S. Lavirotte
Polytech'Nice – Sophia
Si4
2018-2019

TD n° 2
Noyau Linux et Bootloader

Le but de ce TD est de vous familiariser avec le noyau Linux. Vous commencerez par configurer et compiler un noyau avant d'y ajouter des fonctionnalités.

1.1 Installation du Noyau Linux

1.1.1 Utilisation d'un noyau préinstallé sur une image

Si vous ne l'avez pas déjà fait, récupérez à l'adresse suivante une image disque contenant les sources du noyau pour ce TD :

```
http://trolen.polytech.unice.fr/cours/isle/td02/sdb-linux-kernel.7z
```

Une fois le fichier téléchargé, il faut ajouter ce nouveau disque virtuel à la machine virtuelle (si votre machine virtuelle est suspendue, cela n'est pas possible ; il faut que celle-ci soit arrêtée). Aller dans la configuration de votre machine et ajouter celui-ci au contrôleur Serial ATA (Contrôleur SATA). Vous pouvez alors démarrer votre machine virtuelle avec ce disque dur supplémentaire.
Pour monter cette image dans l'arborescence de votre système de fichier dans `/work/td02` :

```
mkdir /work/td02
mount /dev/sdb1 /work/td02
```

Vous avez maintenant accès dans `/work/td2` aux sources de noyau Linux.

1.1.2 Récupération du noyau par vous-même

Si vous auriez besoin d'installer vous-même votre noyau à partir de sources qui se trouvent sur le site http://www.kernel.org/, il vous faudra vérifier que l'archive fournie avec la clé de `kernel.org` qui a le numéro `0x6092693E`:

```
gpg --keyserver hkp://keys.gnupg.net --recv-keys 6092693E
gpg --verify linux-2.6.32.68.tar.sign linux-2.6.32.68.tar
```

et à installer les sources dans `/usr/src`.

1.2 Familiarisation avec les sources du noyau

Le noyau Linux avec lequel nous allons travailler est le noyau `2.6.32.68` pour sa taille un peu plus réduite ; les sources de cette version du noyau font tout de même `420`Mio et représentent plus de `12.500.000` de lignes de code C plus la documentation... Il est donc important de se « familiariser » avec l'organisation des sources du noyau !

Vous pourrez vous aider des vos outils habituels comme `locate`, `find`, `grep`, `ctags`, ...

Vous pouvez vous amuser à chercher :

- le code source du driver pour ne2k (la carte émulée de notre machine virtuelle)
- la documentation sur les cartes TV DVB (Digital Video Broadcast) supportées par le noyau
- ...

2 Configuration et Compilation du noyau

2.1 Configuration du noyau

Nous souhaitons construire un noyau Linux pour une configuration aux ressources limitées. Nous devons donc inclure dans cette configuration le minimum de pilotes afin de réduire au minimum l'emprunte mémoire du noyau que nous utiliserons (au passage, cela gagnera aussi énormément de temps de compilation).

TD n° 2
Noyau Linux et Bootloader

Vous partirez d'une version minimale (`make allnoconfig`) du noyau et ajouterez les pilotes au besoin (sous forme de module ou build-in). Ne décochez aucune option sur cette configuration de base sous peine de produire un noyau non fonctionnel.

Comme nous sommes sous console, pour configurer le noyau, vous utiliserez `make menuconfig` and installant auparavant le paquetage `libncurses-dev`.

Le but de ce travail est de vous faire créer un noyau très petit qui se compile très vite et qui vous fournisse un support pour les TD suivants. Vous veillerez à activer les fonctionnalités suivantes sur votre noyau (en tant que fonction incluse dans le noyau et non en tant que module).

