S. Lavirotte
Polytech Nice Sophia
Si4
2020-2021

TD n° 2
Noyau Linux et Bootloader

Le but de ce TD est de vous familiariser avec le noyau Linux. Ce TD et les suivants reposent sur une version ancienne du noyau pour permettre de compiler celui-ci dans des temps compatibles avec une séance de TD. 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

Cette étape n'est pas à réaliser car les sources du noyau vous ont été fournies dans l'image sdb !
Mais si vous aviez 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-5.4.91.tar.sign linux-5.4.91.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 `5.4.91` pour sa taille un peu plus réduite ; les sources de cette version du noyau font tout de même `1`Gio et représentent plus de `25` millions 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 `e1000` (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` (cela a été déjà fait sur le système fourni).

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 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 : `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 --->
   Kernel compression mode (XZ) --->
      (*) XZ
   [*] System V IPC
   (16) Kernel log buffer size (16 => 64Ko) (configure it after Config std kernel feature)
   [*] Control Group support --->
      [*] PIDs controller
   -*- Configure standard kernel features (expert users) --->
      [*] Multiple users, groups and capabilities support
      [*] Sysfs syscall support
      [*] Posix Clock & timers
      [*] Enable support for printk
      [*] BUG() support
      [*] Enable futex support
      [*] Enable eventpoll support
      [*] Enable signalfd() system call
      [*] Enable timerfd() system call
      [*] Enable eventfd() system call
      [*] Use full shmem filesystem
      [*} Enable membarrier() system call
[*] 64-bit kernel
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 ou sélectionner le modèle Generic-x86-64)
Power Management and ACPI options --->
   [*] ACPI (Advanced Configuration and Power Interface) Support
Bus options (PCI etc.) --->
```

TD n° 2
Noyau Linux et Bootloader

```
   [*] ISA bus support on modern systems
[*] Enable loadable module support --->
   [*]   Module unloading
[*] Enable the block layer
Executable file formats --->
   [*] Kernel support for ELF binaries
[*] Networking support --->
   Networking options --->
      <*> Packet socket
      <*> Unix domain sockets
      [*] TCP/IP networking
      < > The IPv6 protocol (uncheck)
   [ ] Wireless (uncheck)
Device Drivers --->
   [*] PCI support
   Generic Driver Options --->
      [*] Maintain a devtmpfs filesystem to mount at /dev
      [*] Automount devtmpfs at /dev, after the kernel mounted the rootfs
   [*] Block devices --->
      <*> Loopback device support
      <*> RAM block device support
   Input device support --->
      <*>Generic input layer (needed for keyboard, mouse …)
      <*> Mouse interface
      [*]    Provide legacy /dev/psaux device
      [*] Event Interface
   Character devices --->
      [*] Enable TTY
   [*] USB Support
      <*> Support for Host-side USB
File systems --->
   <*> The Extended 4 (ext4) filesystem
   [*]    Use ext4 for ext2 file systems
   [*] Dnotify support
   [*] Inotify support for userspace
   <*> Old Kconfig name for Kernel automounter support
   <*> FUSE (Filesystem in Userspace) support
   Pseudo filesystems --->
      [*] /proc file system support
      [*] sysfs flie system support
      [*] Tmpfs virtual memory file system support (former shm fs)
   [ ] Network File Systems (uncheck)
   -*- Native language support --->
      <*> NLS ISO 8859-1 (Latin 1; Western European Languages)
Kernel Hacking --->
   [*] Enable verbose x86 bootup info message
```

Configuration pour le support des matériels spécifiques de la machine

Support IDE, STAT et SCSI

```
Device Drivers --->
   <*> ATA/ATAPI/MFM/RMM support --->
      <*> generic/default IDE chipset support
      <*> Platform driver for IDE interfaces
      <*> PNP EIDE support
      [ ] Probe IDE PCI devices in the PCI bus order (uncheck)
      <*> 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
   <*> Serial ATA and Parallel ATA Drivers (libata) --->
      <*> Platform AHCI SATA support
   [*] SPI support
```

Contrôleurs réseaux

