COMPILATION D’UN NOYAU LINUX

1. **INTRODUCTION**

Le **noyau Linux** est un [noyau de système d'exploitation](https://fr.wikipedia.org/wiki/Noyau_de_syst%C3%A8me_d%27exploitation) de [type UNIX](https://fr.wikipedia.org/wiki/Type_Unix). Il est utilisé dans plusieurs [systèmes d'exploitation](https://fr.wikipedia.org/wiki/Syst%C3%A8mes_d%27exploitation) dont GNU/Linux et [Android](https://fr.wikipedia.org/wiki/Android). Le noyau Linux est un [logiciel](https://fr.wikipedia.org/wiki/Logiciel) partiellement libre (contenant des [BLOB](https://fr.wikipedia.org/wiki/Binary_large_object) et des modules non-libre - consultez [Linux-libre](https://fr.wikipedia.org/wiki/Linux-libre)) développé essentiellement en [langage C](https://fr.wikipedia.org/wiki/C_(langage)) par des milliers de [bénévoles](https://fr.wikipedia.org/wiki/B%C3%A9n%C3%A9volat) et salariés collaborant sur [Internet](https://fr.wikipedia.org/wiki/Internet).

1. **DÉFINITION**

Le noyau est le cœur du système, c'est lui qui s'occupe de fournir aux [logiciels](https://fr.wikipedia.org/wiki/Logiciel) une [interface de programmation](https://fr.wikipedia.org/wiki/Interface_de_programmation) pour utiliser le matériel. La compilation d’un noyau c’est la traduction du code de ce noyau dans un langage interprétable par le processeur. Elle est une spécificité des systèmes libres, qui n'est possible que parce que l'on dispose des sources du noyau. Cependant, même pour certains Unix commerciaux, il est possible d'effectuer une édition de liens, les modules du noyau étant fournis sous la forme de fichiers objets. La compilation ou l'édition de liens du noyau est une opération technique qui peut surprendre un habitué des systèmes fermés que sont par exemple Windows ou OS/2. Cependant, elle permet d'obtenir un noyau très petit, optimisé pour la machine sur laquelle il tourne, et donc à la fois économe en mémoire et performant.

1. **Informations sur la compilation de noyau linux**

Le noyau Linux a été créé en [1991](https://fr.wikipedia.org/wiki/1991_en_informatique) par [Linus Torvalds](https://fr.wikipedia.org/wiki/Linus_Torvalds) pour les [compatibles PC](https://fr.wikipedia.org/wiki/Compatibles_PC). Initialement conçu pour l'[architecture de processeur](https://fr.wikipedia.org/wiki/Architecture_de_processeur) [x86](https://fr.wikipedia.org/wiki/X86), il a ensuite été [porté](https://fr.wikipedia.org/wiki/Portage_informatique) sur de nombreuses autres, dont [m68k](https://fr.wikipedia.org/wiki/Famille_Motorola_68000), [PowerPC](https://fr.wikipedia.org/wiki/PowerPC), [ARM](https://fr.wikipedia.org/wiki/Architecture_ARM), [SPARC](https://fr.wikipedia.org/wiki/Architecture_SPARC), [MIPS](https://fr.wikipedia.org/wiki/Architecture_MIPS) et [RISC-V](https://fr.wikipedia.org/wiki/RISC-V). Il s'utilise dans une très large gamme de matériel, des [systèmes embarqués](https://fr.wikipedia.org/wiki/Syst%C3%A8me_embarqu%C3%A9) aux [superordinateurs](https://fr.wikipedia.org/wiki/Superordinateur), en passant par les [téléphones mobiles](https://fr.wikipedia.org/wiki/T%C3%A9l%C3%A9phone_mobile) et [ordinateurs personnels](https://fr.wikipedia.org/wiki/Ordinateur_personnel).

