La compilation du noyau

par Romain Guy (Gfx)

Date de publication : 6 Juin 2006

Dernière mise à jour :

Tout le système Linux se trouve architecturé autour d'un noyau ou "kernel" en anglais. Celui-ci contient, outre les fonctions nécessaires au système d'exploitation, des drivers pour vos différents périphériques.

I - Introduction

La gestion des périphériques sous Linux diffère grandement de celle de Windows. En effet, sous Linux, les périphériques se pilotent par des drivers auxquels Linux accède de deux manières : d'un côté les modules du noyau, placés dans le répertoire /lib/modules/preferred, que l'on peut ainsi charger à l'aide des commandes "modprobe" et "insmode". La seconde méthode consiste à placer les drivers au sein même du noyau. Pour cela, il faut recompiler celui-ci. L'incorporation de drivers au noyau s'avère obligatoire si les périphériques doivent être reconnus dès le démarrage (cas des contrôleurs SCSI). Bien entendu, votre système fonctionne parfaitement avec le noyau par défaut. Cependant, celui-ci constitue un noyau générique contenant de nombreuses options dont vous n'aurez sûrement pas l'usage : les retirer permet d'alléger le système. De plus, certaines optimisations peuvent être effectuées... Nous allons donc explorer les diverses étapes constituant la compilation du noyau. L'idée même d'une telle opération déroute souvent les utilisateurs débutants. Pourtant, les sources du noyau sont conçues de manière à ce que la compilation puisse s'effectuer rapidement et sans erreur. Si malgré tout vous provoquez des erreurs empêchant le noyau de se charger, le système ne sera pas inactif pour autant et vous aurez toujours l'occasion de revenir en arrière.

II - A vos marques

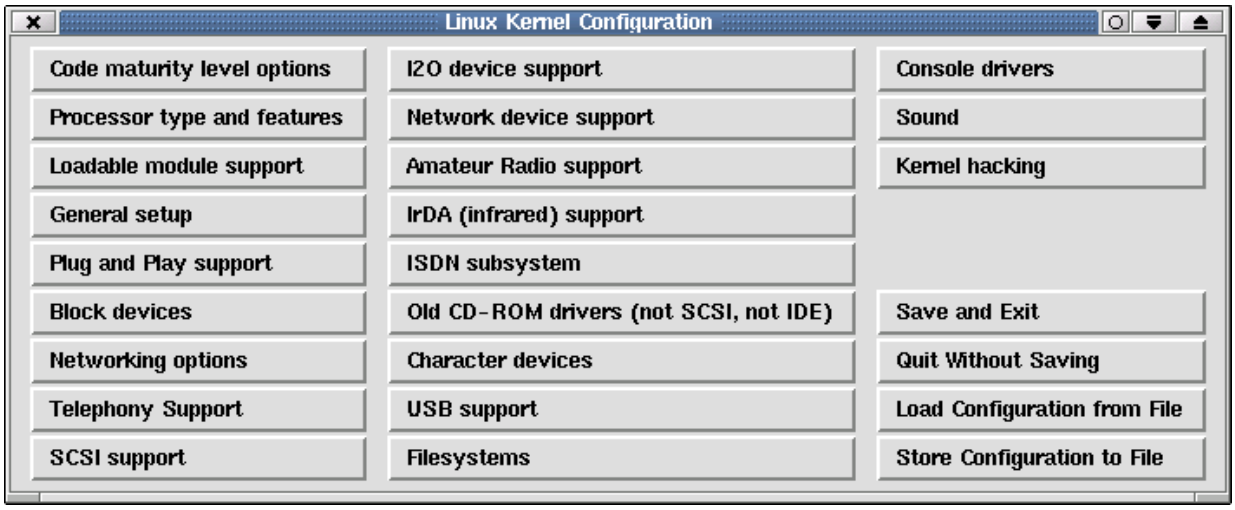
Bien que les programmeurs du noyau soient très actifs, les versions de développement peuvent comporter de nombreuses instabilités. Aussi, avant d'envisager l'exploitation d'une nouvelle version du kernel, soyez sûr que vous ayez besoin de changer de noyau. Ceci est d'autant plus vrai que votre système ne deviendra en aucun cas obsolète si vous n'avez pas recours à la dernière version du noyau. Parmi les applications disponibles sous Linux, une majorité peut s'exécuter avec de très anciennes versions du kernel... Avant de vous attaquer à la compilation, il convient de prendre quelques mesures de sécurité. Premièrement, créez une disquette d'amorçage en utilisant la commande : "mkbootdisk x.x.xx" dans laquelle les "x" désignent le numéro de version du noyau actuel. Pour obtenir ce numéro, tapez "ls -l /lib/modules". Celui qui se situe tout à droite désigne le numéro de version du kernel. Vous pouvez maintenant décider d'exploiter les sources du noyau courant ou d'une mise à jour pour compiler. Pour savoir si les sources sont disponibles sur votre système, tapez "rpm -q kernel-source". Si en retour, une ligne du type kernel-source-xx.xx.xx apparaît, alors les sources se trouvent déjà mises en place. Sinon, installez-les depuis le CD-Rom de votre distribution. Avec une RedHat, nous effectuerons les commandes suivantes :

```
cd /mnt/cdrom/RedHat/RPMS
rpm -i kernel-headers-2.2.12-20.i386.rpm
rpm -i kernel-source-2.2.12-20.i386.rpm
```

