Interventions sur le noyau

Comme le rappelle souvent son créateur Linus Torvalds, le champ de définition de Linux se circonscrit au noyau (`kernel`) et non comme malheureusement l'usage populaire le fait, à une distribution. Maintenu par une communauté de développeurs (et pour "patron" incontesté, son créateur) en licence GNU, le noyau gère les ressources physiques, puis logiques d'un système informatique. Le système Linux Ubuntu est conforme à la norme POSIX, cette API (bibliothèque logicielle ou interface) régissant les demandes de services des processus au noyau.

1. Principes

Le noyau Linux est dit modulaire car il gère de façon dynamique les pilotes de périphériques en fonction des besoins, à l'inverse du noyau de type BSD (FreeBSD, NetBSD et OpenBSD).

Quelques informations complémentaires :

- le site WEB officiel du noyau : http://www.kernel.org
- la version initiale du noyau Ubuntu Lucid Lynx : 2.6.32-22

Le numéro tient compte d'une version majeure (2.6) et d'une révision concernant la correction d'erreurs ou l'ajout de fonctionnalités (32). Le dernier chiffre indique une variation dans la configuration ; vous pouvez parfaitement définir votre propre variation en cas de compilation différente du noyau.

a. Les méthodes

Une modification du noyau n'est pas une opération anodine, elle relève plutôt de la tâche complexe et exceptionnelle. En effet, rares sont les besoins réels d'une modification. On peut cependant classifier trois motivations :

- Le remplacement du noyau pour une version supérieure.
- L'ajout d'une fonctionnalité ou d'un composant.
- Le retrait d'éléments dans le but d'alléger le noyau.

Cette proposition de travail correspond dans ce chapitre à une méthodologie d'enchaînement logique :

- On se base tout d'abord sur un remplacement d'un noyau pour une version plus récente dans le cadre d'une politique de mise à jour.
- On ajoute ou on supprime une fonctionnalité sur ce nouveau noyau afin d'adapter celui-ci à une situation particulière.
- On améliore le noyau afin d'obtenir une version optimisée pour disposer d'un système plus rapide.

Le dernier traitement fait état d'une modification non pas au niveau du noyau `parinitrd` afin d'accélérer le démarrage du système en supprimant les modules inutiles.

b. Préparation de l'environnement

La compilation du noyau nécessite l'installation de paquets spécifiques, à l'instar d'un jardinier préparant son terrain avant toute plantation. La compilation en tant que telle ne présente pas de difficultés. Les raisons des trop fréquents échecs tiennent pour la plupart d'un manque de préparation avant cette compilation.

Les outils de compilation

Ubuntu utilise un paquetage logiciel déjà vu `build-essential`, liste des paquetages considérés comme indispensable pour compiler des logiciels. On peut lui adjoindre le paquet `bin86`, chargé sous Linux de créer le secteur de boot 16 bits et les configurations binaires.

```
aptitude install build-essential bin86
```

Comme toujours, la mise à jour propre du système s'impose :

```
aptitude update
```

```
aptitude safe-upgrade
```

```
aptitude clean
```

Le processus de compilation du noyau possède des outils qui lui sont propres, auquel on ajoute un paquet logiciel particulier permettant d'éviter d'être le **root** lors de la construction du paquet. Plus exactement, on accorde les droits administrateurs au processus de construction du paquet.

```
aptitude install kernel-package fakeroot
```

Toutes les options de compilation sont regroupées dans un fichier de configuration qu'il est plus simple de gérer via une interface :

- de type texte : avec une interface `ncurses`
- de type graphique : avec une interface GTK+ (Gnome) ou QT (KDE)

Dans la situation d'un serveur, on passe par l'interface texte et `ncurses` :

```
aptitude install libncurses5-dev
```

2. Changer le noyau par Aptitude

Le changement de noyau par **Aptitude** et les dépôts Ubuntu s'effectuent de façon triviale lorsqu'il en existe un nouveau : il suffit de lancer la commande `aptitude` avec l'option `full-upgrade`. Elle reste la meilleure méthode et la plus simple d'effectuer un changement sur le noyau, ce qui n'empêche pas un minimum de travail et de méthode.

