

Cours 4 : Séquence de boot et déploiement

Christophe Gonzales

3I015 — Principes et pratiques de l'administration des systèmes

① La séquence de démarrage

1 Le BIOS

Détection du matériel et chargement du boot loader

2 GRUB : le boot loader

chargement et exécution du noyau Linux

3 Le noyau Linux

mise en place de l'infrastructure du système
et lancement d'init

4 Le process `init`

montage des filesystems, lancement des démons,
mise en place des terminaux, éradique les zombies

BIOS : Basic Input Output System

- EEPROM sur la carte mère
- fonctions effectuées lors de la mise sous tension :
 - *Power-On Self-Test (POST)* :
 - vérification initialisation du processeur
 - vérification de la mémoire de base (64 Ko)
 - vérification de l'intégrité de la carte mère
 - init des entrées/sorties : clavier, carte graphique
 - affichage « Press F2 to run setup »
 - *Initialisation de l'ordinateur* :
 - énumération et initialisation des périphériques
 - identification des périphériques amorçables
 - exécution du code de démarrage du périphérique choisi
- accessible via les touches ESC, SUPPR ou F2

GRUB : GRand Unified Bootloader

- permet de choisir le système d'exploitation à lancer
- BIOS charge le secteur d'amorçage (MBR) du périphérique choisi (premiers 512 octets)
- fichier `boot.img` stocké dans le MBR
- `boot.img` s'exécute et charge `core.img` (partie 1)
- `core.img` s'exécute (partie 1.5)
- `/boot/grub/i386-pc/normal.mod` s'exécute (partie 2)
- `/boot/grub/i386-pc/normal.mod` boote un système d'exploitation

exécution de `/boot/grub/i386-pc/normal.mod`

- charge et parse `/boot/grub/grub.cfg`
- affiche menu de démarrage
- laisse l'utilisateur choisir un système à démarrer
- invoque la commande `<< boot >>`

Configuration du boot loader

- `/boot/grub/grub.cfg` :
 - fichier de configuration exploité par grub au boot
 - fichier généré par `update-grub`
- `/etc/default/grub` : configuration générale de grub (hors règles spécifiques aux OS)
- `/etc/grub.d/` : répertoire des scripts utilisés pour générer `grub.cfg`

<code>00_header</code>	<code>10_linux</code>	<code>20_memtest86+</code>	<code>40_custom</code>
<code>05_debian_theme</code>	<code>20_linux_xen</code>	<code>30_os-prober</code>	<code>41_custom</code>

GRUB charge le noyau Linux :
le cœur du système

noyau d'un système d'exploitation

logiciel qui assure :

- communication logiciels \longleftrightarrow matériel
- gestion des process (lancement, ordonnancement...)
- gestion du matériel (mémoire, processeur, périphériques, stockage...)
- gestion des systèmes de fichiers



noyau = ensemble de routines système \neq process
travaille dans l'espace noyau \neq espace utilisateur

Appels système = fonctions :

- appelées depuis un programme de l'espace utilisateur
- exécutées dans l'espace noyau
- retour est effectué dans le programme appelant dans l'espace utilisateur

Exécution du noyau Linux (1/2)

- GRUB \implies chargement du noyau
-

- noyau parse les arguments transmis par GRUB

```
linux /boot/vmlinuz-3.16.0-4-amd64 root=/dev/sda2 ro single  
initrd /boot/initrd.img-3.16.0-4-amd64
```

option	signification
root=	partition où se trouve /
ro	montage de / en read-only \implies fsck
rw	montage de / en read-write \implies \neg fsck
nfsroot=	/ monté par NFS
ip=	configure l'interface réseau pour NFS
nfsaddrs=	idem
ks=	où trouver le fichier de configuration kickstart



Arguments non parsés \implies transmis à `init`

Exécution du noyau Linux (2/2)

- 0 noyau parse les arguments transmis par GRUB
 - 1 initialise un minimum de périphériques
 - 2 création d'un / temporaire en mémoire
 - 3 exécute fonction `startup` = `swapper`
-

- 4 configure la mémoire, les IRQ
- 5 monte l'« init RAM disk »
 - 6 charge des modules / drivers
 - 7 initialise des périphériques
- 8 monte le « vrai » système de fichiers « / »
- 9 exécute `init` : le processus n°1



processus dans le userspace !

