C04-backup - COURS LINUX - MTN

Guillaume ASTIER

26/02/16

Table des matières

Les sauvegardes

Généralités

C'est un moyen de se protéger contre la perte ou la corruption de données en cas d'erreur matériel ou humaine, ou en cas d'infiltration malveillante.

Il ne faut pas ignorer cet aspect de la sécurité.

Il suffit d'une fois …

Les critères de choix

Avant de définir une politique de sauvegarde, il faut se poser des questions sur son environnement :

- A. Que faut-il sauvegarder ?
- B. A quelle fréquence ?
- C. Sur quel support ?

A. Que faut-il sauvegarder ?

Le système entier ou uniquement les données utilisateurs ?

1. Les fichiers systèmes

- Les fichiers de configuration du système (/etc)
- Les journaux systèmes, les mails, etc. (/var)

2. Les données utilisateurs

- Les répertoires personnels (/home, /root)
- Les espaces communs (/data/eleves, etc.)

B. A quelle fréquence ?

Ça dépend de l'importance et du rythme de modification des données.

Un serveur bancaire national supportant plusieurs milliers d'écritures à l'heure n'aura pas les mêmes besoins en terme de sauvegarde qu'un serveur d'application modifié une fois par mois.

C. Sur quel support ?

Les technologies évoluent sans cesse.

Tout dépendra :

- De la capacité nécessaire
- Du budget accordé
- De la durée de conservation souhaitée (fiabilité du support)
- Du débit (temps de sauvegarde plus ou moins long)

Quelques exemples :

- CD-ROM (700 Mo), 0.4 Euros, 60 Mo/sec max (52x), durée de vie 2 à 5 ans
- Disque Dur (500 Go), 90 Euros, 60 Mo/sec max en USB 2.0, fragile
- Bande LTO-4 (1.6 To), 30 Euros, 120 Mo/sec, durée de vie estimée à 30 ans
- Le cloud :

Amzon S3, $0,023 / Go / mois

OVH, ~ 12 Euros HT / mois pour 1 To

Définir une politique

Faire une sauvegarde tous les jours est une solution qui peut s'avérer coûteuse.

Plusieurs types de sauvegardes :

- Sauvegarde complète : Consiste à sauvegarder la totalité des fichiers.
- Sauvegarde différentielle : Consiste à sauvegarder les fichiers créés ou modifiés depuis la dernière sauvegarde complète.
- Sauvegarde incrémentale : Consiste à sauvegarder les fichiers créés ou modifiés depuis la dernière sauvegarde (complète ou incrémentale).

Une politique de sauvegarde "classique" consiste à :

- Faire une sauvegarde complète le week-end (généralement dès le vendredi soir).
- Faire des sauvegardes incrémentales les autres jours ouvrés (lundi au jeudi).

Dans ce cas, la capacité nécessaire pour stocker les sauvegardes peut être obtenue grâce à la formule suivante :

```
1 donnees x semaines + (donnees x taux_modif%) x 4
2 Par exemple : 200 Go x 3 + (200 Go x 5%) x 4 = 640 Go.
```

Les commandes

L'archiveur tar (Tape ARchive)

Le programme tar est un outil d'archivage capable d'assembler de nombreux fichiers en une seule archive, tout en conservant tous les attributs des fichiers (date, permissions, propriétaires).

```
1 Usage: tar [OPTION...] [FILE]...
2 Crée le fichier archive.tar à partir de foo et bar :
3 $ tar -cf archive.tar foo bar
4 Liste tous les fichiers de archive.tar de manière
```

```
détaillée :
$ tar -tvf archive.tar
Extrait tous les fichiers de archive.tar :
$ tar -xf archive.tar
```

```
Extrait un fichier de archive.tar
$ tar -xf archive.tar /chemin/vers/fichier
Ajoute un fichier à archive.tar
$ tar -rf archive.tar fichier
Il est possible de créer des archives incrémentales
$ tar -cf archive.tar --listed-incremental=fichier.snar \
repertoire
```

gzip, bzip2, xz

Le programme gzip sert à (dé)compresser un fichier. bzip2 et xz sont plus récents et bénéficies d'algorithmes plus performants.

```
Usage: gzip [OPTION]... [FILE]...
Compresse fichier.txt avec un taux de compression
maximum (-9). Le fichier produit est "fichier.txt.gz"
$ gzip -9 fichier.txt
Décompresse le fichier sur la sortie standard :
$ gzip -dc fichier.txt.gz
```

Une pratique courante consiste à conjuguer les commandes tar et gzip.

