc(1<<20);
                if (pp[n] == NULL)
                        break;
        }
        printf("malloc failure after %d MiB\n", n);

        for (i = 0; i < n; i++) {
                memset (pp[i], 0, (1<<20));
                printf("%d\n", i+1);
        }

        return 0;
}
```


Notes personnelles

5- Maintenance et métrologie sur des serveurs

Notes personnelles

5.1- Emplacement des fichiers de log

5.1.1- Comment loguer ?

On peut émettre des logs de **deux manières** :

écrire directement dans les fichiers de log
émettre via un service de gestion de logs…

5.1.2- Localisation et format des logs

Les fichiers de log se trouvent **généralement** dans le répertoire **/var/log**.

Parfois regroupés dans un **sous-répertoire propre** à un service : Apache, Samba....

Ces services gèrent **plusieurs fichiers de log**.
Un fichier de log par site hébergé (plus simple pour les statistiques).
Un fichier de log par poste client ou par utilisateur (samba)
Format des messages souvent spécifique à l'application, et paramétrable.

Souvent, on souhaite **séparer** les fichiers de logs de chacun des services...
un fichier error_log, un fichier access_log, et cela pour chaque site hébergé (Apache)
un fichier pour les connexions au service POP, un autre pour le service IMAP...

Dans d'autres cas, on préfère **regrouper** les logs dans des fichiers "par fonction".
Exemple : tout ce qui concerne le mail dans un fichier, l'authentification dans un autre...

Notes personnelles

5.2- Le protocole SYSLOG

5.2.1- Du protocole syslog aux daemons *syslog*

Syslog est un protocole de notification (RFC 5424) largement utilisé dans le monde de la supervision et de l'audit de sécurité.

C'est une **solution standard** d'enregistrement d'évènements système des systèmes Unix et Linux.

Les messages sont émis **par les applications** ou **le noyau** :
Voir la fonction syslog() en C.
Voir la commande **logger**, qui permet de poster des messages depuis le shell.

Plusieurs implémentations du service : le "serveur syslog" (ou *syslog daemon*) :

sysklogd, historiquement utilisé sur la plupart des distributions de GNU/Linux

syslog-ng, implémente des filtres et un format de configuration structuré,

rsyslog, qui a été retenu par plusieurs distributions (Red Hat EL, Debian, Ubuntu...)

Son rôle : **recevoir** des messages, et les **aiguiller** vers une destination
en fonction de leur **provenance** et leur **sévérité**...
et maintenant d'autres éléments figurant dans les messages (tag, expressions...)

Notes personnelles

5.2.2- Format des messages

Avec Syslog, les messages sont toujours formatés comme ceci :

```
Mois Jour HH:MM:SS machine-émettrice TAG: Message à caractère informatif
```

Souvent, le TAG indique : nom-du-programme[PID].

5.2.3- La commande logger

Elle permet de poster des messages au service syslogd local via la socket locale **/dev/log**.

Le message à poster est le dernier argument de la ligne de commande... ou l'entrée standard

```
LOGGER(1)                 BSD General Commands Manual                LOGGER(1)

NAME
     logger - a shell command interface to the syslog(3) system log module

SYNOPSIS
     logger [-isd] [-f file] [-p pri] [-t tag] [-u socket] [message …]
```

Options :

-p : présente la "priority". C'est un couple "facility" . "severity".

-t : présente un "tag", c'est un nom libre. Généralement, c'est programme[PID]

Exemples :

```
# logger -p local0.info -t ESSAI 'Message à caractère informatif'
# ls -l | logger -p local0.debug -t LOGSTDIN   # Trace l'entrée standard
```


Notes personnelles

5.2.4- Priority, facility, level

Elle est composée du couple facility.severity

Facility (émetteur, ou sous-système)

Il existe un ensemble de noms prédéfinis correspondant aux services usuels du monde Unix.

Les sous-systèmes suivants sont disponibles :

auth, **authpriv**, **cron**, **daemon**, **ftp**, **kern**, **lpr**, **mail**, **news**, **syslog**, **user**, **uucp**,

Il n'existe pas de nom comme "apache", ou "http", ou "ldap"... mais on dispose de **local0** à **local7**...

Le niveau de sévérité

Chaque niveau correspond à un état dans lequel se trouve le système (ou le sous-système).

Par ordre décroissant de gravité :

EMERG : Système inutilisable
ALERT : Action corrective indispensable rapidement
CRIT : Conditions critiques
ERR : Des erreurs se produisent
WARNING : Avertissement non bloquant
NOTICE : Il se passe quelque chose de normal, mais qui doit être souligné
INFO : Information pour qui cela intéresserait
DEBUG : Message de débuggage

Notes personnelles

5.3- Les logs avec systemd

5.3.1- Le service systemd-journald

Aujourd'hui, les logs sont collectés par le service **systemd-journald** :
- Kernel (via kmsg)
- Logs système (via syslog)
- Messages émis par l'API "Journal"
- Sorties standard et sortie d'erreur des services
- Enregistrement d'audit (service auditd)

Le service est configuré par **/etc/systemd/journald.conf** (man 5 journald.conf).

Par défaut, les messages sont stockés dans **/run/log/journal**, qui est volatile.
On peut les rendre persistants en créant le répertoire **/var/log/journal**.

C'est le paramètre de configuration "**Storage=**" qui gère cela :
Valeurs possibles : **volatile**, **persistent**, **auto** (par défaut), **none**

Le service **systemd-journald** peut faire suivre les messages au service **Syslog**,
Même si **Storage**=none.

On paramètre cela par "**ForwardToSyslog=**".
Ou par le *kernel-parameter* "**systemd.journald.forward_to_syslog**=".

Du côté de Rsyslogd, cela passe par un "Input Module" (...$ModLoad **imjournal**...).

Notes personnelles

5.3.2- La commande journalctl

La commande **journalctl** permet de consulter le journal écrit par **systemd-journald**.

Options : **-n X** : affiche les X dernières lignes.
-o format : affichage dans un format sympa… (short*, json*...)
-l : listage des lignes au format long
-a : encore plus long, pas de troncature
-x : ajoute (si disponibles) des explications tirées du catalogue de messages
-u unité : spécifie l'unité dont on vuet voir les logs
-p priorité : spécifie le log-level à afficher
--since yyyy-mm-dd [hh:mm:ss] (ou **--until**) : date de début (fin) des messages à afficher

Sans paramètre : lit tout le journal

Paramètres : du type CHAMPS=Valeur pour filtrer les messages affichés.

_SYSTEMD_UNIT=Valeur est synonyme de : **-u** Valeur

```
journalctl _SYSTEMD_UNIT=network.service <==> journalctl -u network.service
```

PRIORITY=Valeur est synonyme de : **-p** Valeur

```
journalctl PRIORITY=err <==> journalctl -p err
```

Voir **systemd.journal-fields**(7) pour la liste des champs utilisables.

Notes personnelles

5.3.3- Purger l'historique des logs

Lorsque le stockage est persistant, les logs doivent être purgés de temps en temps...

Ponctuellement :

```
# journalctl --vacuum-time=30d
```

ou encore

```
# journalctl --vacuum-size=1500M
```

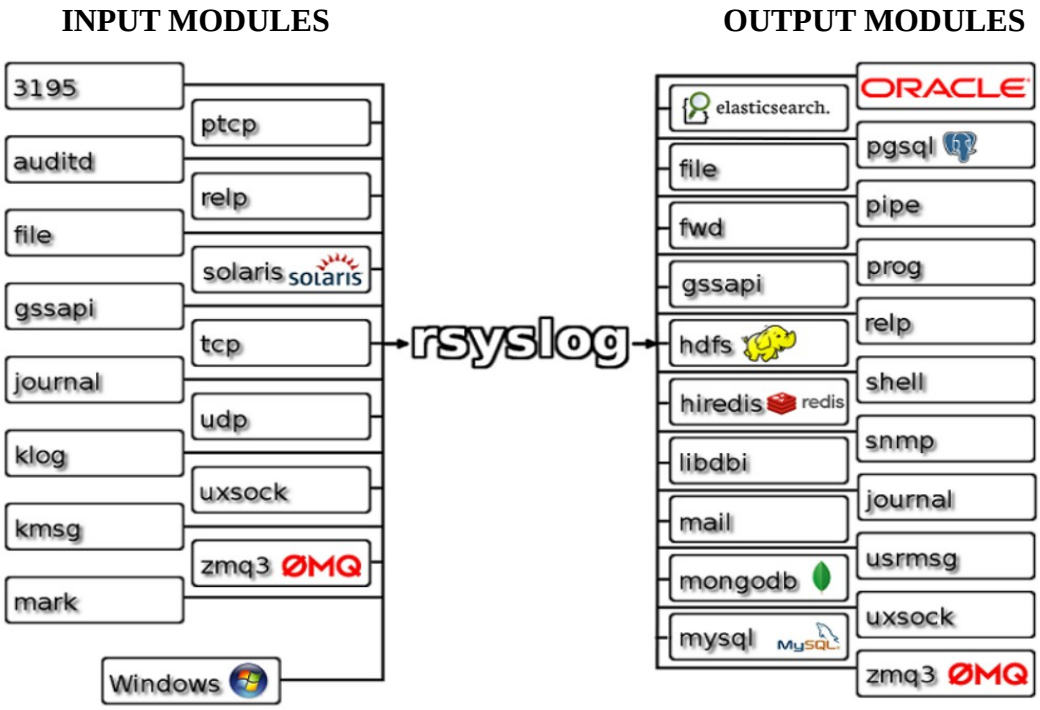
Automatiquement : grâce au fichier de configuration (/etc/systemd/journald.conf) :

```
[Journal]
…
SystemMaxUse=100M
…
```

La valeur par défaut est 10 % du filesystem où repose /var/log/journal, et le paramètre est capé à 4 Go.

Notes personnelles

5.4- Le serveur Rsyslog…



Notes personnelles

5.5- Rsyslogd, le serveur et les modules

Le serveur **rsyslogd** est fourni en standard sur la plupart des distributions modernes.

Attention, il n'est pas installé par défaut sur CentOS 8 !

OS	Version
CentOS 7.9	Rsyslog-8.24
Debian 10	Rsyslog-8.1901
CentOS 8.2	Rsyslog 8.1911

Des packages complémentaires apportent des modules : **I**nput **M**odules, et **O**utput **M**odules

Ce sont des bibliothèques que peut charger le serveur, pour proposer des interfaces complémentaires :

Stockage chiffré (crypto),
Bases de données (libdbi, mysql, pgsql, elasticsearch),
Transmission à distance, chiffrée (gnutls), ou "reliable" (RELP)…
...

```
rsyslog-crypto, rsyslog-elasticsearch, rsyslog-gnutls, rsyslog-gssapi, rsyslog-libdbi, rsyslog-
mmaudit, rsyslog-mmjsonparse, rsyslog-mmnormalize, rsyslog-mmsnmptrapd, rsyslog-mysql, rsyslog-
pgsql, rsyslog-relp, rsyslog-snmp, rsyslog-udpspoof
```


Notes personnelles

5.6- Configuration du service Rsyslogd

5.6.1- Structure de la configuration

La configuration est faite par le fichier **/etc/rsyslog.conf**, qui peut référencer d'autres fichiers :

```
# grep --color -ni '^$Include' /etc/rsyslog.conf
36:$IncludeConfig /etc/rsyslog.d/*.conf
```

Les directives appartiennent à plusieurs catégories distinctes :

Directives globales :
- chargement de modules
- paramétrage de base, limites
- définitions de ports, sockets, noms de fichiers...

Définition de templates :
- concernant les formats de ligne de logs
- concernant les noms des fichiers de log

Définition de règles : **sélecteur** ⇒ **action**
- format compatible avec Syslogd
- format spécifique à Rsyslogd, plus fin.

```
# The imjournal module bellow is now use
$ModLoad imuxsock # provides support for
$ModLoad imjournal # provides access to
#$ModLoad imklog # reads kernel messages
#$ModLoad immark  # provides --MARK-- me

# Provides UDP syslog reception
#$ModLoad imudp
#$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp
#$InputTCPServerRun 514
```

Chacune des catégories peut être implémentée dans **/etc/rsyslog.conf** ou un fichier de **/etc/rsyslog.d**.

Notes personnelles

5.6.2- Format des règles

On définit des **actions** à appliquer aux messages correspondant à un **sélecteur**.

Format d'une Règle : **Sélecteur …… Action**

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog
```

Chaque ligne définit des sélecteurs **émetteur** / **niveau de sévérité** et une **destination**.

La première colonne sert à identifier le message (sélecteur), la seconde sa destination.

Rsyslog accepte aussi des règles dans ce type de syntaxe, mettant en jeu des filtres conditionnels :

```
:programname, startswith, "postfix"             /var/log/maillog
if $programname contains 'postfix' or $programname contains 'amavis' then   /var/log/mail/info.log
```

Cela permet entre autre, de s'intéresser à des éléments plus fins que les sélecteurs.

Notes personnelles

5.6.3- Sélecteurs de messages

La **première colonne** présente un ou plusieurs sélecteurs sur chaque ligne.

Un sélecteur est composé à partir d'un couple sous la forme :

"Émetteur " point "Niveau" : au moins ce niveau
"Émetteur" point-égal "Niveau" : précisément ce niveau
"Émetteur" point "**none**" : aucun message de cet émetteur

De part et d'autre, "*****" joue le rôle de Joker.
"**;**" signifie "**ET**",
"**,**" permet de factoriser.

Exemples :

Codification	Signification
E.N	Provenance=E, niveau >= N
E.=N	Provenance=E, niveau = N
E1,E2.N	Provenance=E1 ou E2, niveau >= N
*.N	Niveau >=N pour toute provenance
E.*	Provenance=E, tout niveau
.	Tout message
.;E1.none;E2.none	Tout message sauf ceux venant de E1 ou E2

Notes personnelles

5.6.4- Les actions

La **seconde colonne** décrit l'action pour chaque message concerné par la partie sélecteur.

Un même message peut avoir plusieurs destinations s'il est concerné par plusieurs sélecteurs.

Actions classiques qui étaient déjà permises par Sysklogd :

Action	Signification
/var/log/messages	Ajout en fin du fichier
-/var/log/messages	Idem mais avec bufferisation (pas de sync à chaque écriture)
Usera, *	Par la commande write, ou wall
@srv-log.actilis.net	Remontée à un serveur distant*(port **udp/514**) (Module imudp)*
@@srv-log.actilis.net	Remontée à un serveur distant*(port **tcp/514**) (Module imtcp)*
|/tmp/fifo	Vers un Pipe (à créer au préalable) (***mkfifo myfifo)***
~	Détruit le message. Il n'est donc pas analysé par les éventuelles règles à suivre.

Rsyslogd apporte d'autres "output modules" : RELP, base de données, exécution de script…

Ces "output modules" nécessitent pour certains (udp, tcp, RELP par exemple) qu'il y ait en face un "Input Module" capable d'être récepteur.

Notes personnelles

5.7- Analyse rapide de la charge système

5.7.1- La charge indiquée par la commande "uptime"

La commande **uptime** indique la "charge système", plus précisément du nombre moyen de processus dans la runqueue (à l'état **runnable** : **R**) ou en attente d'une entrée/sortie (à état **ininterruptible** : **D**).

```
# uptime
 18:00:34 up 81 days, 10:20,  3 users,  load average: 0.18, 0.87, 0.62
```

Les chiffres sont des moyennes :
- sur la dernière minute,
- les 5 dernières minutes,
- le dernier quart d'heure.

Les chiffres sont indépendants du nombre de processeurs.

Lecture de /proc/loadavg

Les chiffres indiqués par **uptime** sont en fait ceux visibles dans **/proc/loadavg**.

```
# cat /proc/loadavg
0.18 0.87 0.62 1/45 11335
```

1/45 : processus (ou threads) en runqueue (< NCPU) / processus existants dans le scheduler.
11335 : numéro du dernier processus créé.

Notes personnelles

Quelques indications

Si le chiffre est de 0 ou près de 0, c'est qu'il n'y a pas de processus "actif".

Cela ne signifie pas qu'il n'y a pas de processus en cours d'exécution.

Un processus en attente d'une action (user) ne contribue pas à l'augmentation de ce chiffre.

Un processus en attente d'un IO contribue à l'augmentation du chiffre.

Un processus en attente de temps CPU contribue à l'augmentation du chiffre.

Le plus souvent, le chiffre est entre 0 et NCPU : il y a de l'activité, mais pas en permanence.

Au delà de NCPU, rien d'alarmant. Mais les programmes s'exécutent moins vite.

Montée en charge et retour à la normale

```
15:22:04 up 81 days,  7:41,  2 users,  load average: 0.00, 0.03, 0.06
15:33:35 up 81 days,  7:53,  3 users,  load average: 0.99, 0.72, 0.35
15:37:01 up 81 days,  7:56,  4 users,  load average: 5.98, 2.39, 1.02
15:38:01 up 81 days,  7:57,  4 users,  load average: 7.79, 3.50, 1.47
15:41:42 up 81 days,  8:01,  2 users,  load average: 3.68, 4.63, 2.44
15:49:00 up 81 days,  8:08,  2 users,  load average: 0.00, 1.07, 1.51
```


Notes personnelles

5.7.2- Identifier le responsable d'un "load average" élevé

Supposons un système montant une charge (load average, avec "**uptime**") très élevé alors qu'il n'y a que très peu de consommation CPU.

Dans ce cas, vmstat ou sar peut aider à compendre : ici, sar montre un iowait très élevé

```
19:35:01          CPU     %user     %nice   %system   %iowait    %steal     %idle
19:45:02          all      0,06      0,00      0,10     99,83      0,00      0,00
19:55:02          all      0,06      0,00      0,12     99,82      0,00      0,00
20:05:02          all      0,06      0,00      0,10     99,84      0,00      0,00
20:15:01          all      0,05      0,00      0,10     99,85      0,00      0,00
20:25:02          all      0,06      0,00      0,10     99,84      0,00      0,00
```

Vmstat montre aussi un **"WA" élevé**, mais **pas (ou trop peu) d'I/O**

```
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 3  1 102020  24476  61580 871292    0    0     4   232  381  971  4  0  0 96
 1  1 102020  24484  61580 870792    0    0   256     0  321 1718  4  0  0 96
 2  1 102020  24484  61580 870792    0    0     0     0  358  852  2  0  0 98
```

Il peut y avoir une charge **iowait** élevée lors de grosses lectures/écritures disque, mais...Ici, c'est simplement un problème d'I/O.

