Compilation noyau

Généralités

Noyau Linux.

Compilation Noyau

Après la restauration du système avec les images, il faut recompiler le noyau.

Deux cas de figures peuvent se passer.

- Le système ne peut ne pas se lancer à cause d'une mauvaise reconnaissance matérielle, il faudra donc recompiler le noyau à partir du SRCD. Dans mon cas, la résolution utilisée par le système n'était pas supportée par mon écran ou le chipset graphique avait changé et je n'avais pas d'affichage. J'ai donc du utiliser le SRCD.
- Le système boot correctement, on a accès à la console, on peut donc recompiler directement à partir du système.

Le système ne demarre pas : Compilation à partir du SRCD

Lancer le SystemRescueCD (SRCD) sur le serveur en appuyant sur entrée.

Monter les volumes logiques correspondant à `/usr` et `/boot` avec les manipulations suivantes.
Activer le groupe de volume LVM nommé Mon_Linux.

```
vgchange -ay
```

Créer le dossier de montage.

```
mkdir /mnt/destination
```

Monter les deux volumes suivants.

```
mount -t ext3 /dev/Mon_Linux/Ma_Racine /mnt/destination/.
mount -t ext3 /dev/Mon_Linux/Mes_Programmes /mnt/destination/usr
mount -t ext3 /dev/Mon_Linux/Mon_Temporaire /mnt/destination/tmp
mount -t ext3 /dev/Mon_Linux/Mes_Logs /mnt/destination/var
```

Après ces manipulations, on peut se positionner en chroot sur le système installé sur le disque dur C'est à dire qu'on change de racine. On va faire les opérations comme si on était sur le système qui vient d'être restauré. A l'inverse, on ne veut pas que les commandes s'exécutent sur le CD Live.

```
chroot /mnt/destination /bin/bash
```

Le prompt en couleur du SRCD doit laisser place à un prompt de couleur blanche sur fond noir. A partir de maintenant, les conséquences des commandes ne sont plus répercutées sur le SRCD.

```
mv /root/sd* /dev
```

On est obligé de récupérer le descripteur de disque car il n'y a pas de daemon udev de lancé pour le système sur le disque avec un système live. Il faut donc récupérer le fichier correspondant au device du système stocké au préalable dans /root après la restauration). Un fichier sda1 doit être présent dans /root. Il a été ajouté après la procédure de restauration.

```
mount /boot
```

Exécuter ensuite une procédure standard de compilation noyau.

Procédure standard de compilation

Si le système boot ou si vous venez de l'étape précédente, voici ce qu'il faut réaliser pour compiler le noyau Linux.

```
cd /usr/src/linux
```

Copie du .config dans `/root`.

```
cp .config /root/config.orig
```

Nettoyage profond des sources

```
make mrproper
```

Recopier le .config sauvegardé.

```
cp /root/config.orig ./.config
```

Paramétrage de la configuration noyau

```
make menuconfig
```

Vérifier et activer les modules suivants.

- Processor type and features
  - SMP ?
  - Processor family (P4, Xeon…)
  - Max CPU
  - Memoire high Memory (4 Go ou jusqu'à 64 Go)
- Device drivers
  - ATA/ATAPI

- SCSI device support
  - SCSI low-level drivers
    - QLAxxx
    - LSI
- SATA
- Fusion MPT device support
  - Support LSI
- Device Drivers
  - Network device support
    - Ethernet (10 or 100Mbit)
      - VIA Rhine support (Carte réseau D-Link DFE-530TX)
    - Fibre Channel driver support
- Character device (tout AGP)
  - /dev/agpgart
- Graphics support
  - Display device support
    - Display panel/monitor support
  - VGA 16-color graphics support
  - Console display driver support
    - Framebuffer console support

Conserver le `.config` généré.