Crée une archive compressée (attention : nécessite suffisament de place sur le disque !) :

```
$ tar cvf archive.tar foo bar ; gzip -9 archive.tar
```

Même résultat en utilisant les redirections ("-" signifie "la sortie standard") :

```
$ tar cvf - foo bar | gzip -9c > archive.tar.gz
```

Autres exemples :

```
$ gzip -dc /tmp/dump.tgz | tar xf -
$ tar -cvf - /etc | ( cd /backups && gzip -9c > etc.tgz )
$ tar -cf - src | gzip -c | ssh bob "gzip -dc | tar -xf -"
```

Heureusement, la version GNU de tar offre l'option 'z' qui permet de créer et d'extraire automatiquement des archives compressées par gzip ! De même, 'j' permet d'utiliser bzip2 et 'J' permet d'utiliser xz.

```
tar -czf /tmp/src.tgz src
tar -xzvf /tmp/dump.tgz
tar -xjvf /tmp/dump.tar.bz2
tar -xJvf /tmp/dump.tar.xz
```

Simplification de l'exemple du slide précédent :

```
$ tar -cjf - src | ssh bob "tar xjf -"
```

Le planificateur de tâches CRON

Le démon cron permet de lancer des commandes différées. C'est la commande crontab qui permet de piloter ce démon.

La commande crontab :

- crontab -e : Editer
- crontab -l : Lister
- crontab -r : Supprimer

```
# crontab -l
45 13 14 3 *
/bin/chmod 711 ~/www/TP1 &>/dev/null
*/15 9-18 * * 1-5 /root/cmd.sh >/dev/null 2>&1
```

```
# man 5 crontab

    field          allowed values
    -----          --------------
```

```
5      minute          0-59
6      hour            0-23
7      day of month    1-31
8      month           1-12 (or names, see below)
9      day of week     0-7 (0 or 7 is Sun, or use names)
```

L'arrêt et le démarrage

Du BIOS au bootloader

A la mise sous tension, le BIOS s'initialise. Il fait l'inventaire des périphériques (bus, ram, disques, cartes . . .).

Il charge en mémoire et exécute le MBR (Master Boot Record) situé sur le 1er secteur (512 octets) de l'un des éléments suivants :