On tentait tout simplement de lire un CD/ROM comportant des erreurs.

Notes personnelles

5.7.3- Le multi-tâche, principe de l'ordonnanceur et des priorités

Chaque processus a une certaine priorité.... inversement liée à son niveau de gentillesse.

Le noyau alloue à chaque processus une part du temps processeur liée à sa priorité. Cette priorité est ré-évaluée à chaque cycle d'exécution d'un processus : plus un processus a récemment utilisé le processeur, plus sa priorité est basse.

L'horloge matérielle, qui génère périodiquement une interruption du processeur, permet au noyau de réviser les priorités des différents processus en fonction de leur niveau de gentillesse "**nice level**".

Quelle que soit son "nice level", un processus qui est "seul" disposera du même temps processeur, et aura des temps d'exécution similaires... qu'il soit privilégié ou pas.

5.7.4- Visualiser la priorité des processus : le "nice level"

On règle donc la priorité des processus indirectement, en jouant sur leur "nice level" (niveau de gentillesse), sachant que plus un processus est "nice", plus il laisse la priorité aux autres.

La commande **nice** (sans argument) donne le niveau de gentillesse du processus courant.

La valeur indiquée est comprise entre **-20** (très prioritaire) et **+19**.(très faiblement prioritaire).

La commande **ps** affiche aussi ces informations, de deux manières :

Dans la colonne **STAT**, "**N**" (Nice) =faiblement prioritaire, "<" = fortement prioritaire.

Dans la colonne "**NI**", c'est tout simplement le nice-level qui est indiqué.

Notes personnelles

5.7.5- Gérer la priorité des processus : modifier le nice level

La commande nice

On peut exécuter une commande au travers de **nice**.

Dans ce cas, le nice level du processus concerné sera de 10, à moins qu'il ne soit précisé en option de la commande nice. (**nice -NICE-LEVEL cmd**)

La commande renice

La commande **renice** permet de modifier le nice level d'un processus en cours d'exécution.

Elle nécessite par contre forcément deux arguments (au minimum).

5.7.6- Modérer priorité d'entrée-sortie d'un processus

Voir la commande "**ionice**"... et le scheduler "**CFQ**".

Notes personnelles

Exemple :

On lance un sous-shell dont au travers de la commande "nice", dont moins prioritaire

```
# nice sh
sh-3.00# ps -o pid,stat,ni,cmd $$
  PID STAT  NI CMD
 8956 SN    10 sh

sh-3.00# renice 19 $$
8956: old priority 10, new priority 19
sh-3.00# ps -o pid,stat,ni,cmd $$
  PID STAT  NI CMD
 8956 SN    19 sh

sh-3.00# renice -20 $$
8956: old priority 19, new priority -20

sh-3.00# ps -o pid,stat,ni,cmd $$
  PID STAT  NI CMD
 8956 S<   -20 sh

sh-3.00# nice
-20
```

Une fois revenu dans le shell père, on affiche le niveau de gentillesse

```
sh-3.00# exit
exit
# nice
0
```


Notes personnelles

5.8- Analyser l'activité du système avec vmstat

Fonctionnalités attendues dans ce genre d'outil :

- Donner des informations moyennes depuis le démarrage
- Permettre l'audit au long d'une période de temps donnée
- Formater l'affichage, utiliser les résultats pour générer des graphiques...
- Charger très peu la machine.

5.8.1- Fonctionnement de vmstat

Syntaxe classique

```
vmstat [-n] [delai en sec [nbr ]]
```

Option **-n** : ne pas ré-afficher les lignes de titre
Delai : temps en secondes entre deux affichages
Nbr : nombre d'itérations

Exemples

600 mesures à raison d'une par seconde

```
vmstat 1 600
```

une mesure toutes les 10 secondes indéfiniment, sans titre

```
vmstat -n 10
```


Notes personnelles

Lire les informations fournies par vmstat :
D'une version à l'autre, **vmstat** évolue et montre parfois de nouvelles informations (cpu/**wa**, cpu/**st**, ...), alors que d'autres disparaissent (procs/w).

```
procs -----------memory------- --swap-- ---io--- --system-- -----cpu------
r  b  swpd  free   buff  cache  si  so   bi   bo   in    cs us sy id wa st
3  0  4164 23416 109732 292156   0   0    4    7  147   361  1  1 99 0  0
```

Procs

r = nombre de processus dans la runqueue (demandeurs de temps cpu),
b = nombre de processus en sommeil non-interruptible (en attente d'une IO),

Memory (exprimé en ko)

swpd = quantité de swap utilisé
free = quantité de mémoire virtuelle libre (RAM + SWAP)
buff = quantité de mémoire utilisée comme tampon d'E/S (*)
cache = quantité de mémoire utilisée comme cache (*)
Sur certaines versions de vmstat, on trouve ici deux colonnes complémentaires :
active = quantité de mémoire récemment utilisée
inactive = quantité de mémoire n'ayant pas été utilisée récemment

Notes personnelles

```
procs -----------memory------- --swap-- ---io--- --system-- -----cpu------
r  b  swpd  free   buff  cache  si  so   bi   bo   in    cs us sy id wa st
3  0  4164 23416 109732 292156   0   0    4    7  147   361  1  1 99 0  0
```

Swap (exprimé en ko/s)

si = quantité de mémoire paginée lue depuis le disque
so = quantité de mémoire paginée écrite vers le disque

Blocs (nombre par seconde)

bi = nombre de blocs reçus (par le noyau) depuis des périphériques de type bloc
bo = nombre de blocs envoyés (par le noyau) vers des périphériques de type bloc

Système (nombre par seconde)

in = nombre d'interruptions (y compris l'horloge)
cs = nombre de changements de contexte (appels système + commutation de tâches)

CPU (pourcentage de temps CPU)

us = temps consommé par les processus
sy = temps passé dans le noyau
id = temps inutilisé (donc libre)
wa = (>2.5.41) temps d'attente des entrées/sortie (disque, réseau, ...)
st = (>2.6.11) temps volé (non alloué) à une machine virtuelle (qui en aurait besoin)

Notes personnelles

5.8.2- Charge processeur et run-queue

```
$ vmstat 1 10
   procs                  memory    swap        io     system      cpu
 r  b  w  swpd  free   buff  cache  si  so   bi   bo   in    cs us sy id
 3  0  0  4164 23416 109732 292156   0   0    4    7  147   361  1  1 99
 0  0  0  4164 23416 109732 292156   0   0    0    0  163   253  6  0 94
 0  0  0  4164 23416 109732 292156   0   0    0    0  480  1700  6  7 87
 1  0  0  4164 23416 109732 292156   0   0    0    0  519  1797  5  4 91
 1  0  0  4164 23416 109732 292156   0   0    0    0  540  1574  5  4 91
 1  0  0  4164 23416 109732 292156   0   0    0   40  527  1248  7  2 91
 1  0  0  4164 23416 109732 292156   0   0    0    0  300   547  5  0 95
 3  0  1  4164 23416 109732 292156   0   0    0    0  162   254  5  0 95
 1  0  0  4164 23416 109732 292156   0   0    0    0  162   235  4  0 96
 1  0  0  4164 23416 109732 292156   0   0    0    0  161   238  4  0 96
```

Les lignes mises en évidence montrent une activité du noyau en terme d'interruptions et de changements de contexte. (Déplacements rapides de la souris en environnement graphique).

Notes personnelles

5.8.3- vmstat et la mémoire : free et vmstat

Combien de mémoire est réellement nécessaire au système ?

La commande "**free**" permet de le savoir assez précisément :

```
# free
             total       used       free     shared    buffers     cached
Mem:        230992  US=112212     118780          0     B=9180    C=52864
-/+ buffers/cache:     U=50168    F=180824
Swap:       264592          0     264592
```

US = *U* + *B* + *C*. C'est donc "**U**" qui indique la quantité de RAM utilisée par le noyau + les processus.

L'option "**-a**" de "**vmstat**" donne une information complémentaire (active/inactive) :

```
# vmstat -a
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free  inact active   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 118780  33376  55728    0    0    37    12   56   50  2  1 95  1  0
```

On retrouve le chiffre concernant l'espace libre similaire à celui de la commande free.

Désormais (depuis RedHat 7), l'affichage de la commande "free" est simplifié :

```
# free
              total        used        free      shared  buff/cache   available
Mem:       16352228     1182560    12950724      580268     2218944    14258084
Swap:             0           0           0
```


Notes personnelles

5.8.4- Vmstat et les disques

La commande **vmstat** affiche les entrées sorties (colonnes I/O) et peut donner des détails par partition

Zoom pour chaque disque

```
# vmstat -d
disk- ------------reads------------ ------------writes----------- -----IO------
       total merged sectors      ms  total merged sectors      ms    cur    sec
ram0       0      0       0       0      0      0       0       0      0      0
...
ram15      0      0       0       0      0      0       0       0      0      0
hda     3380   3291  120332   40344   2541   3559   48800  123464      0     34
hdc        0      0       0       0      0      0       0       0      0      0
fd0        0      0       0       0      0      0       0       0      0      0
md0        0      0       0       0      0      0       0       0      0      0
```

Pour chaque partition

```
# vmstat -p /dev/hda1
hda1          reads   read sectors  writes    requested writes
                4943     116970       6340      50720
# vmstat -p /dev/hda2
hda2          reads   read sectors  writes    requested writes
                 198       1584          0          0
```


Notes personnelles

5.8.5- Statistiques CPU et disque avec iostat

La commande **iostat** affiche des données en temps réel :

Avec l'option **-c**, donne un rapport sur l'utilisation du **CPU**.
Avec l'option **-d**, donne des informations sur les disques similaires à **sar -b**, **par blockdevice**.
Avec les options **-n** (et **-h**), donne des informations sur l'activité NFS.

On peut spécifier l'unité (par défaut en blocs/s) avec **-k** (Ko/s) ou **-m** (Mo/s).

5.8.6- Statistiques processeurs avec mpstat

La commande **mpstat** donne des informations sur l'utilisation du CPU (-u) et sur les interruptions (-I).
Comme **iostat**, elle travaille en temps réel.
On peut spécifier un processeur (-P) en particulier ou visualiser des statistiques complètes.

```
# mpstat -P ALL 1
Linux 2.6.32-279.9.1.el6.x86_64 (mailstore2.actilis.net)   16/09/2012   _x86_64_  (4 CPU)

23:23:07     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
23:23:08     all    0,25    0,00    0,25    0,00    0,00    0,00    0,00    0,00   99,50
23:23:08       0    1,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   99,00
23:23:08       1    0,99    0,00    0,99    0,00    0,00    0,00    0,00    0,00   98,02
23:23:08       2    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00  100,00
23:23:08       3    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00  100,00
```


Notes personnelles

5.8.7- Les modes totalisation

Statistiques concernant les disques

```
# vmstat -D
           20 disks
            6 partitions
         3389 total reads
         3296 merged reads
       120564 read sectors
        40432 milli reading
         2590 writes
         3610 merged writes
        49600 written sectors
       123572 milli writing
            0 inprogress IO
           34 milli spent IO
```


Notes personnelles

Statistiques générales

```
# vmstat -s
      4043712  total memory
      3987232  used memory
      1355384  active memory
      2333644  inactive memory
        56480  free memory
       672880  buffer memory
      2650548  swap cache
       522104  total swap
          512  used swap
       521592  free swap
     43625748 non-nice user cpu ticks
       793730 nice user cpu ticks
     13896548 system cpu ticks
   3910955199 idle cpu ticks
    301222483 IO-wait cpu ticks
       441734 IRQ cpu ticks
      2043517 softirq cpu ticks
            0 stolen cpu ticks
  23111440295 pages paged in
   2028628244 pages paged out
         2986 pages swapped in
         3241 pages swapped out
    437422225 interrupts
   3456327316 CPU context switches
   1276893242 boot time
     55312540 forks
```


Notes personnelles

5.9- System Activity Reporting

5.9.1- Principe du monitoring par sysstat

Le paquetage **Sysstat** fournit des commandes pour collecter et restituer des statistiques.

Des mesures sont acquises régulièrement (tâche planifiée par cron ou daemon), et peuvent être restituées à la demande.

Collecte et gestion des données

sadc : (system activity data collector) il acquière et stocke les données dans des fichiers (non texte) :
Ils sont dans **/var/log/sysstat** sous Debian et sous **/var/log/sa/** sous RedHat.
Leur nom est **saXX** (avec XX = numéro du jour dans le mois).

sa1 : script d'appel à **sadc** (prévu pour être lancé par *cron* ou par un *timer* systemd).
sa2 : production et stockage d'un rapport (**sarXX**) journalier (XX = numéro du jour dans le mois)

Par défaut, 7 jours d'historique sont conservés (voir HISTORY dans **/etc/sysconfig/sysstat**).

Planification standard (RedHat ≤ 7)

```
# run system activity accounting tool every 10 minutes
*/10 * * * * root /usr/lib/sa/sa1 1 1
# generate a daily summary of process accounting at 23:53
53 23 * * * root /usr/lib/sa/sa2 -A
```


Notes personnelles

5.9.2- Afficher les données avec sar

sar : collecte et affiche des enregistrements statistiques préalablement acquis dans des fichiers.
sadf : affiche les données dans des formats computer-readable (-d = csv, -x = xml, -p = format mangeable facilement par Awk), permettant leur post-traitement, leur stockage en base de données...

Sans option, **sar** peut montrer un pic d'activité (CPU, ou disque).

```
12:00:01 AM       CPU     %user     %nice   %system   %iowait    %steal     %idle
...
10:26:01 AM       all      0.28      0.00      0.17      0.46      0.00     99.09
10:28:01 AM       all      0.44      0.00      0.26      1.18      0.00     98.13
10:30:01 AM       all      0.19      0.00      0.15      0.38      0.00     99.28
10:32:01 AM       all      0.86      0.00      0.36     14.92      0.00     83.86
10:34:01 AM       all      0.24      0.00      0.15      0.19      0.00     99.43
10:36:01 AM       all      0.30      0.00      0.18      0.76      0.00     98.76
10:38:01 AM       all      0.39      0.00      0.21      0.96      0.00     98.45
10:40:01 AM       all      0.22      0.00      0.16      0.13      0.00     99.49
...
Average:          all      0.24      0.00      0.12      0.27      0.00     99.37
```

On peut avoir plus de détails sur ce pic, grâce à **sar -b** ou **sar -d**, uniquement si **sadc** est lancé avec l'option **-d** (RHEL 5, sysstat V7)
-S DISK (RHEL 6, sysstat V9), ou **-S XDISK**, pour collecter des statistiques sur les partitions.

Cela engendre du volume et il faut l'activer si on le souhaite ces options dans **/etc/sysconfig/sysstat** (SADC_OPTIONS). En standard, elle ne l'est pas, et "**sar -d**" n'a donc pas de données à produire.

Notes personnelles

5.9.3- Usages de la commande sar

Avec toutes les options ci-dessous, on peut :
spécifier une plage horaire (**-s** debut, **-e** fin)
spécifier un fichier à analyser (**-f**)

Les CPU : sar **-u** (c'est la valeur par défaut)
La run Queue : sar **-q**
La mémoire : sar **-R** et **-r**,
Le swap : sar **-S** (taux d'utilisation du swap), sar **-W** (swap in / swap out)

Détails concernant les accès disques
Avec sar **-b**, on obtient les détails suivants :

```
22:32:01          tps      rtps      wtps   bread/s   bwrtn/s
22:34:01        90,25     74,99     15,26   8561,46    196,01
22:36:01        96,87     73,08     23,79   8443,87    363,41
22:38:01        90,01     73,39     16,62   8513,15    224,43
Moyenne:       542,63    341,34    201,29   8234,23   2978,91
```

Avec sar **-d**, (aidé de l'option -p : pretty device names) on obtient un détail par **bloc-device**.
Les noms des devices sont associés aux noms logiques dans le fichier **/etc/sysconfig/sysstat.ioconf**.

Détails concernant le réseau : sar **-n** XXX
XXX : DEV, EDEV, NFS, NFSD, SOCK, ALL
et IP, EIP, ICMP, EICMP, TCP, ETCP, UDP, SOCK6, IP6, EIP6, ICMP6, EICMP6 et UDP6.

Notes personnelles

5.9.4- Étude de cas

Depuis quelques heures, la machine en question (une machine de sauvegarde / archive) émet des archives de plusieurs centaines de Go (via rsync), sa connexion (100Mbps) est saturée. À 1h 35, la machine émettrice démarre dans le même temps une sauvegarde d'un autre serveur (elle acquière des donnés via rsync).

```
#  sar -s 01:30:00 -e 01:40:00
01:30:01          CPU     %user     %nice   %system   %iowait    %steal     %idle
01:32:01          all     11,30      0,00      7,36      1,05      0,00     80,29
01:34:01          all     11,22      0,00      7,35      1,25      0,00     80,17
01:36:01          all     10,24      0,00      6,60     27,41      0,00     55,75
01:38:01          all      8,72      0,00      5,52     57,51      0,00     28,25
01:40:01          all      8,87      0,00      5,58     56,67      0,00     28,88
01:42:01          all      8,53      0,00      5,57     58,21      0,00     27,69
Moyenne:          all      9,81      0,00      6,33     33,69      0,00     50,17
```

```
# sar -s 01:30:00 -e 01:40:00 -n DEV  | grep -E 'IFACE|eth0'
01:30:01     IFACE   rxpck/s   txpck/s    rxbyt/s    txbyt/s   rxcmp/s   txcmp/s  rxmcst/s
01:32:01      eth0   3671,74   8065,96 247826,11 12209226,84      0,00      0,00      0,00
01:34:01      eth0   3660,54   8043,09 246564,22 12175792,30      0,00      0,00      0,00
01:36:01      eth0   3322,98   7332,28 224129,85 11071811,80      0,00      0,00      0,00
01:38:01      eth0   2868,84   6373,37 193907,81  9588567,57      0,00      0,00      0,00
01:40:01      eth0   2905,13   6453,16 195874,55  9710040,24      0,00      0,00      0,00
01:42:01      eth0   2821,96   6268,70 190119,56  9435780,77      0,00      0,00      0,00
Moyenne:      eth0   3208,53   7089,44 216403,21 10698552,61      0,00      0,00      0,00
```

À retenir : IOWAIT n'est donc pas forcément lié aux blocs-device !