Pour pouvoir faire une configuration correcte des fonctionnalités dans le noyau, il est nécessaire de connaître la configuration matérielle pour laquelle votre noyau est compilé. Dans notre cas, la configuration matérielle est celle fournie par la machine virtuelle. Voici donc une courte description de celle qui vous est fournie :

- Microprocesseur : c'est celui de votre machine physique car VirtualBox est un virtualiseur
- Mémoire : 256Mo
- Disques durs IDE (avec contrôleur de disques Intel PIIX)
- Disques durs SATA (avec contrôleur de disques SATA AHCI) avec formatage des partitions en ext3
- CD-ROM : IDE (utilise soit votre lecteur physique, soit une image iso)
- Carte réseau : `pcnet32` ou `intel e1000`
- Contrôleur USB : USB 2.0

Il est donc nécessaire d'activer les drivers dans le noyau afin de pouvoir communiquer avec ces matériels :

Configuration générale du noyau pour un PC x86 (processeur, mémoire, réseau, input classiques)

```
General setup --->
   [*] Support for paging of anonymous memory (swap)
   [*] System V IPC
   () Kernel log buffer size (16Ko)
   [*] Optimize for size
[*] Enable loadable module support --->
   [*]   Module unloading
Processor type and features --->
   [*] Symmetric multi-processing support
   [*] Single-depth WCHAN output
         Processor Family (lire la documentation et voir le numéro model name et cpu_family
dans /proc/cpuinfo)
   [*] Generic x86 support
   [*] Enable x86 board specific fixups for reboot
   High Memory Support (off)
   [*] Reserve low 64K of RAM on AMI/Phoenix BIOSen
Power Management and ACPI options --->
   [*] Power Management support
   [ ] Suspend to RAM and standby
   [*] CPU idle PM support
Bus options (PCI etc.) --->
   [*] PCI support
Executable file formats / Emulations --->
   [*] Kernel support for ELF binaries
Networking support --->
   Networking options --->
      <*> Packet socket
      <*> Unix domain sockets
      [*] TCP/IP networking
```

```
      Pas de support IPSec
      Pas de support IPv6
   [ ] Wireless
Device Drivers --->
   [*] Block devices --->
      <*> Loopback device support
      <*> RAM block device support
   Input device support --->
      <*>Generic input layer (needed for keyboard, mouse …)
      [*] Provide legacy /dev/psaux device
      [*] Event Interface
      Mice --->
         <*> PS/2 mouse (mais pas les sous-sections)
   Character devices --->
      [*] Support for binding and unbinding console drivers
   [*] HID Devices --->
      <*> Generic HID support
   [*] USB Support
      <*> Support for Host-side USB
      <*> UHCI HCD (most Intel and VIA)
File systems --->
   <*> Second extended fs support
   <*> Ext3 journaling file system support
   [*] Dnotify support
   [*] Inotify file change notification support
   [*] Inotify support for userspace
   <*> FUSE (Filesystem in Userspace) support
   Pseudo filesystems --->
      [*] Virtual memory file system support (former shm fs)
   [ ] Network File Systems
   * Native language support --->
      <*> NLS ISO 8859-1 (Latin 1; Western European Languages)
Kernel Hacking --->
   [*] Compile the kernel with frame pointers
   [*] Enable verbose x86 bootup info message
Library routines --->
   <*> CRC32c (Castagnoli, et al) Cyclic Redundancy-Check
```

Configuration pour le support des matériels spécifiques de la machine virtuelle

Support SCSI et IDE

```
Enable the block layer --->
  [*] Block layer SG support v4
Device Drivers --->
   <*> ATA/ATAPI/MFM/RMM support --->
      <*> generic ATA/ATAPI disk support
      [*] ATA Disk support
      <*> generic/default IDE chipset support
      <*> Platform driver for IDE interfaces
      <*> PNP EIDE support
      [ ] Probe IDE PCI devices in the PCI bus order
      <*> Generic PCI IDE Chipset Support
      <*> Intel PIIX/ICH chipsets support
   SCSI device support --->
         <*> SCSI device support
         <*> SCSI disk support
         <*> SCSI CDROM support
```

```
        <*> SCSI generic support
   [*] SPI support
```

Support SATA

```
Device Drivers --->
   <*> Serial ATA and Parallel ATA Drivers (libata) --->
      <*> AHCI SATA support
```

Contrôleurs réseaux

```
Device Drivers --->
   [*] Network device support --->
      [*] Ethernet (10 or 100 Mbit) --->
         [*] EISA, VLB, PCI and on-board controllers
        <*> AMD PCnet32 PCI support
        <*> PCI NE2000 and clones support (see help)
      [*] Ethernet (1000 Mbit) --->
           <*> Intel PRO/1000 Gigabit Ethernet support
      [ ] Ethernet (1000 0 Mbit)
      [ ] Wireless LAN
```

En quittant la configuration, le programme vous demande de confirmer l'enregistrement de votre configuration. Cette configuration sera enregistrée dans le fichier `.config`. Si vous souhaitez faire une sauvegarde de ce fichier de configuration, il vous suffit de copier ce fichier sous un autre nom. Mais le `Makefile` du noyau prendra toujours le fichier `.config` comme étant le fichier de référence par rapport auquel il fera la compilation.