La compilation du noyau de Linux nécessite de disposer des sources du noyau fourni par votre distribution. En effet, la plupart des distributions modifient ces sources pour y ajouter des fonctionnalités ou des correctifs de sécurité, ce qui permet de ne pas prendre le risque d'utiliser la dernière version officielle du noyau, qui peut ne pas être considérée comme stable car trop récente. Parfois même, ce sont des fonctionnalités non encore incluses dans le noyau officiel qui sont utilisées par les distributions, ce qui rend l'utilisation du noyau générique impossible. Rien ne vous empêche d'utiliser pour autant la dernière version officielle du noyau, mais vous vous exposerez au risque d'avoir un système non fonctionnel, ou au moins à diagnostiquer des problèmes encore inconnus ou d'avoir à mettre à jour d'autres composants de votre distribution manuellement en raison des évolutions que ce noyau peut avoir subi.

Le noyau utilise des constructions spécifiques à GCC (**GNU Compiler Collection)** et ne peut donc être compilé de manière fiable que par ce compilateur (même si d'autres compilateurs compatibles avec GCC sont utilisables moyennant quelques efforts).

**Note :** La documentation du noyau indique que la version de gcc à utiliser pour le compiler est la version 3.2. La raison en est que les développeurs du noyau eux-mêmes changent rarement de compilateur, afin de ne pas avoir à se préoccuper d'éventuels bugs de celui-ci et ainsi de pouvoir se concentrer sur leur travail. Toutefois, les versions plus récentes de GCC sont également tout à fait valables de manière générale, et parfois même elles sont nécessaires pour les architectures les plus récentes. Ainsi, les architectures 64 bits requièrent au minimum GCC 3.4...

1. **Avantages et Inconvénients**
2. **Avantages**

L'un des principaux avantages des programmes compilés est qu'ils sont des unités autonomes qui sont prêtes à être exécutées. Comme ils sont déjà compilés dans des binaires en langage machine, il n'y a pas de deuxième application ou package que l'utilisateur doit maintenir à jour. Si un programme est compilé pour Windows sur une architecture x86, l'utilisateur final n'a besoin que d'un système d'exploitation Windows fonctionnant sur une architecture x86. De plus, un package précompilé peut s'exécuter plus rapidement qu'un interpréteur compilant le code source en temps réel.

Bien que le fait d'être verrouillé dans un package matériel spécifique présente des inconvénients, la compilation d'un programme peut également augmenter ses performances. Les utilisateurs peuvent envoyer des options spécifiques aux compilateurs concernant les détails du matériel sur lequel le programme sera exécuté. Cela permet au compilateur de créer du code en langage machine qui utilise le plus efficacement possible le matériel spécifié, par opposition à un code plus générique. Cela permet également aux utilisateurs avancés d'optimiser les performances d'un programme sur leurs ordinateurs.

1. **Inconvénients**

Étant donné qu'un compilateur traduit le code source dans un langage machine spécifique, les programmes doivent être spécifiquement compilés pour OS X, Windows ou Linux, ainsi que spécifiquement pour les architectures 32 bits ou 64 bits. Pour un programmeur ou un éditeur de logiciels essayant de diffuser un produit auprès du public le plus large possible, cela signifie conserver plusieurs versions du code source pour la même application. Cela entraîne plus de temps consacré à la maintenance du code source et des problèmes supplémentaires lorsque les mises à jour sont publiées.

L'un des inconvénients d'avoir un compilateur est qu'il doit en fait compiler le code source. Alors que les petits programmes codés par de nombreux programmeurs novices prennent un temps insignifiant à compiler, les suites d'applications plus importantes peuvent prendre un temps considérable à compiler. Lorsque les programmeurs n'ont rien d'autre à faire que d'attendre la fin du compilateur, ce temps peut s'additionner, en particulier pendant la phase de développement, lorsque le code doit être compilé pour tester les fonctionnalités et résoudre les problèmes.