Notes personnelles

```
# sar -s 01:30:00 -e 01:44:00 -b
01:30:01          tps      rtps      wtps   bread/s   bwrtn/s
01:32:01       642,10    628,10     14,00  68211,90    178,13
01:34:01       640,25    626,57     13,68  68018,13    171,38
01:36:01       616,06    586,36     29,70  62071,30    372,37
01:38:01       576,74    525,00     51,74  53840,71    598,26
01:40:01       581,37    533,37     47,99  54622,13    582,10
01:42:01       609,14    526,49     82,65  53206,91    961,75
Moyenne:       610,94    570,99     39,95  59995,38    477,27

# sar -s 01:30:00 -e 01:44:00 -q
01:30:01      runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15
01:32:01            1       112      0,64      0,56      0,56
01:34:01            2       113      0,67      0,64      0,58
01:36:01            1       121      1,49      0,88      0,67
01:38:01            2       121      2,29      1,42      0,89
01:40:01            2       121      2,32      1,71      1,06
01:42:01            2       120      2,40      1,93      1,21
Moyenne:            2       118      1,64      1,19      0,83

# sar -s 01:30:00 -e 01:44:00  -R
01:30:01      frmpg/s   bufpg/s   campg/s
01:32:01         0,21      0,20      1,27
01:34:01         1,91     -0,09     -4,50
01:36:01         0,06      4,22      2,02
01:38:01        -2,34     10,74     -6,08
01:40:01         0,92      8,97     -8,21
01:42:01        -0,96      8,63    -20,99
Moyenne:        -0,03      5,45     -6,08
```


Notes personnelles

5.10- Outils de mesure des performances

5.10.1- Statistiques

dstat - versatile resource statistics tool,
ifstat - InterFace STATistics Monitoring,
procps - /proc file system utilities,
tcpstat - network interface statistics reporting tool,
sysstat - sar, iostat and mpstat - system performance tools for Linux.

5.10.2- Benchmarking

bonnie++ - Benchmark suite for hard drive and file system performance,
hdparm - A utility for displaying and/or setting hard disk parameters.
iozone - IOzone Filesystem Benchmark,
iperf - Tool for measuring TCP and UDP bandwidth performance,
netperf - Performance testing tool for TCP/UDP,
netpipe - Protocol independent performance tool,
unixbench - BYTE's UNIX Benchmarks.

5.10.3- Monitoring interactif

top, **htop**, **iotop**, **atop...** - cpu, memory, disk and network monitoring

Notes personnelles

5.10.4- Stress de machine

stress - imposes a configurable amount of CPU, memory, I/O, or disk stress on a POSIX-OS.
fio - I/O benchmark and stress/hardware verification tool
iogen - I/O generator
spew - I/O performance measurement and load generation tool

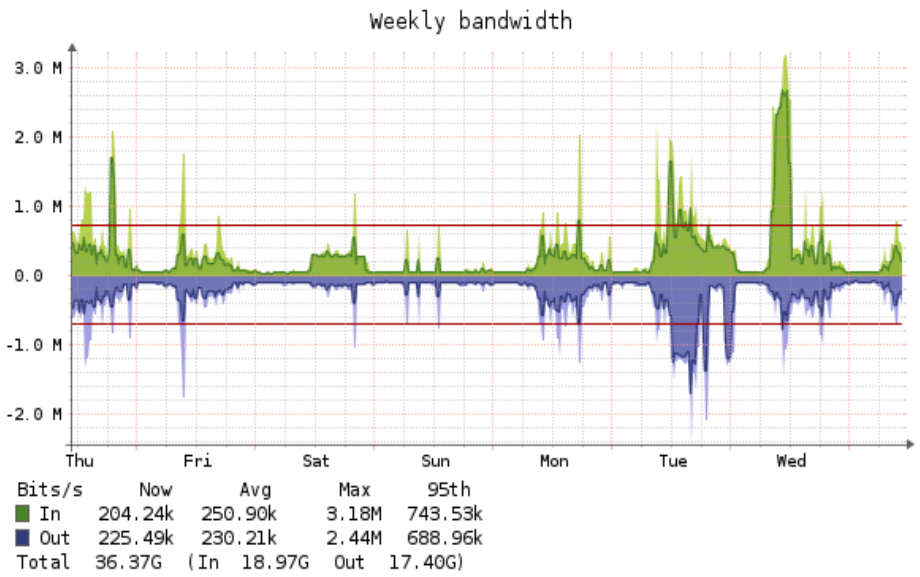
Voir aussi sur http://www.acnc.com/benchmarks.html

Notes personnelles

5.11- Cacti : Présentation

Cacti (www.cacti.net) est un logiciel de métrologie : mesure de performances.

Il est basé sur **RRDTool** (outil de gestion de bases de données cycliques).



Cacti est considéré comme une interface à RRDTool, qui stocke (de manière cyclique) des données de statistiques.

Ces données sont acquises par SNMP ou par des scripts ou programmes s'exécutant sur les cibles.

Cacti peut produire de nombreux graphiques, grâce à des templates, que l'on trouve facilement sur le forum (http://forums.cacti.net/).

C'est un Logiciel Libre utilisant un serveur web (PHP) et une base de données MySQL.

Notes personnelles

5.12- Cacti : installation et découverte

Le package est fourni par CentOS en standard, comme ses quelques dépendances...

- Php & php-cli
- Php-mysql & php-pdo
- Php-gd

```
# yum -y install cacti
```

Petit tour d'horizon :

```
# rpm -ql cacti | grep /etc
/etc/cacti
/etc/cacti/db.php
/etc/cron.d/cacti
/etc/httpd/conf.d/cacti.conf
/etc/logrotate.d/cacti
```

La configuration de la base MySQL accédée par Cacti est faite dans **/etc/cacti/db.php**[25]

C'est une **tâche cron** qui va peupler les bases de données (elle est inhibée par défaut)

La configuration pour Apache est prête, et un redémarrage de "httpd" est nécessaire.

D'après la configuration logrotate, Cacti émet des logs dans **/var/log/cacti/cacti.log**.

25 Attention, il faut donc déjà disposer d'un serveur MySQL !

Notes personnelles

5.13- Cacti : configuration

5.13.1- Base de données

Commencer par créer une base de données sur un serveur MySQL.

```
# mysql -u root -p
mysql> create database cacti;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL ON cacti.* TO cactiuser@'%' IDENTIFIED BY 'le-password';
Query OK, 0 rows affected (0.06 sec)
mysql> flush privileges;
Query OK, 0 rows affected (0.07 sec)
```

Côté configuration Cacti, adapter la configuration dans **/etc/cacti/db.php**

```
/* make sure these values refect your actual database/host/user/password */
$database_type = "mysql";
$database_default = "cacti";
$database_hostname = "le-serveur-mysql";
$database_username = "cactiuser";
$database_password = "le-password";
$database_port = "3306";
$database_ssl = false;
```

Créer le schéma de données dans la base :

```
# mysql -h le-serveur-mysql -u cactiuser -p  cacti < /usr/share/doc//cacti*/cacti.sql
```


Notes personnelles

5.13.2- Moteur & Polling

Le cron est par défaut inhibé. Il faut donc dé-commenter la ligne suivante de **/etc/cron.d/cacti**.

```
*/5 * * * *     cacti   /usr/bin/php /usr/share/cacti/poller.php > /dev/null 2>&1
```

5.13.3- Application Web

Les permissions d'accès sont par défaut limitées à "localhost".

On règle cela dans **/etc/httpd/conf.d/cacti.conf**, notamment dans la partie ci-dessous :

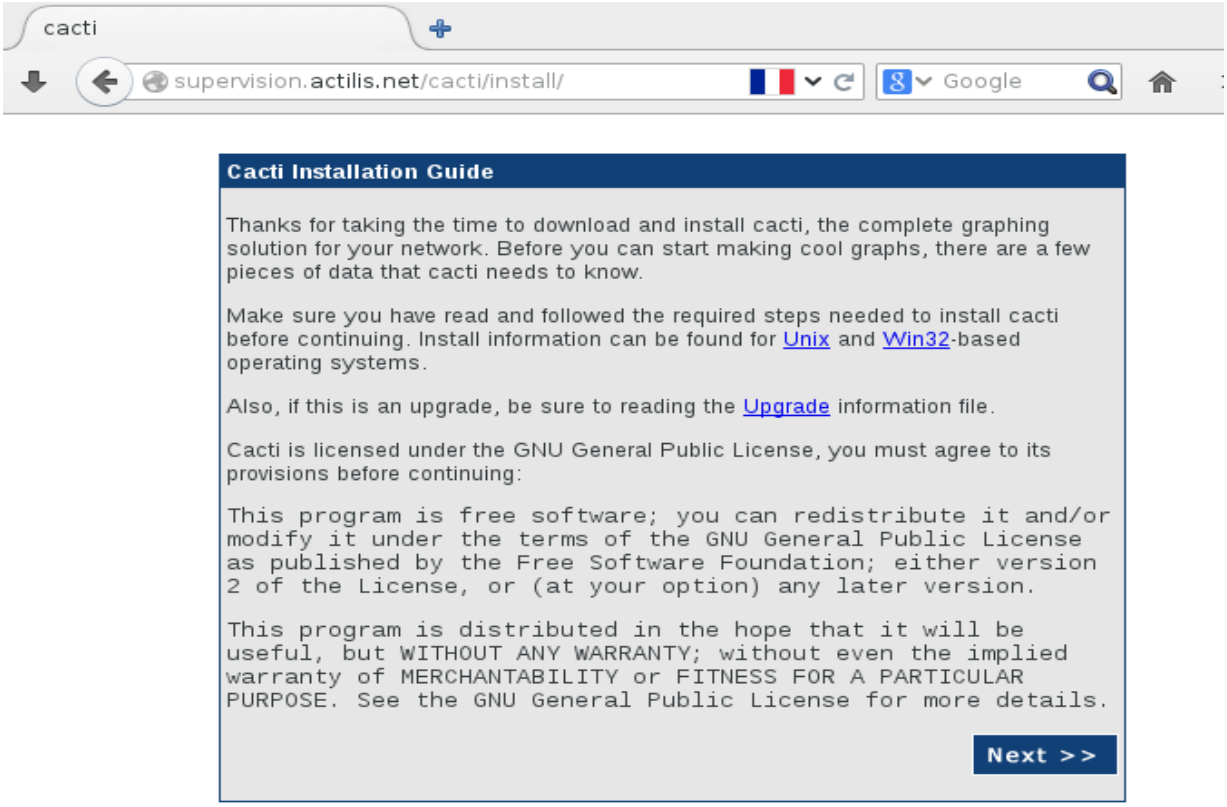
```
Alias /cacti    /usr/share/cacti

<Directory /usr/share/cacti/>
        <IfModule mod_authz_core.c>
                # httpd 2.4
                Require host localhost
        </IfModule>
        <IfModule !mod_authz_core.c>
                # httpd 2.2
                Order deny,allow
                Deny from all
                # Allow from localhost
                Allow from adresse-de-votre-reseau-client
        </IfModule>
</Directory>
```

L'URL sera donc **http://votre-serveur/cacti/**

Il suffit ensuite de se connecter à l'URL http://votre-serveur-cacti/cacti/ pour être redirigé vers la page d'installation (/install/)... Et de vérifier les différents écrans suivants.

Notes personnelles

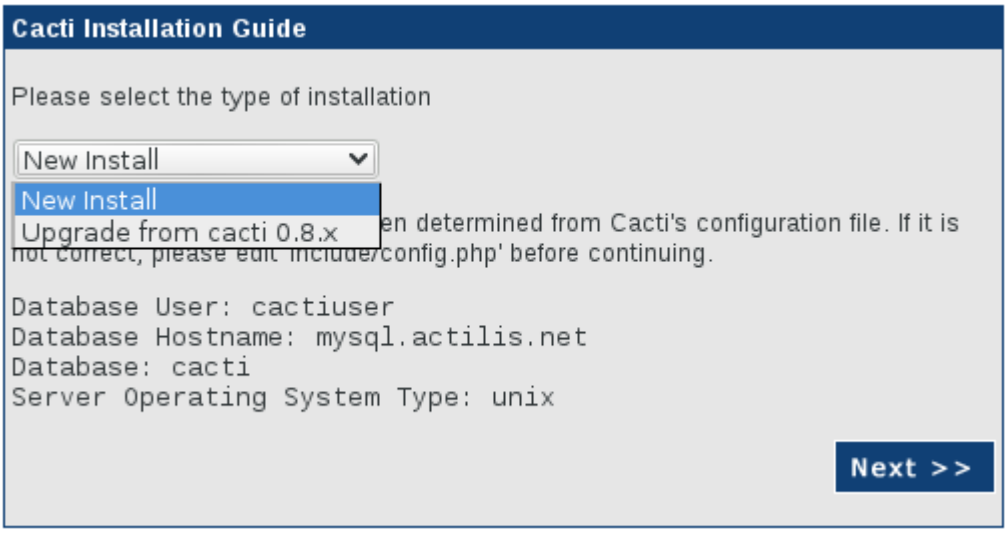


On retrouve ensuite les paramètres saisis dans **/etc/cacti/db.php**

Notes personnelles

Notes personnelles

Notes personnelles



Notes personnelles

Puis on valide les pré-requis.

Si un des éléments est manquant (théoriquement impossible avec la version "rpm" de Cacti), alors il faudra réaliser l'installation manuellement.

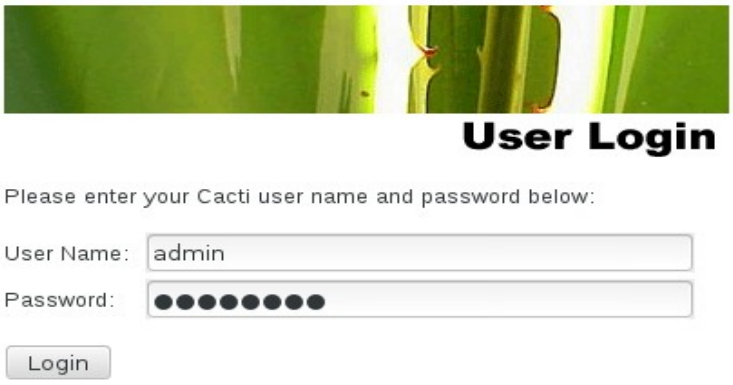
Ici, la machine utilisée est déjà serveur Nagios et dispose donc des éléments concernant SNMP.

Notes personnelles

5.14- Cacti : prise en main

5.14.1- Accès à l'interface

En sortie d'installation, l'écran d'accueil apparaît :

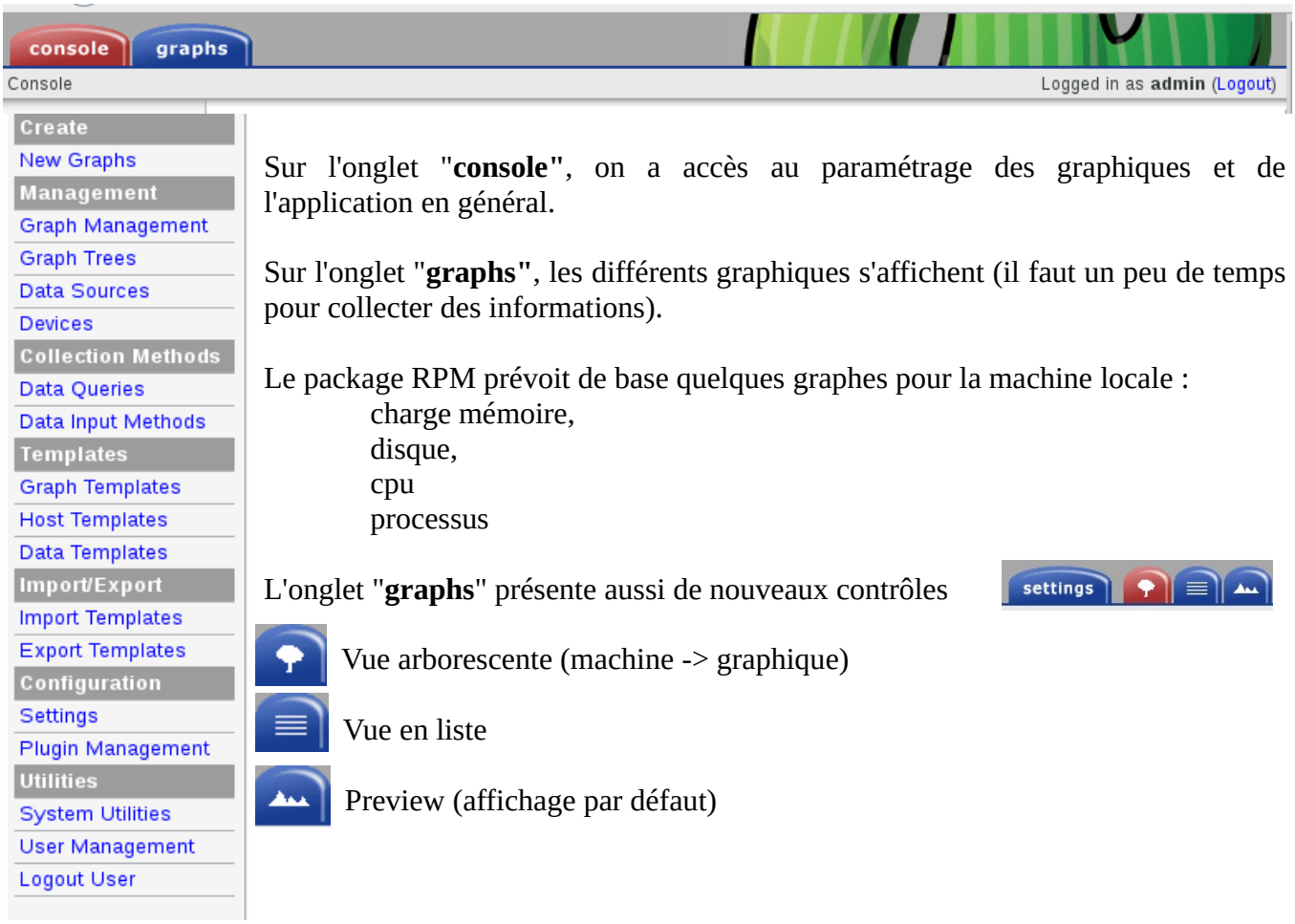


Le login par défaut est "admin", tout comme le mot de passe.
À la première connexion, il sera demandé de le changer.

En cas de problème, passez la requête suivante dans la base de données, elle mettra "passw0rd" comme mot de passe pour "admin" :

```
update user_auth set password=md5('passw0rd') where username='admin';
```


Notes personnelles



Sur l'onglet "**console**", on a accès au paramétrage des graphiques et de l'application en général.

