lkn Chapitre 5. Configuration et construction

Table of Contents

Source: Linux Kernel in a Nutshell: 5. Configuring and Building

APrès avoir téléchargé la source de la version de noyau sélectionnée et l'avoir installé dans un répertoire local, on peut construire le code. La première étape consiste à configurer le noyau avec les options appropriées; le noyau peut ensuite être compilé. Les deux tâches sont effectuées via l'utilitaire **make** standard.

Création d'une configuration

La configuration du noyau est conservée dans un fichier appelé `.config` dans le répertoire supérieur de l'arborescence des sources du noyau. Quand on viens de décompresser le code source du noyau, il n'y aura pas de fichier `.config`, il doit donc être créé. Il peut être créé à partir de zéro, créé en le basant sur la «configuration par défaut», tiré d'une version du noyau en cours d'exécution ou extrait d'une version du noyau de distribution. On étudiera les deux premières méthodes ici, et les deux dernières méthodes dans le Chapitre 8, Personnalisation d'un noyau.

Configuration à partir de zéro

La méthode la plus basique de configuration d'un noyau est d'utiliser la méthode **make config**:

```
cd linux-2.6.17.10
make config
```

```
 make config
scripts/kconfig/conf arch/i386/Kconfig
*
* Linux Kernel Configuration
*
*
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (EXPERIMENTAL) [Y/n/?]
Y

*
* General setup
*
Local version - append to kernel release (LOCALVERSION) []


Automatically append version information to the version string
(LOCALVERSION_AUTO) [Y/n/?]
Y

...
```

Le programme de configuration du noyau passera en revue toutes les options de configuration et vous demandera si vous souhaitez activer cette option ou non. En règle générale, les choix pour chaque option sont affichés au format **[Y/m/n/?]** La lettre majuscule est la valeur par défaut et peut être sélectionnée en appuyant simplement sur la touche Entrée. Les quatre choix sont:

y	Construire directement dans le noyau.
n	Ne pas construire.
m	Construire en tant que module, à charger si nécessaire.
?	Imprimer un bref message descriptif et répéter l'invite.

Le noyau contient près de deux mille options de configuration différentes, donc chaque demande individuelle prendra beaucoup de temps. Heureusement, il existe un moyen plus simple de configurer un noyau: baser la configuration sur une configuration pré-construite.

Options de configuration par défaut

Chaque version du noyau est livrée avec une configuration de noyau “par défaut”. Cette configuration est vaguement basée sur les valeurs par défaut que le mainteneur du noyau de cette architecture considère comme les meilleures options à utiliser. Dans certains cas, c'est simplement la configuration qui est utilisée par le mainteneur du noyau lui-même pour ses machines personnelles. Cela est vrai pour l'architecture i386, où la configuration par défaut du noyau correspond étroitement à ce que Linus Torvalds utilise pour sa principale machine de développement.

Pour créer cette configuration par défaut, procéder comme suit:

```
cd linux-2.6.17.10
make defconfig
```

Un grand nombre d'options de configuration défileront rapidement à l'écran et un fichier `.config` sera écrit et placé dans le répertoire du noyau. Le noyau est maintenant correctement configuré, mais il doit être personnalisé pour la machine afin de s'assurer qu'il fonctionnera correctement.

Modification de la configuration

Maintenant que l'on a créé un fichier de configuration de base, il doit être modifié pour prendre en charge le matériel qui est présent dans le système. Pour plus de détails sur la façon de savoir quelles options de configuration il faut sélectionner pour y parvenir, consulter le Chapitre 8, Personnalisation d'un noyau. Ici, on montrera comment sélectionner les options que l'on souhaite modifier.

Il existe trois différents outils de configuration du noyau interactif: un basé sur un terminal appelé **menuconfig**, un graphique basé sur GTK+ appelé **gconfig**, et un graphique basé sur QT appelé **xconfig**.