Anatomie d'un init RAM disk

```
[root@msLDAP tmp]# egrep initrd /boot/grub/grub.cfg  
initrd /boot/initrd.img-3.16.0-4-amd64
```

```
[root@msLDAP tmp]# cp /boot/initrd.img-3.16.0-4-amd64  
./initrd.gz
```

```
[root@msLDAP tmp]# gunzip ./initrd.gz
```

```
[root@msLDAP tmp]# mkdir tmp2 && cd tmp2
```

```
[root@msLDAP tmp2]# cpio -id < ../initrd  
90952 blocks
```

```
[root@msLDAP tmp2]# ls
```

```
bin      conf    etc     init      lib  
lib64   run     sbin   scripts
```

```
[root@msLDAP tmp2]# cd lib64/x86_64-linux-gnu && ls -l libc.*  
-rwxr-xr-x 1 root root 1729984 août 24 11:32 libc.so.6
```

Le noyau a chargé :

- les routines système
- des drivers pour les périphériques
- des modules essentiels du noyau
- des bibliothèques dynamiques
- les structures de données pour gérer les processus

⇒ les processus peuvent être exécutés

`init` : l'ancêtre de tous les process

- `init` fait partie de `systemd`

Les missions principales de `init` :

- exécute au boot les process/démons nécessaires
 - ⇒ découvrent les périphériques restants
 - ⇒ montent les systèmes de fichiers
 - ⇒ démarrent les services
- détruit les process terminés après la mort de leurs parents
- exécute les process nécessaires à l'arrêt de la machine

- Découpage en unités :
 - service : les process et démons
 - target : points de synchronisation pour le démarrage des services
 - mount, automount : points de montage contrôlés par systemd (*exemple* : `/proc/sys/fs/binfmt_misc`)
 - slice : permet de regrouper des unités
 - autres unités : socket, device, snapshot, timer, swap, path, scope
- Fonctionnement par événements (règles `After=`, `Before=`, `WantedBy=`, *etc.*)
 - ⇒ boot hautement parallélisé !

- Parallélisation
- Gestion simple des dépendances
- Gestion simple des événements
- Analyses possibles :
 - `systemd-analyze blame`
 - `systemd-analyze critical-chain`
- Gestion des process par cgroup : `systemd-cgls`

- Exécutables de `systemd` \implies répertoire `/lib/systemd`
- Fichiers de configuration \implies répertoires :
 - `/lib/systemd/system`
 - `/etc/systemd/system`
- `default.target` : 1ère target que `systemd` essaye d'atteindre

Default.target

```
# This file is part of systemd.  
#  
# systemd is free software; you can redistribute it and/or  
# modify it under the terms of the GNU Lesser General Public  
# License as published by the Free Software Foundation;  
# either version 2.1 of the License, or (at your option) any  
# later version.
```

```
[Unit]
```

```
Description=Graphical Interface  
Documentation=man:systemd.special(7)  
Requires=multi-user.target  
After=multi-user.target  
Conflicts=rescue.target  
Wants=display-manager.service  
AllowIsolate=yes
```

Syntaxe des Unit de systemd

Description=	description des fonctionnalités de l'unité
Documentation=	où se trouve la documentation
Requires=	liste les unités qui doivent être activées avec succès pour que l'unité courante soit aussi activée avec succès. Les unités sont démarrées en parallèle par défaut
Wants=	≈ Requires= mais si les unités échouent, l'unité courante continuera à fonctionner
BindsTo=	≈ Requires= mais l'unité courante stoppera quand l'unité associée stoppera
Before=	les unités indiquées doivent être démarrées pour que l'unité courante puisse démarrer
After=	inverse de Before=
Conflicts=	indique les unités qui ne peuvent fonctionner en même temps que l'unité courante. Si celle-ci démarre, cela stoppera les unités en conflit
Condition...=	conditions à respecter pour que l'unité puisse démarrer

<https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files>

<http://www.freedesktop.org/software/systemd/man/systemd.unit.html>

```
# This file is part of systemd.  
#  
# systemd is free software ; you can redistribute it and/or  
# modify it under the terms of the GNU Lesser General Public  
# License as published by the Free Software Foundation ;  
# either version 2.1 of the License, or (at your option) any  
# later version.
```

```
[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
```

Arbre de dépendence

```
[root@msLDAP /]# systemctl list-dependencies graphical.target
```

```
graphical.target
├─ lightdm.service
├─ multi-user.target
│   └─ dbus.service
│   └─ rc-local.service
│   └─ rsyslog.service
│   └─ systemd-update-utmp-runlevel.service
│   └─ basic.target
│       ├── sockets.target
│       │   ├── dbus.socket
│       │   └─ systemd-initctl.socket
│       └─ sysinit.target
│           ├── dev-hugepages.mount
│           ├── keyboard-setup.service
│           ├── networking.service
│           ├── proc-sys-fs-binfmt_misc.automount
│           └─ local-fs.target
│               ├── -.mount
│               ├── systemd-fsck-root.service
│               ├── systemd-remount-fs.service
│               └─ toto.mount
└─ getty.target
    ├── getty-static.service
    └─ getty@tty1.service
```

```
[Unit]
Description=System Logging Service
Requires=syslog.socket
Documentation=man :rsyslogd(8)
Documentation=http://www.rsyslog.com/doc/
```

```
[Service]
Type=notify
ExecStart=/usr/sbin/rsyslogd -n
StandardOutput=null
Restart=on-failure
```

```
[Install]
WantedBy=multi-user.target
Alias=syslog.service
```

