Les modules

par Romain Guy (Gfx)

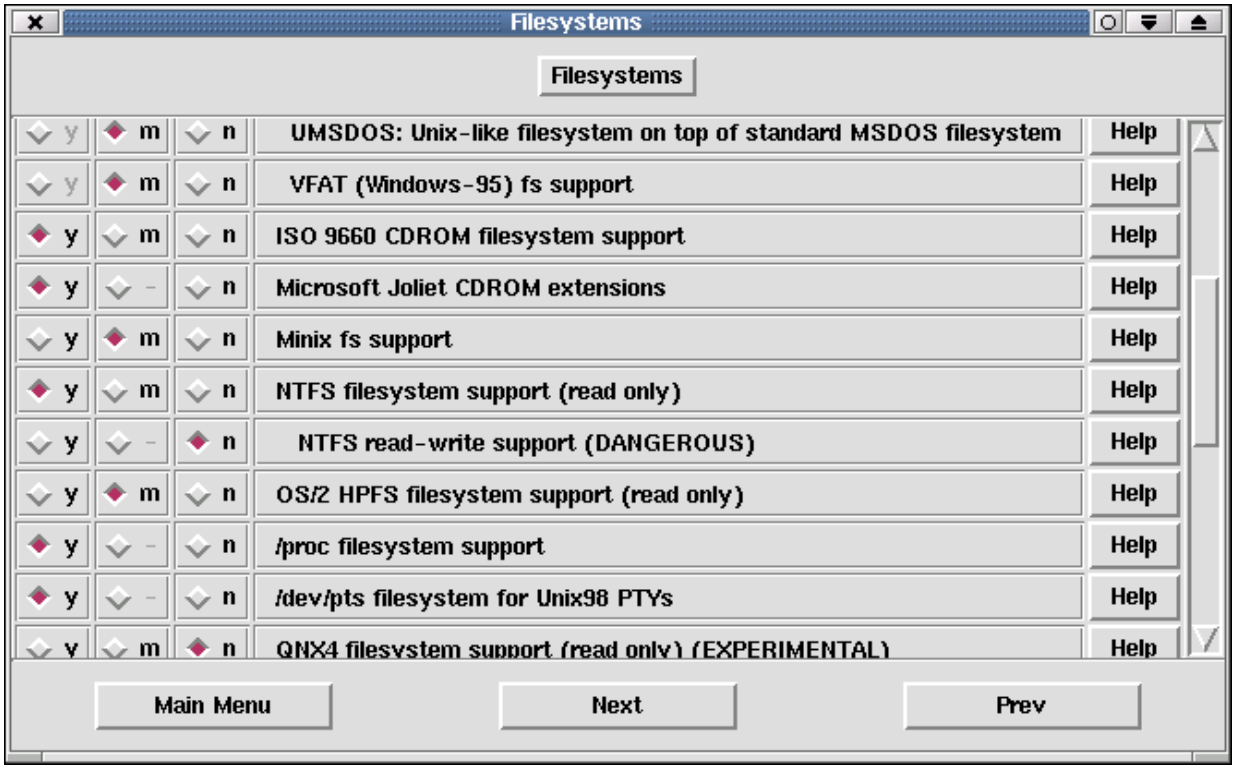
Date de publication : 6 Juin 2006

Dernière mise à jour :

La gestion des périphériques sous Linux, comme sous tout autre système d'exploitation, est réalisée à l'aide pilotes. Ceux-ci peuvent se voir intégrer au noyau statiquement ou sous forme de modules.

I - Introduction

Par définition, le noyau au c#ur du système d'exploitation venu du froid est un noyau monolithique. A l'instar du grand monolithe noir de (((italique)))2001, Odyssée de l'Espace(((fin italique))), le noyau de Linux est un logiciel comprenant tous les éléments garantissant l'accès aux structures de données et aux fonctions du système. A l'opposé des structures monolithiques se trouvent les systèmes d'exploitations basés sur des noyaux minimalistes (on parle souvent de micro-noyaux, comme Neutrino qui est au c#ur de QNX). Ce type de noyaux ne comprend que le strict nécessaire. L'ensemble des autres fonctions disponibles est géré par des processus externes au noyau, dont l'exécution ne se voit décidée qu'en fonction des besoins de l'instant. Les structures minimalistes facilitent énormément l'adjonction de nouvelles fonctions au noyau. On peut mettre ce type de gestion en parallèle avec les applications possédant des plugins.