1. **Applications**

La compilation du noyau n'est pas très difficile. Elle nécessite cependant de répondre correctement aux questions de configuration. Les erreurs peuvent être multiples, et souvent fatales. Il est donc fortement conseillé de disposer d'une disquette de démarrage afin de réparer le système en cas d'erreur. Par ailleurs, il faut toujours conserver le dernier noyau utilisable en sauvegarde dans le répertoire /boot/. Il faut également ajouter une entrée spécifiant ce noyau dans le programme de démarrage (**lilo**), afin de pouvoir sélectionner l'ancien noyau en cas d'erreur. Ces opérations seront décrites en détail plus loin.

1. **Les types de compilations**

Il existe deux (02) types de compilation du noyau

**a. Compilation sans modules externes**

Par compilation sans modules externes, on entend la compilation d'un noyau sans sources externes aux sources officielles du noyau Linux (i.e. sans sources autres que celles fournies dans l’archive Linux-2.6.16.tar.bz2.

Maintenant, vous allez donner un nom à votre noyau.

Par exemple, vous pouvez adopter la convention suivante : kernel-image-version\_du\_noyau-nom\_de\_la\_machine\_numéro\_de\_révision, ce qui donne par exemple kernel-image-2.6.16.2-camel\_1. Cette notation donnera un paquet Debian nommé kernel-image-2.6.16.2-camel\_1\_i386.deb.

Il faut alors s'armer de patience, surtout si vous avez un ordinateur peu puissant ! Sur un ordinateur récent, cela prend plusieurs minutes.

**Exemple de compilation du noyau Linux :**

|  |
| --- |
| **% cd ~/kernel/linux**  **% fakeroot make-kpkg --append-to-version -camel --revision 1 kernel-image** |

**b. Compilation avec modules externes**

Pour des fonctionnalités très particulières ou des drivers de matériels peu répandus, vous aurez peut-être besoin de compiler votre noyau avec des modules externes. On aura alors à ajouter une option à make-kpkg.

On commence par installer les modules externes dans ~/kernel/modules/.

**Exemple d'installation des modules de SHFS** (qui permettent de monter une partition d'une autre machine via SSH)

|  |
| --- |
| **# apt-get install shfs-source** |

|  |
| --- |
| **% cd ~/kernel/**  **% tar jxvf /usr/src/shfs.tar.bz2**  **% export MODULE\_LOC=~/kernel/modules**  **% cd linux/**  **% fakeroot make-kpkg --append-to-version -camel --revision 1 kernel-image**  **% fakeroot make-kpkg --append-to-version -camel modules-image** |

La compilation du noyau se passe en quatre étapes :

* Installation des fichiers sources ;
* Réponse aux questions de configuration ;
* Compilation et installation du noyau ;
* Compilation et installation des modules.

## Installation des sources de Linux

Les sources du noyau sont normalement fournies par votre distribution. Vous pouvez toutefois trouver les sources génériques du noyau « officiel » sur le site [kernel.org](http://www.kernel.org/). Si l'on dispose déjà d'une version complète des fichiers sources, il est envisageable de ne télécharger que les fichiers différentiels de cette version à la version courante (ce que l'on appelle classiquement des « patches »).

Il est recommandé d'installer les sources du noyau dans un autre répertoire que celui où se trouvent les fichiers sources de votre distribution, car ceux-ci contiennent les fichiers d'en-tête C nécessaires à la compilation des programmes. Le remplacement des fichiers sources du noyau imposerait donc, en toute rigueur, de recompiler la bibliothèque C du système. Cette opération comporte des risques.