Bien entendu, le numéro de version varie d'une distribution à l'autre. Une fois mises en place, les sources figurent dans le répertoire /usr/src/linux. Placez-vous dans celui-ci.

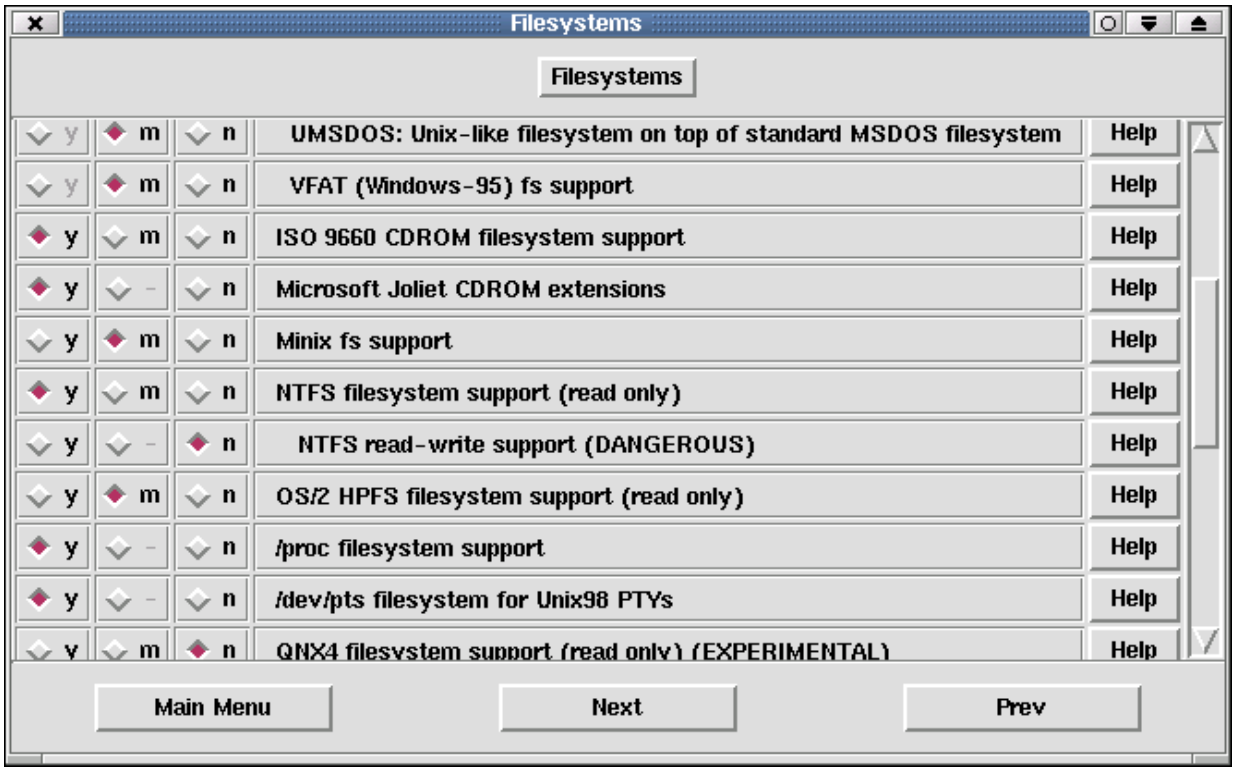
III - Configuration

La sélection des options du kernel peut se faire de deux manières : soit en mode texte dans la console, soit grâce à une interface graphique sous X11. Une fois que vous vous trouvez dans le répertoire des sources, tapez "make menuconfig" pour utiliser la première méthode ou "make xconfig" pour la seconde. La sélection en mode console se montre très longue et particulièrement astreignante, ainsi préférez-lui la méthode sous X. Au bout de quelques secondes, une fenêtre nommée "Linux Kernel Configuration" apparaît. Celle-ci contient la liste des différentes sections du noyau. Une fois dans une section, vous pouvez procéder à divers réglages. Les boutons "Next" et "Main Menu" permettent de passer à la section suivante ou de revenir au menu principal. Faisons les choses dans l'ordre et cliquons sur le bouton de la première section, intitulée "Code maturity level options". La fenêtre change alors pour laisser place aux options de la section. Chaque option peut avoir trois valeurs : Y (Yes, pour compiler dans le noyau), M (Module : compiler en tant que module externe) et N (No, ne pas compiler). Les pilotes dont vous aurez toujours besoin doivent êtres inclus dans le noyau par Y tandis que ceux nécessitant une configuration supplémentaire (IRQ, DMA...) doivent se trouver compilés comme des modules. Il en va ainsi pour la carte son notamment.



Le menu de la configuration du noyau.

La seule option de cette première section s'appelle "Prompt for development and/or incomplete code/drivers". Si vous l'activez, les sections comporteront des options spéciales encore en cours de développement. Si ce n'est par curiosité, évitez de répondre "oui" à cette invite, vous alourdiriez inutilement la phase de configuration (et le kernel). La vraie sélection des options ne commencera que lorsque vous aurez cliqué sur "Next".



Les options du noyau disposent de trois états : Y, M et N.