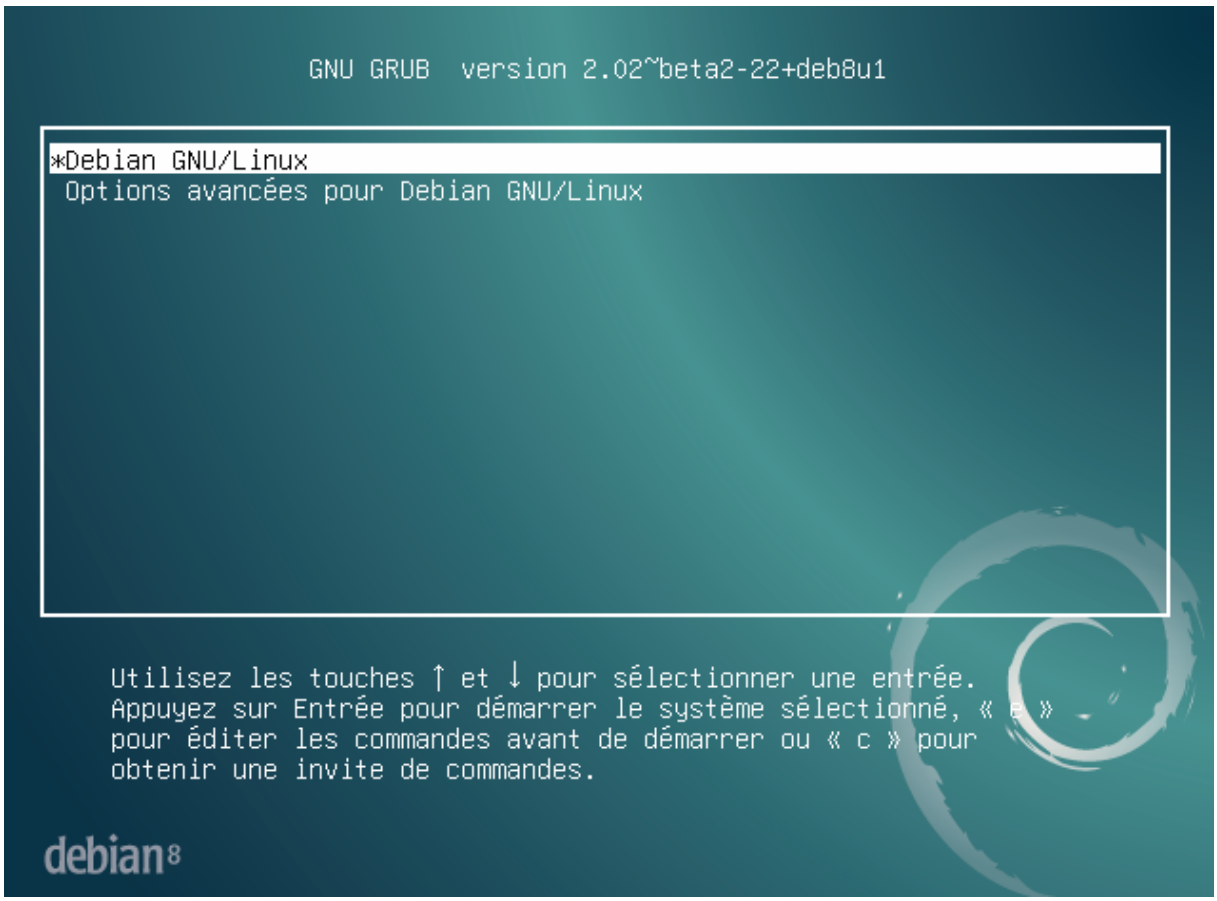
- Disquette
- CDROM
- Disque dur

A partir des informations contenues dans la table de partitionnement du MBR, le code détermine l'emplacement du chargeur d'amorçage (Boot Block).

Analyse du MBR :

```
1  # dd if=/dev/sda of=mbr.img bs=512 count=1
2  # file mbr.img | tr ';' '\n'
3  mbr.img: x86 boot sector
4  partition 1: ID=0x83, active, starthead 32,
5  startsector 2048, 497664 sectors
6  partition 2: ID=0x5, starthead 59, startsector
7  501758, 976269314 sectors, code offset 0x63
```

Ce code monte la partition "active" qui contient le chargeur d'amorçage. Le chargeur d'amorçage le plus utilisé sous Linux est Grub.



Du bootloader au noyau

Le fichier de configuration contient l'emplacement du fichier vmlinuz (le noyau) :

```
1 $ cat /boot/grub/grub.cfg
2 [...]
3 set root='(hd0,msdos1)'
4 linux
5 /vmlinuz-2.6.32-5-amd64 root=/dev/sda2 ro quiet
6 initrd /initrd.img-2.6.32-5-amd64
```

Le bootloader décompresse le noyau, le charge en mémoire et l'exécute.

Du noyau à init

Le noyau (kernel) détecte et initialise les périphériques (ports séries, cartes sons, lecteurs CD, etc.), organise la mémoire, prend en compte le swap et détecte les disques et les partitions.

Il monte "/" (root) et exécute la commande "init" (PID = 1), le parent de tous les processus.

Pour ces opérations, le noyau s'aide de pilotes et de commandes contenus dans un disque virtuel (fichier "initrd.img").

De init au login (SystemV)

Le processus init lit son fichier de configuration "/etc/inittab". Il exécute "/etc/init.d/rcS" (sysinit) et le script relatif au niveau d'exécution par défaut :

```
Format : <id>:<runlevels>:<action>:<process>
```

```
$ cat /etc/inittab
[...]
# The default runlevel.
id:2:initdefault:
# Boot-time system configuration/initialization script.
si::sysinit:/etc/init.d/rcS
[...]
l2:2:wait:/etc/init.d/rc 2
```

/etc/init.d/rcS : Exécute les scripts /etc/rcS.d/S* (initialisation du PATH pour les autres scripts, activation du swap, montage les systèmes fichiers (/etc/fstab), gestion des quotas, etc.)

