Module LP-UE142

Noyau Linux

Pierre Nerzic

IUT de Lannion

1 - Noyau et modules

Le noyau Linux est écrit principalement en C et son source est public. Il évolue assez souvent et on peut le personnaliser en définissant des options (présence ou absence de tel dispositif) et en le recompilant.

L'intérêt est de suivre de près les améliorations et mises à jour du noyau. Faire uname -r : ça donne le numéro de version du noyau, ex 2.6.27-9-generic.

NB : on ne compile le noyau que si on le doit vraiment. C'est une opération potentiellement risquée pour le matériel et pour le système.

Sur Ubuntu, consulter :

https://help.ubuntu.com/community/Kernel/Compile

a) Structure du noyau

Le noyau a deux modes de fonctionnement : le mode privilégié et le mode utilisateur. En mode privilégié, il gère la commutation des processus, la gestion de la mémoire virtuelle et les interruptions de périphériques. En mode utilisateur, il

exécute les appels systèmes des logiciels : créer un fichier, aller dans un répertoire, allouer de la mémoire...

Le noyau contient aussi des parties optionnelles : les modules. Ce sont des librairies de fonctions optionnelles chargées à la demande (dans /lib/modules/2.6.27-9-generic/… extension .ko). En général, ce sont les drivers de périphériques non indispensables lors du boot, ils sont chargés à la demande, selon l'utilisation du système. Par exemple : lirc, le driver de télécommande pour remplacer la souris.

Les modules contiennent aussi deux parties : une partie pour le mode privilégié (routine d'interruption, installation dans le système) et une partie pour le mode utilisateur (réponses aux demandes d'E/S).

Ces modules sont chargés en mémoire et liés au noyau par un mécanisme d'édition des liens statique.

L'ensemble doit être cohérent (construit d'une seule traite) sous peine d'un plantage définitif. En effet, les appels aux fonctions des modules sont représentés par des adresses fixes et non des symboles, voir le fichier /proc/kallsyms. Si

l'adresse est fausse ne serait-ce que d'un octet, le noyau plante sans aucune possibilité de récupération.

<explications sur l'édition des liens statique et dynamique>

b) Modules

Plusieurs commandes permettent de gérer l'édition des liens des modules dans le noyau :

- lsmod : permet de lister l'ensemble des modules présents en mémoire
- modinfo : donne quelques informations sur le module passé en paramètre
- modprobe : permet d'insérer un module en mémoire, ainsi que toutes ses dépendances. Utilise l'autre commande /sbin/insmod.
- rmmod : permet de supprimer un module de la mémoire.
- depmod -a : permet la création d'une liste des modules utilisables avec le noyau (dans /lib/modules/version/modules.dep).

On peut fournir des options à un module lors du chargement. Les options sont listées par modinfo (nom, type et signification). On les fournit par modprobe module option=valeur...

Certains modules à charger au démarrage sont spécifiés dans /etc/modules. Ce sont les modules qui ne seraient pas chargés automatiquement ou qui ne détectent pas le matériel automatiquement.

Au contraire, certains modules sont nuisibles au système : ils font planter la machine, pour différentes raisons : version du module, matériel trop récent... On ne doit donc pas les charger. Il faut mettre leur nom précédé par le mot-clé blacklist dans /etc/modprobe.d/blacklist.

c) Disque RAM initial

On a vu dans un cours précédent, que grub ou lilo charge une image du noyau en mémoire : vmlinuz. C'est un fichier compressé autoextractible qui contient le programme exécutable du noyau. Ce noyau n'est pas très gros, il ne contient pas tous les modules.

En plus du noyau, le boot charge un ramdisk initrd.img qui contient également les modules nécessaires dès le démarrage. En effet, le noyau n'est pas compilé avec

tous les modules. Beaucoup sont optionnels et chargés selon le matériel disponible dans l'ordinateur. Donc, il faut que ces modules soient quand même disponibles, c'est pour ça qu'on les met dans ce ram disk.

On peut configurer ce disque virtuel dans /etc/initramfs-tools/initramfs.conf. Consulter la documentation pour cela, l'opération n'est pas sans risque et c'est en général pour résoudre un problème très spécifique : forcer ou empêcher le chargement d'un module dès le boot.

2 - Compilation du noyau

L'intérêt de cette manipulation : créer un noyau spécifique pour notre système.