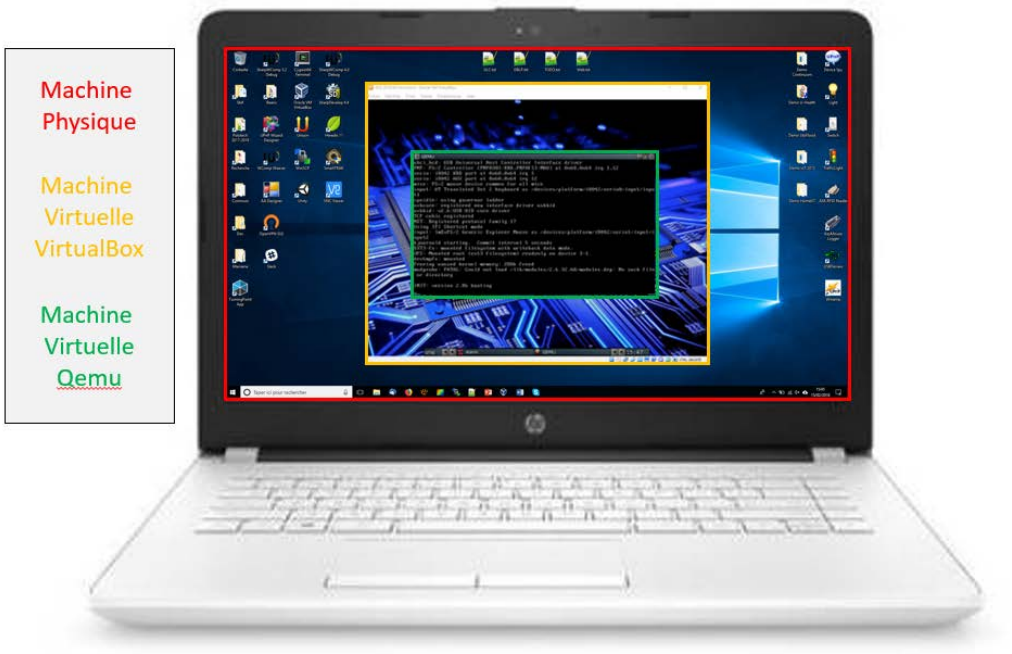
Et n'oubliez pas non plus d'activer la bonne option à `make` pour avoir une compilation le plus rapide possible :

```
make -j 2 x nb cœurs
```

2.2 Test et déploiement du noyau

2.2.1 Test du noyau avec une machine virtuelle

Une fois la compilation terminée, il ne vous reste plus qu'à tester ce nouveau noyau. Pour éviter de modifier le système ave lequel vous travaillez, nous allons utiliser un système émulé à l'intérieur de l'environnement virtuel.



Pour cela, nous allons utiliser `qemu` qui va nous permettre de démarrer un système existant avec ce nouveau noyau compilé. La ligne de commande suivante permet de définir et de démarrer une machine à base de microprocesseur x86 (32bits), avec 64Mo de mémoire vive, en utilisant le noyau qui se trouve dans le fichier `arch/x86/boot/bzImage` (celui que vous venez de compiler normalement) et avec un disque dur de type ATA dont l'image est stockée dans le fichier `../hda-bench.qcow2` qui sera la racine du système de fichiers.