Générer le noyau

La configuration sera positionnée dans `/usr/src/linux/arch/i386/boot/bzImage`.

```
make dep clean bzImage (noyau 2.4)
make clean bzImage (noyau 2.6)
```

2 fichiers créés.

```
setup (c’est le System.map)
system
```

Le kernel se situe dans : `arch/i386/boot/bzImage`.

Copier les fichiers générés en conservant les anciens

```
mv /boot/System.map-2.6.22.8 /boot/System.map-ancien (renommer)
cp System.map /boot/System.map-2.6.22.8
mv /boot/vmlinuz-2.6.22.8 /boot/vmlinuz-ancien (renommer)
cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.22.8 (écraser l’ancien. Changer
i386 par l’architecture ex : x86_64)
```

Compiler les modules

```
make modules
```

La commande compile les modules qui ne sont pas cochés comme intégrés au noyau mais noté en module (M). Elle les positionne dans `/usr/src/linux/lib/`. Elle créée les.ko.

Installer les modules.

```
make modules_install
```

Il fait une simple copie des modules de `/usr/src/linux/lib/` dans `/lib/modules/nomnoyau`.

Vérifier `/etc/modules.conf` ou `/etc/modprobe.conf` sur les noyaux récents (en particulier modules carte réseau et le bonding + fiber channel a mettre (?) en commentaire).

Résoudre les dépendances des modules du noyau

```
depmod –ae –F /boot/System.map-2.6.22.8 2.6.22.8
```

Le dernier paramètre correspond au nom du dossier dans lequel se trouve les modules du kernel Linux (`/lib/modules/2.6.22.8`).

Créer le fichier de ramdisk

```
mv /boot/initrd-2.6.22.8.img /boot/initrd.img-ancien (renommer)
mkinitrd /boot/initrd-2.6.22.8.img 2.6.22.8
```

Le dernier paramètre correspond au nom du dossier dans lequel se trouve les modules du kernel Linux (`/lib/modules/2.6.22.8`). Ce fichier contient les modules bas niveaux nécessaires au démarrage de la machine. Par exemple si on utilise LVM, il faut qu'il soit intégré au démarrage pour pouvoir monter les volumes. De même que le module SCSI doit être présent si on dispose de disque SCSI.

Il ne reste plus que l'installation de GRUB avant de redémarrer pour tester.

Installation GRUB via SRCD

Uniquement si compilation à partir du SRCD.

Installer le GRUB. Modifier le fichier `/boot/grub/menu.lst` en ajoutant les lignes nécessaires pour booter sur le nouveau noyau.

```
grub-install /dev/sda
umount /boot
```

```
mv /dev/sda* /root #ceci car il n'y a pas de daemon udev de lancé pour le système sur le disque avec un système live
```

Sortir du chroot et rebooter.

```
exit
reboot
```

Installation GRUB sur un système normalement booté

Installer le GRUB. Modifier le fichier `/boot/grub/menu.lst` en ajoutant les lignes nécessaires pour booter sur le nouveau noyau.

```
grub-install /dev/sda
reboot
```