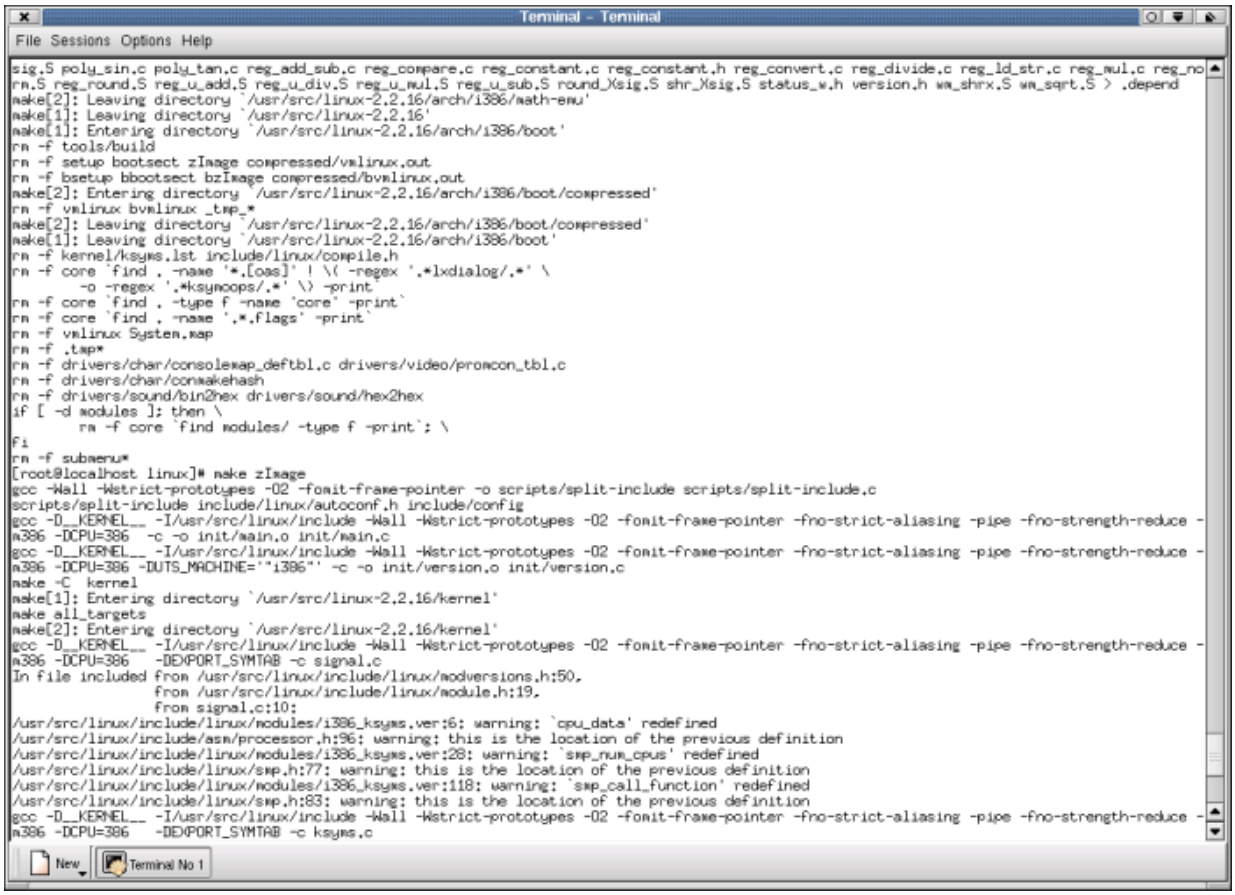
IV - Options principales

La seconde section de configuration permet d'activer diverses optimisations en fonction du processeur. Il arrive par exemple qu'un système Linux tournant sur un Pentium II possède un noyau compilé pour un 486. Les performances sont alors loin d'être optimales. L'option "Math emulation" se montre inutile si vous ne possédez pas un 386 ou 486. L'activation de l'option MTRR résout certains problèmes graphiques et l'option "Symetric multi-processing support" sert à gérer des machines multi-processeurs. La section suivante, "Loadable module support" permet d'activer le support des drivers en tant que modules du noyau. Il se montre impératif de lancer cette option. De la même manière, il est recommandé de déclencher l'option "Kernel daemon support" qui donne le moyen de charger automatiquement des modules au démarrage. Dans la section "General Setup", la seule chose réellement importante à savoir est d'activer le "Kernel support for ELF binaries". Le format binaire ELF a supplanté l'ancien a.out. Sans lui, vous risquez de rencontrer bien des difficultés pour exécuter les applications.

