# Le noyau Linux

## Que fait le noyau Linux ?

Vous avez probablement entendu des gens parler de compilation du kernel ou de construction du kernel, mais qu'est-ce qu'un kernel exactement et à quoi sert-il ? Le kernel est le centre de votre ordinateur. C'est la fondation du système d'exploitation tout entier. Le kernel joue le rôle d'un pont entre le matériel et les applications. Cela signifie que le kernel est (généralement) la seule partie logicielle responsable de la gestion des composants matériels de votre ordinateur. C'est le kernel qui ordonne à votre disque dur de rechercher certaines données. C'est le kernel qui ordonne à votre carte réseau d'envoyer des changements rapides de courant. Le kernel est aussi à l'écoute du matériel. Lorsque la carte réseau détecte un ordinateur distant envoyant de l'information, elle renvoie cette information au kernel. Cela fait du kernel le logiciel le plus simple et le plus complexe de votre ordinateur.

## Travailler avec les modules

La complexité d'un kernel Linux récent est stupéfiante. Le code source pour le kernel représente environ 400 Mo non compressés. Il y a des milliers de développeurs, de centaines d'options et si tout était activé le noyau représenterait plus de 100 Mo à lui tout seul. Afin de conserver une taille raisonnable (ainsi que la quantité de RAM nécessaire pour le kernel), la plupart des options du kernel sont compilées sous forme de modules. Vous pouvez vous représenter ces modules comme des pilotes de périphériques qui peuvent être insérés ou enlevés du kernel à volonté. En fait, la plupart ne sont pas des pilotes de périphériques, mais correspondent à la prise en charge de choses telles que des protocoles réseaux, de systèmes de sécurité ou même de systèmes de fichiers. En résumé, presque toute partie du kernel Linux peut être compilée en tant que module.

Il est important de comprendre que Slackware gèrera automatiquement le chargement des modules pour vous. Lorsque le système démarre, ***udevd***(8) est lancé et commence à analyser le matériel présent. Pour chaque périphérique découvert, il chargera le module correspondant et créera un nœud dans `/dev`. Cela signifie que vous n'aurez pas à charger de modules pour pouvoir utiliser votre ordinateur, sauf cas particulier.

Comment savoir quels modules sont actuellement actifs et comment en ajouter ou en retirer ? Fort heureusement, nous avons une panoplie complète d'outils pour gérer cela. Comme vous avez pu le deviner, l'outil pour lister les modules est ***lsmod***(8).

```
darkstar:~# lsmod
Module                  Size  Used by
nls_utf8                1952  1
cifs                  240600  2
i915                  168584  2
drm                   168128  3 i915
i2c_algo_bit            6468  1 i915
tun                    12740  1
... lignes supprimées ...
```

En plus d'afficher les modules actuellement chargés il affiche la taille de chacun et vous indique quels autres modules l'utilisent.

Il existe deux programmes pour charger des modules : ***insmod***(8) et ***modprobe***(8). Tous deux peuvent charger des modules et signaler toute erreur (comme le fait de charger un module pour un périphérique qui n'est pas présent sur votre système), mais ***modprobe*** est souvent préféré car il peut charger toutes les dépendances d'un module. Son utilisation est très simple.

```
darkstar:~# insmod ext3
darkstar:~# modprobe ext4
darkstar:~# lsmod | grep ext
ext4                  239928  1
jbd2                   59088  1 ext4
crc16                   1984  1 ext4
ext3                  139408  0
jbd                    48520  1 ext3
mbcache                 8068  2 ext4,ext3
```

Supprimer des modules peut être une entreprise risquée, et une fois de plus nous avons deux logiciels pour cela : ***rmmod***(8) et ***modprobe***. Afin de supprimer un module avec modprobe, vous devrez utiliser l'option *-r*.

```
darkstar:~# rmmod ext3
darkstar:~# modprobe -r ext4
darkstar:~# lsmod | grep ext
```

# Pourquoi et comment compiler un noyau

La plupart des utilisateurs de Slackware n'auront jamais besoin de compiler un noyau. Le noyau *huge* (en anglais : “gros”) et le noyau *generic* (“générique”) contiennent potentiellement tout ce dont vous aurez besoin.

