Compilation du noyau

Du point de vue de la compilation, le noyau est presque une application comme les autres, avec un code source, une procédure de compilation et une procédure d'installation.

1. Les composants du noyau

Le noyau Linux est responsable de la gestion du matériel. La notion de pilote de périphérique n'existe pas directement en environnement Linux puisque les éléments permettant de communiquer correctement avec un périphérique sont compris dans le code du noyau. On se rend assez vite compte du confort de cette situation : le noyau récent compris dans une distribution Linux permet de gérer directement l'ensemble des périphériques d'un système sans avoir à installer des pilotes supplémentaires. En contrepartie, le code du noyau pour gérer l'ensemble des périphériques existant a tendance à devenir de plus en plus imposant, et son chargement intégral entraînerait une consommation de mémoire démesurée. Pour cette raison, le noyau a une structure modulaire, et seuls les modules nécessaires au fonctionnement du système sont chargés en mémoire.

a. Le cœur de noyau

Ce que l'on peut appeler le « cœur de noyau » est la partie irréductible du noyau, celle qui sera intégralement chargée en mémoire. Elle ne contient en principe que des éléments dont on est sûr qu'ils seront nécessaires à l'utilisation. Le cœur de noyau est un fichier se trouvant dans le répertoire /boot et dont la taille est de quelques MégaOctets.

b. Les modules

L'importance des modules de noyau

Les modules ont un rôle primordial car beaucoup de fonctions essentielles sont gérées sous forme de modules. Si un noyau ne dispose pas des modules nécessaires au fonctionnement du système, les fonctions afférentes ne seront tout simplement pas disponibles.

Tentative de chargement d'une ressource non supportée

Cet exemple est réalisé sur un système dont le noyau ne supporte pas le format de filesystem ext3.

```
light:/mnt# mount /dev/hda3 partition
mount: unknown filesystem type 'ext3'
light:/mnt#
```

Les modules sont des fichiers portant l'extension **.ko** qui sont chargés en mémoire en fonction des besoins. Des commandes sont disponibles pour consulter la liste des modules chargés, en retirer de la mémoire ou en charger de nouveaux.

Les noyaux de versions anciennes (2.4 notamment) exploitent des fichiers de modules portant l'extension « .o ».

Manipulations ponctuelles des modules

Affichage des modules chargés en mémoire

```
lsmod
```

Affichage des modules disponibles sur le système

```
modprobe -l
```

Les fichiers correspondants à ces modules se trouvent conventionnellement dans un répertoire **/lib/modules** et dans une sous-arborescence du nom noyau courant, tel que renvoyé par la commande **uname -r**.

Retrait d'un module chargé en mémoire

```
rmmod nom_module
  ou
modprobe -r nom_module
```

Où *nom_module* représente le nom du module présent en mémoire tel qu'il a été affiché par la commande **lsmod**. Les deux commandes **rmmod** et **modprobe -r** ont le même résultat.

Chargement d'un module en mémoire

```
insmod fichier_module
  ou
modprobe nom_module
```

Où *nom_module* représente le nom du module tel qu'il serait affiché par la commande **lsmod**, alors que *fichier_module* représente le nom du fichier de module présent sur le disque. En fait, le nom du module est obtenu en retirant l'extension **.ko** au nom du fichier.

Chargement d'un module

Le chargement manuel du module qui manquait précédemment rend possible le montage de la partition ext3.

```
light:/mnt# insmod /lib/modules/2.6.26-2-686/kernel/fs/ext3/ext3.ko
light:/mnt# mount /dev/hda3 partition
light:/mnt# mount
/dev/hda1 on / type ext2 (rw,errors=remount-ro)
(...)
/dev/hda3 on /mnt/partition type ext3 (rw)
light:/mnt#
```

Chargement forcé d'un module

Les modules sont en principe chargés au démarrage en fonction de la détection du matériel présent. Il est toutefois possible de forcer le chargement d'un module en alimentant un fichier de configuration des modules. Tout module mentionné dans un fichier **/etc/modules** sera chargé inconditionnellement au démarrage.

