Compiler le noyau Linux

Gazette Linux n°111 — Février 2005

R. Krishnakumar

Article paru dans le n°111 de la Gazette Linux de février 2005.
Traduction française par Deny <deny CHEZ monaco POINT net>.
Relecture de la traduction française par Joëlle Cornavin <jcornavi CHEZ club TIRET internet POINT fr>.

Table des matières

1. Télécharger le code source du noyau 1
2. Configurer le noyau 2
3. Construire les dépendances 2
4. Créer l'image finale 3
5. Compiler et installer les modules 3
6. Amorcer à partir du nouveau noyau 3
7. Installer manuellement le noyau 3
8. Compilation verbeuse 4
9. Nettoyage des sources du noyau 4
10. Conclusion 4

Cet article servira d'initiation aux nouveaux venus dans le monde du développement Linux qui s'essaient à la compilation du noyau à partir du source. Y seront expliquées les diverses étapes, du téléchargement du source du noyau jusqu'à l'amorçage à partir de la nouvelle image. On trouvera aussi des astuces pour nettoyer le code source, réaliser une compilation verbeuse, etc.

1. Télécharger le code source du noyau

Pour pouvoir compiler un nouveau noyau, il faut en télécharger le code source sur le site www.kernel.org (http://www.kernel.org). Toutes les versions y sont disponibles. Prenons un exemple : supposons que nous voulions compiler la version 2.6.9 du noyau Linux. Nous devons télécharger le code source depuis cette adresse : http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.9.tar.bz2.

Il vaut mieux télécharger la version compressée par bzip, dont le taux de compression est supérieur à celui de son homologue gzip. En conséquence, nous mettrons moins de temps à télécharger. Voici à quoi ressemblera un **wget** sur la ligne de commande :

Une fois le source de la version de noyau requise téléchargé, il faut le décompresser avec **bunzip** et le désarchiver avec **untar** de la façon suivante :

L'option `x` désigne l'e'x'traction de l'archive , `v` signifie « verbeux », `j` spécifie que nous employons la commande **bunzip** avant de désarchiver et `f` indique le nom du fichier en entrée.

Le fichier sera désarchivé dans le répertoire `linux-2.6.9`. Dès que vous avez fini de désarchiver, déplacez-vous avec **cd** dans `linux-2.6.9`.

2. Configurer le noyau

Nous devons configurer le noyau avant de commencer à le compiler. Pendant la phase de configuration, nous sélectionnerons les composants qui feront partie de notre futur noyau. Imaginons par exemple que nous utilisons le système de fichiers ext3. Il nous faudra alors choisir la prise en charge du système de fichiers ext3 au moment de configurer le noyau. En générall, la commande à exécuter est :

Celle-ci invoque l'interface ncurses. D'autres options comme **make xconfig** et **make config** sont disponibles. La première fait apparaître le menu de configuration en mode graphique et la seconde est en mode texte.

Une fois que les composants de notre noyau sont choisis, nous pouvons quitter l'interface de configuration. Reste à sélectionner l'option permettant d'enregistrer la configuation dans le menu de configuration avant de sortir.

Après avoir configuré le noyau comme mentionné ci-dessus, nous trouvons un fichier nommé `.config` dans le répertoire de premier niveau du source : il s'agit du fichier de configuration. Il contient diverses options, ainsi que leurs états (qu'ils soient sélectionnés ou non). Par exemple, si nous choisissons de bénéficier de la prise en charge de PCI dans notre noyau, nous pouvons avoir un élément de la forme :

dans le fichier `.config`. De la même manière, les options que nous avons sélectionnées comme non exigées apparaîtront avec l'argument `not set`. Supposons que nous n'ayons pas sélectionné la prise en charge du système de fichiers XFS dans notre noyau, nous trouverons l'élément suivant dans le fichier `.config` :

Une fonctionnalité très appréciable des noyaux 2.6 est que si nous lançons **make menuconfig** (ou **xconfig** ou **config**) pour la première fois, le menu de configuration affiché est basé sur la configuration de notre noyau actuel. Dans mon cas, j'utilise un système Fedora Core 1 avec un noyau `2.4.22-1.2115.nptl`. Par conséquent, quand j'exécute un **make menuconfig** pour la première fois sur le source, le menu de configuration que j'obtiens contient les options comme indiqué dans `/boot/config-2.4.22-1.2115.nptl`.

3. Construire les dépendances

Cette étape est requise pour les noyaux antérieurs à la série 2.6 (ici, je ne fais référence qu'aux noyaux d'une série stable). Par exemple, avec un noyau 2.4, nous devons construire les dépendances de façon explicite. Saisissons la commande suivante :

Celle-ci crée les dépendances nécessaires. En revanche, pour un noyau 2.6, nous pouvons ignorer cette étape. Les dépendances sont créées automatiquement lors de la création de l'image finale.

4. Créer l'image finale

Il est possible de construire divers types d'images binaires de noyau. Nous pouvons compiler une image de noyau complète ou une version compressée de celle-ci ; habituellement, on choisit la version compressée ou l'image bzImage, que l'on peut créer en saisissant :

Dans les noyaux 2.6, cette étape résoudra également les dépendances et mettra en ½uvre la création d'une image bzimage.