- optimisé pour le processeur : SMP, 64bits... et les périphériques
- machine sans disque (diskless) ou machine légère : très petits disques, processeur lent... voir les cartes mères EPIA de VIA.
- architecture matérielle non standard, consulter par exemple OpenWRT

  http://fr.wikipedia.org/wiki/OpenWrt

a) obtention et patch des sources du noyau

En général, on commence par installer les paquets des sources du noyau : fichiers d'entête .h et sources .c dans /usr/src. Il est ultra recommandé de prendre ceux du système actuel qui sont toujours légèrement modifiés par rapport à ceux qu'on peut trouver sur www.kernel.org.

Il est fréquent de devoir patcher les sources, lorsqu'on découvre des bugs sur le système. Selon la doc, ça marche toujours super bien ; la pratique montre que c'est souvent problématique – la moindre différence avec les sources standards fait planter la série de patches. Pour patcher le noyau, il faut obtenir les sources exacts de la précédente version complète, puis appliquer le patch avec la commande suivante :

- **`bzcat patch-`*`versionN+1`*`.bz2 | patch –p0`**

Il faut appliquer les patchs par ordre de version croissant, à partir du premier patch supérieur à la version courante. (patch-2.6.21.bz2 sur un noyau 2.6.20, puis patch-2.6.22, puis patch-2.6.23…). L'option –p0 indique qu'on laisse les chemins inchangés = on doit travailler dans le bon répertoire : /usr/src dans lequel on trouve linux-*version*.

Quoiqu'il en soit, on dispose des sources dans /usr/src/linux-*VERSION*/. Il faut y travailler en tant que root : sudo ou fakeroot.

b) configuration du noyau

Le travail sur le noyau est le même que sur un logiciel C dans le monde Unix : en commande ligne, avec l'utilitaire make et son fichier Makefile. Le Makefile permet de faire plusieurs choses : nettoyer les répertoires (cibles mrproper et clean), lancer un outil de configuration (cibles xconfig ou menuconfig), compiler (cibles bzImage, modules, modules-install) et installer (cible install).

La configuration actuelle est mémorisée dans .config : c'est une liste de couples NOM=<Y|M|N> (en réalité N est noté par un #commentaire). ITEM=Y fait compiler l'item en dur dans le noyau, M sous forme de module et N ne le compile pas. Il est prudent de recopier ce fichier avant de le modifier.

Sur Ubuntu, la configuration est simplifiée :

i) choisir une configuration

L'un des répertoires contient différentes configurations correctes, des variantes pour la distribution (version i386, version i686...). Recopier le fichier souhaité dans .config (écraser .config).

ii) éditer la configuration

Commencer par modifier le fichier Makefile : ligne EXTRAVERSION=-lpgsr par exemple. Si vous ne faites pas cela, votre nouveau noyau écrasera le noyau actuel.

Commande make xconfig (il faut installer libqt3-headers et libqt3-mt-dev).

La fenêtre permet de choisir les options de compilation. Il faut passer un peu de temps pour tout étudier. Attention aux dépendances entre les options. Ne pas toucher à ce qu'on ne connaît pas parfaitement. Il est fréquent d'obtenir un noyau qui ne fonctionne pas à cause d'un choix d'options incorrect – mais rien ne le signale.

c) compilation du noyau

Sur Ubuntu, ce n'est pas compliqué : la commande make-kpkg gère tout. Il suffit de la lancer par **`sudo make-kpkg --initrd buildpackage`** (qui reconstitue tout) ou **`sudo make-kpkg --initrd kernel_image`** (qui ne reconstitue que l'image vmlinuz et le ramdisk) et elle compile et crée plusieurs paquets .deb (dans le répertoire supérieur) dont linux-image.

Sur Redhat ou autre, c'est plus compliqué, plusieurs commandes avec make :

- **`make dep`**
- **`make bzImage`**
- **`make modules`**

Pour construire le Ramdisk, sous redhat :

- **`/sbin/mkinitrd /boot/initrd-`*`version`*`.img `*`version`**

d) installation et test du noyau

Sur Redhat faire make install modules_install et sur Debian, installer les paquets avec dpkg -i image.deb. Cette commande installe aussi le menu de grub.

En principe on doit trouver :

- Le noyau et son ramdisk dans /boot, avec /boot/grub/menu.lst mis à jour
- Les modules dans /lib/modules/*VERSION*/

Configurer /etc/lilo.conf ou /boot/grub/menu.lst pour ce nouveau noyau (attention à son nom) en laissant les autres noyaux par défaut.