Noyau Linux, modules, et initrd

Ce document regroupe un ensemble de connaissances et de constatations provenant d'expérimentations, relatifs au chargement du noyau Linux, à l'utilisation des modules, et à l'activation ou la détection de certains matériels.

I – Explorer le fichier Initrd

Le fichier **initrd** (système racine initial en RAM, traduit soit ***Init Root Device*** (système racine initial), soit ***Init RAM Disk*** (disque virtuel initial)) est un simple périphérique **loop**, contenant un système de fichier EXT2 et compressé avec **gzip -9**. Pour explorer un système racine initial il faut :

1 – renommer le fichier **initrd** avec une extension **.gz** :

```
mv initrd-2.4.20-28.8 initrd.gz
```

2 – décompresser le fichier **intrd.gz** afin d'obtenir le fichier **initrd** initial :

```
gzip –d initrd.gz
```

3 – mounter le fichier **initrd** en tant que périphérique **loop** (le système de fichier EXT2 est détecté automatiquement) :

```
mount –o loop initrd /mnt/initrd
```

On obtient alors le système racine initial dans le répertoire **/mnt/initrd**. A la racine de cette arborescence se trouve le fichier **linuxrc** : il s'agit du premier script shell exécuté par le noyau. Ce script charge les pilotes primitifs indispensables (accès au système de fichier, aux disques SCSI, à la carte réseau, etc.) se trouvant sous forme de modules dans le répertoire **/lib** de l'arborescence **initrd**. Pour plus d'informations sur la manière de créer un fichier **initrd**, on pourra aller voir le fichier **Documentation/initrd.txt** dans les sources du noyau.

La séquence d'amorçage de Linux utilisant un disque virtuel initial **initrd** est la suivante :

1 – Le noyau est chargé en mémoire, ceci est effectué par le chargeur de démarrage (Grub, LILO, LOADLIN, SysLinux, ISOLinux, PXELinux, etc.). On peut voir le message **Loading...** pendant que ceci arrive.
2 – L'image **initrd** du disque virtuel est chargée en mémoire, à nouveau ceci est réalisé par le chargeur de démarrage. On peut voir le message **Loading** à nouveau quand ceci arrive.
3 – Le noyau est initialisé, y compris la lecture des options de ligne de commande et le montage du disque virtuel en tant que racine.
4 – Le programme **/linuxrc** est lancé sur le disque virtuel initial.
5 – Grâce à **pivot_root** le périphérique racine est changé pour celui spécifié dans les paramètres du noyau.
6 – Le programme **/etc/init** est lancé, et va exécuter la séquence de démarrage paramétrable par l'utilisateur.

II – Pilote NE2000 pour le carte réseau ISA RealTek 8019

Faut-il forcément recompiler le noyau pour disposer du pilote NE2000 ? Le cas concret où le besoin d'intégrer un pilote dans le noyau concerne la carte réseau ISA RealTek 8019 sur l'ordinateur Opsi. Cette carte Plug & Play à l'origine, est accessible par le pilote NE2000 de Linux (module **ne**). Or, la norme NE2000 est justement incompatible avec le Plug & Play. Il en résulte l'obligation de désactiver la fonction Plug & Play de la carte RTL 8019 en utilisant l'utilitaire DOS livré avec la carte, et la nécessité de renseigner le module **ne** sur l'adresse d'E/S et l'IRQ utilisées par la carte. La ligne suivante charge manuellement le module **ne** en indiquant les 2 paramètres de la carte :

```
modprobe ne io=0x280 irq=9
```

Pour automatiser le chargement et le paramétrage du pilote ne sur l'ordinateur Opsi, il faut ajouter les 2 lignes suivantes dans le fichier **/etc/modules.conf** :

```
alias eth0 ne
options ne io=0x280 irq=9
```

Malheureusement ce principe ne permet pas d'installer Linux par le réseau sur Opsi, puisque la carte réseau RealTek 8019 ne sera reconnue qu'une fois que le système racine réel sera correctement chargé. Pour activer la carte ISA RTL 8019 dès le chargement du noyau et de l'initrd, il y a 2 solutions :

- Soit intégrer le pilote NE2000 « en dur » dans le noyau Linux
- Soit enrichir le fichier **initrd** en y ajoutant le module **ne** (qui doit se trouver sur la disquette **drvnet.img**)

III – Intégration de pilotes réseau dans le noyau

Pour intégrer le pilote NE2000 dans le noyau il faut le recompiler en choisissant **NE2000/NE1000 support** dans **Network device support** → **Ethernet (10 or 100Mbit)** (programme ***make menuconfig***).

