Live CD

Table des matières

1 Boot virtuel

```
# apt-get install qemu
# qemu --cdrom cd.iso
```

Qemu n'alloue que 128 Mo de ram aux VM par défaut :

```
Loading /boot/initrd.gz...
Not enought memory to load specified image
```

Si besoin, utiliser :

```
$ qemu -m 256M --cdrom monIso.iso
```

Pour obtenir une console sous X il faut changer la combinaison Ctrl-Alt par Ctrl-Alt-Shift :

```
$ qemu -m 256M -alt-grab --cdrom monIso.iso
```

2 Détail de l'Iso de l'installeur Debian

Dépot debian : *dists/XXX/main/installer-YYY/current/images/cdrom/*

- (A) *debian-cd_info.tar.gz*
- (B) *initrd.gz*
- (C) *vmlinuz*

CD1 debian :

- */isolinux* (A)
- */boot* (A)
- */install* (B)+(C)

3 Construction de l'ISO de l'installeur Debian

On se base sur l'outils DEBIAN-CD (qui utilise un mirroir local).

```
# apt-get install debian-cd
$ cd /usr/share/debian-cd

$ cat CONF.sh
export CODENAME=helium
export MIRROR=/var/lib/mirror/pandore
export TDIR=/var/lib/mirror/tmp
export OUT=/var/lib/mirror/debian-cd-test
export APTTMP=/var/lib/mirror/tmp/apt
export NONFREE=1
export CONTRIB=1
export FORCE_FIRMWARE=1
export amd64_MKISOFS="xorriso"
export amd64_MKISOFS_OPTS="-as mkisofs -r -checksum_algorithm_iso md5,sha1"
export ARCHIVE_KEYRING_PACKAGE=debian-archive-keyring                  // export need to be added here !
ARCHIVE_KEYRING_FILE=usr/share/keyring/useit-archive-keyring.gpg
export DEBOOTSTRAP_OPT="--keyring $TDIR/archive-keyring/$ARCHIVE_KEYRING_FILE"
export INSTALLER_CD=2
export TASK=debian-installer+kernel
export MAXCDS=1

$ . CONF.sh
$ make distclean
$ make status
$ make packagelists
$ make image-trees
$ make image CD=1
```

On cherche à réduire au minimum chacune des étapes. On les réduits en commençant par la dernière afin de garder une iso fonctionelle. Finalement seules les 2 dernières importent vraiment.

3.1 Preparation du répertoire CD1

L'avant dernière étape contruit les répertoires *boot1/isolinux* et *CD1/boot*. Ces opérations sont effectives au bout de 3 appels :

```
0- make image-trees
 1- make_disc_trees.pl
  2- start_new_disc
   3- boot-$ARCH
```

- 1)

  ```
  $ /home/nroche/debian-cd/tools/make_disc_trees.pl
     \home/nroche/debian-cd /var/lib/mirror/pandore \
     /var/lib/mirror/tmp helium "amd64" "xorriso" \
     "-as mkisofs -r -checksum_algorithm_iso md5,sha1 -jigdo-min-file-size 1024 -jigdo-exclude 'README*' \
      -jigdo-exclude /doc/ -jigdo-exclude /md5sum.txt -jigdo-exclude /.disk/ -jigdo-exclude /pics/ \
      -jigdo-exclude 'Release*' -jigdo-exclude 'Packages*' -jigdo-exclude 'Sources*' -jigdo-force-md5 /pool/"
  ```

- 2)

```
$ /home/nroche/debian-cd/tools/start_new_disc \
   /home/nroche/debian-cd \
   /var/lib/mirror/pandore \
   /var/lib/mirror/tmp \
   helium "amd64" 1
```