Cependant, certains utilisateurs peuvent avoir besoin de compiler un noyau. Si votre ordinateur dispose de matériel dernier cri, un nouveau noyau peut offrir une meilleur prise en charge de celui-ci. Parfois, un correctif (*patch*) du noyau peut être disponible pour corriger le problème que vous rencontrez. Dans ces situations, la compilation d'un nouveau noyau probablement justifiée. Les utilisateurs souhaitant simplement utiliser la version la plus à jour ou ceux pensant qu'un noyau personnalisé leur procurera de meilleures performances peuvent bien sûr le faire, mais il est peu probable qu'ils ressentent une amélioration significative.

Si vous pensez que compiler votre propre noyau est une tâche que vous voulez ou devez effectuer, ce chapitre devrait vous guider pour chacune des différentes étapes. Compiler et installer un noyau n'est pas si difficile, mais il existe un certain nombre d'erreurs que vous pourriez commettre en chemin, et plusieurs peuvent empêcher votre ordinateur de démarrer et générer une certaine frustration.

La première étape est de s'assurer que vous avez le code source du noyau installé sur votre système. The paquet pour le code source du noyau est inclus dans la série *“k”* de l'installateur Slackware, ou vous pouvez télécharger une autre version depuis http://www.kernel.org/. Traditionnellement, les sources du noyau sont situées dans `/usr/src/linux`, qui est un lien symbolique pointant vers la

version spécifique du noyau utilisé, mais cela n'est en aucun cas gravé dans le marbre. Vous pouvez copier le code source du noyau potentiellement n'importe où sans aucun problème.

```
darkstar:~# ls -l /usr/src
lrwxrwxrwx  1 root root   14 2009-07-22 19:59 linux -> linux-2.6.29.6/
drwxr-xr-x 23 root root 4096 2010-03-17 19:00 linux-2.6.29.6/
```

L'étape la plus complexe lors de la préparation du noyau est sa configuration. Il existe des centaines d'options, qui pour la plupart peuvent être compilées de manière optionnelle sous forme de modules. Cela signifie qu'il existe des centaines de façons de compiler un noyau. Heureusement, il existe quelques astuces utiles qui peuvent vous éviter trop de soucis. Le fichier de configuration du noyau se nomme `.config`. Si vous êtes particulièrement audacieux, vous pouvez modifier ce fichier avec votre éditeur de texte, mais je recommande très fortement d'utiliser les outils proposés par le noyau pour modifier `.config`.

À moins que vous ne soyez déjà expérimenté avec la configuration du noyau, vous devriez toujours partir d'une base solide et la modifier. Cela vous évitera de passer à côté d'une option importante qui vous obligerait à recommencer encore et encore jusqu'à ce que les choses fonctionnent. Le meilleur des fichiers `.config` repose sur celui utilisé par les noyaux par défaut de Slackware. Vous pouvez les trouver sur les disques d'installation de Slackware ou dans le répertoire `kernels/` de votre 'mirroir' favori.

```
darkstar:~# mount /mnt/cdrom
darkstar:~# cd /mnt/cdrom/kernels
darkstar:/mnt/cdrom/kernels# ls
VERSIONS.TXT  huge.s/  generic.s/  speakup.s/
darkstar:/mnt/cdrom/kernels# ls genric.s
System.map.gz  bzImage  config
```

Vous pouvez facilement remplacer le fichier `.config` proposé par défaut en copiant ou téléchargeant le fichier `config` pour le noyau que vous souhaitez utiliser. Personnellement, j'utilise le noyau generic.s comme base, mais vous pouvez préférer utiliser le fichier de configuration de huge.s. Le noyau générique produit beaucoup de modules et de ce fait un noyau plus petit, mais il requiert assez souvent l'utilisation d'initrd.

```
darkstar:/mnt/cdrom/kernels# cp generic.s/config /usr/src/linux/.config
```

Le fichier de configuration du noyau de Slackware ne comporte pas de point dans son nom, à la différence des fichiers de configuration du noyau en général. Si vous oubliez, ou copiez simplement le fichier `config` dans `/usr/src` quel que soit le fichier `.config` présent, c'est celui-ci qui sera utilisé.