Sur l'onglet "**graphs**", les différents graphiques s'affichent (il faut un peu de temps pour collecter des informations).

Le package RPM prévoit de base quelques graphes pour la machine locale :

- charge mémoire,
- disque,
- cpu
- processus

L'onglet "**graphs**" présente aussi de nouveaux contrôles

 Vue arborescente (machine -> graphique)

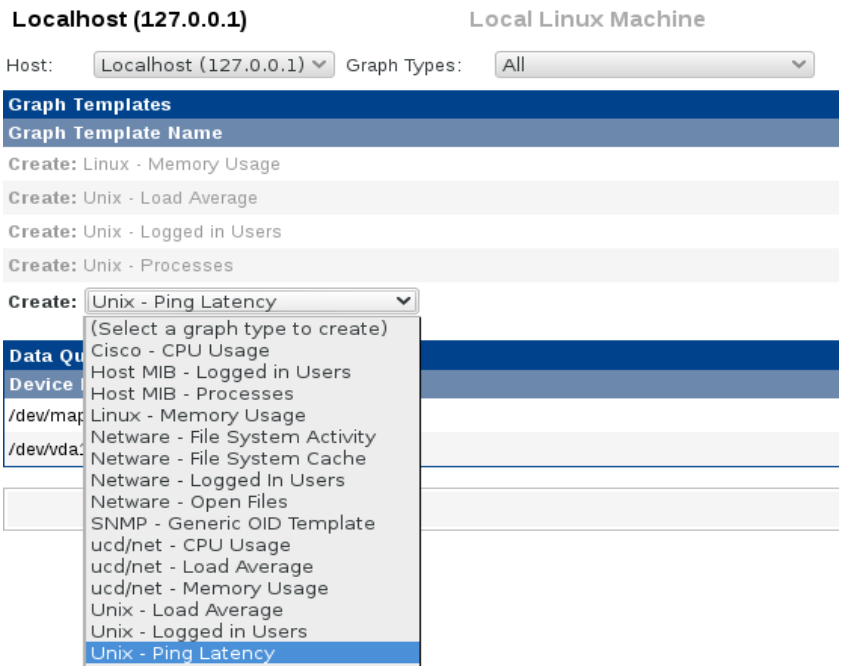
 Vue en liste

 Preview (affichage par défaut)

Notes personnelles

5.14.2- Création de graphiques

On crée un nouveau graphique par "Create / New Graphs"



La première chose à sélectionner est alors le "Host"...

Puis ensuite le type de graph.

Puis ensuite : 

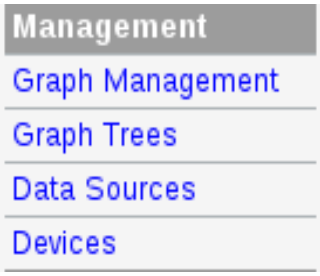
Et après paramétrage de ce graphique (choix de la couleur, du titre, etc...)...

... de nouveau 

Il n'y qu'un "**Host**" dans la liste, on aurait pu en créer un autre ... 

Notes personnelles

5.14.3- Organisation des graphiques



Graph Management : supprimer, déplacer, modifier...

Graph Trees : organiser les graphiques en arbres, contenant des "Items" :
- **Header** : juste un titre
- **Graph** : un graphique directement
- **Host** : nue machine, qu'il faut d'abord créer (Device)

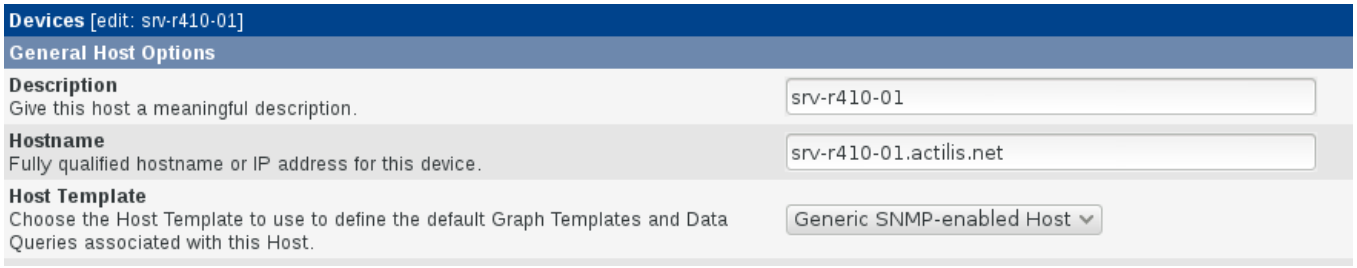
Data Sources : manipuler les paramètres des bases RRD

Devices : manipuler les machines (Hosts)

En principe, on crée des "**Devices**", et ensuite, on les associe à des templates pour créer des graphiques..

puis 

puis valider le formulaire.

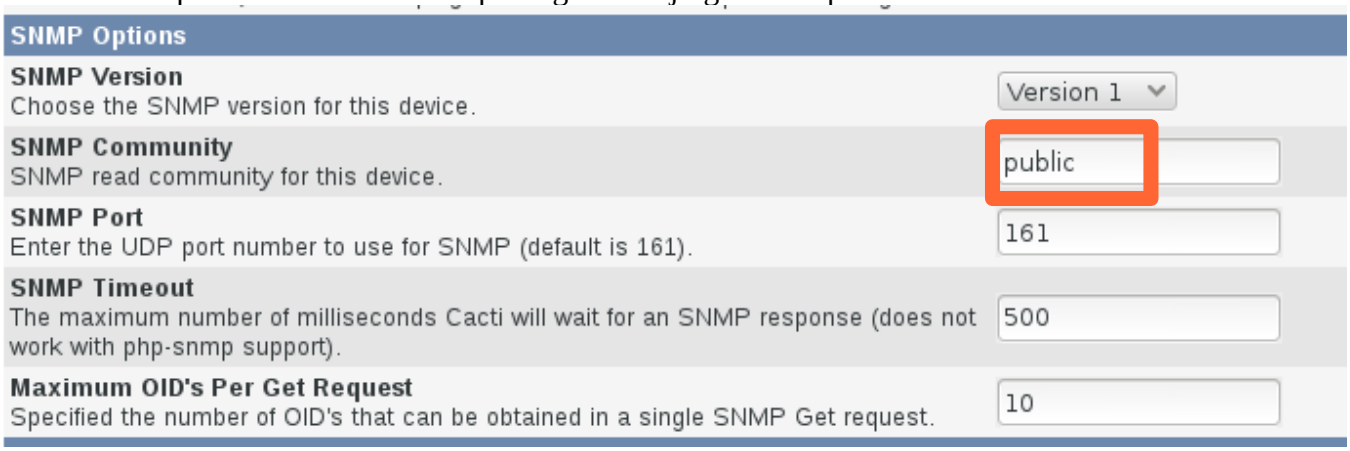


Notes personnelles

Cacti utilise en principe SNMP pour réaliser le polling des hôtes distants.

Il faut donc au préalable **installer un agent SNMP** sur les cibles.

Vérifiez les options SNMP et le fait que l'agent soit joignable depuis le serveur Cacti...



Sinon après l'enregistrement, la sanction est la suivante...

Notes personnelles

5.15- Installer un agent SNMP

Sur une machine à superviser...

RedHat / CentOS :

```
# yum -y install net-snmp[26]
```

Dans **/etc/snmp/snmpd.conf**, contentez-vous de ce contenu :

```
# #       sec.name  source                 community
com2sec  srvcacti ip-du-serveur-cacti/32   public
#
# Map the security name into a group name:
# #       groupName      securityModel securityName
group     MyROGroup         v1        srvcacti
#
# Create a view for us to let the group have rights to:
# #       name           incl/excl     subtree         mask(optional)
view    all             included       .1                80
# Grant the group read-only access to the systemview view.
# #       group      context sec.model sec.level   prefix   read    write  notif
access    MyROGroup    ""      any       noauth    exact     all     none none
#
disk    /       100000
```

Démarrer le service... et vérifier le firewall (port udp/161) et l'adresse d'écoute de l'agent.

```
# service snmpd start ; chkconfig snmpd on
```

26 Debian / Ubuntu : # apt-get install **snmpd**

Notes personnelles

6- Blocage, crash et dépannage d'urgence

Notes personnelles

6.1- Fonctionnement détaillé du boot

6.1.1- Le boot sector "msdos" / MBR

Le **MBR** (premier secteur d'un disque, historiquement appelé "boot sector").

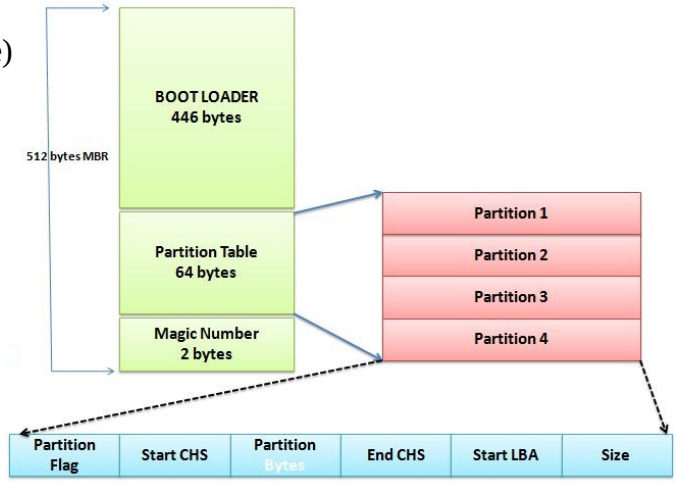
Il a une structure spécifique qu'on a coutume de représenter comme cela (détail en page suivante) :

446 premiers octets : « **IPL** » : peut contenir du code exécutable

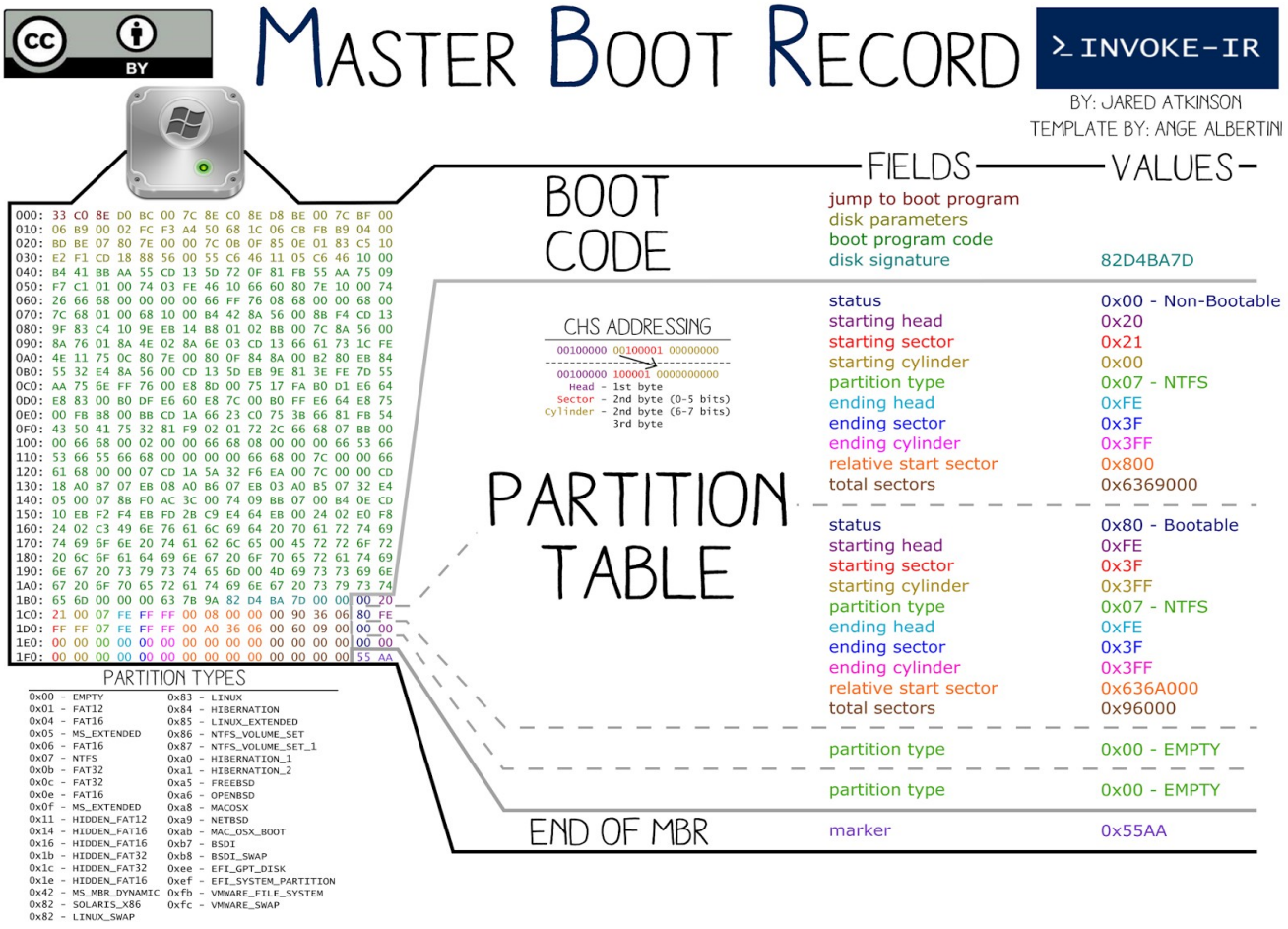
64 octets suivants : table de partition (4 partitions primaires au maximum)

2 derniers octets : magic number (signature)

Soit un total de **512** octets.



Notes personnelles



Notes personnelles

6.1.2- Le démarrage d'un PC

Lorsque le système boote sur un disque dur :

1/ le BIOS cherche et exécute le programme d'amorçage du **MBR** : l'IPL

il est pris sur le boot-sector du disque maitre de l'interface primaire.

2/ Si il n'y a pas de programme d'amorçage sur le MBR,

alors le BIOS exécute celui du premier secteur de la partition "active" (bootable).

C'est le cas pour booter les OS Microsoft.

3/ Quand il n'y en a pas là non plus : "OS not found !"

Ce qu'on doit trouver est appelé le **chargeur** (le "loader") :

LILO
GRUB
...

Notes personnelles

6.1.3- Rôle du chargeur de démarrage

Pour démarrer un système, on doit utiliser un **chargeur de démarrage** ("Boot - Loader")

Il permet la cohabitation de plusieurs OS, ou permet de démarrer un OS de différentes manières.

C'est le chargeur qui **charge le noyau Linux, et lui passe des paramètres**.

LILO : C'était le chargeur historiquement utilisé par Linux,
Il n'est plus distribué en standard dans le monde RedHat et n'évolue plus depuis 2015,
Il a été remplacé par GRUB dans de nombreuses distributions.

GRUB & GRUB2 : Interpréteur de commandes paramétrable en dynamique lors du boot

Évite la « ré-installation » lors des modifications d'options / de menu[27]
GRUB 2 est aujourd'hui le chargeur de démarrage des distributions modernes.

Autres chargeurs : Même si GRUB est le chargeur standard, il existe d'autres :
U-Boot : https://www.denx.de/wiki/U-Boot/
BareBox : https://www.barebox.org/

Un tutoriel générique :
http://etutorials.org/Linux+systems/embedded+linux+systems/Chapter+9.+Setting+Up+the+Bootloader/

27 Avantage et simplicité sont perdus avec GRUB 2 ! GRUB 2 offre d'autres intérêts qui compensent la "difficulté" de configuration

Notes personnelles

6.1.4- Noyau et initrd

Le noyau doit être capable de **monter** son système de fichiers racine (accueilli sur /)

La partition concernée est connue grâce au paramètre "***root=*"** passé par le chargeur.

Pour y accéder, **le noyau a besoin de pilotes** (qu'il peut charger au cours de sa vie)
du contrôleur disque,
pilote pour lire le format du filesystem, ...

sinon "**unable to mount root filesystem : kernel panic !**"

On passe donc souvent par un "intird" (Initial Ramdisk) :

Le ramdisk initial, **dont le noyau teste toujours la présence** en mémoire**,** sert de "mini système" dans le but de charger ces pilotes.

Cet "**initrd**" est lui aussi chargé par le chargeur.

Notes personnelles

6.1.5- Le premier processus

À court terme, le noyau va **créer un premier processus**, pour exécuter **/usr/lib/systemd/systemd**[28]

Le projet "**systemd**" est le système de démarrage des distributions modernes... RedHat / CentOS / Fedora… Debian, Ubuntu 15.4.

Reprendre le contrôle si on a perdu le mot de passe de root : c'est ici que ça se joue !

1/ Ajouter ceci à la ligne de commande du noyau dans GRUB ("kernel" ou "linux16") : **init=/bin/bash**

2/ Booter.

3/ Quand on a accès au shell,
Passer le système de fichiers en RW : **mount -o remount /**
Changer le mot de passe : **passwd**
Si SeLinux est actif, forcer une réfection des labels (au reboot) : **touch /.autorelabel**
Repasser le système de fichier racine en RO : **mount -o remount,ro /**

4/ Continuer un démarrage normal en passant la main au vrai "init" : **exec /sbin/init**
ou **exec /usr/lib/systemd/systemd**, le premier étant un lien vers le second.

28 **Avant "systemd", /sbin/init** était le programme exécuté par le premier processus (ou celui précisé par le paramètre "***init=***")
S'il y a un Ramdisk initial, c'est /**init** qui est exécuté, puis, une fois que "/" est monté, il "*switche*" sur le "vrai /".

Notes personnelles

6.2- Le démarrage, les runlevels/targets, les services/unités

Le démarrage est depuis 2014 (CentOS 7, Debian 8, ...) géré par **systemd**.

Il succède à **SysVInit** (/etc/inittab…) et **Upstart** (/etc/init + initctl).

Il définit des actions permettant grâce à des "unités" (sortes de fiches de tâche), de :

1/ Construire le système de base : monter les disques, nommer la machine, activer le swap...

2/ Démarrer des services : établir une configuration cible (**target**)

C'est un état de la machine, dans lequel certains processus existent, et d'autres non.

Avant **systemd**, on appelait cela un **runlevel**.

Une cible est composée d'unités ("units"… voir **systemctl list-units** ...).

Les unités sont des actions à exécuter, ou des services à lancer

Certaines unités dépendent d'autres.

