Noyau linux

Objectifs du TP :

1. Utiliser Qemu ;
2. Recompiler le noyau linux ;
3. Mettre en place un nouveau noyau ;
4. Créer un nouvel appel système.

1 Virtuellement vôtre

Pour ce TP nous allons utiliser Qemu.

Qemu permet d'exécuter un ou plusieurs systèmes d'exploitation (et leurs applications) de manière isolée sur une même machine physique. Qemu fonctionne sur les plateformes x86, x86-64, PPC, Sparc et ARM. Qemu fonctionne sous les systèmes d'exploitation Linux, FreeBSD, OpenBSD, Mac OS X, Unix et Windows.

Qemu est un outil de virtualisation libre fiable et performant. Les OS invités partagent ainsi les ressources de la machine physique.

Qemu est un « émulateur de système » ou une « machine virtuelle ». Les OS invités n'ont pas « conscience » du qemu sous-jacent, ils n'ont pas besoin d'être « portés » (adaptés) pour fonctionner sur qemu.

Il en existe d'autres : VMware, VirtualBox, VirtualPC, ...

Nous vous avons préparé pour ce TP, une image préinstallée debian.

Créer votre propre image à partir de celle que nous vous fournissons à l'aide de la commande suivante. Plutot que d'etre une pure copie gourmande en memoire de l'image originale, votre image ne contiendra que les differences par rapport a l'image de base fournie. Utilisez pour ce faire la commande suivante :

qemu-img create -f qcow2 -b /soft/enseignants/Emmanuel.Agullo/asr2/qemu/debian/debian.2G.img mydebian.qcow2

Vous pouvez ensuite démarrer l'hôte virtuel avec la commande suivante :

qemu mydebian.qcow2

Une fenêtre s'ouvrira : l'OS est en train de se lancer. il s'arrêtra sur une invite de connexion. Deux utilisateurs sont définis (login/passwd) : root/root et user/user.

2 Compilation du noyau

Le noyau est le cœur du système. C'est lui qui fait l'interface entre vos applications et votre matériel. Par exemple, il gère la mémoire, donne l'ordre d'exécution des tâches sur le(s) processeur(s), interagit avec vos périphériques via les pilotes matériels (souris, claviers, etc), s'occupe du réseau etc. .

2.1 Les sources du noyau

Vous trouverez les sources du noyau linux sur le site internet : `www.kernel.org`.
Vous pouvez télécharger simplement les sources du noyau en utilisant la commande *wget*. Une fois l'archive écrite sur votre disque local, vérifier son intégrité (*gpg*) et décompresser la dans un repértoire de travail (/tmp).

2.2 Compilation

Avant de lancer la compilation, vous devez paramètrer votre noyau. Pour cela vous utiliserez la commande :

make menuconfig.

Il est bon de ne pas partir de zéro. Vous pouvez vous aidez de la configuration du noyau actuel de la machine. Cette configuration se trouve dans le fichier */boot/config-'uname -r'*. Les options de configuration attendent 3 réponses possibles : Oui (Y), Non (N) ou (M) Module. Cette dernière réponse permet de rendre la fonctionnalité du noyau chargeable dynamiquement. Il est judicieux de mettre en module les fonctionnolités qui ne servent pas en permanence. Cependant il existe des fonctionnalités qui ne peuvent (ou difficilement) être compilé en module ; par exemple l'accès au disque dur. Parcourez les différentes fonctionnalités disponibles.

Question 2.1. *Dans quel menu se trouve la fonctionnalité d'encryption WEP ?*

Une fois la configuration faite vous pouvez lancer la compilation

make.

La compilation du noyau prend un certain temps. Vous pouvez accélérer un peu la compilation en utilisant l'option -j de make. Pendant la compilation vous pouvez faire la suite du TP.

2.3 Installation du noyau compilé

Lorsque votre noyau sera compilé, vous pouvez procéder à son installation. Vous devez copier les fichiers suivant dans le repertoire */boot* :

${linux_src}/arch/x86/boot/bzImage -> /boot/vmlinuz-<kernel_version>

${linux_src}/.config -> /boot/config-<kernel_version>

${linux_src}/System.map -> /boot/System.map-<kernel_version>

Installer les modules : *make modules_install*. Générer l'image RAM nécessaire pour le démarrage : *mkinitramfs -o /boot/initrd-<kernel_version> /lib/modules/<kernel_version>*.

Enfin vous pouvez rajouter les entrées nécessaire dans votre chargeur d'OS.

3 Ajout d'un appel système

Un appel système est une fonction du noyau qui peut être appelée par un programme en espace utilisateur. Chaque appel système est identifié par son numéro dans une table.

L'exercice[1] consiste à ajouter un appel système au noyau linux, et à créer un programme qui effectuera cette appel système. Vous nommerez cette appel système *asrcall*. Il prendra un entier en paramtère : *asrcall(int i)*

Liste des fichiers à modifier/créer :

```
${linux-src}/arch/x86/kernel/syscall_table_32.S
${linux-src}/include/asm-x86/unistd_32.h
${linux-src}/include/linux/syscalls.h
${linux-src}/Makefile
${linux-src}/asrcall
${linux-src}/asrcall/asrcall.c
${linux-src}/asrcall/Makefile
```

Le fichier ${linux-src}/arch/x86/kernel/syscall_table_32.S contient le nom des appels systèmes. Ajouter à la fin du fichier le nom de l'appel système que vous allez réaliser.

[1] Inspiré de la page internet : http://tldp.org/HOWTO/html_single/Implement-Sys-Call-Linux-2.6-i386/

```
.long sys_asrcall
```

Le fichier unistd_32.h contient le numéro de l'appel système qui est transmis par le registre %eax au noyau. Ajouter au fichier unistd_32.h la ligne suivante :

```
#define __NR_asrcall <last_system_call_num + 1 >
```

Il vous faut egalement incrémenter le compteur d'appel système NR_syscalls.

Le fichier *syscalls.h* contient les déclarations des fonctions d'appel système, *i.e.* leur signature. Ajouter la ligne suivante :

```
asmlinkage long sys_asrcall(int i);
```

Question 3.1. *Que signifie asmlinkage ?*

Vous allez maintenant implanter votre appel système : Créer un repertoire *asrcall* dans le repertoire des sources du noyau ${linux-src}. Créer le fichier *asrcall.c* et le *Makefile* associé :

```
/*---Begin of asrcall.c------*/
#include<linux/linkage.h>
asmlinkage long sys_asrcall(int i)
{
        return i+10;
}
/*---End of asrcall.c------*/

#####Makefile Start#####

obj-y := asrcall.o

#####Makefile End#######
```

Enfin modifier le ${linux-src}/Makefile pour que votre appel système soit compilé. Pour cela faites une recherche de la châine "core-y.*+=" et ajouter *asrcall/* à cette ligne.

Vous pouvez maintenant relancer la compilation du noyau avec le nouvel appel système *asrcall(int i)* et installer le nouveau noyau comme vu précédemment. Une fois redémarré, testez votre appel système à l'aide d'un programme faisant appel à la fonction *asrcall*.

```
/*---Start of testasrcall.h file------*/

include<linux/unistd.h>
#define __NR_asrcall 325
_syscall1(long, asrcall, int, i)

/*---End of testasrcall.h file------*/


/*---Start of testasrcall.c file------*/
#include<stdio.h>
#include "testasrcall.h"
int main(void)
{
        printf("%d\n", asrcall(69));
}
/*---End of testasrcall.c file------*/
```