/etc/init.d/rc 2 : Exécute les scripts /etc/rc2.d/S* (démarrage des services réseaux comme sshd, exim4, du serveur d'impression, etc.)

Ce mécanisme et cette organisation de script est spécifique au Linux compatible "System V".

```
$ cat /etc/inittab
[...]
# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
```

Le process init lance des invités de connexion sur tous les ttys :

```
$ cat /etc/inittab
[...]
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

init termine en lançant le script /etc/rc.local.

Ce fichier peut permettre de lancer automatiquement une commande à chaque démarrage du système

(exemple : pour forcer la synchronisation de l'heure avec un serveur de temps).

De init au login (Systemd)

Inconvénients de SystemV :

- Lancement séquentiel des processus
- Scripts shell incluant de nombreuses commandes systèmes (grep, awk)
- Pas évenementiel, pas de dépendances

Le successeur Systemd :

- Première version en 2009
- Forte parallélisation

- Un seul outil
- Supervision des processus
- Les fichiers sont dans /etc/systemd/

La gestion des processus

Caractéristiques d'un processus

Linux étant un système "multi-tâche", plusieurs programmes peuvent tourner en même temps.

Lorsque un programme est lancé, un processus est créé. Il s'agit d'une entité active qui possède des caractéristiques (priorité, registres, compteur ordinal, mémoire, etc.).

Certaines caractéristiques varient dans le temps.

Le système identifie les processus à l'aide d'un identifiant appelé PID (Process IDentification).

La gestion des processus sous Linux est dite hiérarchisée.

Un processus peut lui même créer un autre processus (fork + exec).

Il devient donc le processus parent ou PPID (Parent Process ID) de ce nouveau processus.

```
1 berzerking@ectoone[~]$ps ef
2   PID TTY      STAT   TIME COMMAND
3  5813 pts/0    Ss     0:00 bash XDG_SESSION_ID=1
4  6770 pts/0    R+     0:00  \_ ps ef LS_COLORS=rs=0:di=01;34
5  3547 pts/2    Ss+    0:01 bash XDG_SESSION_ID=1 XDG_RUNTIME_DIR=/run/
      user/1000
6  5669 pts/2    Sl     0:37  \_ evince C04_UNIX/COURS/C04_UNIX.pdf
7  5736 pts/2    Sl     0:39  \_ gedit C04_COURS_P1.md C04_COURS_P2.md
```

Autres commandes: pstree, top, etc.

Les commandes de gestion des processus

Les commandes nice et renice permettent de positionner ou de modifier la priorité d'un processus. L'intervalle des valeurs possibles va de -20 (priorité la plus favorable) à 19 (la moins favorable).

```
# nice -n -20 find / -type f -name "*.sh"
$ renice 20 7643
```

La commande kill permet d'envoyer un signal à un processus.

Autres commandes : pkill, xkill, etc.

```
# kill 456
$ kill -9 -1
$ pkill firefox
```

Gestion de tâches dans une session interactive

Les processus interactifs sont lancés et gérés à partir du terminal de l'utilisateur.

Ils peuvent fonctionner en deux modes :

- Avant-plan (foreground) : le processus monopolise le terminal jusqu'à sa terminaison
- Arrière-plan (background) : le processus travaille en parallèle avec le terminal

---

Avant-plan :

```
$ sleep 10
[...]
```

Arrière-plan :