Configuration des modules

Le fichier **/etc/modules.conf** permet de configurer certains modules et notamment de définir des associations forcées entre périphérique et modules.

Exemple de fichier /etc/modules.conf

```
# Association forcée du pilote tg3 avec la carte réseau
alias eth0 tg3
```

À titre de vérification ou pour voir si les associations entre le matériel et les modules se sont bien réalisées, il est possible d'afficher des informations sur les modules chargés avec la commande **modinfo**.

Visualisation des informations liées à un module

On voit notamment le fichier .ko contenant le code du module, quelques informations d'environnement et les alias gérés dynamiquement par le système pour les matériels liés à ce module.

```
root@serveur:/boot$ modinfo r8169
filename:       /lib/modules/2.6.32-24-generic/kernel/drivers/net/r8169.ko
version:        2.3LK-NAPI
license:        GPL
description:    RealTek RTL-8169 Gigabit Ethernet driver
author:         Realtek and the Linux r8169 crew <netdev@vger.kernel.org>
srcversion:     D37E06388C6313C1D062CC3
alias:          pci:v00000001d00008168sv*sd00002410bc*sc*i*
alias:          pci:v00001737d00001032sv*sd00000024bc*sc*i*
alias:          pci:v000016ECd00000116sv*sd*bc*sc*i*
alias:          pci:v00001259d0000C107sv*sd*bc*sc*i*
```

```
alias:          pci:v00001186d00004300sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008169sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008168sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008167sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008136sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008129sv*sd*bc*sc*i*
depends:        mii
vermagic:       2.6.32-24-generic SMP mod_unload modversions
parm:           rx_copybreak:Copy breakpoint for copy-only-tiny-frames (int)
parm:           use_dac:Enable PCI DAC. Unsafe on 32 bit PCI slot. (int)
parm:           debug:Debug verbosity level (0=none, ..., 16=all) (int)
root@serveur:/boot$
```

c. Autour du noyau

Nous savons maintenant que le noyau est constitué d'une entité insécable, et de modules chargés en mémoire à la demande. Lors de la phase de démarrage, le gestionnaire de démarrage charge le noyau, et les modules correspondants à la configuration matérielle du système sont également chargés. Pour accélérer la phase de détection du matériel et le chargement des modules associés, la plupart des systèmes modernes exploitent un ramdisk (disque virtuel dont le support physique est la mémoire) contenant l'ensemble des modules. Ce ramdisk est généré après la compilation du noyau, et est appelé directement par le gestionnaire de démarrage.

d. Gestion des versions du noyau

Le noyau porte un numéro de version de type A.B.C, par exemple 2.6.15. « A » donne la version principale du noyau, actuellement, et pour sans doute encore quelques temps, la version 2. « B » représente la version courante du noyau. Cette valeur est systématiquement paire sur les versions de noyaux stables et impaire sur les versions en développement. Enfin, « C » est incrémenté en fonction des évolutions mineures de noyau, essentiellement les corrections de bogues et les prises en charge de nouveaux matériels.

La commande **uname -r** permet d'afficher la version du noyau en cours d'exécution.

Affichage de la version du noyau courant

```
toto@serveur:~$ uname -r
2.6.32-24-generic
toto@serveur:~$
```

2. Procédure de compilation et d'exploitation

La procédure de compilation doit toujours être consultée dans le fichier **README** présent avec les sources du noyau. Les éléments spécifiques du noyau sont documentés dans un répertoire **Documentation** fourni avec les sources. Le fichier **README** ne documente que la procédure de compilation.

a. Récupération des sources

Le code source du noyau est téléchargeable librement depuis le site « http://www.kernel.org ». Les principales versions y sont disponibles. Les liens « Full source » permettent de télécharger le code source complet du noyau.

Le noyau étant livré sous forme d'une archive tar.bz2, il faut d'abord la décompresser. Comme pour toute compilation d'application, la majeure partie du travail se passera dans le répertoire issu de l'extraction de l'archive.