Si vous souhaitez utiliser la configuration du noyau actuellement en service comme base, vous pouvez le trouver dans `/proc/config.gz`. Ce fichier spécial est lié au noyau actif et comporte l'intégralité de sa configuration sous forme compressée et nécessite que votre noyau ait été compilé avec l'option nécessaire pour le créer.

```
darkstar:~# zcat /proc/config.gz > /usr/src/linux/.config
```

Maintenant que nous avons préparé une configuration de base solide, il est temps de faire les modifications que nous voulons. La construction d'un noyau complet depuis sa configuration jusqu'à la compilation est effectuée par la commande ***make***(1) en y ajoutant des options spécifiques. Chaque

option correspond à une fonction différente.

Si vous passez à une nouvelle version du noyau, vous aurez définitivement besoin de l'option *oldconfig*. Cela parcourera votre fichier `.config` de base et recherchera pour les éléments manquants qui indiquent que la nouvelle version du noyau contient de nouvelles options. Sachant que de nouvelles options sont ajoutées pour quasiment chaque nouvelle version du noyau, c'est généralement une bonne chose à faire.

```
darkstar:/usr/src/linux# make oldconfig
scripts/kconfig/conf -o arch/x86/Kconfig
*
* Restart config...
*
*
* File systems
*
Second extended fs support (EXT2_FS) [M/n/y/?] m
  Ext2 extended attributes (EXT2_FS_XATTR) [N/y/?] n
  Ext2 execute in place support (EXT2_FS_XIP) [N/y/?] n
Ext3 journalling file system support (EXT3_FS) [M/n/y/?] m
  Ext3 extended attributes (EXT3_FS_XATTR) [Y/n/?] y
    Ext3 POSIX Access Control Lists (EXT3_FS_POSIX_ACL) [Y/n/?] y
    Ext3 Security Labels (EXT3_FS_SECURITY) [Y/n/?] y
The Extended 4 (ext4) filesystem (EXT4_FS) [N/m/y/?] (NEW) m
```