Méthode de configuration à partir de la console

La méthode **menuconfig** de configuration d'un noyau est un programme basé sur une console qui offre un moyen de se déplacer dans la configuration du noyau à l'aide des touches fléches du clavier. Pour démarrer ce mode de configuration, entrer:

```
make menuconfig
```

Un écran «Écran menuconfig initial» s'affiche.

Les instructions de navigation dans le programme et la signification des différents caractères sont affichées en haut de l'écran. Le reste de l'écran contenant les différentes options de configuration du noyau.

La configuration du noyau est divisée en sections. Chaque section contient des options qui correspondent à un sujet spécifique. Ces sections peuvent contenir des sous-sections pour divers sujets spécialisés. Par exemple, tous les pilotes de périphériques du noyau se trouvent sous l'option du menu principal **Device Drivers**. Pour accéder à ce menu, déplacerz la touche fléche vers le bas neuf fois jusqu'à ce que la ligne **Device Drivers —>** soit mise en surbrillance.

Appuyer ensuite sur la touche Entrée. Cela nous déplacera dans le sous-menu des pilotes de périphérique et l'affichera.

On peut continuer à descendre dans la hiérarchie des menus de la même manière. Pour voir le sous-menu **Generic Driver Options**, appuyer à nouveau sur Entrée pour afficher les trois options disponibles.

Les deux premières options portent une marque `[*]`. Cela signifie que cette option est sélectionnée

(en raison du fait que * est au milieu des caractères []), et que cette option est une option **yes** ou **no**. La troisième option a un marquage <>, montrant que cette option peut être intégrée dans le noyau **Y**, construite en tant que module **M**, ou complètement ignorée **N**.

Si l'option est sélectionnée avec **Y**, les crochets angulaires contiendront un caractère *. S'il est sélectionné comme module avec un **M**, ils contiendront un caractère **M**. S'il est désactivé avec **N**, ils ne montreront qu'un espace vide.

Donc, si on veut modifier ces trois options pour:

1. intégrer diretement au moment de la compilation dans le noyau les pilotes qui n'auront plus besoin de micrologiciel externe, il faut appuyer sur **Y**;
2. désactiver l'option pour empêcher la création du micrologiciel il faut appuyer sur **N**;
3. créer le chargeur de micrologiciel de l'espace utilisateur en tant que module, il faut appuyer sur **M**.

Une fois les modifications apportées, appuyer sur la touche **Échap** ou sur la **flèche droite**, puis sur la touche **Entrée** pour quitter ce sous-menu. Toutes les différentes options du noyau peuvent être explorées de cette manière.

Lorsqu'on a terminé de faire toutes les modifications que l'on souhaite apporter à la configuration du noyau, quitter le programme en appuyant sur la touche **Échap** du menu principal. Un écran, «Enregistrement des options du noyau» s'affiche, demandant si on souhaite enregistrer la configuration du noyau modifiée.

Appuyer sur **Entrée** pour enregistrer la configuration, ou si on souhaite annuler les modifications apportées, appuyer sur la **flèche droite** pour passer à la sélection **<No>**, puis appuyer sur **Entrée**.

Méthodes de configuration graphique

Les méthodes **gconfig** et **xconfig** de configuration d'un noyau utilisent un programme graphique pour permettre de modifier la configuration du noyau. Les deux méthodes sont presque identiques, la seule différence étant les différentes boîtes à outils graphiques avec lesquelles elles sont écrites. **gconfig** est écrit à l'aide de la boîte à outils GTK + et possède un écran à deux volets.

La méthode **xconfig** est écrite à l'aide de la boîte à outils QT et possède un écran à trois volets.

On peut alors utiliser la souris pour parcourir les sous-menus et sélectionner les options.

Dans la méthode **gconfig**, une case cochée signifie que l'option sera intégrée dans le noyau, tandis qu'un **moins** dans la case signifie que l'option sera créée sous forme de module. Dans la méthode xconfig, une option construite en tant que module sera affichée avec un **point** dans la case.