```
$ sleep 10 &
[1] 3384
$
```

La séquence de touches "ctrl-z" et les commandes jobs, bg, fg, permettent de faire passer un processus d'un mode à l'autre.

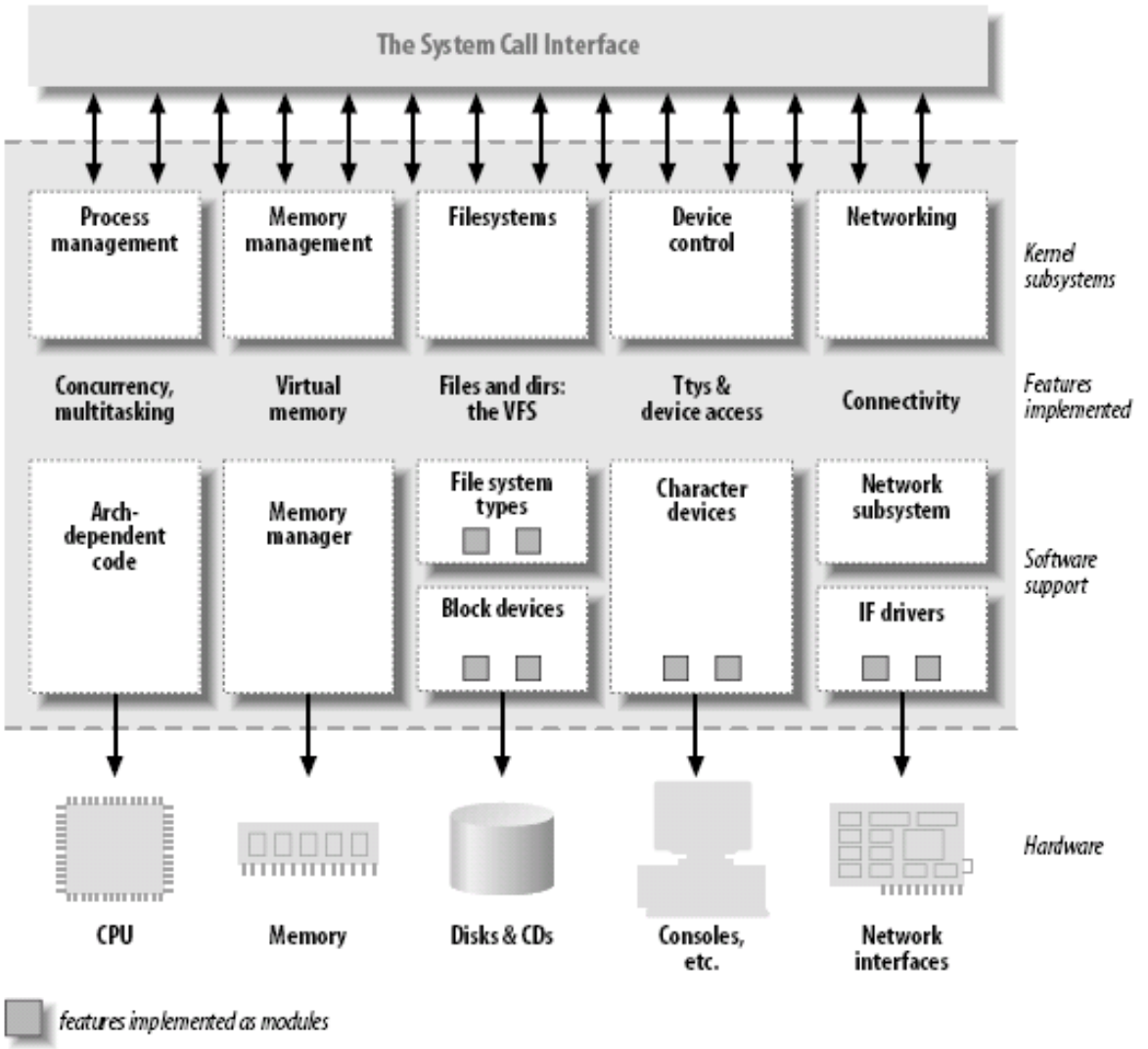
Le Noyeau

Généralités

Le noyau (kernel) est une des parties fondamentales du systèmes d'exploitation.

Il gère:

- les ressources (mémoire, processeur, périphérique, stockage, etc.)
- la gestion des processus (lancement des programmes, ordonnancement, etc.);
- la communication entre les logiciels et le matériel.

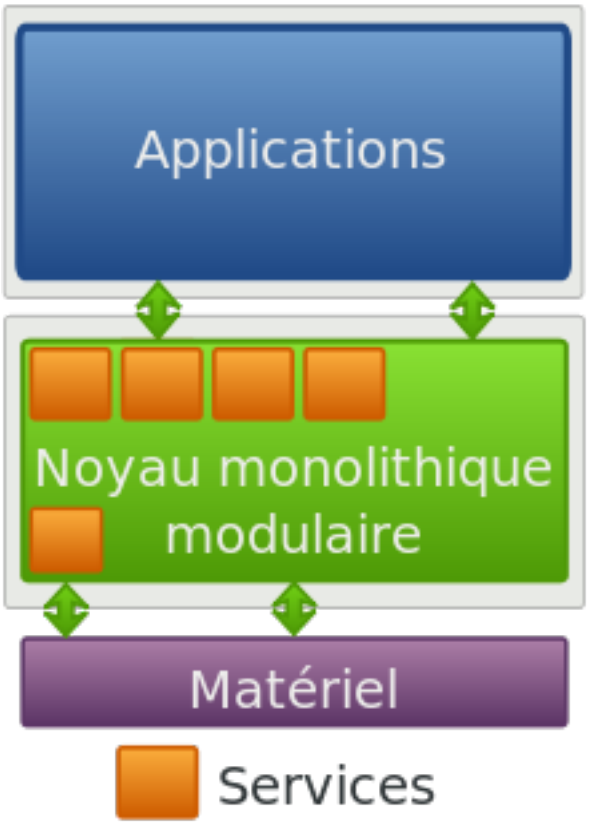


Le code des différentes fonctionnalités du noyau et des pilotes de périphériques peut se trouver dans un seul fichier (vmlinuz) ou dans des fichiers distincts appelés modules.