- 3)

```
$ export BASEDIR=$PWD
$ export DI_CODENAME=helium
$ export MIRROR=/var/lib/mirror/pandore
$ export DEBVERSION=7.0
$ export ARCH=amd64
$ export MKISOFS=xorriso
$ export DISKINFO_DISTRO="distro_Debian"
$ export DEBIAN_KERNEL="kernel_GNU/Linux"
$ mkdir -p /var/lib/mirror/tmp/helium/CD1
$ tools/boot/helium/boot-amd64 1 /var/lib/mirror/tmp/helium/CD1
```

3.2 Construction de l'iso

La dernière étape contruit l'iso. Cette opération est effective au bout de 2 appels :

```
0- make image CD=1
 1- make_image
  2- mkisofs
```

- 1)

```
$ export MKISOFS=xorriso
$ export MKISOFS_OPT="-as mkisofs -r -checksum_algorithm_iso md5,sha1"
$ export MAXJIGDOS=0
$ /home/nroche/debian-cd/tools/make_image \
  "/var/lib/mirror/tmp/helium" \
  "opt_amd64" \
  "/var/lib/mirror/debian-cd-test" \
  "opt_7.0" \
  "opt_/var/lib/mirror/pandore" \
  "xorriso" \
  "-as mkisofs -r -checksum_algorithm_iso md5,sha1" \
  "opt_jidgo" \
  "opt_jidgo_opts"
```

- 2)

```
$ cd /var/lib/mirror/tmp/helium
$ export volid="Debian 7.0 amd64 1"
$ export opts=`cat 1.mkisofs_opts` || true
$ export dirs=`cat 1.mkisofs_dirs` || true
$ export OUT=/var/lib/mirror/debian-cd-test
$ export OUTFILE=monIso
$ export MKISOFS=xorriso
$ export MKISOFS_OPTS="-as mkisofs -r -checksum_algorithm_iso md5,sha1"
$ $MKISOFS $MKISOFS_OPTS -V "$volid" -o $OUT/$OUTFILE.iso $opts $dirs CD1
$ cd -
```

3.3 Résultats

```
Etapes 1 2 3 4     commentaire
--------------------------------------------------------------------------------------
       - - - -  ok  CD|firmw
       - - - 1  ok  CD|firmw|hd
       - - - 2  ok  CD|firmw|hd|chroot failed
       - - 1    ko No packages file /var/lib/mirror/tmp/helium/packages.amd64 for amd64!
       - - 2 2  ok sr0|!firmw (l'installeur boude le CD mais il se monte sans problème)
       - - 3 2  ok sr0|!firmw
       - ! 3 2  ok sr0|!firmw
       ! ! 3 2  ok sr0|!firmw
 !CONF ! ! 3 2  ok sr0|!firmw
       (script) ok sr0|!firmw
```

rq : un warning s'est glissé à un moment donné :

```
xorriso : WARNING : -volid text problematic as automatic mount point name
xorriso : WARNING : -volid text does not comply to ISO 9660 / ECMA 119 rules
```

Le mirroir peut être réduit :

```
$ mkdir -p mirror/dists/helium/main/binary-amd64
$ mkdir -p mirror/pool/main/s/syslinux/
$ cp -r [a_mirror]/dists/helium/main/installer-amd64 \
    mirror/dists/helium/main/
$ cp [a_mirror]/dists/helium/main/binary-amd64/Packages.gz \
    mirror/dists/helium/main/binary-amd64
$ cp [a_mirror]/pool/main/s/syslinux/syslinux-common_4.05+dfsg-6+deb7u1_all.deb \
    mirror/pool/main/s/syslinux/

$ du -sh mirror
375M
```

