[Debian 8] Compiler et installer le kernel 4.7

Les « **Kernel** » sont les **fondements** même d'un Linux – Ils sont maintenus par une équipe de développeur et par **Linus Torvalds** lui-même. Dernier en date au 06/08/2016, la version 4.7 est en version finale !
Tous les kernels peuvent être installés sur votre distribution, à condition d'effectuer une sauvegarde complète de votre machine et de compiler les sources !

Les kernel pour Linux peuvent se trouver à l'adresse suivante : https://www.kernel.org/

Connue de tous, Debian n'est pas très friande des tous derniers kernel. Actuellement, un Debian 8.5 a un kernel 3.16.x ! Toutefois étant que ce kernel est « optimisé » pour Debian et dispose de nombreux patchs de sécurité.
Avec cet article, vous serez en mesure de pouvoir compiler le tout dernier kernel en date pour votre Debian !

La procédure ci-dessous est tirée du site « cyberciti.biz », site sur lequel je me suis aidé pour compiler mon kernel. La source est en fin d'article. De plus, je ne me suis pas attardé sur la personnalisation du kernel – le but ici est de pouvoir compiler un kernel « from scratch » à partir de ses sources.

I. Pré-requis

Sauvegarde !

Avant toute chose, **sauvegardez impérativement** vos documents ou créez une image / snapshot de votre système ! Un kernel défectueux **peut casser votre système** et le rendre totalement inopérant. Les réparations peuvent être très lourdes et extrêmement compliquées… Donc sauvegardez vos données !

Espace de stockage

Compiler un kernel demande **BEAUCOUP** de place – assurez-vous d'avoir **au strict minimum 10 Go d'espace disque de libre** ! **Privilégiez** plutôt **15 Go** de libre pour avoir un peu de débattement si vous ajoutez de nombreux patchs à votre kernel.

Installation des outils nécessaires

La compilation d'un kernel se fait sur une machine disposant de certains outils de compilation, d'interpréteurs et autres. Attention, il vous faut **au moins 500 Mo d'espace libre** sur votre système pour contenir tous ces outils et les dépendances.

```
apt install git build-essential libncurses5-dev xz-utils kernel-package
```

L'installation des paquets peut prendre un certain temps, puisqu'il y a de nombreux paquets à télécharger (des outils et utilitaires de compilation, des librairies…) – tout dépend de la puissance

de votre machine et de vos disques durs.
Lorsque tout est installé, il faut préparer « l'emplacement » où va être travaillé le kernel : tous les fichiers seront stockés dans un dossier, y compris le kernel compilé.

Attention, certains paquets inutiles peuvent être téléchargés et installés, notamment les documentations « texlive » – elles prennent une place assez impressionnante (plus de 600mo !!), vous pouvez donc vous en séparer via cette commande :

```
apt remove texlive-latex-base-doc texlive-latex-recommended-doc texlive-latex-extra-doc texlive-pictures-doc texlive-pstricks-doc
```

Préparation du dossier de travail

Je ne me suis pas embêté pour cette partie, je suis connecté en root et j'ai utilisé mon dossier de sauvegarde dans lequel j'ai créé un sous-dossier spécifique. Libre à vous de placer où vous le souhaitez ce fameux dossier de travail.

```
mkdir /kernel && cd /kernel
```

A l'intérieur, j'ai téléchargé les sources publiques du kernel 4.7 :

```
wget http://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.7.tar.xz
```

Et lorsque l'archive est téléchargée, il faut la décompresser :

```
tar xvf linux-4.7.tar.xz
```

Un dossier va être créé, au nom du kernel – dans ce cas présent (en version 4.7), le dossier s'intitulera « *linux-4.7* »

```
cd linux-4.7
```