3/ Démarrer des tâches supervisées : moyen de se connecter, environnement graphique…

Cela passe par une surveillance de certains processus, et une relance en cas d'arrêt

La commande pour piloter les différentes tâches est **systemctl**… list-units, enable, disable, start, stop...

Notes personnelles

6.3- GRUB 2

La plupart des distributions majeures a migré vers **GRUB2**, totalement réécrit pour palier le manque de flexibilité de la première version de GRUB.

6.3.1- Améliorations de GRUB 2

Fichier de configuration écrit dans un langage proche du shell : **flexibilité** sans précédent ;

Support des **jeux de caractères étendus** : les entrées dans de menu de démarrage peuvent désormais être écrites dans la langue de l'utilisateur ;

Support des tables de partitions modernes telles que GPT ;

Architecture modulaire permettant de supporter aisément de nombreux systèmes de fichiers et technologies de stockage supplémentaires : RAID, **LVM**, etc.

L'installation de RHEL 7 refuse cependant l'absence de partition "/boot" si "/" est sur LVM.

La configuration de GRUB 2 est générée de façon modulaire,

Permet à d'autres paquets d'y ajouter des entrées supplémentaires…

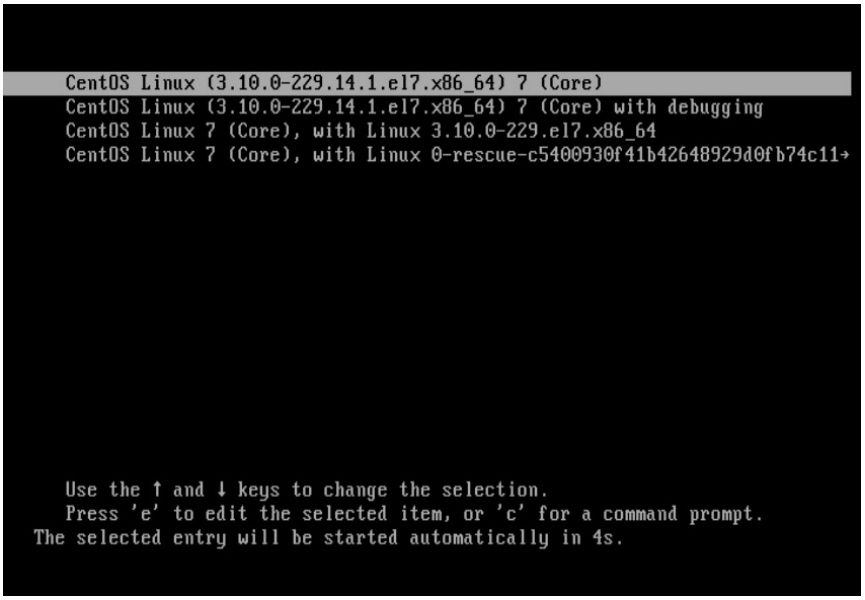
C'est un simple dépôt de fichier dans le répertoire de configuration **/etc/grub.d**

Mais...

Rend inutile la modification manuelle de cette configuration.

Notes personnelles

6.3.2- Le menu proposé



Une ligne par choix décrit dans la configuration.

On peut trouver plusieurs lignes pour le même système, prévoyant de le démarrer avec des versions de noyau ou options différentes.

Un mode de récupération (rescue) est par exemple souvent proposé.

GRUB est souvent configuré pour démarrer un de ces choix au bout d'un certain délai.

On peut interrompre le compte à rebours par n'importe quelle touche pour faire apparaître le menu.

Le choix proposé par défaut et la durée du compte à rebours sont des éléments paramétrables.

Lorsqu'un choix est sélectionné, on peut le valider et démarrer normalement (par Entrée), ou alors l'éditer (par « e ») ou passer en ligne de commande (par « c », voir le paragraphe « Utiliser GRUB2 en mode manuel »).

Notes personnelles

6.3.3- Éditer le menu avant de démarrer

Suite à un appui sur « **e** », les différentes lignes concernant le choix sélectionné sont affichées dans un éditeur minimal similaire à Emacs, et peuvent donc être modifiées au besoin.

Une fois dans cet éditeur,
« **CTRL-x** » : booter
« **ESC** » : annuler
« **CTRL-c** » : ligne de commande

La touche **Tabulation** offre des possibilités de complétion, sur le premier mot (la commande) et le second : fichiers, périphériques...

Les modifications faites ici ne sont pas enregistrées dans le fichier de configuration.

La documentation détaillée de tous les mots clés :
http://www.gnu.org/software/grub/manual/grub.html#Command_002dline-and-menu-entry-commands

Notes personnelles

6.3.4- Utiliser GRUB2 en mode manuel

Il faut fournir à Grub2 toutes les informations permettant de démarrer le système.

Ce sont celles de la capture d'écran du paragraphe « Éditer le menu avant de démarrer » :

les lignes **insmod** chargent les modules GRUB2 nécessaires : part_msdos, xfs;

la ligne **set root** ou **search** indiquent à GRUB2 où il peut trouver les fichiers nécessaires ;

la ligne **linux** (ou linux16) demande à GRUB2 de charger le noyau Linux dont le nom est fourni, en lui passant certaines options : **root=**..., **quiet**, etc. ;

la ligne **initrd** (ou initrd16) demande à GRUB2 de charger le ramdisk initial cité.

L'important est de lire la réponse à chaque étape.

Il suffit ensuite de demander à GRUB2 de booter, grâce à la commande "**boot**", et le système démarre la configuration qu'on a entrée.

Notes :

Booter en utilisant Grub de cette manière n'impose pas de l'installer sur le disque au préalable.
Cet exemple fonctionnerait sur une machine qui démarre avec un "Grub vierge sur CD".
Tout ceci est plus facile avec un keymap français… Celui-ci n'est pas installé par défaut.

Notes personnelles

6.4- Le fichier de configuration de GRUB2

Le fichier lu par GRUB2 est **dynamiquement créé** et son édition n'est donc pas recommandée.

```
# ls -l /etc/grub2.cfg
lrwxrwxrwx. 1 root root 22  5 nov.  13:15 /etc/grub2.cfg -> ../boot/grub2/grub.cfg
```

Il est produit par la commande suivante : **grub2-mkconfig** (ou **update-grub** sous Debian)

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Elle s'appuie sur les directives de **/etc/default/grub** et sur **les "scripts" du répertoire /etc/grub.d**.

```
# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=yes
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="vconsole.keymap=fr rd.lvm.lv=vg00/lv_swap vconsole.font=latarcyrheb-sun16
rd.lvm.lv=vg00/lv_root crashkernel=auto  rhgb quiet"
GRUB_DISABLE_RECOVERY="true
```

Contenu du répertoire **/etc/grub.d** :

```
# ls  /etc/grub.d
00_header  10_linux      20_ppc_terminfo  40_custom  README
00_tuned   20_linux_xen  30_os-prober     41_custom
```


Notes personnelles

6.5- Modifier le menu de GRUB2

6.5.1- Modification du "timeout"

C'est le paramètre **GRUB_TIMEOUT** du fichier **/etc/default/grub**.

6.5.2- Le choix par défaut dans le menu

C'est **grub2-editenv list** qui l'affiche. Pour le définir :

Le paramètre **GRUB_DEFAULT** (/etc/default/grub) peut prendre une de ces 3 valeurs :

1/ le **numéro** d'une entrée (commençant à 0)

2/ le **libellé** d'une entrée (***grep ^menuentry /boot/grub2/grub.cfg | cut -d"'" -f2***)

3/ le mot "**saved**", dans ce cas, le choix par défaut peut être :

spécifié par la commande **grub2-set-default** (numéro ou libellé)

ou

mémorisé au boot si le paramètre **GRUB_SAVEDEFAULT** vaut **true.**

La commande **grub2-set-default** ne nécessite pas de nouvelle exécution de **grub2-mkconfig**.
En effet, elle enregistre, le choix dans l'*Environment Block* : **/boot/grub2/grubenv**.

Notes personnelles

6.5.3- Ajouter son propre noyau

On met à disposition dans /boot le fichier **vmlinuz** et le fichier **initramfs** :

```
# cd /boot
# cp vmlinuz-3.10.0-123.13.2.el7.x86_64 vmlinuz-3.10-test.x86_64
# cp initramfs-3.10.0-123.13.2.el7.x86_64.img initramfs-3.10-test.x86_64.img
```

On crée la configuration

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-123.13.2.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-123.13.2.el7.x86_64.img
Found linux image: /boot/vmlinuz-3.10-test.x86_64
Found initrd image: /boot/initramfs-3.10-test.x86_64.img
...
Found linux image: /boot/vmlinuz-0-rescue-58f7cfe98bdb404fb265fdc5d483c436
Found initrd image: /boot/initramfs-0-rescue-58f7cfe98bdb404fb265fdc5d483c436.img
done
```

On établit le choix par défaut

```
# grub2-set-default "CentOS Linux, with Linux 3.10-test.x86_64"
```

Vérification

```
# grub2-editenv list
saved_entry=CentOS Linux, with Linux 3.10-test.x86_64
```


Notes personnelles

6.5.4- Modifier une option de démarrage du noyau

Cela passe par l'édition de /etc/default/grub

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=yes
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="vconsole.keymap=fr rd.lvm.lv=vg00/lv_swap vconsole.font=latarcyrheb-sun16
rd.lvm.lv=vg00/lv_root crashkernel=auto quiet elevator=deadline"
GRUB_DISABLE_RECOVERY="true"
```

Ici, on a modifié la ligne de commande du noyau (GRUB_CMDLINE_LINUX) :

retrait de "rhgb"

ajout de "elevator=deadline"

Il faut "recompiler" grub.cfg.

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Après le reboot…

```
# dmesg | grep sched
[    0.526578] io scheduler noop registered
[    0.526578] io scheduler deadline registered (default)
[    0.526589] io scheduler cfq registered
```


Notes personnelles

6.5.5- Modifier "juste" une entrée

L'inconvénient de la méthode précédente est qu'elle modifie toutes les entrées de type "Linux".

Pour modifier "juste une entrée", la démarche (qui est en fait un ajout d'une entrée) est la suivante :

Repérer le bloc d'une entrée à modifier grâce à un **grub2-mkconfig (| less ...)**

Copier / coller le texte de cette entrée vers le fichier /etc/grub.d/40_custom

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries. Simply type the
# menu entries you want to add after this comment. Be careful not to change
# the 'exec tail' line above.
menuentry 'CentOS Linux, with Linux 3.10.0-123.el7.x86_64' --class centos --class gnu-linux --class
gnu --class os --unrestricted $menuentry_id_option 'gnulinux-3.10.0-123.el7.x86_64-advanced-
f7f3160b-f78f-4d1e-a78b-d77485733d6e' {
...
```

Attention, le collage doit se faire **au plus tôt à partir de la 3ème ligne**.

C'est grâce à la **seconde ligne** que la 3ème et suivantes intègreront grub.cfg…

… après le prochain **grub2-mkconfig**

Notes personnelles

6.6- Sécuriser le menu de GRUB2

GRUB2 peut être protégé par un mot de passe en clair ou chiffré.

L'accès à un choix du menu peut être réservé à un "super user" et soumis un mot de passe,
ainsi qu'à certains utilisateurs, listés dans certains choix.

6.6.1- Déclaration d'utilisateurs GRUB

Créer un fichier **/etc/grub.d/01_users** (présent si un mot de passe GRUB a été prévu à l'installation).

Y ajouter

```
cat <<EOF
set superusers="francois"
password francois passfanch
password philippe passphil
EOF
```

François est superuser, il peut donc accéder à toutes les entrées.

On peut spécifier (dans **/etc/grub.d/40_custom**) des clauses "menuentry" avec des paramètres :
--unrestricted : une entrée est accessible à tout le monde
--users philippe : Philippe a accès à cette entrée (ainsi que François, qui est "superuser")

Puis… **grub2-mkconfig**...

Notes personnelles

6.6.2- Production d'un mot de passe chiffré

On utilise la commande **grub2-mkpasswd-pbkdf2**.

```
# grub2-mkpasswd-pbkdf2
Entrez le mot de passe :
Entrez de nouveau le mot de passe :
Le hachage PBKDF2 du mot de passe est
grub.pbkdf2.sha512.10000.6A839D73D16E8DF7077CCEA338EE997CA1FE7949883753D31486B3F74AE1ED8607010DE1686
8FE4C783485B865732338164F56F20C7718473873703F54DA8979.1AF9A051AC0EFFA65EA1ABA1F9646DFC50F4263115066A
7E70E8768015B2088E73815AA28171E5F00668552D468CB25523A49C0FB3B10142EE35B284F08DEA59
```

Puis, à la place du mot de passe en clair, dans **/etc/grub.d/01_users**, on la renseigne :

```
cat <<EOF
set superusers="francois"
password_pbkdf2 francois grub.pbkdf2.sha512.10000.6A839D73D16E8DF7077CCEA338EE997CA1FE794
9883753D31486B3F74AE1ED8607010DE16868FE4C783485B865732338164F56F20C7718473873703F54DA8979
.1AF9A051AC0EFFA65EA1ABA1F9646DFC50F4263115066A7E70E8768015B2088E73815AA28171E5F00668552D
468CB25523A49C0FB3B10142EE35B284F08DEA59
password philippe passphil
EOF
```

Sous Debian, ces éléments doivent être écrits dans **/etc/grub.d/40_custom**.

Notes personnelles

6.7- Installer ou réinstaller GRUB2

GRUB2 est normalement installé pendant l'installation du système.

6.7.1- Installer GRUB 2 sur un disque ou une clé USB

La commande "**grub2-install**" permet de refaire le MBR du disque spécifié.

```
# grub2-install /dev/sda
Installing for i386-pc platform.
Installation terminée, sans erreur.
```

Ici, tout se passe bien, on a ré-installé GRUB2 sur le disque où il était présent.

6.7.2- Quand le système ne boote plus…

On peut réaliser la même opération depuis l'image ISO d'installation :

Booter sur l'image "Minimal", puis, **Troubleshooting** et ensuite "**Rescue a CentOS System**" :
À la proposition de montage des filesystems détectés : "Continue", puis 2 fois "OK".
Le système présent sur le disque est alors monté dans "/mnt/sysimage", où l'on "chroote"

```
# loadkeys fr
# chroot /mnt/sysimage
```

enfin, après avoir vérifié "/boot/grub2/devices.map"

```
# grub2-install /dev/sda
```


Notes personnelles

6.8- Le premier processus : (SysV)init, Upstart, Systemd

Jusqu'aux versions 5.x de RHEL, l'exécutable **/sbin/init** venait du projet "**SysVinit**".
Sur RHEL 6, il est remplacé par le programme "**Upstart**" soutenu par Ubuntu.
Depuis RHEL 7, c'est une autre approche : **Systemd**.

Le rôle de "**Init**", "**Upstart**", ou "**Systemd**" est similaire et peut être découpé en 3 étapes :
réaliser des actions de "**bas niveau**" (montage de systèmes de fichiers, activation du swap, ...),
lancer les services (des processus en arrière plan fournissant des services),
mettre en place d'un **moyen de se connecter** au système.

6.8.1- Les runlevels et les targets

Le premier processus lancé par le noyau est souvent appelé "process d'init", il sert à lancer d'autres processus, les services notamment, lors du démarrage du système.

Il gère différents modes dans lesquels certains services doivent exister et d'autres non.

Ces modes étaient appelés des **runlevels** dans "Init" et "Upstart" ou sont des **targets** dans "Systemd".

Les programmes exécutés dans les différents modes sont paramétrés :
par le fichier **/etc/inittab** dans le cas de **SysVinit**.
par les fichiers du répertoire **/etc/init** dans le cas de **Upstart**.
par des "***unités***" (des fichiers définis dans /lib/systemd/system) dans **systemd**.

Notes personnelles

6.8.1.1- Les runlevels

On connait le runlevel courant par la commande "**runlevel**" ou la commande "**who -r**".

Niveau	Rôle
0	Arrêt du système
1	Mode maintenance (single user)
2	Mode multi utilisateur de base (incluant TCP/IP)
3	Multi-utilisateur + RPC (pour NIS, NFS, par exemple)
4	Libre (en principe maintenu comme le 3)
5	Multi utilisateur complet (3) + XDM (X-Window)
6	Reboot

Le **runlevel par défaut** est passé en argument au noyau, ou à défaut lu dans **/etc/inittab** ("initdefault")

6.8.1.2- Les targets de Systemd

Ce sont les répertoires "***.target.wants**" de **/etc/systemd/system**.

```
# ls -ld *target*
drwxr-xr-x. 2 root root 4096  5 nov.  13:15 basic.target.wants
lrwxrwxrwx. 1 root root   37  5 nov.  13:21 default.target -> /lib/systemd/system/multi-user.target
drwxr-xr-x. 2 root root 4096  5 nov.  13:14 default.target.wants
drwxr-xr-x. 2 root root 4096  5 nov.  13:14 getty.target.wants
drwxr-xr-x. 2 root root 4096  5 nov.  13:15 multi-user.target.wants
drwxr-xr-x. 2 root root 4096  5 nov.  13:15 sockets.target.wants
drwxr-xr-x. 2 root root 4096  5 nov.  13:15 sysinit.target.wants
drwxr-xr-x. 2 root root 4096  5 nov.  13:14 system-update.target.wants
```

La "target" par défaut est celle pointée par **/etc/systemd/system/default.target**

Notes personnelles

6.8.2- SysVInit et le fichier inittab

Le fichier **/etc/inittab** définit notamment :

le **runlevel** choisi par défaut (**initdefault**)

l'appel à **rc.sysinit**, qui exécute les tâches de bas niveau
lancé en mode "***sysinit***" (= **wait**, mais uniquement lors d'un boot)

l'appel à **rc**, qui lance tous les services concernés par le runlevel choisi
lancé en mode "***wait***" (on attend qu'il rende la main pour continuer)
utilise des scripts de démarrage installés par les packages fournissant les services
ils sont stockés dans **/etc/init.d** (ou **/etc/rc.d/init.d)**
des liens symboliques sont créés dans /etc/rcX.d, où X est le runlevel concerné

l'appel à **rc.local**, en dernier script de démarrage exécuté (lien symbolique S99local)

le lancement de **processus monitorés par "init"**, comme les terminaux de connexion.
Lancés en mode "***respawn***" (si le processus rend la main, on le relance)

Quelques déclarations d'actions à exécuter sur évènement (**ctrlaltdel**, **powerfail**, **powerokwait**, ...).

Notes personnelles

Exemple de fichier "inittab" épuré de ses commentaires

```
# grep -vE '^#|^ *$' /etc/inittab
id:5:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
x:5:respawn:/etc/X11/prefdm -nodaemon
dns:345:respawn:/usr/sbin/named -u named -f
```

On avertit init d'un changement dans inittab par :

le signal HUP (kill -1 1)
ou par "init q" ou "telinit q".

Notes personnelles

6.8.3- Upstart et les jobs de /etc/init

Avec Upstart, le fichier **/etc/inittab** est presque vide (Red Hat) ou n'existe plus (Ubuntu).

Triggers : Le but est de ne plus utiliser plus les runlevels, mais d'associer des événements à des changements d'état ("**state**") qui se produisent dans le système et dont **init** est informé.

À l'ajout ou au retrait d'un périphérique : le système **udev** prévient le démon **init** de Upstart pour qu'il démarre ou arrête le service associé.

C'est une approche dynamique qui offre divers avantages :

- évite les changements de niveau ou interventions manuelles pour réaliser l'opération.
- améliore le temps de démarrage initial du système (moins de tâches à exécuter au boot)

Tâches : Le démon **init** exécute les fichiers de définition des jobs du répertoire **/etc/init**.

Services : Les packages concernant les services installent les scripts de démarrage dans **/etc/init.d** et les liens symboliques dans **/etc/rcX.d**.

Le système Upstart maintient la compatibilité avec les runlevels.

Notes personnelles

Les jobs sont dans le répertoire **/etc/init**. Il y a un fichier par Job,

ce sont des commandes à exécuter et les évènements qui déclenchent leur exécution.
Il existe deux types de job : **tâches**, et **services**.