⇒ mécanisme simple pour rajouter de nouveaux services

- Compatibilité avec système V :

```
[root@msLDAP /lib/systemd/system]# ls -go runlevel*.target |  
awk 'print substr($0, index($0,$7))'  
runlevel0.target -> poweroff.target  
runlevel1.target -> rescue.target  
runlevel2.target -> multi-user.target  
runlevel3.target -> multi-user.target  
runlevel4.target -> multi-user.target  
runlevel5.target -> graphical.target  
runlevel6.target -> reboot.target
```

- Les runlevels peuvent être passés en paramètre dans grub :

```
linux /boot/vmlinuz-3.16.0-4-amd64 root=/dev/sda2 ro 1
```

- Mais système V obsolète (manque de parallélisation et de flexibilité)

- /etc/fstab convertie en Unit systemd par /lib/systemd/system-generators/systemd-fstab-generator
- Certains générateurs sont exécutés très tôt (avant les Unit)
- Ils placent leurs résultats dans /run/systemd/generator/

```
[root@msLDAP /]# ls -F /run/systemd/generator/
```

```
local-fs.target.requires/      multi-user.target.wants/  
local-fs.target.wants/        remote-fs.target.d/  
-.mount                        toto.mount  
mountall.service.d/          umountfs.service.d/  
mountall-bootclean.service.d/ umountnfs.service.d/  
mountnfs.service.d/
```


-.mount et /etc/fstab

```
# /etc/fstab : static file system information. #
# <file system> <mount point> <type> <options>          <dump> <pass>
/dev/sda1      /                ext4  errors=remount-ro 0      1
/dev/sda2      none            swap  sw                 0      0
/dev/sda3      /toto           ext4  defaults           0      2
```

```
[root@msLDAP /run/systemd/generator]# more ./-.mount
```

```
# Automatically generated by systemd-fstab-generator
```

```
[Unit]
```

```
SourcePath=/etc/fstab
```

```
Documentation=man:fstab(5) man:systemd-fstab-generator(8)
```

```
Before=local-fs.target
```

```
[Mount]
```

```
What=/dev/sda1
```

```
Where=/  

```

```
Type=ext4
```

```
Options=errors=remount-ro
```

```
[root@msLDAP /run/systemd/generator]# ls -go local-fs.target.requires/
```

```
-.mount -> /run/systemd/generator/-.mount
```

```
toto.mount -> /run/systemd/generator/toto.mount
```

```
[root@msLDAP /run/systemd/generator]# more toto.mount
# Automatically generated by systemd-fstab-generator

[Unit]
SourcePath=/etc/fstab
Documentation=man:fstab(5) man:systemd-fstab-generator(8)
Before=local-fs.target
RequiresOverridable=systemd-fsck@dev-sda3.service
After=systemd-fsck@dev-sda3.service

[Mount]
What=/dev/sda3
Where=/toto
Type=ext4
```

Et si le boot se passe mal ?

- démon rsyslogd
démarré par `/lib/systemd/system/rsyslog.service`

- fichier de configuration : `/etc/rsyslog.conf`

```
# First some standard log files. Log by facility.
auth,authpriv.*      /var/log/auth.log
 *.* ;auth,authpriv.none -/var/log/syslog
daemon.*             -/var/log/daemon.log
kern.*               -/var/log/kern.log
lpr.*                -/var/log/lpr.log
mail.*               -/var/log/mail.log
user.*               -/var/log/user.log
```

- Journal de systemd : `journalctl`

- `journalctl -b` : logs du dernier boot
- `journalctl --list-boots` : liste des boots loggués
- `journalctl --since 09:00 --until 10:00`
- `journalctl -u networking.service` : logs d'une unité

- commande `shutdown` = lien symbolique vers `systemctl`

$\Rightarrow \begin{cases} \text{shutdown -h} \rightarrow \text{systemctl poweroff} \\ \text{shutdown -r} \rightarrow \text{systemctl reboot} \end{cases}$