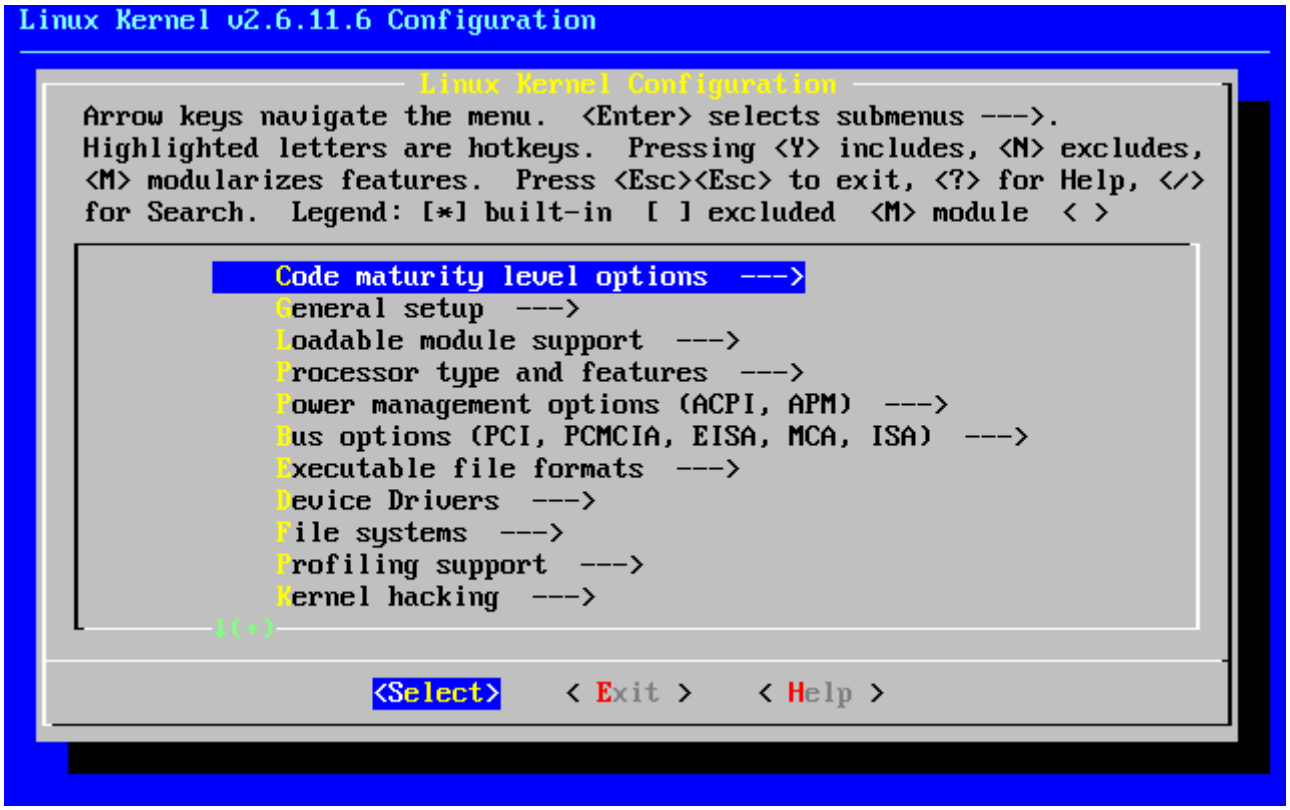
Ici vous pouvez voir que le nouveau noyau que je compile ajoute le support pour un nouveau système de fichiers : ext4. *oldconfig* a revu ma configuration originelle, conservé les anciennes options telle qu'elles étaient et m'a demandé ce que je désire faire pour les nouvelles options. Il est généralement sage de choisir l'option par défaut, mais vous pouvez choisir ce qui vous convient. *oldconfig* est un outil très pratique pour visualiser les nouvelles options de configuration, en faisant le plus adapté pour les utilisateurs qui veulent simplement essayer la dernière version du noyau.

Pour les opérations de configuration plus sérieuses, il existe une multitude d'options. Le noyau Linux peut être configuré de trois façons principales. La première est *config*, qui énumérera chacune des options une par une et vous demander de que vous souhaitez. Cette méthode est si fastidieuse qu'il est peu probable que quiconque l'utilise encore.

```
darkstar:/usr/src/linux# make config
scripts/kconfig/conf arch/x86/Kconfig
*
* Linux Kernel Configuration
*
*
* General setup
*
Prompt for development and/or incomplete code/drivers (EXPERIMENTAL) [Y/n/?]
Y
Local version - append to kernel release (LOCALVERSION) [] -test
Automatically append version information to the version string
(LOCALVERSION_AUTO) [N/y/?] n
Support for paging of anonymous memory (swap) (SWAP) [Y/n/?]
```

Heureusement, il existe deux autres méthodes plus faciles pour configurer votre noyau, *menuconfig* et *xconfig*. Toutes les deux créent un programme basé sur des menus qui vous propose de sélectionner ou dé-sélectionner des options sans avoir à toutes les vérifier. *menuconfig* est la méthode la plus utilisée, et c'est celle que je recommande. *xconfig* n'est utile que si vous essayez de compiler un noyau depuis un environnement graphique via ***X***. Les deux sont très proches, et nous n'allons documenter que *menuconfig*.