Récapitulatif des commandes de compilation noyau

Installation noyau methode debian

```
uname -a
uname -r
uname -v
df -h
cd /usr/src/
ll
dpkg -l binutils
apt-get install binutils bzip2
dpkg -l binutils
dpkg -l bzip2
dpkg -l | grep kernel
cat /etc/group
cat /etc/group | less
cat /proc/cpuinfo
apt-cache search ^linux-source
apt-cache search ^kernel-source
apt-cache search ^linux-source
apt-get install lshw
lshw > /root/infos.txt
vi /root/infos.txt
apt-cache search ^linux-source
apt-get install linux-source-2.6.26
tar jxvf linux-source-2.6.26.tar.bz2
chmod 700 linux-source-2.6.26
apt-get remove linux-source-2.6.26
df -hT
apt-get clean
```

```
df -hT
ln -s /usr/src/linux-source-2.6.26/ linux
vi /usr/src/linux/Documentation/Changes
apt-get install gcc make binutils util-linux module-init-tools oprofile
e2fsprogs procps udev libncurses5-dev kernel-package initramfs-tools
cramfsprogs debconf-utils build-essential
cd linux
ls -a
apt-get clean
man apt-get
cp -i /boot/config-2.6.24-1-686 .
ll
make menuconfig
diff /boot/config-2.6.24-1-686 .config
cp .config /root/saveconfig
mv /root/saveconfig /root/config-2.6.26
ll /root/
make-kpkg --initrd --revision1 --append-to-version=-i2ldebian kernel-image
make-kpkg --initrd --revision 1 --append-to-version=-i2ldebian kernel-image
cd ..
dpkg --install linux-image-2.6.26-i2ldebian_1_i386.deb
cd linux
make-kpkg clean
cd arch/x86/boot/
uname -a
reboot
uname -a
cp /usr/src/linux/.config /root/saveconfigOK-2.6.26
cd
cp /usr/src/linux-image-2.6.26-i2ldebian_1_i386.deb /root/
cd
du -sh *
mkdir noyau
mv linux-image-2.6.26-i2ldebian_1_i386.deb noyau/
mv saveconfigOK-2.6.26 noyau/
mv config_i2ldebian_2.6.26 noyau/
reboot
vi infos.txt
uname -a
lsmod
cd noyau/
reboot
cd noyau/
cd /usr/src/linux/arch/x86/boot/
du -sh *
cd ..
du -sh *
du -sh boot/
vi boot/install.sh
htop
vi /boot/grub/menu.lst
```

```
vi /etc/fstab
du -sh boot/
ll boot/
ll /lib/modules/
ll /lib/modules/
cd /usr/src/
cd linux
vi Makefile
cd /boot/
rm config-2.6.26-i2lmain
rm vmlinuz-2.6.26-i2lmain
rm System.map-2.6.26-i2lmain
cd noyau/
cd ..
cd /usr/src/
cd linux
cd
```

Installation noyau manuelle

```
apt-get remove --purge linux-image-2.6.26-i2ldebian_1_i386.deb
apt-get remove --purge linux-image-2.6.26-i2ldebian_1
uname -a
apt-cache search linux-image
apt-get remove --purge linux-image-2.6.26-i2ldebian
vi /boot/grub/menu.lst
cd /usr/src/linux
ll .config
make mrproper
cp /root/noyau/saveconfigOK-2.6.26 ./.config
ll .config
vi Makefile
make menuconfig
cd ..
du -sh *
rm linux-image-2.6.26-i2ldebian_1_i386.deb
du -sh *
cd linux
ll arch/x86/boot/
ll drivers/
make
make modules_install
ll /lib/modules/
du -sh /usr/src/linux/arch/x86/boot/
du -sh /usr/src/linux/arch/x86/boot/*
ll /boot/
cp -i arch/x86/boot/bzImage /boot/vmlinuz-2.6.26-i2ldebian
cp -i System.map /boot/System.map-2.6.26-i2ldebian
cp -i .config /boot/config-2.6.26-i2ldebian
mkinitramfs -o /boot/initrd.img-2.6.26-i2ldebian /lib/modules/2.6.26-
```

```
i2ldebian
cd /
mv vmlinuz vmlinuz.old
ln -si boot/vmlinuz-2.6.26-i2ldebian vmlinuz
mv initrd.img initrd.img.old
ln -si boot/initrd.img-2.6.26-i2ldebian initrd.img
ln -si boot/System.map-2.6.26-i2ldebian System.map
cd /boot/grub/
cp menu.lst menu.lst.orig
vi menu.lst
update-grub  //ajoute le mode single-user automatiquement tant qu'on ecrit
au-dessus de ###END DEBIAN
vi menu.lst
uname -a
reboot
uname -a
cd /boot/
cd
history > noyau/lstcdes.txt
```

Création noyau d'un master (méthode succincte et incomplète)

On laisse la restauration se finir c'est l'heure de prendre le café.