Ces deux méthodes inviteront à enregistrer la configuration modifiée lorsqu'on quitte le programme, et offrent la possibilité d'écrire cette configuration dans un fichier différent. De cette façon, on peut créer plusieurs configurations différentes.

Construire le noyau

Maintenant qu'on a créé une configuration de noyau qui correspond aux besoins, on peut construire le noyau:

```
make

  CHK     include/linux/version.h
  UPD     include/linux/version.h
  SYMLINK include/asm -> include/asm-i386
  SPLIT   include/linux/autoconf.h -> include/config/*
  CC      arch/i386/kernel/asm-offsets.s
  GEN     include/asm-i386/asm-offsets.h
  CC      scripts/mod/empty.o
  HOSTCC  scripts/mod/mk_elfconfig
  MKELF   scripts/mod/elfconfig.h
  HOSTCC  scripts/mod/file2alias.o
  HOSTCC  scripts/mod/modpost.o
  HOSTCC  scripts/mod/sumversion.o
  HOSTLD  scripts/mod/modpost
  HOSTCC  scripts/kallsyms
  HOSTCC  scripts/conmakehash
  HOSTCC  scripts/bin2c
  CC      init/main.o
  CHK     include/linux/compile.h
  UPD     include/linux/compile.h
  CC      init/version.o
  CC      init/do_mounts.o
...
```

En exécutant **make**, le système de construction du noyau utilisera la configuration que sélectionnée pour construire un noyau et tous les modules nécessaires pour prendre en charge cette configuration. [1)] Pendant la construction du noyau, **make** affichera les noms de fichiers individuels de ce qui se passe actuellement, ainsi que tous les avertissements ou erreurs de build.

Si la construction du noyau s'est terminée sans erreur, on a réussi à créer une image du noyau. Cependant, il doit être installé correctement avant d'essayer de démarrer à partir de celui-ci. Voir Chapitre 6, Installation et démarrage à partir d'un noyau pour savoir comment procéder.

Options de construction avancées

Le système de construction du noyau permet de faire bien plus de choses que de simplement construire le noyau et les modules complets. Le chapitre 11, Référence de la ligne de commande de construction du noyau inclut la liste complète des options fournies par le système de construction du noyau. Dans cette section, on va discuter de certaines de ces options de construction avancées. Pour

voir une description complète de la façon d'utiliser d'autres options de build avancées, se reporter à la documentation du noyau sur le système de build, qui se trouve dans le répertoire `Documentation/kbuild/` des sources.

Construire plus rapidement sur des machines multiprocesseurs

Le système de construction du noyau fonctionne très bien en tant que tâche qui peut être divisée en petits morceaux et confiée à différents processeurs. Ce faisant, on peut utiliser toute la puissance d'une machine multiprocesseur et réduire considérablement le temps de génération du noyau.

Pour construire le noyau de manière **multithread**, utilisez l'option **-j** du programme **make**. Il est préférable de donner un numéro à l'option **-j** qui correspond au double du nombre de processeurs dans le système. Donc, pour une machine avec 2 processeurs présents, utiliser:

```
make -j4
```

et pour une machine à quatre processeurs, utilise:

```
make -j8
```

Si on ne transmet pas de valeur numérique à l'option **-j**

```
make -j
```

le système de génération créera un nouveau thread pour chaque sous-répertoire de l'arborescence du noyau, ce qui peut facilement empêcher la machine de répondre et prendre beaucoup plus de temps pour terminer la génération. Pour cette raison, il est recommandé de toujours passer un nombre à l'option **-j**.

Construire seulement une partie du noyau

Lorsqu'on effectue le développement d'un noyau, on souhaitez parfois créer uniquement un sous-répertoire spécifique ou un seul fichier dans toute l'arborescence du noyau. Le système de construction du noyau permet de le faire facilement. Pour créer sélectivement un répertoire spécifique, le spécifier sur la ligne de commande de génération. Par exemple, pour créer les fichiers dans le répertoire drivers `/usb/serial`, entrer:

```
make drivers/usb/serial
```

Cependant, l'utilisation de cette syntaxe ne construira pas les images finales du module dans ce répertoire. Pour ce faire, utiliser l'argument **M=**:

```
make M=drivers/usb/serial
```

qui va construire tous les fichiers nécessaires dans ce répertoire et lier les images finales du module.