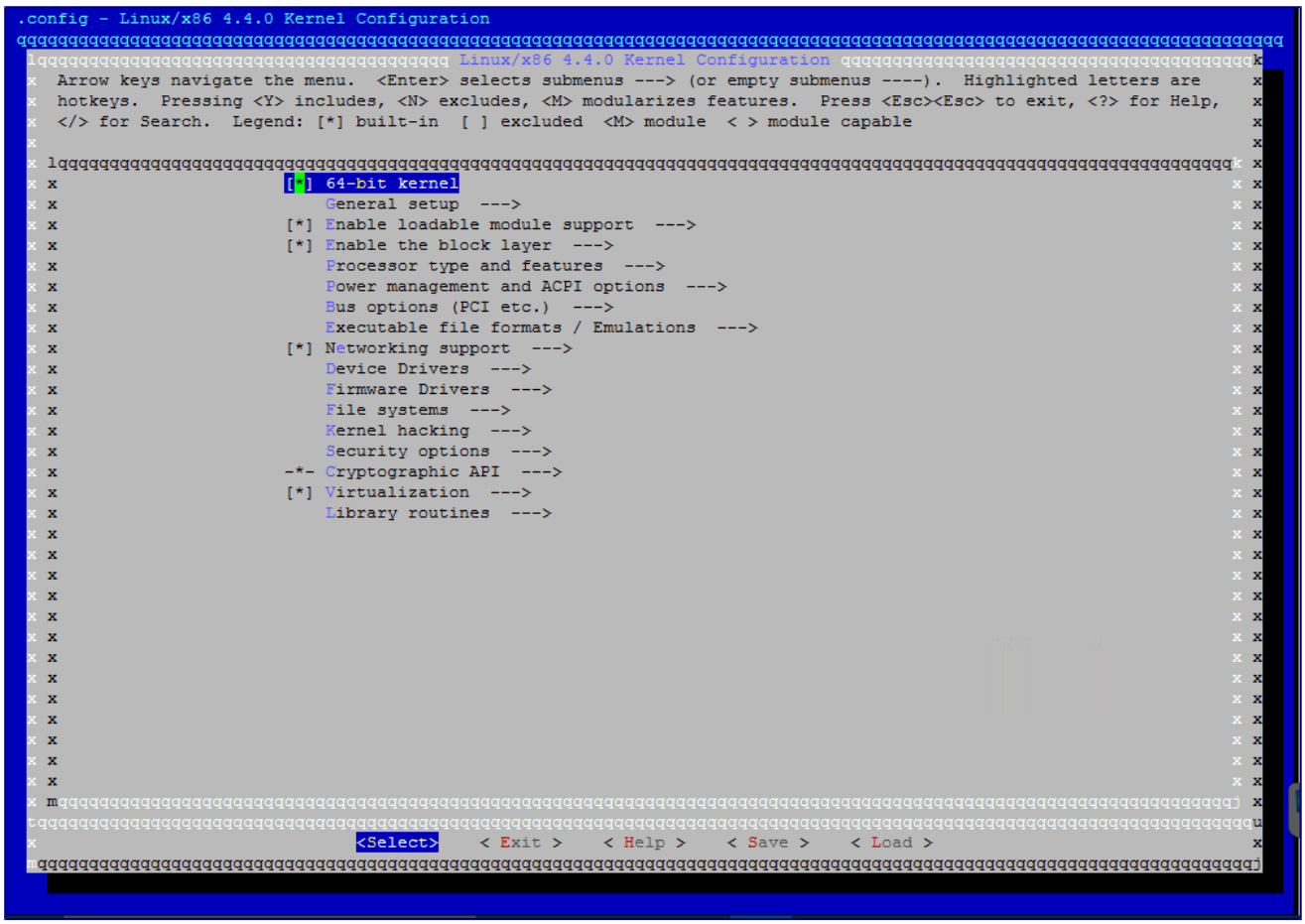
Préparation de la compilation

Puisque les sources sont extraites et prêtes à être compilées, une étape préliminaire est nécessaire – il faut copier l'actuel fichier de configuration du kernel pour bénéficier d'une base « saine ».

```
cp /boot/config-$(uname -r) .config
```