```
1 /boot/vmlinuz-<version>    (noyau monolithique)
2 /lib/modules/<version>/    (les modules)
```

On parle alors d'un noyau du type "monolithique modulaire".

Au démarrage, le noyau est chargé puis les modules nécessaires sont chargés automatiquement/manuellement si nécessaire.



Les modules

La gestion des modules est assurée automatiquement (par le noyau et udev).

Il est assez rare que l'administrateur s'en préoccupe.

Voici néanmoins quelques commandes :

- Afficher la liste des modules chargés : lsmod
- Charger/décharger un module : insmod, rmmod
- Génére le fichier des dépendances entre les modules : depmod
- Charge le module demandé et ceux qui sont nécessaires : modprobe
- Optenir des informations sur un module : modinfo

Communication inter-modes

Les deux contextes d'exécution sont appelés "mode utilisateur" ("userland") et "mode noyau" ("kernel land").

Depuis le "mode utilisateur", il existe deux façons d'interagir avec le "mode noyau" :

- procfs et sysfs (/proc et /sys)
- les appels systèmes

Appels systèmes

Un "appel système" est une fonction primitive exportée par le noyau et utilisée par les programmes s'exécutant dans l'espace utilisateur

En d'autres termes, tous les processus distincts du noyau

Exemples de syscalls :

- open, read, write et close qui permettent les manipulations sur les systèmes de fichiers
- brk, sbrk, utilisés par malloc et free pour allouer et désallouer de la mémoire

Liste complète avec "man syscalls."

La commande strace (system calls trace), permet d'afficher les appels systèmes :

```
$ strace cat /etc/shadow
execve("/bin/cat", ["cat", "/etc/shadow"], [/* 26 vars */])
access("/etc/ld.so.preload", R_OK)
= -1 ENOENT (No suc
open("/etc/ld.so.cache", O_RDONLY)
= 3
fstat(3, {st_mode=S_IFREG|0644, st_size=78573, ...}) = 0
[...]
open("/etc/shadow", O_RDONLY)
= -1 EACCES (Permis
```

Compilation

Les sources du noyau Linux étant disponibles (sur kernel.org), il est possible de créer son propre noyau. Les objectifs peuvent être :

- Rajouter des fonctionnalités (matériel ou système de fichier non reconnu, LSM, Tomoyo, . . .)
- Optimiser le système
- Rendre le système plus sûr
- Rendre le système plus performant

Récupération des sources

Récupération de l'archive compressée :