Après la compilation, nous pouvons trouver l'image du noyau dans le chemin `arch/i386/boot/bzImage` dans le cas d'une image destinée à un processeur de type 386 (Pentium®, AMD®, etc.).

5. Compiler et installer les modules

Si, dans la section configuration, nous avons sélectionné des composants devant être intégrés en tant que modules du noyau, il faut maintenant les compiler. Pour compiler les modules, exécutons la commande :

Cette commande compilera les composants (qui sont sélectionnés pour la compilation des modules) sous forme de modules. Dans un noyau 2.4, on obtiendra des fichiers `.o` des composants correspondants. Dans un noyau 2.6 cependant, le fichier de sortie sera un module `.ko`. Par exemple, avec l'option du pilote réseau des cartes Realtek® à compiler à titre de module, nous pouvons, après avoir exécuté un **make modules**, trouver dans `driver/net/` un fichier appelé `8139too.o` dans le cas d'un noyau 2.4 et `8139too.ko` pour un noyau 2.6.

Après la compilation des modules, il est à présent temps de les installer. pour ce faire, exécutez la commande :

en tant que super-utilisateur (root). Cette commande permet d'installer les modules et les autres fichiers nécessaires dans le répertoire `/lib/modules/2.6.9`.

6. Amorcer à partir du nouveau noyau

L'installation des modules terminée, nous pouvons entreprendre une procédure d'installation automatique du noyau pour l'exécutable du noyau. Saisissons simplement :

Cette commande permet de mettre à jour l'image du noyau dans le répertoire `/boot`, d'actualiser le fichier de configuration du chargeur d'amorçage (`lilo.conf` ou `grub.conf`), puis d'effectuer les actions nécessaires pour rendre le nouveau noyau amorçable.

Ensuite, il faut redémarrer le système. Au prochain amorçage de la machine, le menu d'amorçage nous donnera la possibilité de démarrer à partir du nouveau noyau que vous avons construit. Choisissons cette option et voilà !, nous amorçons dans un noyau que nous avons compilé !

7. Installer manuellement le noyau

Au cas où **make install** ne fonctionne pas ou si pour toute autre raison, nous ne pouvons pas effectuer une installation automatique, il faut envisager une installation manuelle du noyau. Par exemple, si nous utilisons le gestionnaire d'amorçage grub, il faut copier la bzImage dans la partition `boot` puis modifier le fichier `/etc/grub.conf` pour répercuter la présence de la nouvelle image. Avec le gestionnaire d'amorçage lilo ,il faut copier bzImage dans l'emplacement réservé à l'amorçage, puis modifier le fichier `lilo.conf` et exécuter ensuite la commande **lilo** pour être sûr que nous ayons la possibilité d'amorcer depuis notre nouvelle image lors de notre prochain démarrage. Voici les étapes à suivre en tant que super-utilisateur si nous faisons appel au chargeur d'amorçage lilo :

Après cette commande, ajoutons la ligne suivante dans `/etc/lilo.conf` :

Puis, lançons lilo :

Redémarrons ensuite la machine. À l'invite de commande de lilo, saisissez **2.6.9-kernel** comme option d'amorçage et nous démarrons avec le noyau personnalisé nouvellement compilé.

8. Compilation verbeuse

Nous constatons que la compilation du noyau est très silencieuse. L'écran affiche très peu d'informations sur ce qui se passe pendant la compilation.

Pour savoir quelles sont les commandes utilisées lors de la compilation, nous devons indiquer l'option de compilation `verbose` comme ci-dessous :

Cette option permet d'afficher le détail des commandes exécutées pendant la compilation. Voici un extrait de la sortie de la compilation :

9. Nettoyage des sources du noyau

Après avoir initié la compilation sur le source, si nous voulons nettoyer les fichiers objet et autres fichiers temporaires, exécutons la commande suivante :

Cette commande supprime la plupart des fichiers générés mais conserve le fichier de configuration.

Pour un nettoyage complet, c'est-à-dire si nous voulons que le source reviennt à l'état dans lequel il était avant de démarrer la compilation, saisissez :

Cette commande supprime tous les fichiers générés, le fichier de configuration ainsi que divers fichiers de sauvegarde. Tous les changements apportés au source seront ainsi annulés. Après cette étape, le source sera rigoureusement identique à ce qu'il était juste après téléchargement et désarchivage.

10. Conclusion

Nous avons vu comment obtenir le source du noyau Linux, le configurer, construire une image et des modules, amorcer à partir du noyau nouvellement compilé et effectuer une compilation verbeuse. Nous avons également examiné comment nettoyer les fichiers temporaires et autres fichiers de configuration

créés pendant la compilation. La prochaine étape dans la construction d'un noyau serait d'en modifier le source et d'essayer de l'expérimenter.

Krishnakumar aime « bidouiller » le noyau Linux. Il travaille pour Hewlett-Packard® et est également titulaire d'une maîtrise de technologie du Collège gouvernemental d'Ingénierie de Thrissur (Inde).