Et ainsi, le (futur) kernel de votre système est prêt à recevoir ses tweaks / patchs et autres routines ! Pour ce faire, vous devez lancer la commande :

```
make menuconfig
```



Je ne détaillerai pas cette partie – je ne la connais pas du tout, je n'ai pas eu le temps de la travailler. J'ai donc laisser tous les paramètres par défaut de mon ancien « .config » du kernel Debian de base. Faites très attention lors de l'utilisation de ce menu ! Un faux-pas et vous pouvez rendre votre kernel totalement inutilisable et donc votre machine aussi !

Même si vous n'avez aucune option à modifier, vous devez quand même utiliser le « *make menuconfig* » pour créer le fichier « .config ». Sélectionnez donc « < **Save** >« , validez les modifications et enfin sélectionnez « < **Exit** >« .

Lancement de la compilation

Le moment est arrivé ! La compilation va bientôt pouvoir se lancer ! Une dernière commande est à effectuer pour remettre de l'ordre dans les fichiers et retrouver une arborescence « propre ».
Cette opération peut durer quelques minutes.

```
make-kpkg clean
```

```
root@czsweb01:/backupczs/kernel/linux-4.4# make-kpkg clean
exec make kpkg_version=13.014+nmu1 -f /usr/share/kernel-package/ruleset/minimal.mk clean
====== making target minimal_clean [new prereqs: ]======
This is kernel package version 13.014+nmu1.
test ! -f .config || cp -pf .config config.precious
test ! -e stamp-building || rm -f stamp-building
test ! -f Makefile || \
            make    ARCH=x86_64 distclean
make[1]: Entering directory '/backupczs/kernel/linux-4.4'
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config include/generated
  CLEAN   .config .config.old
make[1]: Leaving directory '/backupczs/kernel/linux-4.4'
test ! -f config.precious || mv -f config.precious .config
rm -f modules/modversions.h modules/ksyms.ver scripts/cramfs/cramfsck scripts/cramfs/mkcramfs
root@czsweb01:/backupczs/kernel/linux-4.4#
```

Enfin, la **compilation est maintenant** possible :

```
make-kpkg --initrd --revision=1.CZS kernel_image kernel_headers --jobs x
```

- – –**initrd** : essentiel pour avoir un pseudo-OS au démarrage de votre machine – c'est grâce au initrd que vous voyez apparaître toutes les inscriptions au démarrage de votre machine Linux et par la suite (indirectement) faire booter votre système.
- – –**revision=1.CZS** : Numéro de la révision du kernel – à changer selon vos besoins et vos envies
- **kernel_image kernel_headers** : la compilation produira les fichiers « kernel_image » et « kernel_headers », nécessaire pour créer votre kernel « bootable » pour le système.
- **– -jobs x** : cette variable permet d'optimiser l'utilisation des ressources de votre machine pour la compilation. « x » correspond au nombre suivant : il s'agit du nombre de CPU de votre machine +1.
  Exemple : Si votre VM a deux CPU virtuels, « – -jobs x » devra donc être « – -jobs 3 ». *Merci à Geoffrey pour son commentaire et cette information !*

```
root@czsweb01:/backupczs/kernel/linux-4.4# fakeroot make-kpkg --initrd --revision=1.CZS kernel_image kernel_headers
exec make kpkg_version=13.014+nmu1 -f /usr/share/kernel-package/ruleset/minimal.mk debian DEBIAN_REVISION=1.CZS  INITRD=YES
====== making target debian/stamp/conf/minimal_debian [new prereqs: ]======
This is kernel package version 13.014+nmu1.
test -d debian            || mkdir debian
test ! -e stamp-building || rm -f stamp-building
install -p -m 755 /usr/share/kernel-package/rules debian/rules
for file in ChangeLog  Control  Control.bin86 config templates.in rules; do                                    \
          cp -f  /usr/share/kernel-package/$file ./debian/;                                 \
       done
cp: cannot stat '/usr/share/kernel-package/ChangeLog': No such file or directory
for dir  in Config docs examples ruleset scripts pkg po;  do                                        \
         cp -af /usr/share/kernel-package/$dir  ./debian/;                                 \
       done
test -f debian/control || sed        -e 's/=V/4.4.0/g'  \
               -e 's/=D/1.CZS/g'         -e 's/=A/amd64/g'  \
       -e 's/=SA//g'  \
       -e 's/=I//g'                                \
       -e 's/=CV/4.4/g'                            \
       -e 's/=M/Unknown Kernel Package Maintainer <unknown@unconfigured.in.etc.kernel-pkg.conf>/g'                    \
       -e 's/=ST/linux/g'      -e 's/=B/x86_64/g'    \
               -e 's/=R//g'    /usr/share/kernel-package/Control > debian/control
test -f debian/changelog ||  sed -e 's/=V/4.4.0/g'       \
          -e 's/=D/1.CZS/g'        -e 's/=A/amd64/g'       \
          -e 's/=ST/linux/g'     -e 's/=B/x86_64/g'         \
          -e 's/=M/Unknown Kernel Package Maintainer <unknown@unconfigured.in.etc.kernel-pkg.conf>/g'
     \
           /usr/share/kernel-package/changelog > debian/changelog
chmod 0644 debian/control debian/changelog
test -d ./debian/stamp || mkdir debian/stamp
make -f debian/rules debian/stamp/conf/kernel-conf
make[1]: Entering directory '/backupczs/kernel/linux-4.4'
```