Voici le script résultant :

```
#!/bin/bash
set -e

export BASEDIR=$PWD
export DI_CODENAME="wheezy"
export MIRROR="$PWD/mirror"
export DEBVERSION="7.0"
export ARCH="i386"
export DISKINFO_DISTRO="distro_Debian"
export DEBIAN_KERNEL="kernel_GNU/Linux"
export volid="Debian 7.0 amd64 1"
export MKISOFS="xorriso"
export MKISOFS_OPTS="-as mkisofs -r -checksum_algorithm_iso md5,sha1"

# cleaning
###########
rm -fr tmp
rm -f monIso.iso

# prepare ISO
```

```
##############
mkdir -p tmp/$DI_CODENAME/CD1
tools/boot/$DI_CODENAME/boot-$ARCH 1 $PWD/tmp/$DI_CODENAME/CD1

# build ISO
############
cd tmp/$DI_CODENAME
export opts=`cat 1.mkisofs_opts`
export dirs=`cat 1.mkisofs_dirs`
$MKISOFS $MKISOFS_OPTS -V "$volid" -o $BASEDIR/monIso.iso $opts $dirs CD1
cd -

# test ISO
###########
qemu-system-x86_64 --cdrom monIso.iso
```

4 Construction from scratch

4.1 Démarrer depuis une ISO

Fichiers à récupérer :

```
mkdir inputs
cd inputs
MIRROR=http://ftp2.fr.debian.org/debian
wget $MIRROR/dists/wheezy/main/installer-i386/current/images/cdrom/debian-cd_info.tar.gz
wget $MIRROR/dists/wheezy/main/installer-i386/current/images/cdrom/initrd.gz
wget $MIRROR/dists/wheezy/main/installer-i386/current/images/cdrom/vmlinuz
wget $MIRROR/pool/main/s/syslinux/syslinux-common_4.05+dfsg-6+deb7u1_all.deb
cd -
cp ???/splash.png inputs/
```

Script :

```
#!/bin/bash
set -e
#set -x

rm -fr CD boot syslinux
rm -f monIso.iso

# Linux
mkdir -p CD/boot
cp inputs/vmlinuz inputs/initrd.gz inputs/debian-cd_info.tar.gz CD/boot

# Boot loader
echo "\\tools\\loadlin.exe \\boot\\vmlinuz initrd=initrd.gz" | todos > CD/boot/install.bat

# sysLinux (pour récupérer 2 fichiers)
mkdir syslinux/
dpkg --fsys-tarfile inputs/syslinux*.deb > syslinux.tar
tar -C syslinux/ -xf syslinux.tar ./usr/lib
rm syslinux.tar

MKISOFS_OPT="-isohybrid-mbr syslinux/usr/lib/syslinux/isohdpfx.bin"
```

```
# isoLinux
mkdir -p boot/isolinux
tar -C boot/isolinux -zxf inputs/debian-cd_info.tar.gz
MKISOFS_OPT="$MKISOFS_OPT -partition_offset 16"
# Not everything in the tarball is isolinux stuff
mv boot/isolinux/g2ldr* boot
mv boot/isolinux/setup.exe boot
mv boot/isolinux/win32-loader.ini boot
rm boot/isolinux/*.withgtk

# Modify win32-loader.ini
sed -i "s|install/|boot/|" boot/win32-loader.ini
sed -i "/^default_desktop/ d" boot/win32-loader.ini

cp syslinux/usr/lib/syslinux/isolinux.bin boot/isolinux/
cp syslinux/usr/lib/syslinux/vesamenu.c32 boot/isolinux/

#rm -r boot/isolinux/desktop
sed -i "s|%install%|boot|" boot/isolinux/*.cfg
rm -f boot/isolinux/amd*.cfg boot/isolinux/desktop/amd*.cfg

# should be 640x480
pngtopnm < inputs/splash.png | ppmquant 16 | \
 ppmtolss16 "#ffffff=7" "#000000=0" > boot/isolinux/splash.rle
pngtopnm < inputs/splash.png | ppmquant 16 | pnmtopng > boot/isolinux/splash.png
sed -i "s|built on|built $BUILD_DATE; d-i|" boot/isolinux/f1.txt

# ouvre un navigateur sous windows (je pense)
todos > CD/autorun.inf <<EOF
[autorun]
open=setup.exe
EOF

# au cas où (pour la suite)
cp -f inputs/menu.cfg inputs/txt.cfg boot/isolinux/ || true
tar -C CD -zxf root.tgz || true

# Common mkisofs options when creating CDs
MKISOFS_OPT="$MKISOFS_OPT -J -joliet-long"
MKISOFS_OPT="$MKISOFS_OPT -cache-inodes"

# Add the normal options to make an ElTorito bootable CD/DVD
MKISOFS_OPT="$MKISOFS_OPT -b isolinux/isolinux.bin"
MKISOFS_OPT="$MKISOFS_OPT -c isolinux/boot.cat"
MKISOFS_OPT="$MKISOFS_OPT -no-emul-boot"
bls=4 # Specify 4 for BIOS boot, don't calculate it
MKISOFS_OPT="$MKISOFS_OPT -boot-load-size $bls"
MKISOFS_OPT="$MKISOFS_OPT -boot-info-table"
MKISOFS_OPT="$MKISOFS_OPT boot"

xorriso -as mkisofs -r -checksum_algorithm_iso md5,sha1 \
 -V 'Nico 0.0 i386 1' -o monIso.iso \
 $MKISOFS_OPT CD
```

```
qemu -m 512M -ctrl-grab --cdrom monIso.iso
```