Si on travaille sur les sources de noyau copiées lors de l'installation du système, le répertoire de travail devrait être /usr/src/linux. La documentation se trouvera alors naturellement dans /usr/src/linux/Documentation. En cas de travail sur des sources nouvelles, un répertoire neutre est recommandé.

b. Génération du fichier de réponse

La compilation s'effectue en fonction des informations données dans un fichier **.config** qui se trouve dans la racine du répertoire des sources. Ce fichier indique pour chaque élément du noyau s'il doit être présent dans le cœur de noyau, présent sous forme de module, ou absent du noyau compilé.

Selon le système employé, plusieurs moyens sont à notre disposition pour générer ce fichier de réponse.

Génération du fichier de réponses : commandes possibles	
make config	Pose la question à l'utilisateur pour chacun des modules.
make menuconfig	Présente une interface texte améliorée.
make xconfig	Présente une interface graphique.
make gconfig	Présente une interface graphique.
make defconfig	Génère un fichier de réponse en s'appuyant sur toutes les valeurs de compilation par défaut.
make oldconfig	Génère un fichier de réponse en s'appuyant sur un fichier .config déjà utilisé pour une version plus ancienne du noyau.

Si la compilation du noyau ne présente pas de difficulté particulière, le renseignement du fichier de réponse requiert une compétence étendue et la connaissance précise du matériel.

Exemple de création du fichier de réponses

La compilation précise du noyau nécessite une connaissance de toutes les technologies matérielles gérées par ce noyau.

```
[root@beta linux-2.6.34.4]# make config
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/docproc
  HOSTCC  scripts/basic/hash
  HOSTCC  scripts/kconfig/conf.o
(...)
PentiumPro memory ordering errata workaround (X86_PPRO_FENCE) [Y/n/?] y
HPET Timer Support (HPET_TIMER) [Y/n/?] y
Maximum number of CPUs (NR_CPUS) [8] (NEW) 8
SMT (Hyperthreading) scheduler support (SCHED_SMT) [Y/n/?] y
Multi-core scheduler support (SCHED_MC) [Y/n/?] y
Preemption Model
  1. No Forced Preemption (Server) (PREEMPT_NONE)
> 2. Voluntary Kernel Preemption (Desktop) (PREEMPT_VOLUNTARY)
  3. Preemptible Kernel (Low-Latency Desktop) (PREEMPT)
choice[1-3]: 2
Reroute for broken boot IRQs (X86_REROUTE_FOR_BROKEN_BOOT_IRQS) [N/y/?] (NEW) n
Machine Check / overheating reporting (X86_MCE) [Y/n/?] n
Toshiba Laptop support (TOSHIBA) [M/n/y/?] n
Dell laptop support (I8K) [M/n/y/?] m
Enable X86 board specific fixups for reboot (X86_REBOOTFIXUPS) [N/y/?]n
(...)
CRC-CCITT functions (CRC_CCITT) [M/y/?] m
CRC16 functions (CRC16) [M/y/?] m
CRC calculation for the T10 Data Integrity Field (CRC_T10DIF) [N/m/y/?] (NEW) m
CRC ITU-T V.41 functions (CRC_ITU_T) [M/y/?] m
CRC32 functions (CRC32) [Y/?] y
CRC7 functions (CRC7) [N/m/y/?] (NEW) n
CRC32c (Castagnoli, et al) Cyclic Redundancy-Check (LIBCRC32C) [Y/m/?] y
#
# configuration written to .config
#


[root@beta linux-2.6.34.4]#
```

Premières lignes du fichier de configuration

```
[root@beta linux-2.6.34.4]# head -15 .config
#
```

```
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.34.4
# Mon Aug 16 16:57:44 2010
#
# CONFIG_64BIT is not set
CONFIG_X86_32=y
# CONFIG_X86_64 is not set
CONFIG_X86=y
CONFIG_OUTPUT_FORMAT="elf32-i386"
CONFIG_ARCH_DEFCONFIG="arch/x86/configs/i386_defconfig"
CONFIG_GENERIC_TIME=y
CONFIG_GENERIC_CMOS_UPDATE=y
CONFIG_CLOCKSOURCE_WATCHDOG=y
CONFIG_GENERIC_CLOCKEVENTS=y
[root@beta linux-2.6.34.4]#
```