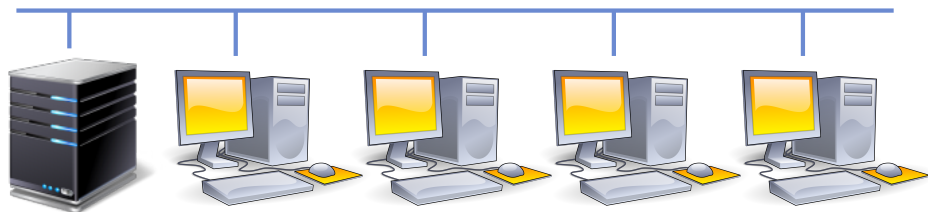
- `systemctl poweroff`

$\Leftrightarrow \text{systemctl start poweroff.target --irreversible}$

\Leftrightarrow changer de runlevel en système V.

$\Leftrightarrow \text{telinit 0}$

② Déploiement



Problème : comment installer linux sur toutes les machines ?

- 1 installation à la main avec un DVD ❌
- 2 installation automatique via ethernet ✔️

Point de vue de la machine à installer

- récupération d'une adresse IP
- récupération du noyau via le réseau
- exécution du noyau linux
- récupération d'un script d'installation (FAI)
- récupération/montage des packages à installer
- exécution des scripts d'installation

FAI : Fully Automatic Installation

Point de vue du serveur d'installation

- serveur d'adresses IP `dhcp`
- serveur de fichiers `tftp` pour le noyau et son chargeur
- rendre accessible un noyau linux ainsi qu'un chargeur de noyau (\approx grub)
- serveur de packages `nfs` ou `apache2` ...
- scripts d'installation : `FAI`

Installation d'un serveur dhcp (1/2)

- éditer le fichier `/etc/default/isc-dhcp-server` :

```
# Path to dhcpd's config file (default : /etc/dhcp/dhcpd.conf).
DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

# Path to dhcpd's PID file (default : /var/run/dhcpd.pid).
DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid

# Additional options to start dhcpd with.
# Don't use options -cf or -pf here ; use DHCPD_CONF/ DHCPD_PID
instead
#OPTIONS=""
# On what interfaces should the DHCP server (dhcpd) serve DHCP
requests ?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="enp0s3"
INTERFACESv6=""
```

Installation d'un serveur dhcp (2/2)

- éditer le fichier `/etc/dhcp/dhcpd.conf` :

```
option domain-name "3i015_dhcp" ; # nom du réseau
default-lease-time 3600 ;          # leasing = temps de validité
max-lease-time 7200 ;             # des addresses IP
authoritative ;                   # serveur DHCP officiel du réseau
subnet 192.168.X.0 netmask 255.255.255.0 { # X = votre n° de machine
    option routers 192.168.X.2 ;      # la passerelle
    option broadcast-address 192.168.X.255 ; # le broadcast
    host msLDAP {                     # affectation d'une
        hardware ethernet 08:00:27:5c:de:3a ; # adresse IP
        fixed-address 192.168.X.1 ; }    # fixe à ssLDAP
    pool {
        next-server 192.168.X.2 ;      # permet tftp/bootp
        filename="fai/pxelinux.0" ;  # boot loader
        range 192.168.X.150 192.168.X.200 ; # plage d'adresses
        allow unknown-clients ; }     # allouées dynamiquement
}
```

- leasing : temps pendant lequel l'adresse IP affectée est valide
- subnet : le réseau servi par dhcpd
- Possibilité d'affecter des adresses IP fixes et/ou dynamiques
- tftp/bootp : mécanisme de récupération du noyau à distance
⇒ pxelinux.0 : image de démarrage (≈ grub)

Trivial File Transfert Protocol

- Package `tftpd-hpa`
- Fichier de configuration : `/etc/default/tftpd-hpa`
- Fichiers à exporter dans répertoire `/srv/tftp`
Dans `dhcpd.conf` : `filename="fai/pxelinux.0"`
⇒ fichier `/srv/tftp/fai/pxelinux.0`

- Package `syslinux-common` : boot loader \approx grub
 - ⇒ fichiers dans `/usr/lib/syslinux/modules/`
 - ⇒ fichier utilisé par FAI :
`/usr/lib/syslinux/modules/bios/ldlinux.c32`
- Package `pxelinux` : boot loader via le réseau
 - ⇒ fichiers dans `/usr/lib/PXELINUX`
 - ⇒ fichier utilisé par FAI : `/usr/lib/PXELINUX/pxelinux.0`