Faire le ménage dans les menus : au final seuls *menu.cfg* et *txt.cfg* semble suffire. Si l'on change l'image, il faut idéalement en trouver une de 640x480 (sinon elle est dupppliquée).

Remarque : (https ://lists.debian.org/debian-boot/2012/03/msg00401.html)

```
> libisofs: WARNING : Can't add /debian to Joliet tree. Symlinks can
> only be added to a Rock Ridget tree.

These warnings are caused by option -J which produces a MS-Windows Joliet
directory tree. They are harmless, except if you want to extract all
files on MS-Windows and want to bring them onto a system that supports
symbolic links.
```

4.2 Remplacement du noyau

Retirer `quiet` et `vga=788` dans *txt.cfg* pour voir les logs du noyau et ajouter `init=/bin/bash` pour prendre la main au plus tôt.

On peut remplacer le noyau et booter sans problème :

```
$ cp /boot/vmlinuz-3.2.0-4-686-pae inputs/vmlinuz
```

Mais alors il manque le driver pour monter le CD.

4.3 Remplacement de l'Initrd

L'idée est de ne pas monter tout le système en mémoire.

```
# mkinitramfs -o initrd.gz
```

Au boot on peut monter le CD :

```
(initramfs) mount -t iso9660 /dev/sr0 /root
```

Ajouter `root=/dev/sr0` dans *txt.cfg* pour monter le CD automatiquement.

Au cas où pour bidouiller :

```
# gunzip initrd.gz
# mkdir ramfs
# cd ramfs
# cat ../initrd | cpio -i

# echo "" > etc/conf.d/resume

# find . | cpio --create --format='newc' > ../initrd
$ cd ..
$ gzip initrd
```

4.4 Ajouter un sytème au CD

On utilise debootstrap puis on fait le ménage (cf https ://wiki.ubuntu.com/ReducingDiskFootprint).

```
# debootstrap --variant=minbase wheezy root http://ftp2.fr.debian.org/debian
# chroot root
# mount -t proc proc /proc
# mount -t sysfs sysfs /sys
```

```
// minimum addons
# passwd
# apt-get install locales
# dpkg-reconfigure locales
# apt-get install localepurge
# apt-get install console-tools console-data
# ln -s i386/azerty/fr-latin9.kmap.gz /usr/share/keymaps/defkeymap.kmap // pour loadkeys -d

// X + browser
# apt-get install xorg
# apt-get install dillo

// free many megabytes as possible (/2)
# apt-get install localepurge
# localepurge
# apt-get clean

# rm /var/lib/apt/lists/*
# rm /var/cache/apt/*
# rm -r /usr/share/doc/*
# rm -r /usr/share/man/*

// 10 more megabyte into /usr/share/i18n ??

# umount /proc
# umount /sys
# ^D
# tar -zcf root.tgz -C root .
```

root/etc/fstab :

```
none            /proc           proc    defaults        0       0
none            /sys            sysfs   noauto          0       0
```

```
$ cp /boot/vmlinuz-3.2.0-4-686-pae inputs/vmlinuz
# mkdir root/lib/modules
# cp -fr /lib/modules/3.2.0-4-686-pae/ root/lib/modules/
# cd root
# find . | cpio --create --format='newc' > ../initrd
# cd ..
# gzip initrd
$ cp initrd.gz inputs/initrd.gz
```