Généralement, les fichiers sources de Linux sont installés dans le répertoire /usr/src/linux/. Certaines distributions font une copie des fichiers d'en-tête du noyau qui ont servi pour la génération de la bibliothèque C dans les répertoires /usr/include/linux/, /usr/include/asm/ /usr/include/asm-generic. Si ce n'est pas le cas, il faudra éviter d'écraser le contenu du répertoire /usr/src/linux/ lors de l'installation des sources du nouveau noyau. Une solution est par exemple d'installer les fichiers du noyau dans le répertoire /usr/src/linux<version>/ et de ne pas toucher au répertoire des sources originels /usr/src/linux/. Les commandes suivantes permettront d'extraire les sources dans le répertoire dédié au sources de Linux :

cd /usr/src

tar xvfz linux-2.6.13.2.tar.gz

Si l'on dispose déjà d'une version complète des fichiers sources, et que l'on désire appliquer un patch, il faut décompresser le fichier de patch avec la commande suivante :

gunzip fichier.gz

où fichier.gz représente le fichier de patch compressé (en supposant qu'il ait été compressé à l'aide de gzip). L'application du patch se fait de la manière suivante :

patch -p1 < fichier

Cette commande doit être lancée à partir du répertoire des sources du noyau (par exemple /usr/src/linux-2.6.13.2/). Dans cette ligne de commande, fichier représente le nom du fichier de patch précédemment décompressé, et l'option -p1 indique au programme patch d'utiliser les noms de répertoires relatifs au répertoire parent (à savoir le répertoire contenant le répertoire des sources du noyau, donc /usr/src/ dans notre cas). Si le patch doit être appliqué depuis un autre répertoire, il faudra éventuellement modifier l'option -px passée en paramètre au programme patch, où x est le nombre de niveaux de répertoires à ignorer pour l'application du patch. Consultez la page de manuel patch pour plus de détails sur cette option.

## Choix des options de configuration du noyau

La configuration du noyau peut se faire à l'ancienne avec la commande suivante :

make config

Cette commande pose une série de questions auxquelles il faut pouvoir répondre correctement du premier coup. On n'a pas le droit à l'erreur ici, faute de quoi il faut tout reprendre à zéro.Il est nettement préférable d'utiliser la version texte, qui fournit les options de configuration sous la forme de menus. Cela peut être réalisé avec la commande suivante :make menuconfig

Certains préféreront l'une des versions X11 du programme de configuration, que l'on peut obtenir avec les commandes make xconfig ou make gconfig

Sachez cependant que certaines options ne sont toutefois pas correctement proposées dans ce mode, et que la version textuelle du programme de configuration reste recommandée.

Quelle que soit la méthode utilisée, il faut répondre par 'Y' (pour « Yes »), 'N' (pour « No ») ou 'M' (pour « Module ») lorsque c'est possible. 'Y' et 'M' incluent la fonctionnalité courante dans le noyau, 'N' la supprime. 'M' permet d'utiliser la fonctionnalité en tant que module du noyau. En général, l'utilisation des modules permet d'alléger le noyau car les fonctionnalités sont chargées et déchargées dynamiquement. Cependant, les fonctionnalités nécessaires au démarrage de Linux, comme les gestionnaires de disques et systèmes de fichiers par exemple, ne doivent en aucun cas être placées dans des modules, car le système ne pourrait alors pas démarrer.

## Compilation et installation du noyau

Une fois la configuration du noyau réalisée, la compilation peut être lancée. Pour cela, il suffit de lancer la simple commande make dans le répertoire /usr/src/linux.

Une fois la compilation achevée, il faut installer le nouveau noyau. En pratique, cela revient à conserver la version originale du noyau installé par votre distribution. Pour cela, il faut en faire une copie de sauvegarde.

Il faut également indiquer au gestionnaire d'amorçage qu'il faut qu'il donne maintenant la possibilité de démarrer l'ancienne version du noyau sous ce nouveau nom. Pour LILO, il suffit d'éditer le fichier /etc/lilo.conf et d'y ajouter une nouvelle configuration. En pratique, cela revient à dupliquer la configuration du noyau actuel et à changer simplement le nom du noyau à charger (paramètre « image » de la configuration dans /etc/lilo.conf) et le nom de la configuration (paramètre « label »). Vous devrez aussi rajouter l'option « prompt » si elle n'y est pas déjà, afin que LILO vous demande la configuration à lancer à chaque démarrage. Dans notre exemple, le nom du noyau à utiliser pour la configuration de sauvegarde sera vmlinuz.old. De même, si la configuration initiale de Linux porte le nom « linux », vous pouvez utiliser le nom « oldlinux » pour la configuration de sauvegarde.