Une fois recompilé, on dispose bien d'un noyau possédant le pilote NE2000 permettant d'accéder à la carte ISA RealTek 8019, mais comment lui indiquer manuellement l'adresse d'E/S et l'IRQ de la carte ? Pour cela on va utiliser le paramètre **ether** à passer au noyau dès son chargement. Le format du paramètre **ether** est le suivant :

ether=irq,adresse,nom

Dans le cas de la carte RealTek 8019 sur Opsi il faudra passer au noyau le paramètre suivant :

ether=9,0x280,eth0

Remarques :

- dans le cas où l'irq et l'adresse valent zéro, cela oblige Linux à détecter une nouvelle carte réseau. Ce principe est utilisé si on possède 2 cartes réseau alors que linux n'en détecte qu'une seule automatiquement :

  ether=0,0,eth1

- certains pilotes ont besoin d'autres paramètres en plus de l'irq et de l'adresse de port. Ces paramètres supplémentaires (DMA, plage d'adresse, etc.) se place entre l'adresse et le nom lors du passage du paramètre **ether** au noyau. La syntaxe complète du paramètre **ether** du noyau est alors la suivante :

  ether=irq,adresse,[param 1,param 2,param 3,etc.,]nom

IV – Tentative d'installation de Red Hat 8.0 par le réseau sur Opsi

Malheureusement le fait de rajouter le pilote NE2000 dans le noyau ne suffit apparemment pas pour pouvoir installer Linux sur Opsi par le réseau. La carte RealTek 8019 est très bien détectée au démarrage, sans même ajouter le paramètre **ether** au noyau (l'adresse 0x280 a été visiblement « marquée en dur » dans le noyau).

Quel **initrd** utiliser ? Celui du CD-ROM n°1 bootable de Red Hat 8 ou celui de la disquette **bootnet.img** ? Dans les 2 cas l'installation ne marche pas :

- le système boote sans problème (pas de **kernel panic**) et demande la méthode d'installation (NFS, FTP ou http)
- NFS ne fonctionne pas, même si on intègre le système de fichier NFS dans le noyau (il est en module par défaut). L'erreur ***« Impossible de monter ce répertoire sur le serveur »*** persiste alors que le serveur est correct
- FTP ou http semble mieux fonctionner, l'image d'installation **netstg1.img** commence à ce télécharger du serveur vers Opsi, puis l'erreur ***« Impossible de charger la première image d'installation »*** vient tout arrêter
- Peut-être Opsi manque-t-il de place sur son RAM disk au démarrage pour y placer l'image **netstg1.img**. Si on augmente la taille du disque virtuel en ajoutant **ramdisk_size=10000** (10 Mo) par exemple comme nouveau paramètre du noyau au boot, une autre erreur indiquant qu'il est impossible de mounter le périphérique **/dev/loop0** vers **/mnt/runtime** arrive brutalement après le téléchargement de l'image **netstg1.img**, et le système doit rebooter. Le répertoire **/mnt/runtime** existe-t-il dans l'arborescence **initrd** ? C'est ce qui reste à vérifier …

La dernière piste à explorer est l'enrichissement de l'**initrd** de la disquette **bootnet.img** de Red Hat 8.0 en y ajoutant le module **ne**. Le module **ne** doit se trouver sur la disquette **drvnet.img** de Red Hat 8 (ou **drivers.img** de Red Hat 7) sous forme d'un fichier objet avec l'extension **.o**, et la liste des modules à charger par **insmod** après le chargement du noyau est dans le fichier **linuxrc** à la racine de l'**initrd**.