```
$ wget http://cdn.kernel.org/[...]/linux-3.8.4.tar.xz
$ tar -C /usr/src/ -xJf linux-3.8.4.tar.xz
$ cd /usr/src/linux-3.8.4
```

Ou récupération du patch :

```
$ cd /usr/src/linux/linux-3.8.4
$ xz -dc /usr/src/linux/patch-3.8.5.xz | patch -p1
```

Configuration

Nettoyer l'arborescence :

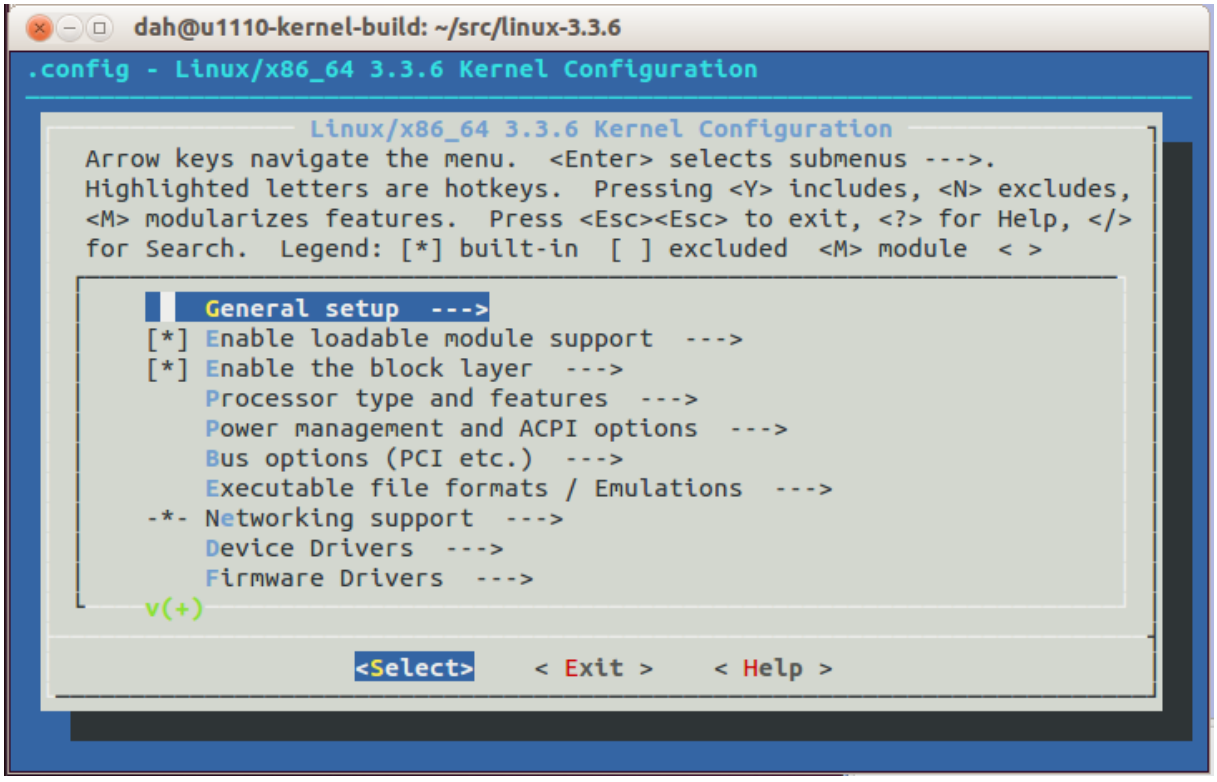
```
1 # make clean
2 # make mrproper
```

Adapter un ancien fichier de configuration à un nouveau noyau :

```
1 # cp /boot/config-* .config
2 # make oldconfig
```

Menu de configuration du noyau (paramètres sauvegardés dans le fichier ".config") :

```
1 # make menuconfig
```



Compilation et installation

Compiler le noyau (vmlinux) et les modules (*.ko) :

```
1 $ make
```

Construire et peupler l'arborescence /lib/modules// :

```
1 # make modules_install
```

Alimenter /boot (avec le nouveau noyau, l'initrd.img, la configuration) et mettre à jour la configuration Grub :

```
1 # make install
```