On change la racine sur le terminal 1 `chroot /mnt/image /bin/bash`. La racine de ce terminal devient la racine du systeme que l'on vient de restaurer.

On monte le pseudo systeme de fichier `/proc`.

```
mount /proc
```

On recompile le noyau.

```
cd /usr/src/linux
make menuconfig
Vérifier les options processeur (mémoire haute type et nombre de
processeurs).
Vérifier le support du port parallèle
Vérifier le support de la disquette.
Vérifier le support des disques IDE  + chipset IDE (lspci –v)
Vérifier le nombre de disque et CDROM SCSI + le driver low level scsi
(attention qlogic ce n'est pas la cf fusion).
Vérifier les cartes réseau et le bonding
Le type de souris /clavier (ps2/usb)
Verifier Carte son chipset graphique
Verifier support usb
Sauvergarder la config
make dep clean bzImage
```

```
cp System.map /boot/System.map-2.4.32
cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.32
make modules
make modules_install
```

Vérifier `/etc/modules.conf` (en particulier modules carte réseau et le bonding + fiber channel a mettre (?) en commentaire).

```
depmod –ae –F /boot/System.map-2.4.32 2.4.32
rm /boot/initrd-2.4.32.img
mkinitrd /boot/initrd-2.4.32.img 2.4.32
```

Réinstaller le chargeur de démarrage.

```
ALT+F2 ->umount /mnt/image/boot
ALT+F1->mount /boot
ALT+F1->grub-install /dev/sda
```

Vérifier les droits sur `/tmp` (`drwxrwxrwt` sinon les positionner avec chmod 777 /tmp)

```
chmod o+t /tmp
```

Vérification du fichier `/etc/fstab` (en particulier la partition de boot). Sauvegarder et vérifier ce que pointe `/dev/cdrom` du fichier (`ls –al /dev/cdrom`). Si on utilise l'emulation scsi-ide il devra etre un lien symbolique vers `/dev/scd0` si on ne l'utilise pas (Fedora Core) un lien vers `/dev/hd(a|b|c|d)` en fonction du controleur.

Vérification du fichier `/etc/grub.conf` en particulier le cdrom avec le module ide-scsi (hdx=ide-scsi) dans la ligne du kernel de la legacy pour une RH9.

Vérfication du fichier `/boot/grub/device.map` (périphérique disque dur a positionner en scsi (ligne (hd0) /dev/sdxy)).

Vérification du fichier `/etc/sysconfig/grub` (idem que précédemment).

Utiliser Kudzu pour le matériel.

```
/etc/init.d/kudzu start
/etc/init.d/kudzu stop
rm /dev/cdrom1
```

enlever la ligne du cdrom1 dans `/etc/fstab`.

```
cd /etc/
grep –R "#serveur_" *
Et remplacer toutes occurrences par les informations demandées
ATTENTION A LA CASSE sur ''/etc/krb5.conf'' et config authentification
radius apache.
```

```
cd /var/www
```

```
grep –R "#serveur_" *
```

Et remplacer l'occurence par l'information demandée.

```
mv ifenslave ifenslave.old
mv ifenslave ifenslave
```

On demonte tout et on redémarre.

```
umount /boot
umount /proc
exit (on sort du chroot)
umount /mnt/image/var
umount /mnt/image/tmp
umount /mnt/image/opt
umount /mnt/image/home
umount/mnt/image/data
umount /mnt/usbstor1
umount /mnt/image
reboot
```

et enlever le cdrom ^^.