Vérifiez la version de votre noyau (version courte) :

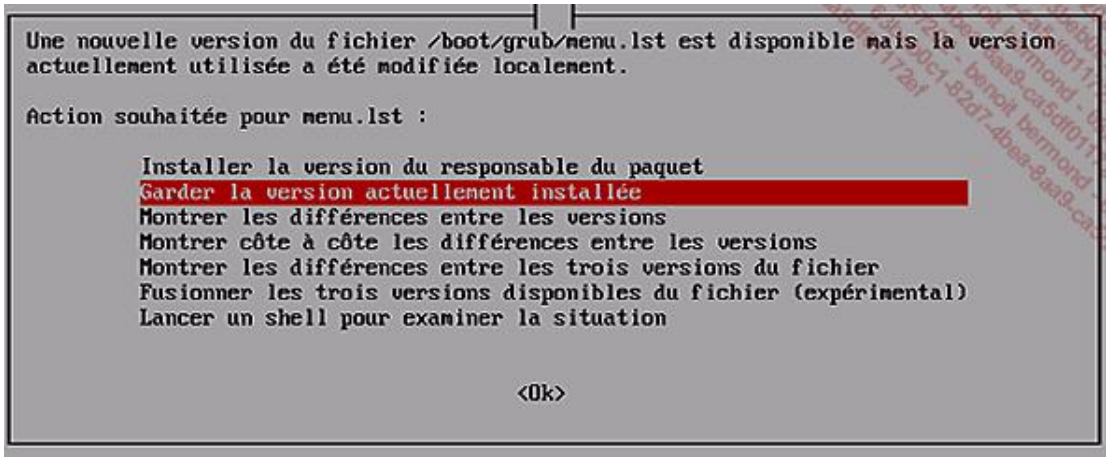
```
uname -r
```

Rappel : l'option `safe-upgrade` ne met à jour que les paquets installés. L'ambiguïté réside dans le numéro de paquet. Si l'on prend exemple sur une distribution serveur initiale avec le noyau `2.6.32-22`, le système considère que ce dernier comme installé alors que le noyau `2.6.32-22-server` (notre noyau de mise à jour au moment de la rédaction de l'ouvrage, il peut changer bien sûr) ne l'est pas. Un `safe-upgrade` ne changera donc pas la version du noyau alors qu'un `full-upgrade` le fera :

```
aptitude full-upgrade
```

a. Modification pour GRUB

Lors de l'installation, une demande concerne le fichier `/boot/grub/grub.cfg` : elle consiste à offrir une nouvelle ligne de menu pour GRUB au démarrage du système.



- Choisissez non pas de garder la version installée (cela ne changerait rien) mais **Installer la version du responsable du paquet**.

  Ce choix lance l'utilitaire `update-grub` qui remplit à la suite du fichier initial les éléments de démarrage du nouveau noyau.

- Redémarrez votre système et choisissez la nouvelle configuration.

b. Nettoyage

Pour ne pas encombrer le système, un nettoyage des précédents paquets s'impose, l'ensemble totalisant près de 90 Mo :

```
aptitude purge linux-image-2.6.32-21-server
```

```
aptitude purge linux-ubuntu-modules-2.6.32-21-server
```

```
aptitude purge linux-restricted-modules-2.6.32-21-server
```

La dernière commande est à omettre si vous n'aviez pas installé de pilotes propriétaires en accès réservé. Si vous aviez installé les en-têtes du noyau :

```
aptitude purge linux-headers-2.6.32-21-server
```

Dans le même ordre d'idée, si vous en avez besoin pour la nouvelle version du noyau :

```
aptitude install linux-headers-`uname -r`
```

L'utilisation de la commande `uname` permet de s'affranchir de la version installée et apporte la certitude d'installer la bonne version.