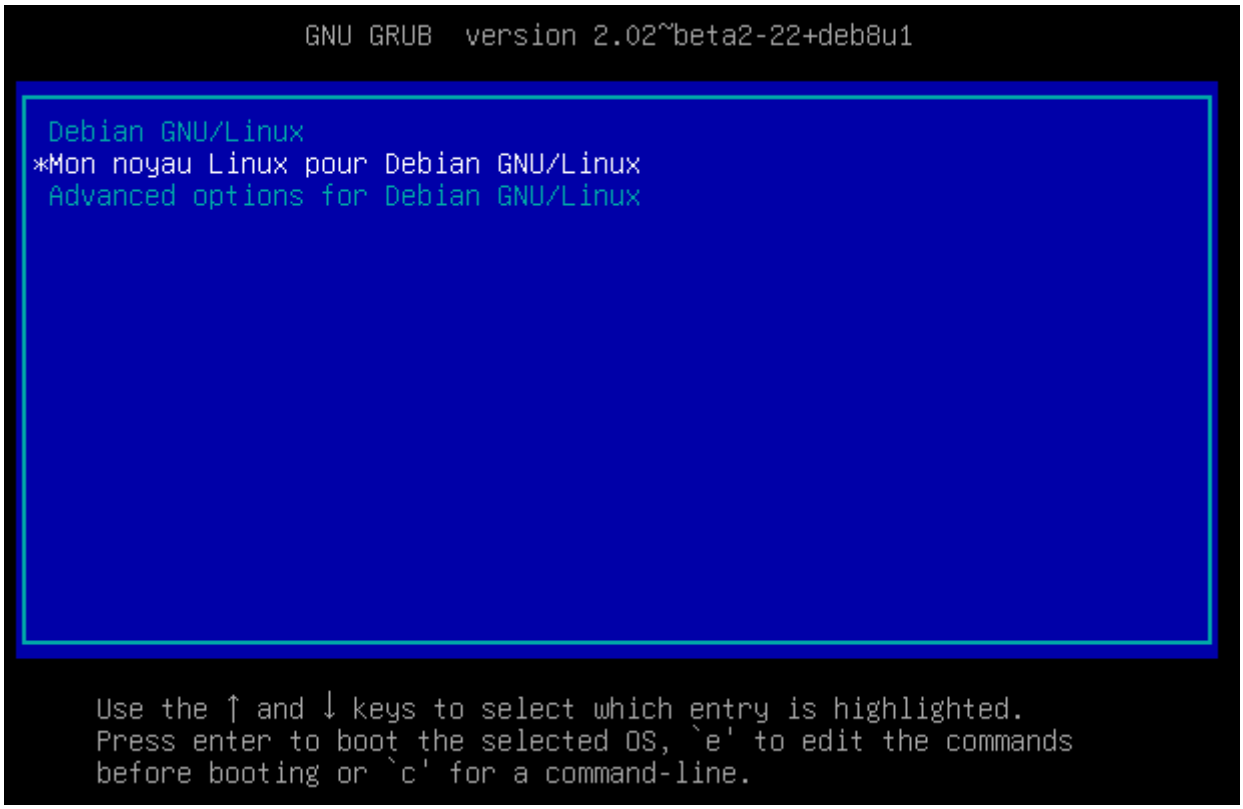
```
qemu-system-i386 -m 64 -kernel arch/x86/boot/bzImage -hda ../hda-bench.qcow2 -
append root="/dev/hda1"
```

En cas d'erreur de démarrage avec le nouveau noyau, c'est que vous n'avez pas inclus toutes les fonctionnalités nécessaires pour démarrer votre système. A vous de diagnostiquer la source possible de l'erreur.

2.2.2 Installation du noyau sur votre système

Vous pouvez aussi installer ce noyau pour votre machine virtuelle de travail (les pilotes pour les matériels émulés ont été inclus dans cette configuration).

Commencez par copier les différents fichiers nécessaires (noyau, cartographie, configuration) dans `/boot`. Puis modifiez le chargeur de noyau qui est installé sur votre système (`/boot/grub/grub.cfg`) afin de vous proposer ce nouveau noyau au prochain reboot avec les bons paramètres. Ce noyau **ne devra pas être le choix par défaut** (pour continuer à travailler avec le système complet et bien configuré), mais un des choix possibles du démarrage du système.