```
# init-system-dbus - tell init

start on started messagebus

task
exec /bin/kill -USR1 1
```

Une tâche : exécutée une fois et retourne à l'état attente (« waiting ») après son exécution.

C'est un peu l'équivalent des commandes définies en ***wait*** dans le fichier **/etc/inittab**.

Un service : supervisé et doit être relancé (*respawned*) si il s'arrête de manière inattendue.

C'est l'équivalent des commandes définies en ***respawn*** dans le fichier **/etc/inittab**.

```
[0:root@srv-formation 18:28:37 ~]# cat /etc/init/httpd.conf
# httpd - Apache Web Server
#
# Starts httpd whitout forking

start on never and stopped rc RUNLEVEL=[2345]

stop on starting rc RUNLEVEL=[016]

console output
respawn
respawn limit 10 120
exec /usr/sbin/httpd -DNO_DETACH
```

On dialogue avec Upstart grâce à la commande "initctl".
Elle permet de lister, démarrer, arrêter, les jobs, et prendre en compte des nouvelles déclarations.

Notes personnelles

6.8.4- Gérer l'évènement ctrl-alt-del

Linux détecte l'événement "**ctrl-alt-del**" lors de l'appui simultané sur CTRL-ALT-SUPPR.

Cela correspond au signal "2" (INT) adressé au processus numéro 1 Init.

On peut décider de l'action à effectuer à la réception de l'événement. Par défaut, c'est un shutdown, pour arrêter "proprement" le système. Tout le monde est donc capable par erreur ...de déclencher, au mauvais moment, un reboot (propre) du serveur.

Avec SysVInit, c'est la ligne "ctrlalddel" de **/etc/inittab** qui exécute le "shutdown -r"
Avec Upstart, c'est dans /etc/init/control-alt-delete.conf.
Avec Systemd, c'est /usr/lib/systemd/system/ctrl-alt-del.target la "cible".

Pour inhiber ce comportement :
Dans tous les cas, on peut aussi utiliser l'option "**-a**" de la commande **shutdown**.
permet de n'accepter un arrêt que si un administrateur est connecté sur la console du système.
"un administrateur" = "un utilisateur dont le nom est listé dans **/etc/shutdown.allow**".

Ce fichier n'existe pas par défaut, un shutdown par "CTRL-ALT-SUPPR" n'est donc autorisé que quand "root" est connecté à la console. Le fichier doit contenir des noms d'utilisateurs habilités (un par ligne), et la commande limite actuellement à 32 utilisateurs déclarés dans ce fichier.

Notes personnelles

6.9- Prise en main de systemd

Désormais, c'est "**systemd**"qui pilote le processus de démarrage et les services, entre-autres.

Il a le même rôle que "**sysvinit**"et "**Upstart**" mais offre bien des fonctionnalités supplémentaires...

Systemd, c'est aussi et surtout un certain nombre de nouveaux outils :

Commandes en "*ctl" :
bootctl,
busctl,
coredumpctl,
hostnamectl,
journalctl,
localectl,
loginctl,
machinectl,
systemctl,
timedatectl

Les classiques :
halt,
init,
poweroff,
reboot,
runlevel, shutdown,
telinit,
udevadm

Commandes en "systemd-*" :
systemd-analyze,
systemd-ask-password,
systemd-cat,
systemd-cgls,
systemd-cgtop,
systemd-coredumpctl,
systemd-delta,
systemd-detect-virt,
systemd-escape,
systemd-firstboot,
systemd-hwdb,
systemd-inhibit,
systemd-loginctl,
systemd-machine-id-setup,
systemd-notify,
systemd-nspawn,
systemd-path,
systemd-run,
systemd-stdio-bridge,
systemd-tmpfiles,
systemd-tty-ask-password-agent

Notes personnelles

6.10- Connaître l'état du système avec systemctl

On peut de manière plus générale prendre connaissance de l'état du système avec "**is-system-running**"

```
# systemctl is-system-running
degraded
```

Note : le code retour est alors différent de 0… (utile pour scripter / superviser)

États possibles :

initializing	première phase du démarrage, avant d'atteindre "basic.target" ou l'arrivée en maintenance
starting	en cours de démarrage, avant que la file d'attente des tâches soit vide pour la première fois
running	le système est opérationnel
degraded	le système est opérationnel, mais une ou plusieurs unités sont en échec
maintenance	le système a été mis en maintenance (ou emergency)
stopping	procédure d'arrêt en cours

L'action "**status**" permet alors de comptabiliser les problèmes.

```
# systemctl status
● srv-vbackup01.actilis.net
    State: degraded
     Jobs: 0 queued
   Failed: 1 units
    Since: mer. 2016-01-13 13:53:48 CET; 3 weeks 2 days ago
…
```

Quand le service "Failed" est identifié, et inhibé/redémarré : "**systemctl reset-failed le-service**"

Notes personnelles

6.10.1- Équivalences systemctl / service / chkconfig

La commande **systemctl** remplace le couple **service** et **chkconfig**, qu'il faut désormais oublier.

Syntaxe :

```
# systemctl action unité.type-d-unité
```

ou

```
# systemctl action unité
```

Les actions similaires à la commande "**service**" et "**chkconfig**"

systemctl...	**service / chkconfig**
start : démarrer	service nom start
stop : arrêter	service nom stop
restart : redémarrer	service nom restart
try-restart : redémarrer seulement s'il tourne	service nom condrestart
reload : recharger la configuration	service nom reload
status / **is-active** : état / démarré/stoppé	service nom status
enable / **disable** : activer/désactiver,	chkconfig nom on / off
status / **is-enabled** : statut du service,	chkconfig --list service

Le listage :

systemctl list-unit-files	--type service	chkconfig --list
systemctl list-units	--type service --all	service –status-all
systemctl list-units	--type mount	
systemctl list-units	--type timer	

Notes personnelles

6.11- Les unités de systemd

Les unités sont définies par des fichiers se trouvant dans un des répertoires suivants :

Répertoire	Description
/usr/lib/systemd/system/	Unités distribuées avec les packages RPM
/run/systemd/system/	Unités créées dynamiquement (runtime), l'emportent sur celles de /usr/lib
/etc/systemd/system/	Unités créées par l'administrateur, l'emportent sur celles de /run/

Il existe plusieurs types d'unités (donnés par les suffixes) :

Type	Extension	Description
Service unit	**.service**	A system service.
Target unit	**.target**	A group of systemd units.
Automount unit	**.automount**	A file system automount point.
Device unit	**.device**	A device file recognized by the kernel.
Mount unit	**.mount**	A file system mount point.
Path unit	**.path**	A file or directory in a file system.
Scope unit	**.scope**	An externally created process.
Slice unit	**.slice**	A group of hierarchically organized units that manage system processes.
Snapshot unit	**.snapshot**	A saved state of the systemd manager.
Socket unit	**.socket**	An inter-process communication socket.
Swap unit	**.swap**	A swap device or a swap file.
Timer unit	**.timer**	A systemd timer.

Notes personnelles

6.11.1- Les targets... sorte de runlevels

Les **runlevels** de "sysvinit" et "Upstart" **disparaissent**… et deviennent des **targets** dans "**systemd**".
⇒ Les **runlevels** connus auparavant pointent vers des **targets**.

```
# cd /usr/lib/systemd/system ; ls -ld runlevel?.target
lrwxrwxrwx. 1 root root 15  5 nov.  13:14 runlevel0.target -> poweroff.target
lrwxrwxrwx. 1 root root 13  5 nov.  13:14 runlevel1.target -> rescue.target
lrwxrwxrwx. 1 root root 17  5 nov.  13:14 runlevel2.target -> multi-user.target
lrwxrwxrwx. 1 root root 17  5 nov.  13:14 runlevel3.target -> multi-user.target
lrwxrwxrwx. 1 root root 17  5 nov.  13:14 runlevel4.target -> multi-user.target
lrwxrwxrwx. 1 root root 16  5 nov.  13:14 runlevel5.target -> graphical.target
lrwxrwxrwx. 1 root root 13  5 nov.  13:14 runlevel6.target -> reboot.target
```

La **"target" par défaut** est celle pointée par **/etc/systemd/system/default.target**.

```
# ls -l /etc/systemd/system/default.target
lrwxrwxrwx 1 root root 37 15 oct.  22:39 /etc/systemd/system/default.target -> /lib/systemd/system/multi-
user.target
```

On peut l'afficher par systemctl get-default … et la modifier par systemctl set-default.

Lister les targets définies

```
# systemctl list-units --type=target
```

Changer de target courante :

```
# systemctl isolate nom.target
```


Notes personnelles

6.11.2- Définition d'une unité

Dans **systemd**, **toute chose est une unité**.

Une unité est définie par un **fichier** écrit dans un format similaire aux ".ini" :

```
[section]
      Directive=   (vide) pour la réinitialiser
      Directive=valeur
      Directive=valeur1,valeur2
```

6.11.2.1- Structure d'une unité

Quelque soit son type, on trouve les sections **[Unit]** et **[Install]** **:** **systemd.unit(5)**

[**Unit**] : définitions indépendantes du type d'unité,
Description= , **Documentation**=...
Requires=, **Wants=**, Conflicts : dépendances...
Before=, **After**=, **Condition***= ... ordonnancement, contrôles de pré-requis...

[**Install**] : actions réalisées lors de l'ajout/retrait du service
voir... ***systemctl enable unité.type*** / ***systemctl disable unité.type***

En fonction de son type, un trouve une section spécifique au type *:*
Voir **systemd.service(5), systemd.mount(5), systemd.timer(5), ...**
Exemples : sshd.service, tmp.mount, dnf-makecache.timer, ...

Notes personnelles

6.11.2.2- Dépendances entre unités

La commande s**ystemctl list-dependencies** permet de lister les dépendances
On peut aussi les grapher avec l'outil "**systemd-analyze**" et "**dot**" (package graphviz):

```
# systemd-analyze dot --require --from-pattern='*.target' | dot -Tsvg > targets.svg
```

La section "[Unit]" : permet d'indiquer de quelle unité nous dépendons
Requires= liste des autres unités dont **dépend absolument** le service décrit.
À l'activation de cette unité, celles mentionnées le sont aussi
À la désactivation d'une des unités citées (ou **échec de démarrage**), celle-ci l'est aussi

Voir aussi **BindsTo=** :
Si nous nous arrêtons, alors les dépendances sont arrêtées.
Si une des dépendances s'arrête, alors nous arrêtons (et toutes les autres sont arrêtées)

Wants= : similaire à Requires (mais **plus souple**).
Pas d'impact sur le demandeur en cas d'échec de la dépendance.

La section "[Install]" : permet d'indiquer l'unité qui dépend de celle-ci

RequiredBy, **WantedBy**= (section [Install] uniquement)

Cette approche utilise alors des répertoires "**.service.wants**" ou "**.service.requires**"

Notes personnelles

6.11.3- Surcharge d'une unité

Les unités sont définies par les packagers

dans **/lib/**systemd/system/**unité.type**

L'administrateur peut définir ses propres unités :

dans **/etc/**systemd/system/**unité.type**

L'administrateur peut définir des surcharges d'unités fournies par un packager :

dans un répertoire **/etc/**systemd/system/**unité.type.d/**fichier.conf

⇒ La création de surcharges est facilitée par la commande **systemctl edit unité.type**

Crée le répertoire,
Crée le fichier de surcharge

Format des fichiers de surcharge : contiennent eux-aussi des sections et directives.

Notes personnelles

6.11.4- Les unités de type target

Comme toute unité, **une target est définie par un fichier**.

Les fichiers définissant des targets ont le suffixe "**.target**" (ici **multi-user.target**)

```
# cat /lib/systemd/system/multi-user.target
…

[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes

[Install]
Alias=default.target
```

Note : il n'existe pas de section de type **[Target]**.

Les directives mentionnées dans une **target** sont héritées par toutes les unités qu'elle groupe.
Ici, "**Requires**" et "**After**" s'appliquent donc à tous les services groupés par **multi-user.target**.

Notes personnelles

6.11.5- Les unités de type service

Un service est défini par un **fichier ".service"**. Exemple : *httpd.service*
Voir ... **systemd.unit(5)**, **systemd.service(5)**

Format similaire aux fichiers ".ini",
[section]
Directive= (vide) pour la réinitialiser
Directive=valeur
Directive=valeur1,valeur2

Les sections :
[**Unit**] : définitions indépendantes du type d'unité
Description= , **Documentation**=...
Requires=, **Wants=**, Conflicts : dépendances...
Before=, **After**=, **Condition***= ... ordonnancement, contrôles de pré-requis...

[**Install**] : actions réalisées lors de l'ajout/retrait du service
voir... ***systemctl enable service*** / ***systemctl disable service***

[**Service**] : pour un service... décrit le processus à superviser... **systemd.service(5)**

Notes personnelles

6.11.5.1- Description d'un service : la section [Service]

Les directives principales sont les suivantes :

Type= *simple*, forking, oneshot, dbus, notify, idle

simple : la commande lancée est supposée le père du service (sshd, rsyslogd...)
les autres unités sont lancées en parallèle.

Oneshot : similaire à ***simple***,
mais le processus doit rendre la main avant que systemd ne poursuive.

Idle : similaire à ***simple***,
mais la tâche n'est lancée que lorsque les autres unités sont dispatchées.

forking : la commande lancée est supposée forker au lancement (named...)

dbus, **notify** : similaires à ***simple***,
mais on attend que le processus se présente sur le bus DBUS,
ou émette une notification.

PidFile= indique le fichier contenant numéro de processus du service

Notes personnelles

Exec*= commandes à exécuter

ExecStart= la ligne de commandes... pour démarrer le service

ExecReload= la ligne de commandes... pour le rechargement du service

ExecStop= la ligne de commandes... pour arrêter un service

ExecStartPre= / **ExecStartPost**= / **ExecStopPost**= ...

Restart= : redémarrage en cas de mort (à l'exception d'un systemctl stop ou restart)
RestartSec= : délai d'attente avant redémarrage

Environment= : permet de spécifier des variables d'environnement (et leur valeur)

EnvironmentFile= pointeur vers un fichier définissant l'environnement du processus.

On peut préfixer le nom de fichier par un "-" (tiret)
L'absence du fichier n'engendre alors pas d'erreur.

On peut spécifier plusieurs fichiers : une clause EnvironmentFile pour chacun

Notes personnelles

6.11.5.2- Installer un nouveau service

Créer un fichier dans **/etc/systemd/system/** nommé **monservice.service**

```
[Unit]
Description=Mon service

[Service]
ExecStart=/usr/bin/bash /path/to/mon-script

[Install]
WantedBy=multi-user.target
```

Attention, "mon-script" doit commencer par "# !/bin/bash".

...man **systemd.service** (5)

Puis pour l'activer...