Attention : les caractères encadrant la commande ne sont pas des apostrophes mais des apostrophes inversées (touches [Alt Gr] **7**, suivies d'un espace pour la matérialisation du caractère).

L'ancien répertoire des modules peut ne pas être vide et donc avoir échappé au nettoyage :

```
rm -Rf /lib/modules/2.6.32-21-server/
```

3. Construction d'un autre noyau

a. Charger les sources

Pour Ubuntu comme pour les autres distributions Linux, la véritable source (sans jeu de mots) et la plus récente se trouve sur le site kernel.org. Malgré son intérêt indéniable, cette méthode est à proscrire car ne comportant pas les réglages nécessaires spécifiques à la distribution Ubuntu. On lui préfère le chargement par les dépôts Ubuntu.

Honnêtement, très rares sont les cas où une demande ne saurait être satisfaite par l'installation des dernières sources stables. Par contre, là encore, deux possibilités existent :

- Soit classiquement par `aptitude install linux-source`,
- Soit par l'utilitaire GIT, système de contrôle de version chargé de gérer le code source par les développeurs du noyau Ubuntu.

Le premier ne donnera au mieux que le code source du noyau en cours : autant utiliser les sources sur kernel.ubuntu.com. Le paquetage git-core étant installé, la première étape charge une copie locale des sources du noyau à partir du dépôt :

```
git-clone git://kernel.ubuntu.com/ubuntu/ubuntu-lucid.git ubuntu-lucid
```

La totalité prend environ 190 Mo, ce qui peut nécessiter un peu de temps... :

```
root@duncan:/home/max# git clone
git://kernel.ubuntu.com/ubuntu/ubuntu-lucid.git ubuntu-lucid
Initialized empty Git repository in /home/max/ubuntu-lucid/.git/
remote: Counting objects: 1537902, done.
remote: Compressing objects: 100% (271510/271510), done.
remote: Total 1537902 (delta 1273707), reused 1517992 (delta 1254607)
Receiving objects: 100% (1537902/1537902), 443.66 MiB | 246 KiB/s, done.
Resolving deltas: 100% (1273707/1273707), done.
Checking out files: 100% (31201/31201), done
```

Plus tard, pour garder l'arborescence à jour vous devrez faire :

```
cd /root/ubuntu-lucid
```

```
git pull
```

Linus Torvalds conseille de prendre un utilisateur différent du root et de compiler le noyau en dehors du répertoire des sources, ce qui n'est pas le cas ici.

Il ne reste plus qu'à mettre en place l'arborescence dans le répertoire des sources (attention : bien mettre le chemin absolu) :

```
ln -s /root/ubuntu-lucid/ /usr/src/linux
```