L'option 'm' désigne la compilation des modules

Ainsi, un noyau monolithique comme celui de notre cher Linux rend complexe l'ajout de nouveaux composants systèmes. Théoriquement, chaque extension matérielle additive devrait impliquer la création et la compilation d'un nouveau noyau. Malheureusement, l'architecture de Linux rendait fort difficile la modification de cette caractéristique. C'est pour cela qu'une voie intermédiaire fut explorée: les modules.

Les modules sont des programmes chargés et liés dynamiquement, c'est à dire au cours de l'exécution, au noyau. Les modules peuvent être également retirés de la mémoire lorsque le noyau n'a plus besoin de faire appel à leurs services.

II - Pilotes modulaires ou pilotes statiques ?

Lors de la compilation du noyau, de nombreuses options vous sont offertes. Et une grande partie des fonctionnalités proposées peuvent se voir intégrées de manière statique ou modulaire au noyau. Nous allons explorer les avantages et les inconvénients de l'utilisation poussée des modules. Tout d'abord, le temps de chargement d'un noyau fortement modulaire se voit très diminué, la plupart des pilotes n'étant lancés que lorsqu'ils sont nécessaires. De plus, un noyau modulaire est plus générique, car moins lié au matériel. Il sera donc possible de faire migrer facilement un tel noyau d'une machine à l'autre. Enfin, les développeurs de pilotes peuvent plus facilement tester leurs travaux puisqu'ils peuvent à loisir démarrer ou arrêter les pilotes.

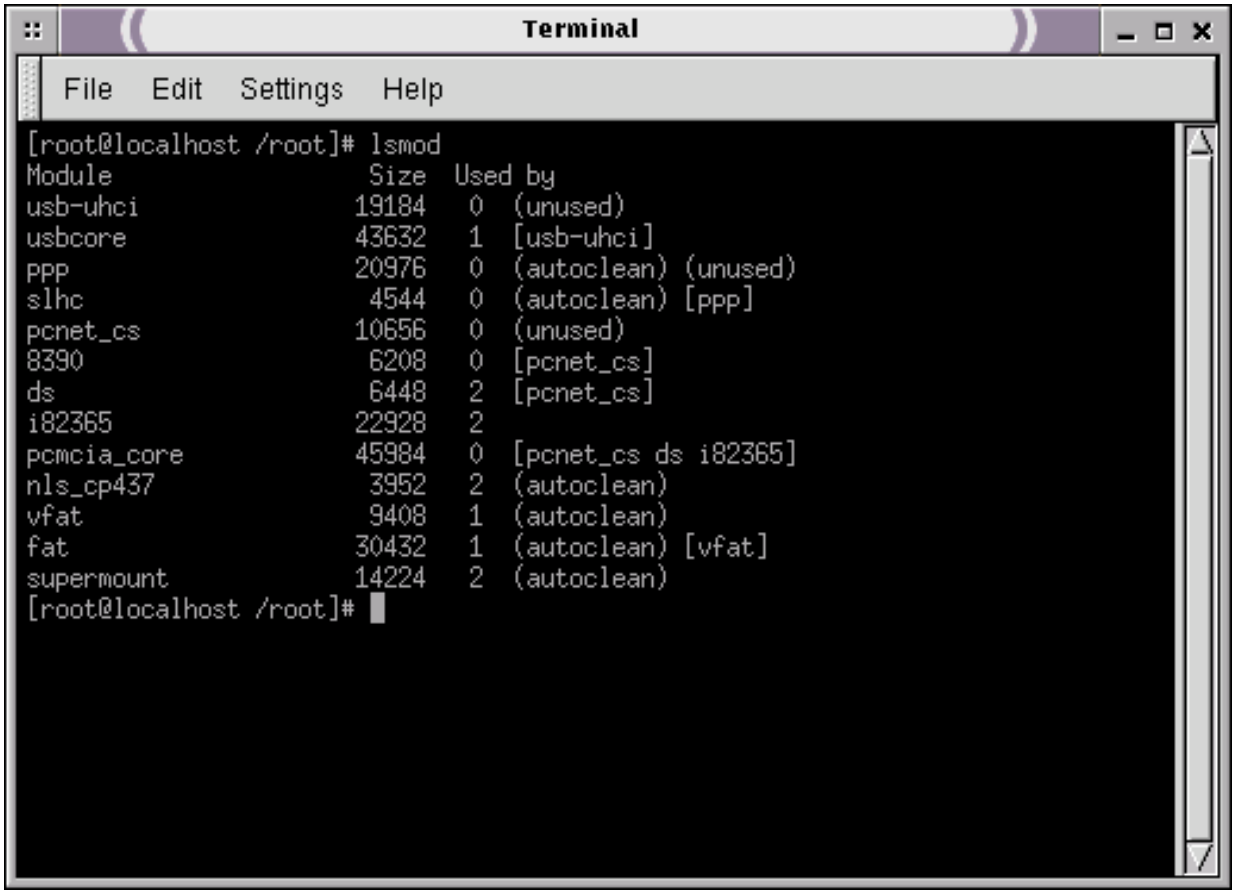
Cependant, l'emploi de modules n'a pas exclusivement des avantages. La gestion de structure de données supplémentaires engendrées par la mise en place des modules entraîne une augmentation sensible de la consommation mémoire. En outre, les modules pénalisent les performances des pilotes car le noyau ne peut plus s'adresser directement à eux. Il doit au préalable passer par des tableaux d'adressage avant d'atteindre le code du pilote.