⇒ installer `syslinux-common` et `pxelinux`

Fully Automatic Installation

- fichiers de configuration : 2 répertoires :
 - `/etc/fai` : configuration du *nfs root filesystem*
paramètres pour créer le répertoire `/srv/fai/nfsroot`
(mini système linux complet utilisé par les clients)
précise les *paths* et les dépôts à utiliser...
 - `/srv/fai/config` : *config space*
paramètres de l'installation des clients
- `fai-setup` crée le répertoire `/srv/fai/nfsroot`
- `config space` : fonctionne avec des *classes*
(par exemple, DEBIAN, FAIBASE, *etc*)
1 classe \implies directives d'installation
différentes classes \implies différentes installations
 \implies possibilité d'installer différentes distributions
un client \in certaines classes \implies installation spécifique

Au démarrage de l'installation du client :

- 1 Le client récupère son adresse IP via DHCP, puis son boot loader, son noyau Linux et son init ramdisk via tftp
- 2 son système minimal est le `nfsroot` du serveur
- 3 Il réalise les tâches suivantes :
 - a récupération du config space
 - b définition des classes et variables
 - c évaluation de `FAI_ACTION`
 - d installation initiale
 - e partitionnement, création des filesystems, montages
 - f communication avec un miroir debian
 - g debconf + installation de packages
 - h exécutions de scripts de customisation

Configuration de FAI : vue d'ensemble

- 1 Installer les packages : `fai-server` et `fai-storage` (pour la doc)
- 2 Configurer le `nfsroot` (répertoire `/etc/fai`)
- 3 Créer et configurer le *config space* (`/srv/fai/config`)
- 4 Exécuter `fai-setup -v` pour créer le `nfsroot` de FAI
- 5 Exécuter `fai-chboot -B -I default` pour créer le fichier de config du *boot loader*, copier le noyau linux et son init ramdisk dans `/srv/tftp/fai`

... Le serveur est installé. Au démarrage, les clients s'installent.

- 3 fichiers importants à éditer :
 - 1 Fichier `/etc/fai/fai.conf` :
 - Définit pour les clients où se trouve le *config space*
 - Indique comment écrire les logs d'installation
 - 2 Fichier `/etc/fai/nfsroot.conf` :
 - Configuration pour créer le `nfsroot` (où se trouvent les dépôts de packages utiles pour créer `nfsroot`, *etc.*)
 - 3 Fichier `/etc/fai/apt/sources.list` :
 - Les noms des dépôts copiés dans `nfsroot`
 - (qui seront utilisés par les clients lors des installs)

- Variables d'environnement du shell
- SERVER : adresse IP/nom du serveur FAI utilisé par les clients
- FAI_CONFIG_SRC : config space utilisé par les clients
- LOGUSER : l'utilisateur qui sauvegardera les logs d'installation
- LOGSERVER : serveur où seront sauvegardés les logs
- FAI_LOGPROTO : protocole pour sauvegarder les logs

```
SERVER=192.168.X.2
```

```
FAI_CONFIG_SRC=nfs://$SERVER/srv/fai/config
```

```
LOGUSER=fai
```

```
LOGSERVER=$SERVER
```

```
FAI_LOGPROTO=ssh
```

- Création du nfsroot par debootstrap :
bootstrap un système debian de base
 - FAI_DEBOOTSTRAP : dépôt à utiliser pour installer les packages du nfsroot (en salle de TME, poste n°2 = miroir Debian)
 - FAI_DEBOOTSTRAP_OPTS : les options à passer à debootstrap
-

```
FAI_DEBOOTSTRAP="buster  
http://ftp.lip6.fr/pub/linux/distributions/debian"  
FAI_DEBOOTSTRAP_OPTS="--exclude=info --include=aptitude  
--no-check-gpg"
```



dans FAI_DEBOOTSTRAP, "buster" spécifié avant http://


- Format apt/sources.list classique
- En TME, machine n°2 = miroir debian

elle s'appelle localement ftp.lip6.fr :

```
deb http://ftp.lip6.fr/pub/linux/distributions  
/debian buster main contrib non-free
```



ci-dessus : 1 seule ligne dans le fichier apt/sources.list !