Prenez un peu de temps pour étudier avec attention toutes les possibilités qui s'offrent à vous durant la sélection des options du noyau. N'hésitez surtout pas à cliquer sans arrêt sur les boutons "Help" qui affichent une description concise de chaque option. Une fois revenu au menu principal, cliquez sur "Save and exit".

V - Compilation

Enfin nous pouvons compiler ! Avant tout, il faut effacer toute trace d'une éventuelle précédente compilation en tapant : "make dep clean". Ensuite, la commande "make zImage" va créer le noyau proprement dit. Si vous avez sélectionné de trop nombreuses options, vous obtiendrez un message d'erreur vous informant que le noyau s'avère trop volumineux. Dans ce cas, utilisez le paramètre "bzImage" plutôt que "zImage". Normalement, l'étape de compilation ne devrait poser aucun problème. Une fois le noyau créé, le plus dur est fait, mais il reste néanmoins quelques actions à accomplir. Premièrement, la création des modules : "make modules modules_install". Le noyau doit ensuite être installé. Dans le respect de certaines conventions, nous allons déposer le noyau, appelé zImage, dans le répertoire /boot avec le nom vmlinuz-xx.xx.xx. Nous lançons donc "cp arch/i386/boot/zImage /boot/vmlinuz-2.2.17" (en admettant que nous ayons compilé un noyau de version 2.2.17). Dans le cas d'un kernel trop volumineux, tapez "cp arch/i386/boot/bzImage /boot/vmlinuz-2.2.17".