Commandes pour intégrer des modules/drivers additionnels lors de la compilation et l'installation du noyau

Méthode Debian et méthode manuelle

Je pars du principe qu'on est dans `/usr/src` et qu'on a déjà récupéré les sources du noyau.

```
tar jxvf linux-source-2.6.26.tar.bz2 #extraire les sources du noyau.
ln -s /usr/src/linux-source-2.6.26/ linux #lier linux à nos sources.
vi /usr/src/linux/Documentation/Changes #vérifier qu'on a tout ce qui faut
pour compiler en vérifiant la section "requirements".
cp -i /boot/config-2.6.24-1-686 ./.config #récupérer si possible une
configuration déjà faite. Sinon, moi j'installe lshw pour connaître le
matériel et je configure le noyau avec l'aide de cet outil.
zcat /proc/config.gz > /usr/src/linux/.config #récupérer configuration
actuel du noyau.
make menuconfig #outils de personnalisation du fichier .config pour
sélectionner les parties qu'on veut intégrer au noyau en dur, en tant que
module, ou pas du tout.

vi Makefile #Mettre une valeur à EXTRAVERSION pour positionner les modules
dans une autre arborescence de dossier que celle en cours.
```

Ensuite on lance la compilation. Le fichier `.config` est utilisé à ce moment. C'est la partie suivante.

Méthode Debian

Compilation avec ajout de module externe à son noyau (module nvidia, module de son...) Récupérer la source du module via apt ou à la main et extraire le contenu du module dans `/usr/src/modules/`.

```
make-kpkg --initrd --revision 1 --append-to-version=-i2ldebian kernel-image
--added_modules=nommodule #compile le noyau avec le module externe et
création du paquet .deb.
```

Enfin :

```
dpkg --install linux-image-2.6.26-i2ldebian_1_i386.deb #installe le noyau et
met grub à jour.
```

Méthode manuelle

Procédure standard de la compilation d'un noyau.

```
make #compilation du noyau.
make modules_install #installation des modules dans /lib/modules/nomkernel.
cp -i arch/x86/boot/bzImage /boot/vmlinuz-2.6.26-i2ldebian #copie du noyau.
cp -i System.map /boot/System.map-2.6.26-i2ldebian #copie du fichier de
mapping.
cp -i .config /boot/config-2.6.26-i2ldebian #copie de la configuration pour
la sauvegarder.
mkinitramfs -o /boot/initrd.img-2.6.26-i2ldebian /lib/modules/2.6.26-
i2ldebian #création du fichier de RAM disque.
vi /boot/grub/menu.lst #modifier le grub en fonction d'un exemple (changer
uniquement les versions noyau en général).
update-grub #mettre à jour grub
```

On ne peut ajouter un module pendant l'installation. C'est seulement après l'avoir fait qu'on gère l'ajout d'un module supplémentaire. C'est la partie suivante.

Intégrer des modules/drivers additionnels mais de manière séparée en supposant le noyau déjà compilé

Méthode Debian et manuelle

Le noyau est déjà compilé (avec les commandes make, make modules_install...) mais on veut ajouter un module qui n'est pas présent (très souvent pour supporter un matériel particulier).

Procédure classique de compilation d'un module (détaillé dans les fichiers README ou INSTALL avec les sources). Récupérer le module du constructeur du module sur son site officiel par exemple. En utilisateur de base :

```
tar xvp archive
./configure --help //voir les options de configurations
./configure --prefix=endroit_ou_installer
make //compiler les sources
su -c root 'make install' //installation
```

Les commandes précédentes vont ajouter le module automatiquement dans l'arborescence des modules du noyau `/lib/modules/2.6.26…../kernel/drivers/categorie_modules….`

Une fois cette opération réalisée, on peut charger le module avec la commande

```
modprobe nommodule.
lsmod #pour vérifier.
```

Si tout fonctionne bien, on peut faire en sorte que le module soit chargé au démarrage en ajoutant le nom du module au fichier `/etc/modprobe.conf` (qui a remplacé le fichier `/etc/modules` mais pas dans toutes les distributions).