```
# systemctl enable monservice.service
```

...man **systemctl**.... ou voir la page suivante.

Pour vérifier ou lister un service : **systemctl cat mon.service**

On peut aussi utiliser **systemctl edit**, mais attention : application immédiate !

Notes personnelles

6.12- Diagnostic de configuration réseau

6.12.1- Validation de la configuration

driver : dmesg, module...
diagnostiquer une interface non reconnue
identifier et charger le bon module

lien : ethtool, mii-tool
diagnostiquer un problème de lien, de câble déconnecté
paramétrer : la vitesse du lien, contrôle de flux, full/half duplex

adressage : ifconfig, ping, **ip**
valider une adresse IP, un masque
tester l'accessibilité d'un hôte distant,

routage : route, traceroute, **ip**
vérifier et valider une route,
ajouter, supprimer une entrée dans la table de routage

sockets : netstat, lsof
lister les sockets en écoute,
identifier le processus responsable d'une socket

Netfilter... : iptables

Notes personnelles

6.12.2- Commandes

dmesg, ethtool
ifconfig, **ip**
ifup, **ifdown**,
route, **netstat, lsof**
ping, traceroute
host, **getent**

6.12.3- Configuration des interfaces

/etc/sysconfig/network (RedHat, Mandriva, ...)
/etc/sysconfig/network-scripts/**ifcfg-*** (RedHat, Mandriva, ...)
/etc/hostname (Debian, Ubuntu, ...)
/etc/network/interfaces (Debian, Ubuntu, ...)

6.12.4- Résolution de noms

/etc/nsswitch.conf,
/etc/resolv.conf,
/etc/hosts,

Notes personnelles

6.13- Problèmes de résolution de nom côté client

6.13.1- Configurer les mécanismes de résolution de nom côté client

Comprendre et maîtriser les librairies NSS*, modifier l'ordre des services de résolution de nom
Format de **/etc/nsswitch.conf**
Lister les méthodes (librairies filles) disponibles

Connaître les fichiers de configuration des différentes méthodes clientes :

files : /etc/hosts, /etc/passwd, /etc/group, …

dns : /etc/resolv.conf

ldap : /etc/ldap.conf, /etc/nss_ldap.conf
Avec RedHat 6 : nslcd.conf

6.13.2- Tester et diagnostiquer les problèmes de résolution de noms

Savoir utiliser les bonnes commandes de diagnostic :

getent
host
ldapsearch ?

Notes personnelles

7- Optimisation des performances

Notes personnelles

7.1- Régler les paramètres des disques

7.1.1- Information sur les paramètres des disques

La commande **hdparm** est un utilitaire d'information sur les disques (essentiellement IDE/UDMA).

Visualiser les paramètres des disques

Le mode "-i" : selon le noyau,

```
# hdparm -i /dev/sda

/dev/sda:

 Model=ST9500423AS, FwRev=0002SDM1, SerialNo=6WR01DZ7
 Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs RotSpdTol>.5% }
 RawCHS=16383/16/63, TrkSize=0, SectSize=0, ECCbytes=4
 BuffType=unknown, BuffSize=16384kB, MaxMultSect=16, MultSect=16
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=976773168
 IORDY=on/off, tPIO={min:120,w/IORDY:120}, tDMA={min:120,rec:120}
 PIO modes:  pio0 pio1 pio2 pio3 pio4
 DMA modes:  mdma0 mdma1 mdma2
 UDMA modes: udma0 udma1 udma2 udma3 udma4 udma5 *udma6
 AdvancedPM=yes: unknown setting WriteCache=enabled
 Drive conforms to: unknown:  ATA/ATAPI-4,5,6,7

 * signifies the current active mode
```


Notes personnelles

Le mode "-I" : selon le périphérique

```
# hdparm -I /dev/sda

/dev/sda:

ATA device, with non-removable media
        Model Number:       ST9500423AS
        Serial Number:      6WR01DZ7
        Firmware Revision:  0002SDM1
        Transport:          Serial
Standards:
        Used: unknown (minor revision code 0x0029)
        Supported: 8 7 6 5
        Likely used: 8
Configuration:
        Logical         max     current
        cylinders       16383   16383
        heads           16      16
        sectors/track   63      63
        --
        CHS current addressable sectors:   16514064
        LBA    user addressable sectors:  268435455
        LBA48  user addressable sectors:  976773168
        Logical  Sector size:                   512 bytes
        Physical Sector size:                  4096 bytes
        Logical Sector-0 offset:                  0 bytes
        device size with M = 1024*1024:      476940 MBytes
        device size with M = 1000*1000:      500107 MBytes (500 GB)
        cache/buffer size  = 16384 KBytes
        Nominal Media Rotation Rate: 7200
… …
```

7.1.2- Modifier les paramètres des disques

Hdparm permet aussi d'améliorer ou de dégrader les performances d'un disque, de le mettre en veille (économie d'énergie), de gérer son niveau sonore, de mesurer (de manière grossière) ses performances...

Notes personnelles

Visualisation des paramètres

```
# hdparm /dev/hda

/dev/hda:
 multcount    = 16 (on)
 IO_support   =  1 (32-bit)
 unmaskirq    =  0 (off)
 using_dma    =  1 (on)
 keepsettings =  0 (off)
 readonly     =  0 (off)
 readahead    = 256 (on)
 geometry     = 16383/255/63, sectors = 156301488, start = 0
```

Modification d'un paramètre : activation du DMA :

```
# hdparm -d1 /dev/hda

/dev/hda:
 setting using_dma to 1 (on)
 using_dma    =  1 (on)
```

Notes personnelles

Visualisation des performances

Après chaque modification d'un paramètre, on l'habitude de tenter un test de performance :

```
# hdparm -tT /dev/hda

/dev/hda:
 Timing cached reads:   1424 MB in  2.00 seconds = 711.80 MB/sec
 Timing buffered disk reads:  138 MB in  3.03 seconds =  45.60 MB/sec
```

Aujourd'hui, avec la virtualisation, hdparm ne veut plus dire grand chose et est considérée comme obsolète (https://launchpad.net/ubuntu/natty/i386/hdparm).

Notes personnelles

7.2- Les accès disques et le noyau : choisir le scheduler I/O

Pour évaluer les performances des entrées/sorties, on devrait considérer ces quatre éléments :

le taux de transfert
ratio entre les opérations de lecture et celles d'écriture
taux de requêtes par seconde
proximité des données (localisation des données sur le support).

7.2.1.1- Schedulers sont disponibles sur un noyau précompilé

Au démarrage du noyau, on peut lister ceux compilés :

```
# dmesg | grep scheduler
[    1.005297] io scheduler noop registered
[    1.005299] io scheduler deadline registered
[    1.005333] io scheduler cfq registered (default)
```

7.2.1.2- Choisir l'io-scheduler au démarrage : l'option elevator=

C'est une option de démarrage du noyau, qui doit donc être fournie par le chargeur (GRUB).
Voir la documentation "linux:Documentation/kernel-parameters.txt" :

```
603    elevator=       [IOSCHED]
604                    Format: {"anticipatory" | "cfq" | "deadline" | "noop"}
```

!! Anticipatory n'est plus fourni sur les noyaux récents (disparu en 2.6.32).

Notes personnelles

7.2.1.3- Les différents IO schedulers

Ils définissent des stratégies de gestion des I/O par le noyau.

Les schedulers d'entrées-sorties disponibles avec le noyau 2.6 (et 3.X) sont les suivants :

Anticipatory (**as**) (***< 2.6.32***) : consiste à temporiser certaines I/O de manière à les regrouper et les réaliser de manière agrégée ou ré-ordonnée (but : réduire les temps d'accès aux disques).

Offre de bonnes performances dans certains cas : Serveur web, poste de travail bureautique
Détériore les performances sur les disque disposant d'un cache, et avec les contrôleurs RAID.

Deadline (**deadline**) : minimise les temps de latence pour chaque requête. Ré-ordonne les requêtes en utilisant 5 files d'attentes. Pas forcément optimal sur des configuration multi-disque ou multi-filesystem, mais comportement proche du temps réel. Adapté aux bases de données demandant des accès disque intensifs.

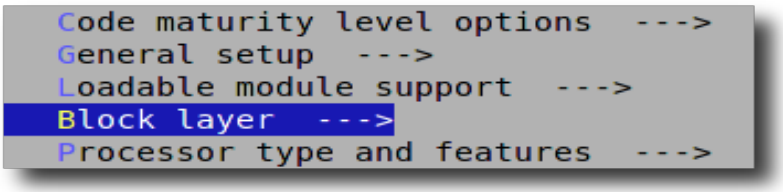
Completely Fair Queuing (**cfq**) : offre de bonnes performances pour la plupart des applications (autant en débit qu'en temps de latence). À choisir sur les machines multi-processeurs et disposant de plusieurs disques. Permet de choisir la priorité données aux processus sur les IO (**ionice**).

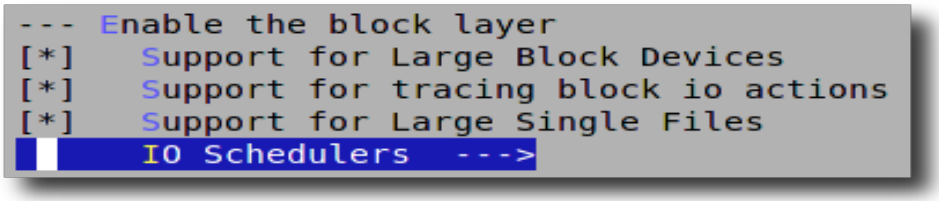
NOOP (**noop**) : il réalise un simple FIFO et utilise le moins possible de CPU par transaction I/O. Il s'appuie sur les fait que les performances des I/O sont du ressort du block device. Il est donc adapté aux memory-disks et controleurs "intelligents" et disposant de cache.

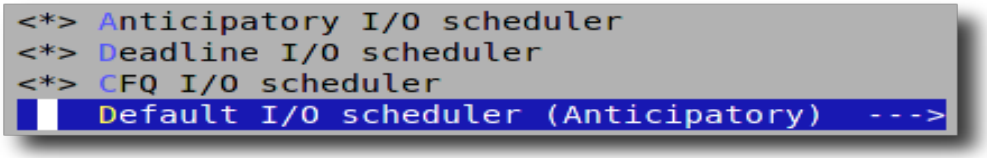
Notes personnelles

7.2.1.4- Compiler les différents I/O schedulers

Menu principal des sources du noyau







Notes personnelles

7.3- Répartir les I/O sur plusieurs filesystems

Saturation du scheduler ?
- charge WA (Wait) très élevée pendant longtemps
- goulet d'étranglement.

Stratégie choisis mal adaptée à l'ensemble de l'arborescence ?
- pas possible de spécifier un scheduler par répertoire
- nécessite donc plusieurs filesystems

Séparer les périphériques physiques ?
- profiter des caches de plusieurs disques
- choisir un scheduler adapté à chaque usage

Mesurer la charge d'I/O induite par l'activité de clients différents

Exemple d'un serveur de mail mutualisé :
- un grand nombre d'accès concurrents
- des opérations parfois longues (ouverture de "grosses" boites à lettres
  - séparer les filesystems des "gros clients" améliore les performances globales

Notes personnelles

7.4- Tester et optimiser les performances du réseau

Le MTU
Paramétrable par ifconfig, il peut avoir une influence sur les débits des lignes xDSL

Les paramètres noyau lié au réseau
Tout ce qui concerne les timeout, sur les sockets notamment (backlog, etc...)

De petits guide s
http://www.speedguide.net/articles/linux-tweaking-121

Linux TCP autotuning
http://kaivanov.blogspot.fr/2010/09/linux-tcp-tuning.html

TxQueuelen
http://fasterdata.es.net/host-tuning/linux/

Voir le document (2006) : "The Performance analysis of Linux Networking".

Notes personnelles

8- Supervision

Notes personnelles

8.1- Nagios : architecture et installation

Nagios est un système de **supervision d'équipements et de serveurs** composé de plusieurs éléments.

Pour les distributions RLEL/ CentOS, il est packagé sur plusieurs dépôts, dont **EPEL**
Pour la famille Debian, le package nagios3 est disponible en standard.

8.1.1- Nagios Core : le serveur nagios

```
# yum install nagios                # apt-get install nagios
```

8.1.2- Nagios est une application web

Elle s'appuie sur Apache / PHP et est composée d'un ensemble de scripts CGI et contenus HTML

Le package Nagios prévoit la configuration d'Apache (**/etc/httpd/conf.d/nagios.conf**).

Il faut un redémarrage d'Apache pour prendre en compte cette nouvelle "application web".

```
# service httpd restart                    # service apache2 restart
```

8.1.3- Des plugins

Ce sont des exécutables (C, shell, perl, …) utilisés par le serveur Nagios pour collecter l'information.
Par défaut, aucun n'est installé.

Un meta-package "**nagios-plugins-all**" (nagios-plugins-basic, common, standard) qui les installe.

Notes personnelles

8.2- Nagios : premier lancement et connexion au service

8.2.1- Première connexion et lancement du service

Le service "nagios" doit être lancé pour commencer la collecte d'informations.

```
# service nagios restart
```

On pointe un navigateur sur http://votre-serveur/nagios/

Le nom d'utilisateur est "**nagiosadmin**" tout comme le mot de passe.

Les informations d'authentification sont stockées dans **/etc/nagios/passwd**,

On peut (doit) modifier le mot de passe d'origine grâce à la commande "**htpasswd**".

```
# htpasswd /etc/nagios/passwd nagiosadmin [29]
```

29 Sur Ubuntu 14.4, le fichier est /etc/nagios3/htpasswd.users, la commande de modification est donc :
htpasswd /etc/nagios3/htpasswd.users nagiosadmin

Notes personnelles

8.3- Nagios : configuration

8.3.1- Structure et philosophie

La configuration principale est dans **/etc/nagios/nagios.cfg**. (/etc/nagios3/nagios.cfg)

Elle est basée sur des objets, que l'on associe entre-eux pour décrire les hôtes et services à superviser.

Les objets principaux sont :

des hôtes (**host**), qui peuvent être regroupé (**hostgroup**)

des éléments à superviser (**service**), qui font appel à des "check_command"

des commandes (**command**), décrites par un nom (**command_name**) et une "**command_line**"

On peut faire des inclusions de sous-fichier

```
cfg_file=/etc/nagios/unfichier.cfg
```

et de sous-répertoires :

```
cfg_dir=/etc/nagios/conf.d
```


Notes personnelles

8.3.2- Les templates

Ce sont des objets "non instanciés" (**register** 0) sur lesquels s'appuieront d'autres objets, grâce à "**use**".

```
# Fichier /etc/nagios/conf.d/templates.cfg

define service{
                name SSH
                use local-service
                service_description SSH
                check_command   check_ssh
                register 0
        }


define service{
                name HTTP
                use local-service
                service_description HTTP
                check_command   check_http
                register 0
        }
```

Un template peut être utilisé pour des services, des machines... d'autres templates.

On peut **(devrait absolument) s'appuyer sur des templates** pour simplifier la configuration.

Notes personnelles

Un template de machine "**linux-server**" est défini dans le fichier **/etc/nagios/objects/templates.cfg**.

Il est lui-même basé sur le template "**generic-host**".

```
# Linux host definition template - This is NOT a real host, just a template !

define host{
      name                        linux-server
      use                         generic-host
      check_period                24x7
      check_interval              5
      retry_interval              1
      max_check_attempts          10
      check_command check-host-alive
      notification_period         workhours
      notification_interval       120
      notification_options        d,u,r
      contact_groups              admins
      register                    0
      }
```


Notes personnelles

8.3.3- Les machines

Le but des objets "host" est de définir des machines à superviser.

On peut s'appuyer sur un template pour utiliser ses paramètres,

```
# Fichier /etc/nagios/conf.d/machines.cfg
define host{
     use linux-server
     host_name serveur1
     alias SRV-1
     address 10.44.56.1
}
```

"serveur1" prend tous les paramètres du template "linux-server".

et même surcharger des paramètres du template.

```
define host{
     use linux-server
     host_name serveur2
     alias SRV-2
     address 10.44.56.2
     check_interval 10
}
```

"serveur2" prend tous les paramètres définis dans le template "linux-server", mais l'un d'entre-eux, *check_interval* vaut est surchargé (10 minutes au lieu des 5 définis dans le template).

Notes personnelles

8.3.4- Machines / Services

On associe des services à des machines.

```
# Fichier /etc/nagios/conf.d/services.cfg
define service{
     host serveur1,serveur2,pcfrancois
     use SSH
}
define service{
     host serveur1,serveur2
     use HTTP
}
```

Cela évite de dupliquer des énumérations de services similaires pour N machines.

Notes personnelles

8.3.5- Les Hostgroups

L'interface propose un regroupement des machines :

```
# Fichier /etc/nagios/conf.d/groupes.cfg
define hostgroup{
        hostgroup_name  Serveurs
        alias           Linux Servers du Reseau
        members         serveur1,serveur2
        }

define hostgroup{
        hostgroup_name  Portables
        alias           Linux Portables
        members         pcfrancois
        }
```


Notes personnelles

9- Annexes

Notes personnelles

9.1- Mise en place d'un miroir de paquets local

Il s'agit d'un répertoire d'un serveur accessible via HTTP ou FTP par les clients Yum mettant à disposition l'ensemble des packages RPM d'une distribution.

9.1.1- Récupération du contenu d'un miroir officiel

On contacte un miroir "amont", par FTP, HTTP, ou mieux : Rsync.

```
#!/bin/bash
PATH=$PATH:/usr/local/bin
BASEDIR=/data/CentOS
MIRROR=rsync://distrib-coffee.ipsl.jussieu.fr/pub/linux/centos
VERSION=6.3

cd $BASEDIR && [ -e ${VERSION} ] || mkdir ${VERSION}
cd $BASEDIR/$VERSION
[ `pwd` != "$BASEDIR/$VERSION" ] && exit 1

rsync -rlptDvz -P --delete  --delete-excluded  \
                  --exclude 'cr/*' \
                  --exclude '*src*.iso' --exclude "*of*.iso" \
                  --exclude "**bin-DVD**"  \
                  --exclude "**LiveCD**"  --exclude "**LiveDVD**"  \
                  --exclude "ppc"  --exclude "s390"  --exclude "s390x"  \
                  --exclude "ia64" --exclude "alpha" --exclude "i386"  \
                  --exclude "**fasttrack**"  \
                  --exclude "**SRPMS**"  \
    $MIRROR/$VERSION/ .
chown -R root.root .
```


Notes personnelles

9.1.2- Accéder au dépôt via un serveur web

Du côté d'Apache, déclarer un VirtualHost (/etc/httpd/conf.d/repo.conf) pour servir de miroir Yum

```
<VirtualHost *:80>
        ServerAdmin admin@actilis.net
        DocumentRoot /data/CentOS

        Servername  mirror.actilis.net

        ErrorLog /var/log/apache/mirror.actilis.error_log
        CustomLog /var/log/apache/mirror.actilis.access_log combined

        ## Tout est interdit par defaut dans /data/CentOS
        <Directory /data/CentOS>
                Options Indexes FollowSymLinks
                AllowOverride All
                order allow,deny
                Allow from all
                ErrorDocument 403 http://www.actilis.net/
        </Directory>
</VirtualHost>
```

9.1.3- Accès au dépôt via NFS

La solution la plus simple est "d'exporter" le répertoire de stockage du dépôt, puis, sur les clients, de réaliser un simple montage.

La clause "baseurl" utilisera alors "file://" au lieu de "http://" ou ftp://".

Notes personnelles

9.2- Structure détaillée d'un paquetage RPM

RPM est un procédé de packaging utilisé par un bon nombre de distributions.

C'est aussi un format de fichiers, documenté ici : http://www.rpm.org/max-rpm-snapshot/

9.2.1- Composition d'un fichier RPM

Ce sont des fichiers d'archives proche de CPIO découpé en 4 "sections" :

```
struct rpmlead {
    unsigned char magic[4];
    unsigned char major, minor;
    short type;
    short archnum;
    char name[66];
    short osnum;
    short signature_type;
    char reserved[16];
} ;
```

un en-tête : "**lead**"
informe sur le format, le nom et la version du package,
le type (binaire / source), l'os cible, la version de RPM utilisée...
la taille du lead est de **96 octets,**
dont 66 réservés au nom du package (y compris sa version).

une structure de "**signatures**" (8e ad e8)
taille du package, taille de son payload, digest SHA1 du header,
digest md5... visibles par "**rpm --checksig --verbose**"

une structure de "**header**"
contient tous les **RPMTAGS**, c'est à dire les informations disponibles par "**rpm -qi**"

vient ensuite le contenu du paquetage : "**payload**" (l'archive, au format cpio gzippé).

Notes personnelles

9.2.2- Détail des différentes "sections"

9.2.2.1- Le "lead" : 96 octets

L'exemple suivant se décompose comme indiqué :