Taille indicative du fichier de configuration

```
[root@beta linux-2.6.34.4]# wc -l .config
3641 .config
[root@beta linux-2.6.34.4]#
```

La configuration des modules pour une version de noyau installée doit se trouver dans un fichier config-*version* dans le répertoire /boot.

Visualisation des fichiers de configuration des noyaux

```
root@serveur:/boot$ ls config*
config-2.6.27-11-generic  config-2.6.32-21-generic  config-2.6.32-24-generic
config-2.6.28-16-generic  config-2.6.32-22-generic
config-2.6.31-21-generic  config-2.6.32-23-generic
root@serveur:/boot$ cat config-2.6.32-24-generic
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.32-24-generic
# Thu Aug 19 01:38:31 2010
#
CONFIG_64BIT=y
# CONFIG_X86_32 is not set
CONFIG_X86_64=y
CONFIG_X86=y
CONFIG_OUTPUT_FORMAT="elf64-x86-64"
CONFIG_ARCH_DEFCONFIG="arch/x86/configs/x86_64_defconfig"
CONFIG_GENERIC_TIME=y
CONFIG_GENERIC_CMOS_UPDATE=y
CONFIG_CLOCKSOURCE_WATCHDOG=y
(...)
root@serveur:/boot$
```

c. Compilation du noyau et des modules

La compilation se réalise le plus simplement du monde en tapant la commande **make** depuis le répertoire racine des sources. La durée de l'opération dépend de la puissance de la machine sur laquelle elle est réalisée, mais une bonne heure est souvent nécessaire. Pour un noyau en version 2.6, la commande **make** provoque la compilation du noyau et des modules.

Compilation du noyau et des modules

C'est parti pour une heure ou deux...

```
[root@beta linux-2.6.34.4]# make
scripts/kconfig/conf -s arch/x86/Kconfig
  CHK     include/linux/version.h
  UPD     include/linux/version.h
  CHK     include/generated/utsrelease.h
```

```
  UPD     include/generated/utsrelease.h
  CC      kernel/bounds.s
  GEN     include/generated/bounds.h
  CC      arch/x86/kernel/asm-offsets.s
(...)
  CC      arch/x86/kernel/cpu/cpufreq/speedstep-lib.o
  CC      arch/x86/kernel/cpu/cpufreq/speedstep-smi.o
  LD      arch/x86/kernel/cpu/cpufreq/built-in.o
  CC [M]  arch/x86/kernel/cpu/cpufreq/powernow-k8.o
  CC [M]  arch/x86/kernel/cpu/cpufreq/acpi-cpufreq.o
  CC [M]  arch/x86/kernel/cpu/cpufreq/speedstep-centrino.o
  CC [M]  arch/x86/kernel/cpu/cpufreq/p4-clockmod.o
  CC      arch/x86/kernel/cpu/mcheck/mce.o
  CC      arch/x86/kernel/cpu/mcheck/mce-severity.o
  CC      arch/x86/kernel/cpu/mcheck/mce_intel.o
(...)
```

L'exécution de la commande **make** provoque la compilation du noyau et de ses modules. Elle appelle aussi la commande **depmod** qui génère le fichier **modules.dep** de dépendance des modules.

d. Installation des modules

Les modules sont installés par la commande spécifique **make modules_install**. Ils sont copiés dans un répertoire **/lib/modules**, sous un répertoire correspondant à la version du noyau.