Attention ! **La compilation d'un kernel prend un vrai temps** !! Vous pouvez facilement passer plusieurs heures à compiler votre noyau !
Pour ma part, j'ai mis plus de 34 minutes (machine virtuelle, 1vCPU, 2Go de RAM, disque au format RAW sur disques SSD) – encore une fois, tout dépend de la puissance de votre machine.

II. Installation du kernel

Après la compilation, votre nouveau **noyau** est **prêt à l'emploi** !! Vous pouvez dès à présent l'installer sur votre machine pour pouvoir l'exploiter.
Il faut installer les deux paquets « .deb » que vous venez de compiler :

```
dpkg -i linux_headers-4.7_amd64.deb && dpkg -i linux_image-4.7_amd64.deb
```

Attention, le nom des fichiers peut changer en fonction de ce que vous avez entrer dans la commande de compilation, notamment quant à l'attribut « – -revision ».
L'installation se fait rapidement pour les deux paquets – peu après l'installation, vous n'êtes pas obligé de redémarrer votre machine ! En effet, depuis les kernel 4.x, vous pouvez « redémarrer à chaud » votre machine et donc prendre en compte les modifications sans devoir redémarrer le serveur.

Pour se faire, il faut utiliser le paquet « *kexec-utils*« .

```
apt install kexec-tools && systemctl kexec
```

Vos sessions SSH vont alors se couper, puisque le serveur va redémarrer les services sans redémarrer physiquement.
Source

Toutefois, vous n'êtes pas obligé d'utiliser ce paquet si vous préférez l'ancienne méthode, à savoir le redémarrage « normal » via un « reboot ».

Au démarrage du serveur, lorsque vous arriverez sur le grub, dirigez-vous dans « *Options avancées* » (*Advanced settings*), vous devriez avoir une liste avec 4 options : deux lignes pour le kernel 4.7 et les deux autres lignes sont pour votre autre kernel.

J'ai sélectionné le kernel 4.7 – la machine a booté puis immédiatement, j'ai effectué la commande ci-dessous :

```
uname -a
```

```
root@kernelczs:~# uname -a
Linux kernelczs 4.6.0 #1 SMP Mon May 16 13:41:33 CEST 2016 x86_64 GNU/Linux
```

Comme le dirait Apache, « It Works ! » 🙂