Compiler un kernel 5

Pour ceux qui veulent bénéficier d'un kernel 5 sans passer passer par un utilitaire graphique, un ppa, un kernel d'une autre distribution, etc afin que celui-ci sont réellement adapté à leur machine.

How-to proposé pour le kernel 5.1.9, similaire pour les autres kernels. Testé sur une MINT 18.3 à jour (aussi une DEBIAN SID), pas de raison que ce soit différent pour une MINT 19. Section shell, ligne de commandes

La machine de test

DELL LATITUDE E6410, Mint 18.3

```
libres09@libres09-Latitude-E6410 ~ $ hostnamectl
..
  Operating System: Linux Mint 18.3
            Kernel: Linux 4.10.0-38-generic
      Architecture: x86-64
```

```
libres09@libres09-Latitude-E6410 ~ $ uname -a
Linux libres09-Latitude-E6410 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

Préparer la machine

Installer les paquets nécessaires à la compilation

```
sudo apt-get install libncurses-dev libssl-dev  libelf-dev
```

Récupérer le dernier kernel

Récupérer l'archive contenant les sources du dernier kernel.

Ces sources sont disponibles ici. http://cdn.kernel.org/pub/linux/kernel/v5.x/

On travaillera dans le répertoire **kernel-5**.

```
mkdir kernel5
cd kernel5
wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.1.9.tar.gz
```

Récupérer aussi sa signature (https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.1.9.tar.sign) permettra de tester l'intégrité de l'archive.

Décompresser l'archive

```
tar xvf linux-5.1.9.tar.gz
cd linux-5.1.9/
```

Copier le fichier de configuration d'un kernel 4 générique

Le fichier .config

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

Pour ceux qui veulent modifier des options de compilations : voir section plus bas.

```
make menuconfig
```

On s'appuie sur le fichier de configuration d'un kernel 4 générique dans un premier temps pour ne pas avoir à remplir par nous mème toutes les options de la compilation. Ce kernel ne sera pas le plus optimisé mais c'est le moyen le plus sûr pour que ça marche du premier coup.

Lancer la compilation

Ce processus, en fonction des machines, prendra un certain temps. Autant utiliser le nombre maximal de processeurs que l'on dispose.

De combien de processeurs mon PC dispose-t-il ?

```
echo $(nproc)
4
```

Donc on peut lancer la compilation en les utilisant tous (choisir l'une des 2 lignes, elles font la même chose)

```
 make -j $(nproc)
 make -j 4
```

Pour n'utiliser que 2 processeurs sur les 4, par exemple

```
 make -j 2
```

La compilation d'un noyau est un processus relativement long. Sur ce i5, 4 cœurs, elle prend plus de 2 heures, donc prévoyez de l'occupation. Elle peut tourner en arrière plan sans problème.
Généralement, pas de soucis particulier, au pire lire l'erreur, il s'agit souvent d'un paquet de développement à installer (dont le nom est stipulé)...

Installation des modules de ce nouveau kernel

Une fois la compilation terminée, il faut installer tous les modules de ce noyau

```
sudo make modules_install
```

Les modules sont installés ici

```
ls /lib/modules/5.1.9/
build              modules.builtin      modules.devname  modules.symbols.bin
kernel             modules.builtin.bin  modules.order    source
modules.alias      modules.dep          modules.softdep
modules.alias.bin  modules.dep.bin      modules.symbols
```

Installation du kernel

On peut enfin installer le dernier kernel

```
sudo make install
```

On peut vérifier son installation

```
ls /boot
config-4.19.0-5-amd64      initrd.img-4.9.0-9-amd64   vmlinuz-4.19.0-5-amd64
config-4.9.0-9-amd64       initrd.img-5.1.9           vmlinuz-4.9.0-9-amd64
config-5.1.9               System.map-4.19.0-5-amd64  vmlinuz-5.1.9
config-5.1.9.old           System.map-4.9.0-9-amd64   vmlinuz-5.1.9.old
grub                       System.map-5.1.9
initrd.img-4.19.0-5-amd64  System.map-5.1.9.old
```

Les opérations suivantes sont obsolètes : sudo update-initramfs -c -k 5.0.0 && sudo update-grub . Donc on laisse tomber...

Prise en charge du kernel 5

Normalement il sera configuré comme kernel par défaut dès le redémarrage de la machine. Donc on reboote