III - Faites la chaîne

Une particularité fort intéressante des modules est incarnée par le chaînage de modules, ou inter-dépendance modulaire. Prenons l'exemple du module gérant le système de fichiers VFAT. Ce module nécessite pour fonctionner d'avoir accès au module de système de fichiers FAT. Si le chaînage des modules peut compliquer le traitement des informations, cette technique apporte une souplesse supplémentaire aux développeurs du système. L'opportunité d'étendre les capacités d'une fonction est très importante. Pour autant, les dépendances entre modules n'interdisent pas toutes les combinaisons possibles d'intégration au noyau. On pourra par exemple avoir la gestion VFAT en tant que module et la gestion FAT compilée statiquement dans le noyau, ou bien les deux gestionnaires compilés statiquement. La seule interdiction existante est l'intégration du noyau concerne l'intégration du pilote VFAT au sein du noyau et du pilote FAT en tant que module.

IV - Utiliser les modules

Quatre commandes principales existent pour utiliser les modules: insmod, rmmod, lsmod et modprobe. La première d'entre elles, "insmod", sert à insérer un module dans le noyau. L'exécutable "rmmod" permet de retirer un module du noyau et "lsmod" d'afficher une liste de tous les modules chargés en mémoire. Enfin, la commande "modprobe", la plus puissante, regroupe "insmod" et "rmmod". "modprobe" va même plus loin en gérant les chaînes de modules lors des chargements. Par exemple, si vous désirez charger le module VFAT, pour reprendre l'exemple précédent, "modprobe" va au préalable démarrer le module FAT.



Liste des modules chargés

Tous les modules installés sur votre système sont présents dans le répertoire /lib/modules. Ou plus exactement dans le répertoire désigné par le numéro de version de votre noyau, placé dans le répertoire /lib/modules. Le répertoire correspondant à votre noyau contient à son tour des sous-répertoires alloués aux divers modules. L'emplacement des modules étant fixe, et ce quelle que soit votre distribution de Linux, l'ajout d'un module ne dépend absolument pas de votre position dans l'arborescence. Prenons l'exemple d'un ordinateur portable disposant d'une carte son Cirrus Logic CS4281. Le module correspondant dans le noyau 2.2.17 se nomme "cs4281". L'installation du support sonore dans un tel environnement pourra être réalisée de la sorte:

```
insmod cs4281
```