```
$ hexdump -n 100 -vC  snort-2.8.0.2-1.RH5.i386.rpm
00000000  ed ab ee db 03 00 00 00  00 01 73 6e 6f 72 74 2d  |..........snort-|
00000010  32 2e 38 2e 30 2e 32 2d  31 00 00 00 00 00 00 00  |2.8.0.2-1.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 05  |................|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  8e ad e8 01                                       |....|
```

M1 à M4 : C'est le Magic de ces fichiers, permettant notamment à la commande "file" de les identifier. Valeurs : 0x**ED**, 0x**AB**, 0x**EE**, 0x**DB**. Le type MIME officiel n'utilise que M3 et M4 :

```
# grep rpm /usr/share/magic.mime
>2      beshort         0xeedb          application/x-rpm
```

MAJ(or) et **MIN**(or) : indiquent la version du format utilisé. (3.0 => 0300)
TYPE : 0000 (rpm binaire) ou 0001 (rpm source)
ARCH : 0001 => i386
NOM : sur 66 octets
OS : 0001 => Linux
SIG(nature) : 00 05 => C'est en fait le type de signature. Le type 5 est celui des RPMs version 3.00
Padding : 16 octets valant 0x00.

En rouge, c'est le Magic Number d'une structure de header (8e ad e8) à l'offset 0x60 (soit 96).

Notes personnelles

9.2.2.2- Les headers de signature : indexes de contenu

```
$ hexdump -s 96 -n 192 -vC  snort-2.8.0.2-1.RH5.i386.rpm
00000060  8e ad e8 01 00 00 00 00  00 00 00 05 00 00 00 54  |...............T|
```

Après le Magic Number, et de longueur variable, leurs champs sont les suivants :

Magic Number de démarrage d'une structure d'en-tête: (8e ad e8)

Version (1 octet) : 01

4 octets réservés : 00 00 00 00

4 octets indiquant le nombre d'entrées d'index dans la section signature (ici 05).

4 octets indiquant le nombre d'octets de données dans la signature. Ici 0x54 => **84**.

À suivre... les entrées d'index : ici, nous attendons 05 blocs de 16 octets.

Voir page suivante.

Notes personnelles

Chaque entrée d'index est composée de 4 éléments de 32 bits :

```
00000070  00 00 00 3e 00 00 00 07  00 00 00 44 00 00 00 10  |...>.......D....|   (E)
00000080  00 00 01 0d 00 00 00 06  00 00 00 00 00 00 00 01  |................|   (A)
00000090  00 00 03 e8 00 00 00 04  00 00 00 2c 00 00 00 01  |...........,....|   (B)
000000a0  00 00 03 ec 00 00 00 07  00 00 00 30 00 00 00 10  |...........0....|   (C)
000000b0  00 00 03 ef 00 00 00 04  00 00 00 40 00 00 00 01  |...........@....|   (D)
```

un tag de signature : sa valeur, convertie en décimal, possède une signification visible dans le fichier "**rpmlib.h**".

```
/** \ingroup signature
 * Tags found in signature header from package.
 */
enum rpmtagSignature {
    RPMSIGTAG_SIZE      = 1000, /*!< internal Header+Payload size in bytes. */
    RPMSIGTAG_LEMD5_1   = 1001, /*!< internal Broken MD5, take 1 @deprecated legacy. */
    RPMSIGTAG_PGP       = 1002, /*!< internal PGP 2.6.3 signature. */
    RPMSIGTAG_LEMD5_2   = 1003, /*!< internal Broken MD5, take 2 @deprecated legacy. */
    RPMSIGTAG_MD5       = 1004, /*!< internal MD5 signature. */
    RPMSIGTAG_GPG       = 1005, /*!< internal GnuPG signature. */
    RPMSIGTAG_PGP5      = 1006, /*!< internal PGP5 signature @deprecated legacy. */
    RPMSIGTAG_PAYLOADSIZE = 1007,/*!< internal uncompressed payload size in bytes. */
    RPMSIGTAG_BADSHA1_1 = RPMTAG_BADSHA1_1,     /*!< internal Broken SHA1, take 1. */
    RPMSIGTAG_BADSHA1_2 = RPMTAG_BADSHA1_2,     /*!< internal Broken SHA1, take 2. */
    RPMSIGTAG_SHA1      = RPMTAG_SHA1HEADER,    /*!< internal sha1 header digest. */
    RPMSIGTAG_DSA       = RPMTAG_DSAHEADER,     /*!< internal DSA header signature. */
    RPMSIGTAG_RSA       = RPMTAG_RSAHEADER      /*!< internal RSA header signature. */
};
```

Les 3 derniers (**03 e8**, **03 ec**, et **03 ef**) correspondent par exemple à **1000** (SIZE), **1004** (MD5), et **1007** (PAYLOAD SIZE).

un type : dans l'exemple, on trouve les types 4, 6 et 7 :

NULL = 0 , CHAR = 1 , INT8 = 2 , INT16 = 3 ,
INT32 = 4 , INT64 = 5 , **STRING = 6** (terminée par \0),
BIN = 7 (terminée par \0) , STRING_ARRAY = 8

un offset : c'est l'emplacement où commence l'enregistrement.
un compteur : nombre d'éléments contenus. Toutes les STRING ont un compte de 1, et les STRING_ARRAY ont un compte donnant le nombre de chaînes qu'elles contiennent.

Notes personnelles

Suivent ensuite les signatures, jusqu'à trouver une nouvelle structure d'en-tête (8e ad e8)

```
000000c0  34 31 63 38 38 33 65 31  38 38 39 63 64 66 62 63  |41c883e1889cdfbc|
000000d0  35 62 66 63 37 32 65 62  31 66 33 62 62 34 66 62  |5bfc72eb1f3bb4fb|
000000e0  31 32 63 65 61 37 35 61  00 00 00 00 00 1d c3 14  |12cea75a........|
000000f0  cb d1 62 ea 52 5f 2a e3  5b 45 aa c3 b2 6a 2b b2  |..b.R_*.[E...j+.|
00000100  00 9b f3 9c 00 00 00 3e  00 00 00 07 ff ff ff b0  |.......>........|
00000110  00 00 00 10 00 00 00 00  8e ad e8 01 00 00 00 00  |................|
```

Contenu des signatures : (pour l'exemple) dans l'ordre des lettres :

e) 1 entrée de type **07** (binaire, tag : 00 3e = 62,) à l'offset 0x44 (=68), taille = **0x10** = 16

```
00000070  00 00 00 3e 00 00 00 07  00 00 00 44 00 00 00 10  |...>.......D....|   (E)
```

```
$ hexdump -s $(( 192 + 68 )) -n 16 -vC  snort-2.8.0.2-1.RH5.i386.rpm
00000104  00 00 00 3e 00 00 00 07  ff ff ff b0 00 00 00 10  |...>............|
```

Il y a ensuite 4 octets "NULL" (\0) pour tomber sur un compte rond à 8 octets près.
==> Pas grand chose de lisible à l'œil nu, …

Notes personnelles

a) 1 entrée de type **06** (chaine, tag : 01 1d = 269) à l'offset 0... termine donc par un "\0".

```
00000080  00 00 01 0d 00 00 00 06  00 00 00 00 00 00 00 01  |................|   (A)
```

Cette entrée se trouve au début des signatures (donc à l'offset 0xc0 (= 192)). Sa longueur est 0x2b = 43 maximum, (car (**b**)commence en 0x2c= **44**)

```
$ hexdump -s 192 -n 43 -vC  snort-2.8.0.2-1.RH5.i386.rpm
000000c0  34 31 63 38 38 33 65 31  38 38 39 63 64 66 62 63  |41c883e1889cdfbc|
000000d0  35 62 66 63 37 32 65 62  31 66 33 62 62 34 66 62  |5bfc72eb1f3bb4fb|
000000e0  31 32 63 65 61 37 35 61  00 00 00 00              |12cea75a....|
$ rpm --checksig --verbose /tmp/snort-2.8.0.2-1.RH5.i386.rpm
/tmp/snort-2.8.0.2-1.RH5.i386.rpm:
    Header SHA1 digest: OK (41c883e1889cdfbc5bfc72eb1f3bb4fb12cea75a)
    MD5 digest: OK (cbd162ea525f2ae35b45aac3b26a2bb2)
```

b) 1 entrée de type **04 (INT 32 = 4 octets)**, tag = 03 e8 (= 1000 : **SIZE**), à l'offset 0x2c = 44

```
00000090  00 00 03 e8 00 00 00 04  00 00 00 2c 00 00 00 01  |...........,....|   (B)
```

```
$ hexdump -s $(( 192 + 44 )) -n 4 -vC  snort-2.8.0.2-1.RH5.i386.rpm
000000ec  00 1d c3 14                                       |....|
$ ls -l snort-2.8.0.2-1.RH5.i386.rpm
-rw-rw----. 1 fmicaux fmicaux 1950764 10 oct.  22:51 snort-2.8.0.2-1.RH5.i386.rpm
```

==> 0x **1dc314** = **1 950 484** (contre **1950764** pour le fichier)... c'est en fait la taille du fichier moins 280 octets = 96 (lead) + 184 (signatures).

Notes personnelles

c) 1 entrée de type **07 (binaire)**, tag = 03 ec (=1004 : **MD5**), offset 0x30 = 48, longueur 16

```
000000a0  00 00 03 ec 00 00 00 07  00 00 00 30 00 00 00 10  |...........0....|   (C)
```

C'est le Diget MD5 du package :

```
$ hexdump -s $(( 192 + 48 )) -n 16 -vC  snort-2.8.0.2-1.RH5.i386.rpm
000000f0  cb d1 62 ea 52 5f 2a e3  5b 45 aa c3 b2 6a 2b b2  |..b.R_*.[E...j+.|
```

```
$ rpm --checksig --verbose /tmp/snort-2.8.0.2-1.RH5.i386.rpm
/tmp/snort-2.8.0.2-1.RH5.i386.rpm:
    Header SHA1 digest: OK (41c883e1889cdfbc5bfc72eb1f3bb4fb12cea75a)
    MD5 digest: OK (cbd162ea525f2ae35b45aac3b26a2bb2)
```

d) 1 entrée de type **03 (INT 32 = 4 octets)**, tag = 03 ef (=1007 : **PAYLOAD SZ**), offset 0x40 = 64

```
000000b0  00 00 03 ef 00 00 00 04  00 00 00 40 00 00 00 01  |...........@....|   (D)
```

```
$ hexdump -s $(( 192 + 64 )) -n 4 -vC  snort-2.8.0.2-1.RH5.i386.rpm
00000100  00 9b f3 9c                                       |....|
```

=> 10 220 444 octets... C'est ce que pèse **l'archive décompressée**.

```
$ rpm2cpio snort-2.8.0.2-1.RH5.i386.rpm  |wc -c
10220444
```


Notes personnelles

9.2.2.3- Le Header

Nous débutons à l'offset **280** par un nouveau Magic Number (8e ad e8) d'une structure d'en-tête.

```
$ hexdump -s $(( 280 )) -n 128 -vC  snort-2.8.0.2-1.RH5.i386.rpm
00000118  8e ad e8 01 00 00 00 00  00 00 00 45 00 00 72 d1  |...........E..r.|
```

On retrouve les champs VERSION (01), suivis de 4 octets réservés (00 00 00 00), suivi du nombre d'entrées (00 00 00 45), suivi la taille du "store", qui est de 00 00 72 d1 (soit **29 393** octets).

On a 0x45 tags (soit 69...on a donc 69 lignes comme les 5 lignes débutant par "128" à "188").

```
… 0X45 = 69 lignes de headers au total …
00000128  00 00 00 3f 00 00 00 07  00 00 72 c1 00 00 00 10  |...?......r.....|
00000138  00 00 00 64 00 00 00 08  00 00 00 00 00 00 00 01  |...d............|
00000148  00 00 03 e8 00 00 00 06  00 00 00 02 00 00 00 01  |................|
00000158  00 00 03 e9 00 00 00 06  00 00 00 08 00 00 00 01  |................|
00000168  00 00 03 ea 00 00 00 06  00 00 00 10 00 00 00 01  |................|
00000178  00 00 03 eb 00 00 00 04  00 00 00 14 00 00 00 01  |................|
00000188  00 00 03 ec 00 00 00 09  00 00 00 18 00 00 00 01  |................|
```

Les tags...

Le premier TAG (**type 00 00 03 E8**), soit **1000** en décimal (*RPMTAG_NAME*), de type **00 00 00 06** (String) se trouve à un offset de **00 00 00 02**, ...après les entrées d'index, soit à un offset global de de **2** (offset) + **16** (en-tête) + **16 * 69** (entrées d'index)... soit **1120 + 2**.

Notes personnelles

Vu qu'il s'agit d'une structure de données comme celles des signatures, les 16 octets de chaque "ligne" correspondent à une entrée d'index... dont les 4 premiers octets sont un RPMTAG, chacun ayant une signification...

Le header contient toutes les informations sur le paquetage et il existe près de 175 tags différents (voir /usr/include/rpm/**rpmtag.h**).

```
72      RPMTAG_NAME                     = 1000, /* s */
73  #define RPMTAG_N            RPMTAG_NAME     /* s */
74      RPMTAG_VERSION                  = 1001, /* s */
75  #define RPMTAG_V            RPMTAG_VERSION  /* s */
76      RPMTAG_RELEASE                  = 1002, /* s */
77  #define RPMTAG_R            RPMTAG_RELEASE  /* s */
78      RPMTAG_EPOCH                    = 1003, /* i */
79  #define RPMTAG_E            RPMTAG_EPOCH    /* i */
80      RPMTAG_SUMMARY                  = 1004, /* s{} */
81      RPMTAG_DESCRIPTION              = 1005, /* s{} */
82      RPMTAG_BUILDTIME                = 1006, /* i */
83      RPMTAG_BUILDHOST                = 1007, /* s */
84      RPMTAG_INSTALLTIME              = 1008, /* i */
85      RPMTAG_SIZE                     = 1009, /* i */
86      RPMTAG_DISTRIBUTION             = 1010, /* s */
87      RPMTAG_VENDOR                   = 1011, /* s */
"/usr/include/rpm/rpmtag.h" 430 lignes --20%--
```

Type 06 : string, terminé par un "\0"... Les 3 premiers tags sont intéressants !

```
$ hexdump -s $(( 280 + 1120 + 2 )) -n 128 -vC  snort-2.8.0.2-1.RH5.i386.rpm
0000057a  73 6e 6f 72 74 00 32 2e  38 2e 30 2e 32 00 31 00  |snort.2.8.0.2.1.|
```


Notes personnelles

9.2.2.4- L'archive (le payload)

Elle commence à la section suivante :

```
$ hexdump -s $(( 280 + 1120 + 29393 )) -n 200 -vC  snort-2.8.0.2-1.RH5.i386.rpm
00007849  1f 8b 08 00 00 00 00 00  00 03 d4 1a 6b 73 db 36  |............ks.6|
00007859  32 5f ab 5f b1 51 9c 3a  cd 99 7a d9 89 13 79 dc  |2_._.Q.:..z...y.|
```

1f 8b : Il s'agit de données compressées par gzip avec la méthode "deflate" (08).

```
$ dd if=snort-2.8.0.2-1.RH5.i386.rpm skip=$(( 280 + 1120 + 29393 )) ibs=1 of=/tmp/out
1919971+0 enregistrements lus
3749+1 enregistrements écrits
1919971 octets (1,9 MB) copiés, 1,87073 seconde, 1,0 MB/s

$ file /tmp/out
/tmp/out: gzip compressed data, from Unix
```

Il s'agit précisément d'une archive CPIO (au format SVR4 avec checksum CRC) compressée par gzip.

Notes personnelles

9.2.3- Outillage

On peut décompresser cette archive directement grâce à l'outil "**rpm2cpio**".

La visualisation de tous les tags peut être faite en spécifiant lors des requêtes (**rpm -q**) un format spéficique (**--queryformat**).

Exemple :

```
# rpm -q --queryformat 'Le paquetage %{NAME}, version %{VERSION} contient les fichiers suivants : [%
{FILENAMES} (%{FILESIZES} bytes), ]et a été installé %{INSTALLTIME:date}\n' xinetd

Le paquetage xinetd, version 2.3.14 contient les fichiers suivants : /etc/rc.d/init.d/xinetd (2497
bytes), /etc/sysconfig/xinetd (376 bytes), /etc/xinetd.conf (1001 bytes), /etc/xinetd.d/chargen-
dgram (1157 bytes), /etc/xinetd.d/chargen-stream (1159 bytes), /etc/xinetd.d/daytime-dgram (1157
bytes), /etc/xinetd.d/daytime-stream (1159 bytes), /etc/xinetd.d/discard-dgram (1157 bytes),
/etc/xinetd.d/discard-stream (1159 bytes), /etc/xinetd.d/echo-dgram (1148 bytes),
/etc/xinetd.d/echo-stream (1150 bytes), /etc/xinetd.d/tcpmux-server (1212 bytes),
/etc/xinetd.d/time-dgram (1149 bytes), /etc/xinetd.d/time-stream (1150 bytes),
/usr/sbin/inetdconvert (7374 bytes), /usr/sbin/xinetd (164660 bytes), /usr/share/doc/xinetd-2.3.14
(4096 bytes), /usr/share/doc/xinetd-2.3.14/CHANGELOG (42721 bytes),
/usr/share/doc/xinetd-2.3.14/COPYRIGHT (2087 bytes), /usr/share/doc/xinetd-2.3.14/INSTALL (957
bytes), /usr/share/doc/xinetd-2.3.14/README (5724 bytes), /usr/share/doc/xinetd-2.3.14/empty.conf
(1284 bytes), /usr/share/doc/xinetd-2.3.14/sample.conf (4964 bytes),
/usr/share/man/man5/xinetd.conf.5.gz (11301 bytes), /usr/share/man/man8/xinetd.8.gz (2816 bytes),
/usr/share/man/man8/xinetd.log.8.gz (1350 bytes), et a été installé lun 10 déc 2007 18:58:04 CET
```


Notes personnelles

Index lexical

Notes personnelles

Notes personnelles

Notes personnelles