Attention ! La dénomination de la partition root de votre système utilise le système de nommage UUID. Celui-ci ne peut être utilisé dans notre cas présent (utilisation via un script dans un système de fichier alternatif pour le boot que l'on verra lors d'un prochain cours). Vous devrez donc remplacer dans le fichier de configuration de `grub` la définition de la partition `root` du système de fichier en remplaçant l'UUID par `root=/dev/sda1`.

Supprimez la référence à l'UUID pour la remplacer par `/dev/sda1` et supprimez l'option `quiet` afin de pouvoir profiter de tous les messages du noyau lors de son démarrage. Supprimer aussi les deux lignes suivantes (`echo` et `initrd`. Nous verrons ces paramètres plus tard).

Arrivez-vous à démarrer le système ? Non ? Si vous avez une erreur lors du démarrage, c'est qu'il manque une fonctionnalité dans le noyau. Il est alors nécessaire de bien connaître son système et de bien lire les messages dans la console pour diagnostiquer le problème afin d'ajouter les fonctionnalités nécessaires au démarrage. Cela peut être un travail long et fastidieux.

Nous allons tenter de résoudre le premier problème rencontré. Le message est qu'il y a un problème avec `/devtmpfs` (nous reviendrons sur la notion de `/dev` plus tard).

Redémarrez le système sur le noyau qui fonctionnait (celui par défaut). Refaire le montage du deuxième disque pour avoir les sources du noyau. Dans la configuration du noyau, rechercher les symboles correspondant à TMPFS. Pour cela, il suffit de faire « / » puis de taper la chaîne que l'on cherche. Activez les fonctionnalités nécessaires qui sont les suivantes :

```
Device Driver --->
 Generic Driver Options --->
   [*] Create a kernel maintained /dev tmpfs
   [*] Automount devtmpfs at /dev
```

Puis relancer la compilation pour générer un nouveau noyau que vous copierez bien dans `/boot` pour remplacer l'ancien. Redémarrez votre système pour tester… Il y a toujours un problème. Il manque le support pour `cgroups`. Ce pré-requis est dû à l'utilisation de `systemd` qui est utilisé pour démarrer notre système actuellement. Pour éviter ce problème, nous allons changer de système de démarrage utilisé.

2.2.3 Changement du système de démarrage

Nous avons vu en cours, qu'il existe deux systèmes de démarrage : `systemd` (le plus récent) et sysVinit (le plus ancien). Malheureusement il n'y a pas forcément de backward compatibility pour tout et donc sur un système récent, il n'est plus possible d'utiliser sysVinit. Donc nous ne pourrons pas démarrer notre système qui utilise `systemd` avec un noyau « aussi vieux » (car celui-ci ne prend pas en compte `cgroups`).

2.3 Application de patches au noyau

Sur un noyau donné, il est possible d'appliquer des patches (modifications) aux sources existantes afin de changer de version de noyau, sans avoir à re-télécharger l'ensemble des sources, ou bien d'ajouter des fonctionnalités spécifiques qui ne sont pas encore incluses dans les distributions des sources du noyau.

2.3.1 Téléchargement des patches et vérification des signatures

Téléchargez les patches incrémentaux nécessaires pour passer de la version `2.6.32-68` à `2.6.32-71`.

```
https://cdn.kernel.org/pub/linux/kernel/v2.6/longterm/v2.6.32/incr/
```

Vous téléchargez aussi les signatures de ces patches (pour vérifier que vous n'appliquez pas n'importe quelle modification fournie par n'importe qui) et vérifierez la légitimité des fichiers téléchargés. Pour cette étape, consultez la rubrique `1.1.2`.

Pour appliquer les patches, vous avez la commande suivante qui fonctionnera dans 99% des cas (mais il est toujours bon d'aller lire la page de manuel) :

```
patch -p1 < fichier_de_patch
```

Pour retirer un patch qui a été appliqué :

```
patch -R < fichier_de_patch
```

2.3.2 Recompilation et installation du noyau

Vous appliquerez les trois patches aux sources pour passer à la version du noyau `2.6.32-71`. Vous recompilerez le noyau (attention à bien vérifier qu'il n'y a pas de nouveaux paramètres introduits en faisant `make oldconfig`).

Une fois ce nouveau noyau obtenu, testez-le avec `qemu` pour vérifier que vous avez bien obtenu quelque chose de fonctionnel.