Une fois le fichier lilo.conf mis à jour, il faut vérifier que l'on peut bien charger l'ancien système. Pour cela, il faut réinstaller LILO et redémarrer la machine. La réinstallation de LILO se fait exactement de la même manière que son installation, simplement en l'invoquant en ligne de commande : lilo

Si LILO signale une erreur, vous devez corriger immédiatement votre fichier lilo.conf et le réinstaller.

Le gestionnaire d'amorçage utilisé vous propose alors de choisir le système d'exploitation à lancer. Il faut ici sélectionner la configuration de sauvegarde pour vérifier qu'elle est accessible et fonctionne bien. Le système doit alors démarrer en utilisant la copie sauvegardée du noyau. Si cela ne fonctionne pas, on peut toujours utiliser le noyau actuel en sélectionnant le noyau initial et en corrigeant la configuration du gestionnaire d'amorçage.

Lorsque vous vous serez assurer que le système peut démarrer avec la sauvegarde du noyau, vous pourrez installer le nouveau noyau. Son image a été créée par make dans le répertoire /usr/src/linux/arch/i386/boot/, sous le nom bzImage. L'installation se fait donc simplement par une copie dans /boot/ en écrasant le noyau actuel vmlinuz :

cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz

Il faut également copier le fichier System.map du répertoire /usr/src/linux/ dans le répertoire /boot/ : cp System.map /boot

Ce fichier contient la liste de tous les symboles du nouveau noyau, il est utilisé par quelques utilitaires systèmes.

Si vous utiliser LILO, il vous faudra le réinstaller à nouveau pour qu'il prennent en compte le nouveau noyau. Cela se fait avec la même commande lilo

Cette opération n'est en revanche pas nécessaire avec le GRUB.

Encore une fois, il faut redémarrer la machine avec la commande reboot et vérifier que le nouveau noyau fonctionne bien. S'il ne se charge pas correctement, c'est que les options de configuration choisies ne sont pas correctes. Il faut donc utiliser le noyau sauvegardé, vérifier ses choix et tout recommencer. Attention cependant, cette fois, il ne faut pas recommencer la sauvegarde du noyau puisque cette opération écraserait le bon noyau avec un noyau défectueux.

Si le nouveau noyau démarre correctement, il ne reste plus qu'à installer les modules.

## Compilation et installation des modules

Si le système redémarre correctement, on peut installer les modules. Il n'est pas nécessaire de prendre les mêmes précautions pour les modules que pour le noyau. Il suffit donc ici de lancer la commande suivante dans le répertoire /usr/src/linux/ :

make modules\_install

Au préalable, il est recommandé de décharger tous les modules présents en mémoire. Cela peut être réalisé à l'aide de la commande modprobe et de son option -r.

Les modules sont installés dans le répertoire /lib/module/version/, où version est le numéro de version du noyau courant. Il est possible que des modules d'autres versions du noyau existent dans leurs répertoires respectifs. Si vous n'en avez plus besoin, vous pouvez les effacer. Attention cependant si vous avez installé des modules additionnels non fournis avec le noyau dans ces répertoires, vous pourriez encore en avoir besoin.

Comme on l'a déjà vu, les modules sont utilisés par le chargeur de module du noyau, grâce à la commande modprobe. Cette commande a besoin de connaître les dépendances entre les modules afin de les charger dans le bon ordre. Il faut donc impérativement mettre à jour le fichier /lib/modules/version/modules.dep à chaque fois que l'on installe les modules, à l'aide de la commande suivante : depmod -a