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

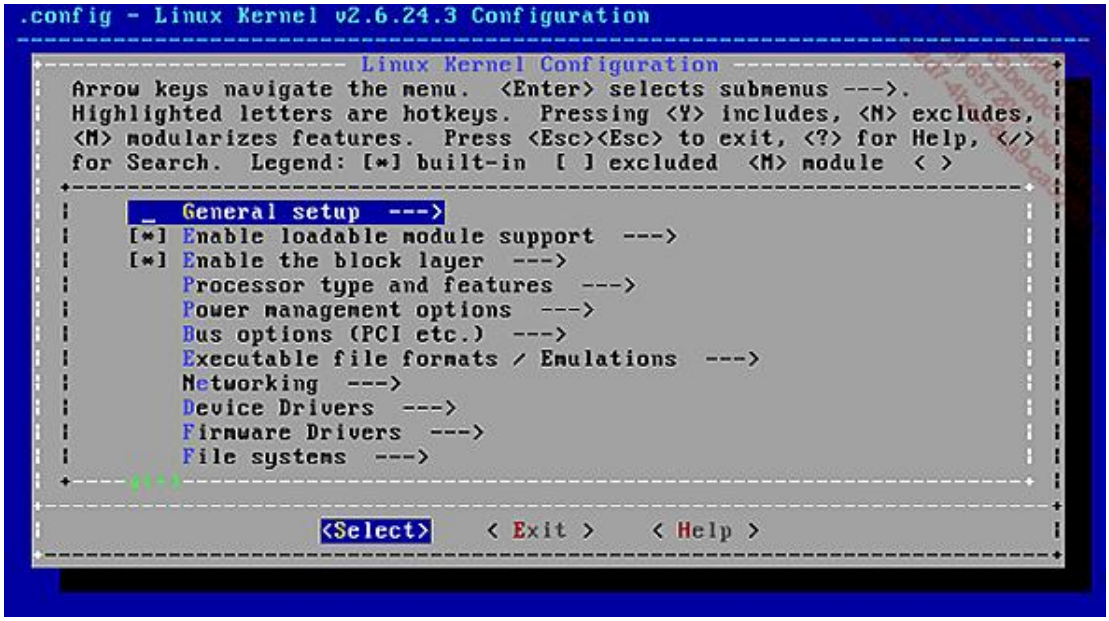
b. Compiler le nouveau noyau

L'idée de base réside dans la construction d'un noyau sous la forme d'un paquetage logiciel, facile à installer et... à désinstaller.

Dans le cas de l'installation des sources par `aptitude install linux-source`, vous auriez pu prendre le fichier de configuration `config-*` se trouvant dans `/boot` puisque, vraisemblablement, la version est identique. Dans l'exemple, la version installée est la version 2.6.32-21 alors que celle par git porte le numéro 2.6.32-22.

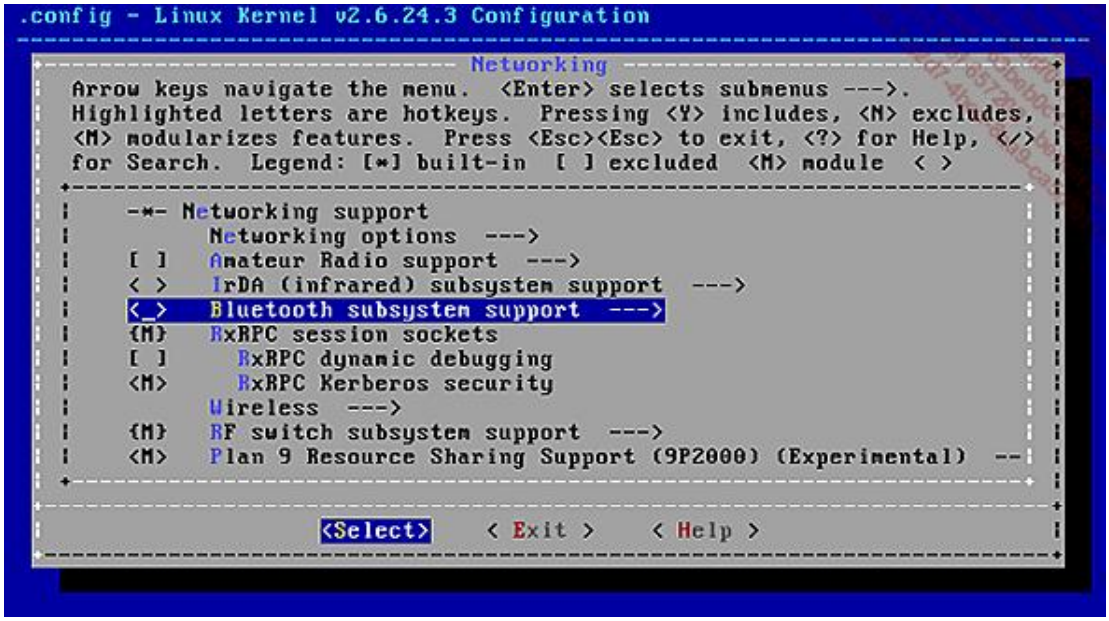
Une fois cette manipulation effectuée, vous lancez le menu de configuration. Dans ce menu, une étoile indique une option activée, un M un chargement par module (chargé en fonction du besoin) :

```
make menuconfig
```