- Répertoire du config space : `/srv/fai/config`
 - Répertoire à créer
 - Initialisation rapide : copier les fichiers de
`/usr/share/doc/fai-doc/examples/simple/`
-  Faire en sorte que le config space appartienne à l'utilisateur `fai`

Anatomie du *config space*

- contenu du répertoire `/srv/fai/config` :

```
[root@msLDAP /]# ls -F /srv/fai/config
basefiles/ class/ debconf/ disk_config/ files/
hooks/ package_config/ scripts/ tests/
```

- `class` : définitions des variables de classe
- `disk_config` : instructions de partitionnement
- `basefiles` : image minimale du système Linux à installer
- `package_config` : les paquets à installer
- `debconf` : instructions de configuration des paquets
- `hooks` : programmes appliqués à chaque étape de l'install
- `scripts` : scripts de postinstall
- `files` : fichiers utilisés par les différents scripts

Définition des classes

```
[root@msLDAP /]# ls /srv/fai/config/class
10-base-classes 50-host-classes CENTOS.var DEBIAN.var FAIBASE.var
```

- Fichiers préfixés par un numéro : sélection des classes
- Fichiers avec suffixe « .var » : définition de variables

```
[root@msLDAP /]# cat /srv/fai/config/class/50-host-classes
#! /bin/bash
# assign classes to hosts
# use a list of classes for our demo machine
case $HOSTNAME in
demohost|client*)
    echo "FAIBASE DEBIAN DHCP DEMO" ; ;
xfcehost)
    echo "FAIBASE DEBIAN DHCP DEMO XORG XFCE" ; ;
puma)
    echo "FAIBASE DEBIAN DHCP RAID_XEN_VIRTUAL" ; ;
*)
    echo "FAIBASE DEBIAN DHCP" ; ;
esac
```

Variables définies par les classes

```
[root@msLDAP /]# cat /srv/fai/config/class/DEBIAN.var
CONSOLEFONT=
# clavier US
KEYMAP=us-latin1
.....
```

```
[root@msLDAP /]# cat /srv/fai/config/class/FAIBASE.var
# Set UTC=yes if your system clock is set to UTC (GMT),
# and UTC=no if not.
UTC=yes
TIMEZONE=Europe/Berlin
# root password for the new installed linux system ;
# md5 and crypt are possible
# pw is "fai"
ROOTPW='$1$kBnWc0.E$djxB128U7dMkr1tJHPf6d1'
```

Bien sélectionner les classes et les valeurs des variables !

Partitionnement : disk_config

```
[root@msLDAP /]# ls /srv/fai/config/disk_config
CENTOS FAIBASE FAISERVER
```

⇒ définition du partitionnement par classe

```
[root@msLDAP /]# cat /srv/fai/config/disk_config/FAIBASE
# example of new config file for setup-storage
# <type> <mountpoint> <size> <fs type> <mount options>
<misc options>

disk_config disk1 disklabel :msdos bootable :1 fstabkey :uuid

primary /      10G      ext4 rw,noatime,errors=remount-ro
logical swap  200-1G  swap  sw
logical /tmp   100-1G  ext4 rw,noatime,nosuid,nodev createopts="-L
```



documentation : man setup-storage (package
fai-setup-storage)

Sélection des packages : package_config

```
[root@msLDAP /]# cat /srv/fai/config/package_config/FAIBASE
PACKAGES aptitude
fai-client
cron
.....

PACKAGES aptitude I386
linux-image-generic initramfs-tools

PACKAGES aptitude DEBIAN
emacs
gcc
make
```



possibilité de sélectionner des packages en fonction de classes spécifiques (hardware (ci-dessus : I386), etc.)

Configuration automatique des packages : debconf

- Certains packages doivent être configurés (cf. slapd)

```
[root@msLDAP /]# cat /srv/fai/config/debconf/DEBIAN
locales locales/default_environment_locale select en_US.UTF-8
locales locales/locales_to_be_generated multiselect en_US.UTF-8 UTF-8
keyboard-configuration keyboard-configuration/modelcode string pc105
keyboard-configuration keyboard-configuration/xkb-keymap select us
keyboard-configuration keyboard-configuration/variant select USA
keyboard-configuration keyboard-configuration/model select Generic 105-key
keyboard-configuration keyboard-configuration/layoutcode string us
keyboard-configuration keyboard-configuration/optionscode string ctrl :nocaps
```

- debconf-get-selections liste les possibilités :

```
[root@msLDAP /]# debconf-get-selections
# Encrypted admin password :
slapd slapd/internal/adminpw password
# Disposition du clavier :
# Choices : Français, Français - Français (variante obsolète)
keyboard-configuration keyboard-configuration/variant select Français - Fra
.....
```

Les hooks

- Installation FAI \implies *tâches*
(confdir, setup, defclass, defvar, action, partition, etc.)
cf. le transparent \ll FAI : philosophie côté client \gg
- les hooks sont dans `/srv/fai/config/hooks`
- nom du hook : `tâche.class`
- hook : exécuté avant la tâche

```
[root@msLDAP /]# ls /srv/fai/config/hooks
debconf.CENTOS instsoft.DEBIAN updatebase.DEBIAN
```

```
[root@msLDAP /]# cat /srv/fai/config/hooks/updatebase.DEBIAN
#! /bin/bash
if [ -n "$APT_PROXY" ]; then
    echo "Acquire::http::Proxy \"$APT_PROXY\" ;" >
        $target/etc/apt/apt.conf.d/02proxy
else
    rm -f $target/etc/apt/apt.conf.d/02proxy
fi
```