Lancer make menuconfig depuis un terminal vous présentera une interface basée sur curses, comme vous pouvez le voir ci-dessous. Chaque section du noyau dispose son propre sous-menu, et vous pouvez naviguer avec les touches fléchées.




Si vous compilez un noyau d'une version identique à un proposé par Slackware vous devez modifier l'option *"Local version"*. Elle se trouve dans le sous-menu *"General setup"*. Si vous ne modifiez pas cette option le noyau que vous compilerez remplacera tous les modules utilisés par les noyaux par défaut. Cela peut rapidement rendre votre système inutilisable.

Une fois que vous avez fini de configurer votre noyau, il est temps de passer à l'étape de la compilation. Il existe différentes méthodes pour cela, mais la plus efficace est d'utiliser *bzImage*. En utilisant cette option avec la commande ***make*** la compilation du noyau débutera et vous verrez un flot de messages s'afficher dans votre terminal jusqu'à l'accomplissement de tout le processus ou si une erreur fatale se produit.

```
darkstar:/usr/src/linux# make bzImage
scripts/kconfig/conf -s arch/x86/Kconfig
  CHK     include/linux/version.h
  CHK     include/linux/utsrelease.h
  SYMLINK include/asm -> include/asm-x86
  CALL    scripts/checksyscalls.sh
  CC      scripts/mod/empty.o
  HOSTCC  scripts/mod/mk_elfconfig
```

```
  MKELF   scripts/mod/elfconfig.h
  HOSTCC  scripts/mod/file2alias.o
... plusieurs centaines de lignes suivent ...
```

Si le processus engendre une erreur, vous devriez vérifier la configuration de votre noyau en premier lieu. Les erreurs de compilation sont le plus souvent générées par un problème dans le fichier `.config`. Si tout s'est bien passé nous n'avons pas encore fini, il reste à préparer les modules.

```
darkstar:/usr/src/linux# make modules
  CHK     include/linux/version.h
  CHK     include/linux/utsrelease.h
  SYMLINK include/asm -> include/asm-x86
  CALL    scripts/checksyscalls.sh
  HOSTCC  scripts/mod/file2alias.o
... plusieurs milliers de lignes suivent ...
```

Si la compilation du noyau et des modules se termine correctement, il ne nous reste qu'à les installer. L'image du noyau doit être recopiée dans un endroit spécifique, généralement le répertoire `/boot`, et vous devez lui donner un nom unique pour éviter d'écraser une autre image du noyau présente au même endroit. Habituellement les images du noyau sont nommées `vmlinuz` en y ajoutant leur numéro de version et une référence locale.

```
  darkstar:/usr/src/linux# cat arch/x86/boot/bzImage > /boot/vmlinuz-
release_number-local_version
  darkstar:/usr/src/linux# make modules_install
```

Une fois ces étapes complétées, vous aurez une nouvelle image du noyau située dans `/boot`'et de nouveaux modules dans le répertoire `/lib/modules`. Afin de pouvoir utiliser ce nouveau noyau vous aurez besoin d'éditer `lilo.conf`, de créer un initrd (uniquement si vous avez besoin de charger un ou plusieurs modules au démarrage) et d'exécuter la commande ***lilo*** pour mettre à jour votre gestionnaire de démarrage. Lorsque vous redémarrerez, si tout se passe sans accroc, vous devriez avoir une option pour démarrer votre système avec votre tout nouveau noyau. Si quelque chose ne marche pas, vous passerez sûrement un certain temps à régler le problème.

# Navigation

**Chapitre précédent : Gérer les mises à jour**

# Sources

- Source originale : http://www.slackbook.org/beta
- Publication initiale d'Alan Hicks, Chris Lumens, David Cantrell, Logan Johnson

- Traduction initiale de escaflown
- Traduction de Ellendhel

slackbook, kernel

From:
http://docs.slackware.com/ - **SlackDocs**

Permanent link:
**http://docs.slackware.com/fr:slackbook:linux_kernel**

Last update: **2017/06/16 06:30 (BST)**