Lorsqu'on crée un seul répertoire de l'une des manières indiquées, l'image finale du noyau n'y est pas liée. Par conséquent, toutes les modifications apportées aux sous-répertoires n'affecteront pas l'image finale du noyau, ce qui n'est probablement pas ce que l'on désire. Exécuter une finale:

```
make
```

pour que le système de construction vérifie tous les fichiers d'objets modifiés et fasse correctement le lien final de l'image du noyau.

Pour construire uniquement un fichier spécifique dans l'arborescence du noyau, le passer simplement comme argument à faire. Par exemple, si on souhaite créer uniquement le module du noyau `drivers/usb/serial/visor.ko`, entrer:

```
make drivers/usb/serial/visor.ko
```

Le système de construction créera tous les fichiers nécessaires pour le module du noyau `visor.ko` et fera le lien final pour créer le module.

Source à un endroit, sortie à un autre

Parfois, il est plus facile d'avoir le code source de l'arborescence du noyau dans un emplacement en lecture seule (comme sur un CD-ROM ou dans un système de contrôle de code source) et de placer la sortie de la construction du noyau ailleurs, de sorte qu'on ne dérangera pas l'arbre source d'origine. Le système de génération du noyau gère cela facilement, en ne nécessitant que le seul argument **O=** pour lui dire où placer la sortie de la génération. Par exemple, si la source du noyau se trouve sur un CD-ROM monté sur `/mnt/cdrom/` et que l'on souhaite placer les fichiers créés dans un répertoire local, entrer:

```
cd /mnt/cdrom/linux-2.6.17.11/
make O=~/linux/linux-2.6.17.11
```

Tous les fichiers de construction seront créés dans le répertoire `~/linux/ linux-2.6.17.11/`.

Cette option **O=** doit également être transmise aux options de configuration de la build afin que la configuration soit correctement placée dans le répertoire de sortie et non dans le répertoire contenant le code source.

Cross compilation

Il est très utile de construire le noyau d'une manière croisée pour permettre à une machine plus puissante de construire un noyau pour un système embarqué plus petit, ou tout simplement de vérifier une construction pour une architecture différente pour s'assurer qu'une modification du code source n'a ne pas casser quelque chose d'inattendu. Le système de construction du noyau permet de spécifier une architecture différente du système actuel avec l'argument **ARCH=**. Le système de génération permet également de spécifier le compilateur spécifique que l'on souhaite utiliser pour la génération en utilisant l'argument **CC=** ou une chaîne d'outils de compilation croisée avec l'argument **CROSS_COMPILE**.

Par exemple, pour obtenir la configuration du noyau par défaut de l'architecture **x86_64**, il faut entrer:

```
make ARCH=x86_64 defconfig
```

Pour construire le noyau entier avec une chaîne d'outils ARM située dans `/usr/local/bin/` il faut entrer:

```
make ARCH=arm CROSS_COMPILE=/usr/local/bin/arm-linux-
```

Il est utile même pour un noyau non compilé de manière croisée de changer ce que le système de build utilise pour le compilateur. Des exemples de cela utilisent les programmes **distcc** ou **ccache**, qui aident tous deux à réduire considérablement le temps nécessaire pour construire un noyau. Pour utiliser le programme **ccache** dans le cadre du système de génération, entrer:

```
make CC="ccache gcc"
```

Pour utiliser à la fois **distcc** et **ccache**, entrer:

```
make CC="ccache distcc"
```

[1)]

Les versions de noyau antérieures à la version 2.6 nécessitaient l'étape supplémentaire de créer des modules pour construire tous les modules de noyau nécessaires. Ce n'est plus nécessaire.