A présent on boot, mais le système n'est accessible qu'en lecture seule.

```
# loadkeys fr
```

Archiver puis vider le contenu des répertoires à basculer en lecture/écriture :

```
# rm root/tmp
# ln -s /var/tmp root/tmp
# tar -zcf root/etc.tgz -C root etc
# tar -zcf root/var.tgz -C root var
# rm -fr root/etc/*
# rm -fr root/var/*
```

Modifier l'initrd afin de décompresser les répertoires dans des disques RAM : *ramf/init* :

```
# Move virtual filesystems over to the real filesystem
mount -n -o move /sys ${rootmnt}/sys
mount -n -o move /proc ${rootmnt}/proc

echo "////////////////// extract /var //////////////////"
mount -t tmpfs -o nosuid,size=5m,mode=755 tmpfs /root/var
mount -t tmpfs -o nosuid,size=200m,mode=755 tmpfs /root/var
tar -zxf root/etc.tgz -C root
tar -zxf root/var.tgz -C root
echo "////////////////// init endings //////////////////"
#/bin/sh

# Chain to real filesystem
```

4.5 Squashfs

```
# apt-get install squashfs-tools
```

Générer la partition squashfs

```
# mv root/var.tgz .
# mv root/etc.tgz .
# mksquashfs root squashfs -m 256K
```

Regénérer l'initrd en ajoutant le module *squashfs* dans le fichier */etc/initramfs-tools/modules* :

```
# echo "squashfs" >> /etc/initramfs-tools/modules
# mkinitramfs -o initrd.gz
```

(initramfs-tools(8) est trés bien documenté mais j'ai pas trouvé la façon propre)
MODULES=list permet de restreindre les modules à charger
Replace `mountroot()` function into *ramf/init/scripts/local* :

```
mountroot()
{
 wait_for_udev 10

 mount -t iso9960 /dev/sr0 /cdrom
 mount -t squashfs -o loop /cdrom/squashfs /root/

 mount -t tmpfs -o nosuid,size=5m,mode=755 tmpfs /root/etc
 mount -t tmpfs -o nosuid,size=200m,mode=755 tmpfs /root/var
 tar -zxf /cdrom/etc.tgz -C root
 tar -zxf /cdrom/var.tgz -C root
}
```

Retirer `root=/dev/sr0` de *txt.cfg*. Ajouter `init=/bin/bash` permet de prendre la main juste après l'initrd.

`xinit` permet de lancer X avec un terminal mais celui-ci est figé. Il manque les drivers (`lspci -k`) de la carte graphique (entre autres) et les utilitaires du noyau (modprobe...).

```
# mkdir lib/modules
# cp -fr /lib/modules/3.2.0-4-686-pae lib/modules
```

TODO : `apt-get install kmod` Peut-être à cause de `debootstap --variant=minbase`?

4.6 Résumé

```
ISO
 isolinux  (récupéré depuis le mirroir debian)
 vmlinuz   (récupéré sur ma machine)
 initrd    (construit via mkinitramfs)
 root      (construit via debootstrap et mkfsquachfs)
  etc      (construit via tar)
  var      (construit via tar)
```

4.7 Réduire la taille du système

L'autre idée c'est d'utiliser un montage CramFs (ou mieux : SquashFS) en plus après le montage du CD.

```
$ mksquashfs root root.squashfs -b 256K         // 128K par défaut => plus lent mais taille identique (/2)
$ mount -t squashfs -o loop root.squashfs mnt/  // loop car ne provient pas directement d'un device
```