Il est intéressant aussi de pouvoir enlever un ancien module qui ne fonctionne pas bien.

```
rmmod nommodule #suppression en live.
blacklist nommodule #à mettre dans /etc/modprobe.d/blacklist pour supprimer
de manière pérenne.
```

De plus en plus, les modules externes sont fournis avec le système DKMS (Dynamic Kernel Module Support) pour la gestion dynamique des modules. Ce système développé par l'équipe Dell permet de conserver un module additionnel d'un noyau dans les autres noyaux. Ainsi, à chaque mise à jour ou compilation d'un nouveau noyau le module additionnel fait partie du nouveau noyau obtenu. On utilise cet outil via le gestionnaire de paquet de la distribution ce qui limite les manipulations manuelles et simplifie l'installation.

Méthode/outils pour ajouter un patch au noyau

Méthode Debian et méthode manuelle

Récupérer le patch. Le décompresser avec tar. Ensuite, patcher un noyau requière de se placer à la racine des sources du noyau (/usr/src/linux) puis d'entrer la commande patch.

```
patch -p1 < /chemin/vers/le/patch
```

La lecture de man patch est recommandée.

Est-il possible d'automatiser de manière périodiques toutes les étapes précédentes (détection de l'apparition d'une

nouvelle version ou d'un nouveau patch du noyau, récupération des sources, configuration (basé sur l'ancienne configuration par exemple), compilation et éventuellement installation ? L'installation ne sera pas automatiquement lancée par soucis de stabilité du système

Méthode Debian et manuelle

Il est bien sûr possible d'automatiser toutes les tâches sans exception notamment à l'aide d'un script bash. Pour une partie des tâches c'est intéressant (récupération automatique des nouvelles sources, position dans le bon dossier, pourquoi pas alerte mail lorsque ceci est fait...) mais l'installation automatique est selon moi très peu intéressante, surtout sur un système en production.

Si des outils libres (scripts, paquets) existent, préciser l'URL permettant d'y accéder et joignez une copie de la page html relative à l'URL à votre compte-rendu