Exemple :

Dans le cas d'un serveur, il n'est pas nécessaire de garder le support pour la radio amateur, les systèmes utilisant l'infrarouge et le bluetooth dans le menu **Networking** :



La sortie après modifications propose automatiquement la sauvegarde de la configuration et écrit le fichier `.config` dans `/usr/src/linux`.

Ce fichier de configuration fait état d'une configuration du noyau ayant trait à la virtualisation par XEN. Comme vous ne disposez pas des sources de celui-ci, vous devez en effacer la référence dans le fichier de configuration. On peut passer par le menu (voir ce qui suit), mais vous pouvez (autre méthode) éditer directement le fichier :

- Éditez le fichier `/usr/src/linux/.config` par VIM.
- Effectuez une recherche sur XEN par les touches [Echap] `/XEN`.
- Supprimez la ligne commençant par `CONFIG_XEN=y`.

Maintenant, vous pouvez lancer l'utilitaire de construction de paquets Debian du noyau à partir des sources. D'abord avec l'option `clean` pour vérification :

```
make-kpkg clean
```

Puis pour la construction des paquets pour le noyau (sur une seule ligne et cela va être un peu long...) :

```
fakeroot make-kpkg --initrd
 --append-to-version=perso32 kernel_image
 kernel_headers modules_image
```

La commande `make-kpkg` regroupe plusieurs commandes : `make dep`, `make bzimage`, `make modules`. À la fin, le résultat se trouve sous la forme de deux paquets que vous installerez par la commande `dpkg -i` :

```
linux-headers-2.6.32.22+drm33.2perso32
linux-image-2.6.32.22+drm33.2perso32
```

Depuis la version Hardy Heron, une nouvelle façon de compiler les sources utilise la commande `AUTOBUILD=1 fakeroot debian/rules binary-debs`. Par expérience, réservez cette méthode si vous installez les sources par `apt-get build-dep linux-image-`uname -r`` et `apt-get source linux-image-`uname -r`` au lieu de `git`. Vous trouverez plus de renseignements sur le WIKI Ubuntu : https://help.ubuntu.com/community/Kernel/Compile

4. Accélérer le démarrage du système

a. Principe

Ubuntu utilise le mécanisme `initrd` (voir le chapitre Préalable à l'installation) qui charge un système de fichiers et le monte en mémoire pour détecter et charger les pilotes nécessaires au système de fichiers sur le disque dur. Le fichier au format compressé contenant ce FS de départ se trouve dans le répertoire `/boot` et a pour nom `initrd.img-(numéro du noyau)`.

Au fur et à mesure des versions Ubuntu, cette image `initrd` a tendance à prendre du poids pour la recherche d'une meilleure compatibilité matérielle. Sur votre système, le chargement de la distribution comporte des pilotes a fortiori connus de vous, on peut donc éliminer les pilotes inutiles. Le but est d'accélérer le chargement du système sachant que l'on ne touche pas aux sources du noyau à la différence de la précédente méthode.

b. Méthodologie de réalisation

La démarche consiste à récupérer le système de fichiers, le décompresser, enlever les éléments inutiles, le compresser à nouveau et le faire prendre en compte par le système. Je prends l'exemple d'une distribution en poste de travail, avec un noyau `2.6.32-22-generic` (en `root` bien sûr).