Le répertoire files

```
[root@msLDAP /]# tree /srv/fai/config/files
```

```
etc
├── default
├── dhcp
│   └── dhcpd.conf
│       └── FAISERVER
├── fai
│   ├── apt
│   │   └── sources.list
│   │       └── FAISERVER
│   ├── fai.conf
│   │   └── FAISERVER
│   └── nfsroot.conf
│       └── FAISERVER
├── motd
│   └── FAIBASE
├── rc.local
│   └── FAISERVER
└── selinux
    ├── config
    └── CENTOS
```

⇒ reproduit l'arborescence des fichiers de config

⇒ copier vos fichiers système dans cette arborescence

Le répertoire scripts

- scripts de postinstallation
 - 1 sous-répertoire par classe (éponyme)
-

```
[root@msLDAP /]# ls /srv/fai/config/scripts/DEBIAN
10-rootpw 20-capabilities 30-interface 40-misc
```

```
[root@msLDAP /]# cat
/srv/fai/config/scripts/DEBIAN/10-rootpw
#! /bin/bash

# save maximum error code
error=0 ; trap 'error=$((($ ?>$error ?$ ?:$error))' ERR

# set root password
$ROOTCMD usermod -p $ROOTPW root

exit $error
```

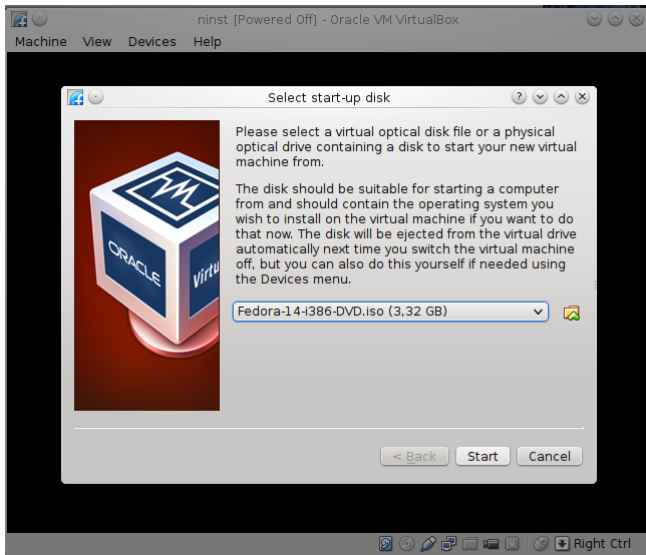
En TME, vous aurez :

- ➊ à installer des serveurs dhcp et tftp
 - ➋ à installer les packages de FAI
 - ➌ à configurer les fichiers dans `/etc/fai`
 - ➍ à créer le *config space*
 - ➎ à mettre à jour tous les fichiers de `/srv/fai/config`
 - ➏ à exécuter `fai-setup -v`
 - ➐ à exécuter `fai-chboot -B -I default`
- ... Voilà, le serveur de déploiement est prêt...

Installation d'une nouvelle machine virtuelle

- 1 Créer une nouvelle machine virtuelle via VirtualBox :
 - créer un disque dur virtuel maintenant
 - sélectionner VDI
 - taille allouée dynamiquement
- 2 Éditez les paramètres de la machine :
 - Système : sélectionner le nombre de processeurs
séquence de boot : disque dur puis réseau
 - Réseau : activer au moins 1 carte
- 3 Démarrer la nouvelle machine

Démarrage de la nouvelle machine (1/4)



- cliquer sur « cancel »

Démarrage de la nouvelle machine (2/4)

```
Intel UNDI, PXE-2.1
PXE Software Copyright (C) 1997-2000 Intel Corporation
Copyright (C) 2010 Oracle Corporation

CLIENT MAC ADDR: 08 00 27 C3 39 46  GUID: 05EB0975-9CFB-44AA-9848-ECBB26672E9D
CLIENT IP: 192.168.1.152  MASK: 255.255.255.0  DHCP IP: 192.168.1.1
GATEWAY IP: 192.168.1.1

PXELINUX 4.02 2010-07-21  Copyright (C) 1994-2010 H. Peter Anvin et al
!PXE entry point found (we hope) at 9DDC:0104 via plan A
UNDI code segment at 9DDC len 199E
UNDI data segment at 9C59 len 1830
Getting cached packet  01 02 03
My IP address seems to be C0A80198 192.168.1.152
ip=192.168.1.152:192.168.1.1:192.168.1.1:255.255.255.0
BOOTIF=01-08-00-27-c3-39-46
SYSUUID=05eb0975-9cfb-44aa-9848-ecbb26672e9d
TFTP prefix: pxelinux/
Trying to load: pxelinux.cfg/default                                ok
Loading vmlinuz.....
Loading initrd.img....._
```

⇒ récupération adresse IP, puis noyau ...