Visualisation des répertoires contenant les modules

Chaque version de noyau installé a son répertoire de modules correspondant.

```
root@serveur:~$ ls /lib/modules/
2.6.27-11-generic  2.6.28-16-generic  2.6.32-22-generic
2.6.27-7-generic   2.6.31-21-generic  2.6.32-23-generic
2.6.27-9-generic   2.6.32-21-generic  2.6.32-24-generic
root@serveur:~$
```

e. Installation du noyau

Le noyau hors-modules se trouve dans le répertoire des sources dans une arborescence **arch/x86/boot** pour les versions 32 bits ou **arch/ia64/boot** pour les versions 64 bits sous le nom **bzImage**. Son installation dans le système en production se fait en copiant simplement ce fichier dans le répertoire **/boot**. Le nom utilisé par défaut (bzImage) est tout à fait exploitable, mais il est préférable de le renommer pour tenir compte de la version compilée.

Des compilations réalisées avec des versions anciennes de noyau peuvent générer un fichier zImage et non bzImage. Le préfixe z ou bz indique le format de compression du fichier noyau (gzip pour z et bzip2 pour bz).

Un noyau nouvellement compilé doit toujours être installé en plus du noyau existant. Ne jamais remplacer un noyau qui fonctionne par un nouveau noyau.

Installation du noyau

L'usage veut que le fichier de noyau ait un nom normalisé qui reflète sa version.

```
root@serveur# cp arch/x86/boot/bzImage /boot/vmlinuz-2.6.15
root@serveur#
```

f. Création du ramdisk des modules

Il faut mettre à disposition du noyau un ramdisk contenant l'ensemble des modules compilés pour la nouvelle version. Ce ramdisk nécessite un fichier image, qui peut être construit avec deux commandes différentes en fonction de la génération du système employé. La commande historique est **mkinitrd**. Elle tend à disparaître au profit de **mkinitramfs**.

Création d'un ramdisk avec la commande mkinitrd

```
mkinitrd nom_image version
```

Création d'un ramdisk avec la commande mkinitramfs

```
mkinitramfs -o nom_image version
```

Où *nom_image* représente le nom du fichier image de ramdisk à créer, et *version* le numéro de version du noyau. Ce numéro correspond en fait au répertoire des modules situé dans /lib/modules.

Exemple de création d'un ramdisk

```
root@serveur:/boot$ mkinitrd /boot/initrd-2.6.28.img 2.6.28
root@serveur:/boot$ file initrd-2.6.28.img
initrd.img-2.6.32-24-generic: gzip compressed data, from Unix, last modified: Fri
Aug 20 07:54:31 2010
root@serveur:/boot$
```

Le fichier ramdisk est en fait une archive cpio compressée au format gzip. Les commandes de création de ramdisk génèrent directement leurs fichiers à ce format.

Un système récent ne devrait proposer que la commande mkinitramfs. Si toutefois mkinitrd était disponible aussi, elle ne devrait pas être utilisée. mkinitrd s'appuie sur devfs et non udev, et ne supporte pas les disques sata.

g. Configuration du gestionnaire de démarrage

Il ne suffit pas d'avoir compilé le noyau et de l'avoir placé au bon endroit, encore faut-il que le gestionnaire de démarrage soit configuré pour être capable de charger ce noyau. Il conviendra donc d'ajouter une entrée au gestionnaire de démarrage en conséquence. Attention, il ne faut rien retirer à la configuration du gestionnaire de démarrage : on ne touche pas à ce qui marche déjà. Il suffit d'ajouter une entrée dans le fichier de configuration du gestionnaire en se basant au besoin sur les entrées déjà présentes.

Ajout d'une entrée au gestionnaire de démarrage

La préservation des entrées existantes permet de toujours pouvoir démarrer sur une configuration stable.

```
# noyau fonctionnel d'origine
title           Debian GNU/Linux, kernel 2.6.26-2-686
root            (hd0,0)
kernel          /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro quiet
initrd          /boot/initrd.img-2.6.26-2-686

# noyau ajouté à tester
title           ESSAI - modules statiques
root            (hd0,0)
kernel          /boot/vmlinuz-2.6.20 root=/dev/hda1 ro quiet
initrd          /boot/initrd.img-2.6.20
```