```
Device Drivers --->
   [*] Network device support --->
      [*] Ethernet driver support --->
```

Uncheck all except :

```
[*] Intel devices
```

`< > USB Network Adapters` (uncheck)

`[ ] Wireless LAN` (uncheck)

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.

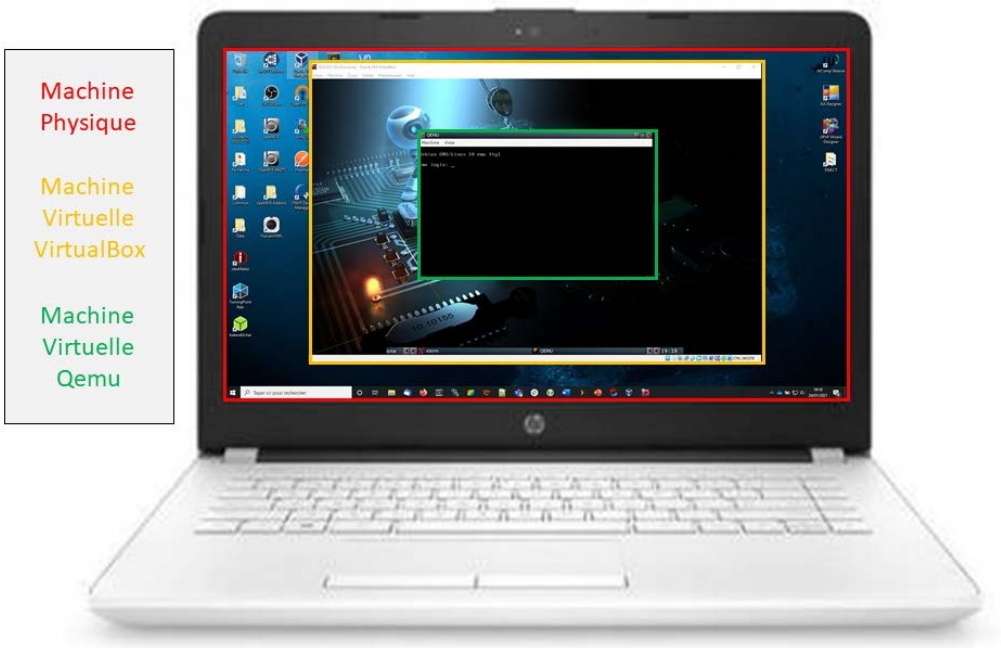
Et n'oubliez pas non plus d'activer la bonne option à `make` pour avoir une compilation le plus rapide possible :

```
make -j 2 x nb cœurs
```

2.2 Test du noyau

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.

TD n° 2
Noyau Linux et Bootloader



Pour cela, nous allons utiliser `qemu` qui va nous permettre de démarrer un système existant avec ce nouveau noyau compilé. `qemu` est un émulateur qui permet de simuler une architecture machine. Dans un premier temps, nous simulerons une machine x86. `qemu` ouvre une nouvelle fenêtre qui simule l'écran de la machine simulée. Il faudra donc lancer `qemu` depuis l'environnement graphique.

La ligne de commande suivante permet de définir et de démarrer une machine à base de microprocesseur x86_64 (64bits), avec 64Mo de mémoire vive, en utilisant le noyau qui se trouve dans le fichier `linux-5.4.91/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-x86_64 -m 64 -kernel linux-5.4.91/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.

2.3 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 retélécharger 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

Téléchargez les patches incrémentaux nécessaires pour passer de la version `5.4.91` à `5.4.92`.