On pourrait, pour plus de sûreté, employer modprobe:

```
modprobe cs4281
```

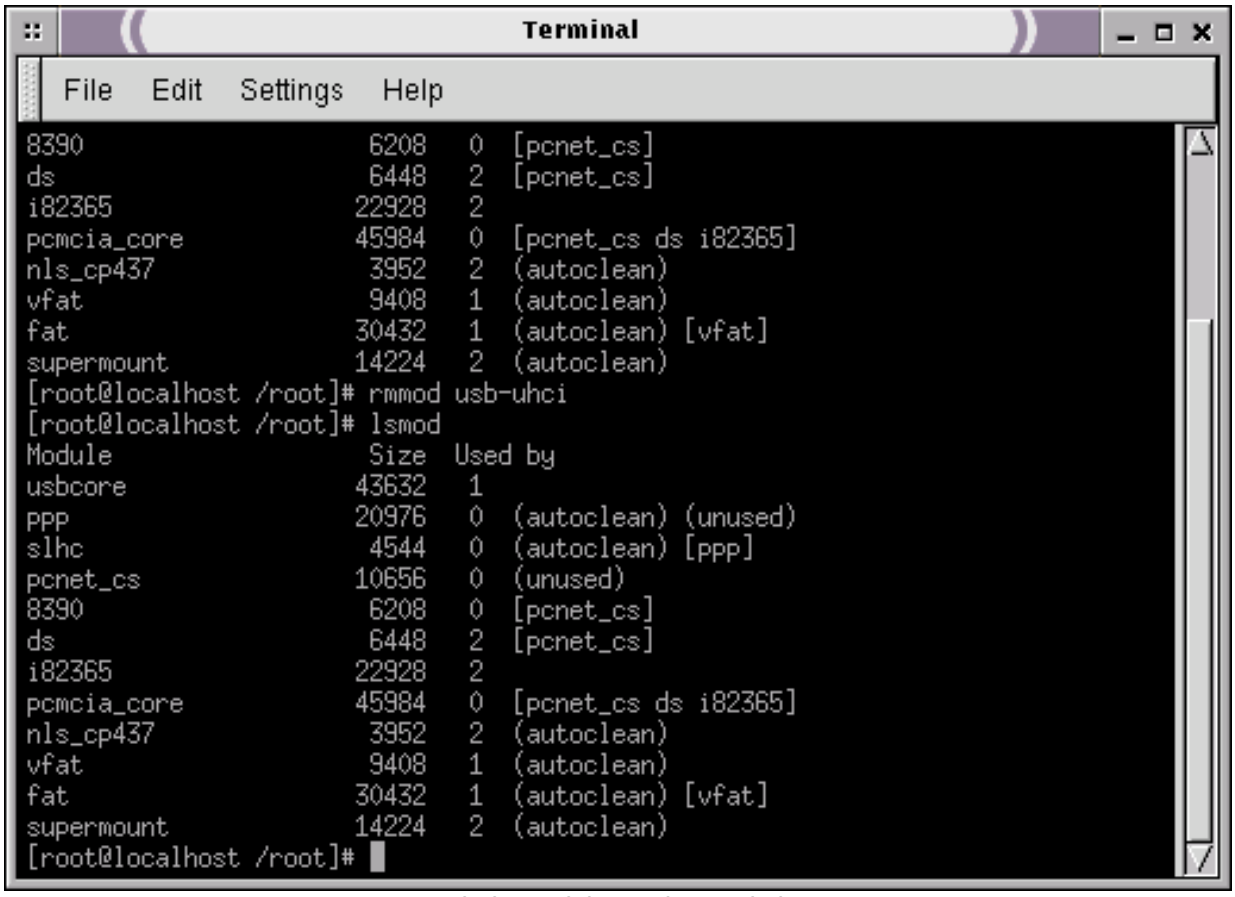
A l'issu de cette commande, le module cs4281 aura pris place dans le noyau. Pour retirer le module du noyau, il vous suffira d'invoquer la commande:

```
rmmod cs4281
```

De même à l'aide de modprobe:

```
modprobe -r cs4281
```

Notez toutefois qu'un module ne peut être déchargé tant qu'il est exploité par d'autres processus. Ceci peut être vérifié à l'aide de "lsmod". L'exécution de cette commande indique le nombre de références attachées au module. Lorsque ce nombre est à zéro, il est alors possible de le décharger. Par ailleurs, la commande "rmmod -a" retire de la mémoire tous les modules dont les compteurs de références sont à zéro.



Retrait du module en charge de l'USB

V - Buffy contre les daemons

L'intérêt des modules serait finalement assez limité s'il n'existait aucun moyen d'automatiser leur chargement et leur déchargement. La solution se nomme "kerneld". "kerneld" est un démon, plus clairement un programme s'exécutant en tâche de fond, informé par le noyau des besoins en modules du système. En effet, dès que le noyau détermine qu'un module est nécessaire (lors du montage d'une partition NFS par exemple), le démon kerneld en est informé. Le chargement du module de gestion NFS sera alors réalisé. Dans une optique similaire, kerneld est apte à décharger les modules. Le retrait d'un module peut s'effectuer de deux manières. Premièrement, un module peut être en mode "autoclean", auquel cas le démon les décharge dès son réveil. La seconde méthode employée par kerneld est la vérification du temps d'inactivité (ou idle). Lorsque le temps imparti est écoulé, kerneld part en chasse et supprime tous les modules inemployés. Si kerneld est très pratique, son utilisation conjointe avec la commande modprobe ou insmod peut poser problèmes: kerneld ne déchargera jamais un module qu'il n'a pas lui-même ajouté au noyau.

Les modules apportent confort et souplesse à l'administration d'un système Linux. Même si les modules ne résolvent pas tous les problèmes, et même si un noyau purement monolithique peut améliorer les performances globales, leur utilisation est vivement indiquée. Ne manquez pas de consulter les man concernant les diverses commandes liées aux modules, vous y trouverez de multiples informations pratiques.