```
sudo /sbin/reboot
```

Et ça passe nickel:

```
libres09@libres09-Latitude-E6410 ~/kernel5/linux-5.1.9 $ hostnamectl
...
  Operating System: Linux Mint 18.3
```

```
           Kernel: Linux 5.1.9
     Architecture: x86-64
```

```
libres09@libres09-Latitude-E6410 ~/kernel5/linux-5.1.9 $ uname -a
Linux libres09-Latitude-E6410 5.1.9 #1 SMP Tue Jun 18 10:36:54 CEST 2019
x86_64 x86_64 x86_64 GNU/Linux
```

Personnaliser la compilation du kernel

Avant la compilation, on peut bien entendu personnaliser les options du fichier de configuration afin d'optimiser ce kernel.

Pour cela 3 utilitaires

- make menuconfig : pas besoin de serveur X, nickel pour du ssh par exemple, du tty
- make xconfig : le même mais avec interface graphique de Qt (nécessite un serveur X)
- make gconfig : le même mais avec interface graphique de GTK+ 2.0 (nécessite un serveur X)

Sur mon PC, par exemple, si je regarde quels drivers sont utiles pour ma carte graphique

```
inxi -G
Graphics:
  Device-1: Intel Haswell-ULT Integrated Graphics driver: i915 v: kernel
  Display: x11 server: X.Org 1.20.4 driver: modesetting unloaded: fbdev,vesa
  resolution: 1366x768~60Hz
  OpenGL: renderer: Mesa DRI Intel Haswell Mobile v: 4.5 Mesa 18.3.6
```

Image

Je remarque que le module nécessaire est le **i915**, je n'ai pas besoin des Nvidia, des AMD, il me suffit de ne pas les compiler pour gagner de la place... Il suffit de naviguer dans le menu de configuration du kernel est de désactiver tout ce qui n'est pas nécessaire, aussi bien les drivers inclus au kernel que les modules.

Idem, je peux enlever tous les drivers de la section Net, sauf celui de ma puce WIFI et carte Ethernet... Idem pour le son.... C'est ainsi qu'on optimise...

Marqués par une astérix (*) : le driver sera inclus au kernel lui-même. Par besoin de le charger (modprobe). Si marqué par un **M**, le driver sera compilé, mais ne sera pas inclus au kernel. Il faudra le charger pour le prendre en compte. On parle de module.

Vu la taille des disques actuels, si vous n'êtes pas sûr de vous, ne touchez pas aux options de configuration, celles issues de la copie du fichier .config d'un kernel existant et générique feront très bien l'affaire. Plus intéressant, sur des machines dont la place est restreinte, ou si la sécurité est un critère important, ou pour les systèmes embarqués.

Faites attention, si vous supprimez un module important, exemple le support de l'ext4, vous avez toutes les chances ne pas pouvoir démarrer le système.

Mon système plante ! Je suis perdu ! Pas de soucis, rebooter ; depuis le grub sur le kernel précédent

et revoyez vous configuration de compilation.

Erreur de keyring à la config

```
CC      certs/system_keyring.o
make[2]: *** No rule to make target 'debian/certs/benh@debian.org.cert.pem',
needed by 'certs/x509_certificate_list'.  Stop.
Makefile:951: recipe for target 'certs' failed
make[1]: *** [certs] Error 2
```

si vous rencontrez cette erreur. Vous pouvez corriger rapidement le fichier de configuration via cette commande sed

```
sed -ri '/CONFIG_SYSTEM_TRUSTED_KEYS/s/=.+/=""/g' .config
```

(j'ai rencontré cette erreur sous DEBIAN, pas sous MINT)

Conclusion

Donc voilà. Donc vous voyez, plutôt que de piocher des kernels à droite à gauche, sans savoir ce qu'ils ont dans le ventre, il n'est pas compliqué de compiler le sien.

Approfondir Changelog-5.0 : Voici les nouveautés / changements du noyau 5 : https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/ChangeLog-5.0

The Linux Kernel

Résumé

J'avais dit quelques lignes, donc je respecte mon engagement

```
wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.1.9.tar.xz
tar xvf linux-5.1.9.tar.xz
cd linux-5.1.9/
cp -v /boot/config-$(uname -r) .config
make menuconfig
make -j $(nproc)
sudo make modules_install
sudo make install
sudo reboot
```