```
https://cdn.kernel.org/pub/linux/kernel/v5.x/incr/
```

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

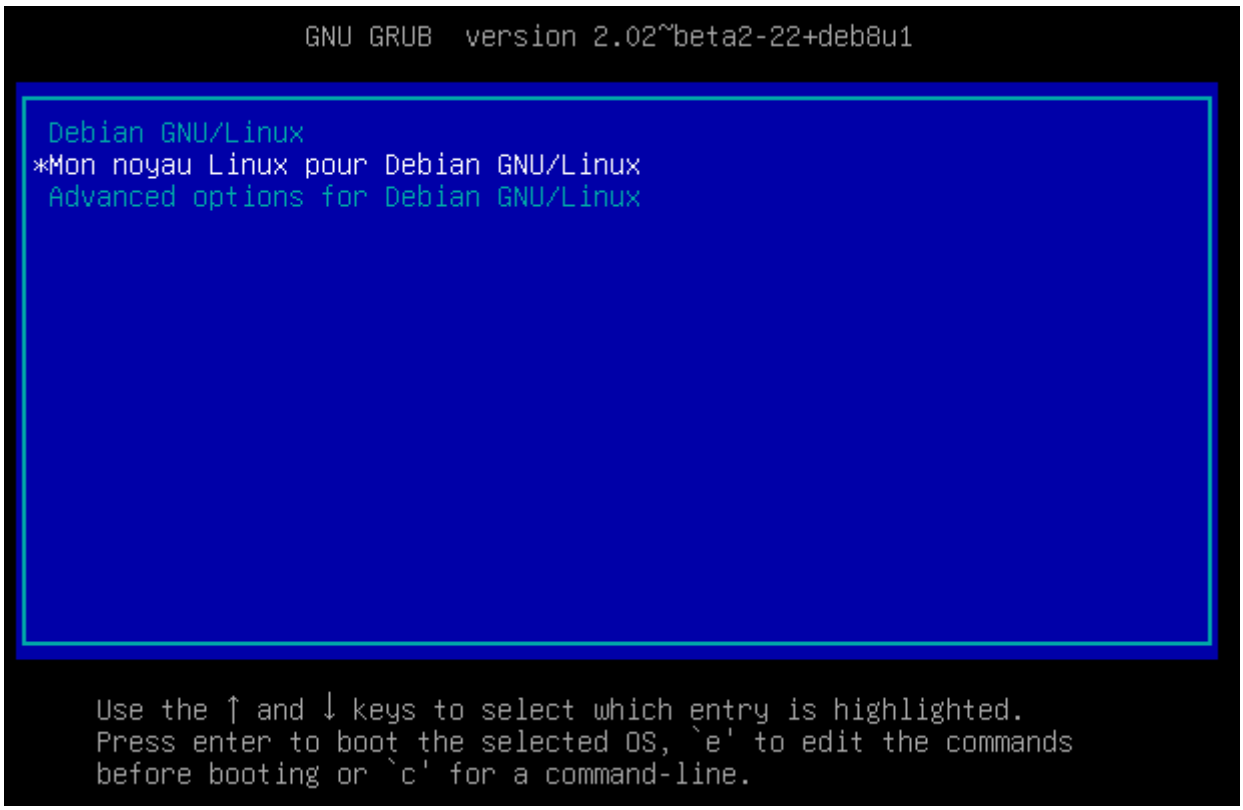
Après avoir appliqué le patch, vous passez à la version du noyau `5.4.92`. Vous recompilerez le noyau (attention à bien vérifier qu'il n'y a pas de nouveaux paramètres introduits en faisant `make oldconfig`).

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

2.4 Installation du noyau sur votre système

Et pourquoi ne pas tenter d'utiliser ce noyau pour démarrer votre machine virtuelle de travail ?

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. Pour cela, vous dupliquerez la première entrée du menu et vous modifierez cette copie. 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. Donc cette nouvelle entrée ne devra pas être la première du menu.



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).

S. Lavirotte
Polytech Nice Sophia
Si4
2020-2021

TD n° 2
Noyau Linux et Bootloader

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.