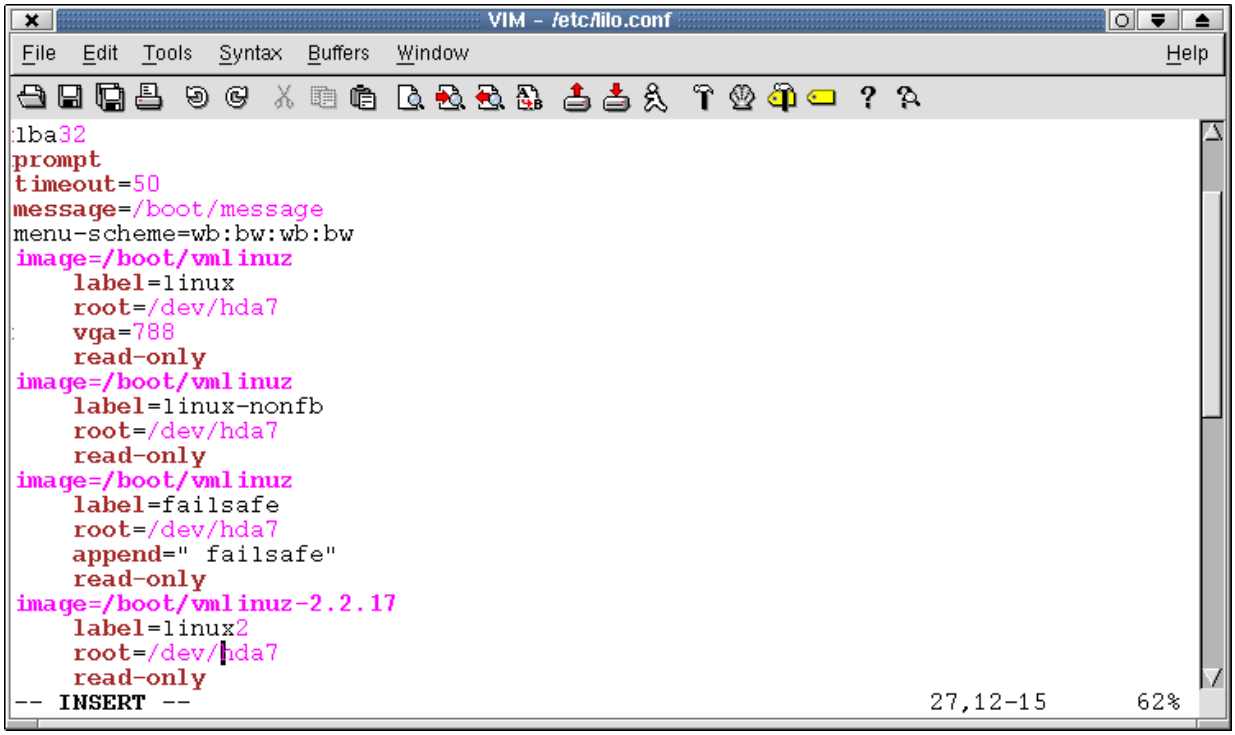


La compilation... évidemment, cela peut faire peur !

Pour démarrer ce noyau, nous allons réaliser une disquette d'amorçage. Il existe deux moyens d'en concevoir une. Soit en passant par la commande "dd" (la disquette ne peut alors être lue sous Linux) ou en employant "makebootdisk". La première commande s'utilise ainsi : "dd if=arch/i386/boot/zImage of=/dev/fd0" et la seconde : "makebootdisk 2.2.17". Finalement, nous devons modifier le fichier de configuration de LILO (cas le plus courant). Il vous faudra éditer le fichier /etc/lilo.conf. Vous pouvez remplacer le noyau utilisé par LILO ou en ajouter un. Pour ce faire, placez les lignes suivantes dans le fichier:

```
image=/boot/vmlinuz-2.2.17
  label=linux2
  root=/dev/hda1 #dépend du système
  read-only
```

Enregistrez le fichier puis relancez LILO grâce à la commande... "lilo" ! Il ne vous reste plus qu'à redémarrer l'ordinateur en tapant "shutdown -r now" (ou "reboot").



Modification de LILO pour booter le nouveau kernel.