Démarrage de la nouvelle machine (3/4)

```
-----  
Fully Automatic Installation - FAI  
4.3.1+deb8u1 (c) 1999-2015  
Thomas Lange <lange@informatik.uni-koeln.de>  
-----  
  
Calling task_confdir  
Kernel currently running: Linux 3.16.0-4-amd64 x86_64 GNU/Linux  
Kernel parameters: BOOT_IMAGE=vmlinuz-3.16.0-4-amd64 initrd=initrd.img-3.16.0-4-  
amd64 ip=dhcp root=/dev/nfs nfsroot=192.168.1.1:/srv/fai/nfsroot aufs FAI_FLAGS=  
verbose,sshd,reboot FAI_ACTION=install FAI_CONFIG_SRC=nfs://192.168.1.1/srv/fai/  
config  
Reading /tmp/fai/boot.log  
-
```

⇒ début de l'installation via FAI ...

Démarrage de la nouvelle machine (3/4)

```
-----  
Fully Automatic Installation - FAI  
  
4.3.1+deb8u1 (c) 1999-2015  
Thomas Lange <lange@informatik.uni-koeln.de>  
-----  
  
Calling task_install  
Calling task_partition  
Starting setup-storage 1.7  
Using config file: /var/lib/fai/config/disk_config/FAIBASE  
Parted could not read a disk label (new disk?)  
Executing: parted -s /dev/sda mklabel msdos  
  DEGRADED MODE. Incomplete RAID LVs will be processed.  
  Creating directory "/run/lock/lvm"  
  Finding all volume groups  
  No volume groups found  
Executing: parted -s /dev/sda mklabel msdos  
Executing: parted -s /dev/sda mkpart primary "ext3" 1048576B 10738466815B  
Executing: parted -s /dev/sda set 1 boot on  
Executing: parted -s /dev/sda mkpart extended "" 10738466816B 12885952511B  
Executing: parted -s /dev/sda mkpart logical "linux-swap" 10738467840B 118122096  
63B  
Executing: parted -s /dev/sda mkpart logical "ext3" 11812210688B 12885952511B  
_
```

⇒ suite de l'installation via FAI ...

Le BIOS

- http://fr.wikipedia.org/wiki/Basic_Input_Output_System
- <http://www.commentcamarche.net/contents/728-bios-c-est-quoi-comment-y-acceder>

GRUB

- http://fr.wikipedia.org/wiki/GRand_Unified_Bootloader
- http://www.gnu.org/software/grub/manual/html_node/
- http://moi.vonos.net/linux/Booting_Linux_on_x86_with_Grub2/#booting-linux-on-x86-using-grub2

Le noyau linux

- http://fr.wikipedia.org/wiki/Noyau_de_système_d'exploitation
- http://fr.wikipedia.org/wiki/Noyau_Linux
- http://en.wikipedia.org/wiki/Linux_startup_process

L'init RAM disk

- <http://www.linuxinfor.com/french/man4/initrd.html>
- <http://www.thegeekstuff.com/2009/07/how-to-view-modify-and-recreate-initrd-img/>
- <http://en.wikipedia.org/wiki/Initrd>
- http://en.wikipedia.org/wiki/GNU_C_Library

Systemd

- `man systemd`
- <http://linuxide.com/linux-how-to/systemd-boot-process/>
- <http://lea-linux.org/documentations/Systemd>
- <http://www.freedesktop.org/wiki/Software/systemd/>

Documentations générales sur le boot

- <http://oldfield.wattle.id.au/luv/boot.html>
- http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch07_:_The_Linux_Boot_Process
- <http://tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html>

Le serveur DHCP et sa configuration

- http://linux.about.com/od/commands/l/blcmdl5_dhcpopt.htm
- http://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol
- `man 5 dhcpd.conf`

Installation par PXE / FAI

- <http://fai-project.org/fai-guide-fr/>
- <https://wiki.debian.org/FAI>
- <https://debian-handbook.info/browse/stable/sect.automated-installation.html>