Voici un script shell qui simplifie la compilation kernel. Il utilise un mode graphique d'ailleurs nommé Xdialog. Le script suivant aurait besoin d'une modification pour prendre en compte la récupération des sources de manière automatique et on pourrait supprimer la gestion graphique et l'interactivité mise en place. Ensuite, on peut l'exécuter périodiquement à l'aide d'une tâche cron.

```
#!/bin/sh

# This little script is aimed at easing the Linux kernel compilation and
# installation. No guarantee is given on its suitability to your own system
# though.

function check_error() {
        ret=$?
        if (( $ret != 0 )) ; then
                echo "Error while building the kernel..."
                exit $ret
        fi
}

TITLE="Linux kernel compilation"
MAKE="/tmp/make.$$"
TEMP="/tmp/kernel-compilation.log"

echo >$TEMP
/bin/ln -s /usr/bin/make $MAKE
check_error
```

```
(
/bin/sleep 1

cd /usr/src/linux
check_error

Xdialog --title "$TITLE" --yesno "make mrproper first ?" 0 0
if (( $? == 0 )) ; then
        echo "------------------------------------------------------------
------------"
        echo "Making mrproper..."
        echo "------------------------------------------------------------
------------"
        $MAKE mrproper 2>&1
        check_error
fi

if [ -f /usr/src/linux/.config ] ; then
        Xdialog --title "$TITLE" --yesno "Configure the kernel ?" 0 0
        if (( $? == 0 )) ; then
                echo "------------------------------------------------------
--------------------"
                echo "Making xconfig..."
                echo "------------------------------------------------------
--------------------"
                $MAKE xconfig 2>&1
                check_error
        fi
else
        echo "------------------------------------------------------------
------------"
        echo "Making xconfig..."
        echo "------------------------------------------------------------
------------"
        $MAKE xconfig 2>&1
        check_error
fi
echo "------------------------------------------------------------------
----"
echo "Making depends..."
echo "------------------------------------------------------------------
----"
$MAKE dep 2>&1
check_error
echo "------------------------------------------------------------------
----"
echo "Cleaning up..."
echo "------------------------------------------------------------------
----"
$MAKE clean 2>&1
check_error
```

```
echo "------------------------------------------------------------------------
----"
echo "Making kernel..."
echo "------------------------------------------------------------------------
----"
$MAKE bzImage 2>&1
check_error
echo "------------------------------------------------------------------------
----"
echo "Making modules..."
echo "------------------------------------------------------------------------
----"
$MAKE modules 2>&1
check_error
echo "------------------------------------------------------------------------
----"
echo "Installing modules..."
echo "------------------------------------------------------------------------
----"
$MAKE modules_install 2>&1
check_error

VERSION=`/bin/grep UTS_RELEASE /usr/src/linux/include/linux/version.h |
/bin/awk --source '{ print $3 }'`
echo -n "VERSION=" >/tmp/version.$$
echo $VERSION >>/tmp/version.$$
. /tmp/version.$$
/bin/rm -f /tmp/version.$$
echo "------------------------------------------------------------------------
----"
echo "Installing kernel v$VERSION... "
echo "------------------------------------------------------------------------
----"
/bin/cp -f /usr/src/linux/.config /boot/config-$VERSION 2>&1
/bin/cp -f /usr/src/linux/System.map /boot/System.map-$VERSION 2>&1
/bin/cp -f /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-$VERSION 2>&1
if [ -f /boot/module-info-`uname -r` ] && ! [ -f /boot/module-info-$VERSION
] ; then
        /bin/cp -f /boot/module-info-`uname -r` /boot/module-info-$VERSION
2>&1
fi
Xdialog --title "$TITLE" --yesno "Make the new kernel the current one ?" 0 0
if (( $? == 0 )) ; then
        echo "------------------------------------------------------------------
------------"
        echo "Making v$VERSION the current kernel... "
        echo "------------------------------------------------------------------
------------"
        /bin/ln -sf /boot/System.map-$VERSION /boot/System.map 2>&1
        /bin/ln -sf /boot/vmlinuz-$VERSION /boot/vmlinuz 2>&1
        if [ -f /boot/module-info-$VERSION ] ; then
```

```
                /bin/ln -sf /boot/module-info-$VERSION /boot/module-info
2>&1
        fi
        if [ -f /boot/map ] && [ -f /sbin/lilo ] && [ -f /etc/lilo.conf ] ;
then
                Xdialog --title "$TITLE" --yesno "Install the new kernel
with lilo ?" 0 0
                if (( $? == 0 )) ; then
                        echo "-----------------------------------------------
----------------------------"
                        echo "Running lilo... "
                        echo "-----------------------------------------------
----------------------------"
                        /sbin/lilo 2>&1
                fi
        fi
fi
echo ""
echo "done !"
) | /bin/cat >$TEMP &

PID=$!

Xdialog --title "$TITLE" --no-button --smooth --tailbox $TEMP 40 80
/usr/bin/killall $MAKE 2>/dev/null
kill $PID 2>/dev/null
/bin/rm -f $MAKE
echo >$TEMP
```

Lien d'accès :
http://www.google.com/codesearch?hl=fr&q=kernel+compil+show:L1u61kgVIDo:CC5wg5RO01Q:HQ0-r8RknkI&sa=N&cd=3&ct=rc&cs_p=http://gentoo.osuosl.org/distfiles/Xdialog-2.2.1.tar.bz2&cs_f=Xdialog-2.2.1/samples/kernel

Si on n'utilise pas cette architecture de script, il existe toujours la méthode classique d'un script shell qui utilise le gestionnaire de paquetage (apt-get sous Debian) pour récupérer et installer le dernier noyau automatiquement. Il récupérera tout seul l'ancienne configuration déjà présente.

```
apt-get -y dist-upgrade #mise à jour y compris le kernel (par défaut le
kernel n'est pas pris en compte).
```

C'est certainement la méthode la plus simple même si peut-être pas forcément la plus optimisée...