- Copiez le fichier `initrd` dans un répertoire particulier :

```
mkdir /root/initrd

cd /root/initrd

cp /boot/initrd.img-2.6.32-generic .
```

- Déterminez le type exact du fichier (surtout sa méthode de compression) :

```
file initrd.img-2.6.32-22-generic
```

Le retour de la commande annonce normalement un fichier compressé au format `gzip`. La commande effectuant la décompression est un peu compliquée :

```
gunzip < initrd.img-2.6.32-22-generic | cpio -i -make-directories
```

Explications :

`gunzip` : commande de restauration.

`< initrd.img-2.6.32-22-generic` : redirection de l'entrée standard pour le fichier `initrd` "alimente" la commande

`gunzip`.

`| cpio -i -make-directories` : tube, c'est-à-dire envoi du résultat de la sortie de la commande `gunzip` en entrée de la commande `cpio`.

La commande `cpio` extrait les éléments de l'archive en créant les répertoires si besoin. On se passe maintenant du fichier initial :

```
rm initrd.img-2.6.32-22-generic
```

Le répertoire contient les éléments suivants :

`Initramfs` possède un fichier de configuration situé dans `conf/initramfs.conf` :

```
#
# initramfs.conf
# Configuration file for mkinitramfs(8). See initramfs.conf(5).
#
#
# MODULES: [ most | netboot | dep | list ]
#
# most - Add all framebuffer, acpi, filesystem, and harddrive drivers.
#
# dep - Try and guess which modules to load.
#
# netboot - Add the base modules, network modules, but skip block devices.
#
# list - Only include modules from the 'additional modules' list
#
MODULES=most

# BUSYBOX: [ y | n ]
#
# Use busybox if available.
#
BUSYBOX=y

#
# NFS Section of the config.
#

#
# BOOT: [ local | nfs ]
#
# local - Boot off of local media (harddrive, USB stick).
#
# nfs - Boot using an NFS drive as the root of the drive.
#
BOOT=local

#
# DEVICE: ...
#
# Specify the network interface, like eth0
#
DEVICE=eth0

#
# NFSROOT: [ auto | HOST:MOUNT ]
#
NFSROOT=auto
```

On y retrouve l'utilisation de la `busybox` comme alternative à un problème de chargement et le démarrage par NFS. La section qui nous intéresse est la première avec les choix pour les modules :

- `most` : charge tous les pilotes.
- `dep` : détecte les pilotes nécessaires à charger.
- `Netboot` : utilise les modules liés au réseau.
- `List` : ne prend que les modules d'une liste additionnelle.

Le deuxième choix suffit largement :

```
MODULES=dep
```

La suite se passe dans le répertoire `lib/modules/2.6.24-19-generic/kernel`. Des pilotes sont inutiles s'ils sont absents de votre système.

Dans `fs`, on peut supprimer les systèmes de fichiers non utilisés :

- `jfs`
- `reiserfs`
- `xfs`

Dans drivers, les pilotes non utilisés :

- `ieee1394` (suivant équipement)
- `parport` (les imprimantes ne sont plus sur port parallèle)
- `virtio` (pour la virtualisation)

Une fois ces modifications (ou d'autres) effectuées, il faut refaire l'opération inverse, c'est-à-dire remettre le système de fichiers dans une archive :

```
cd /root/initrd
```

```
find ./ | cpio -H newc -o > ../initrdnouveau
```

Par rapport à l'ancien fichier, nous sommes passés de 42165 blocs à 39449 blocs. Ensuite, la compression :

```
cd ..
```

```
gzip initrdnouveau
```

Pour remplacer le nouveau système `initrd` :

```
cd /boot
```

```
cp /root/initrdnouveau.gz .
```

Modifiez l'entrée dans `/boot/grub/menu.lst` (ou mieux : dupliquez l'entrée pour plus de sécurité) au niveau de la ligne :

```
initrd /boot/initrdnouveau.gz
```

Redémarrez avec votre nouvel `initrd`. La vitesse de démarrage n'est pas foncièrement évidente sans chronomètre, mais en tout cas, votre processus `initrd` est plus "propre" et adapté à votre configuration.