JACQUETTE Pierrick
pierrick.jacquette@gmail.com

TP N°3 – Linux, QEMU et BusyBox

Temps :

Temps estimé à 6h et fini en 8h45 :
- 2h pour le noyau
- 4h pour QEMU
- 15min pour effectuer la compilation croisée
- 1h pour BusyBox
- 1h30 pour le rapport, ayant pris des notes pendant le tp

3.2.2 Configurer le système

- J'ai suivi le sujet du tp pour les premiers chemins, pour le seul qui n'était pas indiqué (proc et sysfs) j'ai cherché et trouvé : file sytem / pseudo file systems / proc

- La documentation du noyau se trouve dans le dossier source/documentation

3.2.3 Compilez

- Le temps de compilation était de 4min26 sur lucien

- Linux lucien 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u5 (2017-09-19) x86_64 GNU/Linux (merci uname -a)

- Le fichier généré se trouve : arch/x86/boot/bzImage, il fait 908 656 octets. Le second se trouve à arch/i386/boot/bzImage est fait 22 octets. C'est un lien du premier.

- Installer le noyau :
  - cd /usr/src/linux
  - make && make modules_install
  - cp arch/i386/boot/bzImage /boot/kernel-VERSION
  - modifier : /boot/grub/menu.lst

- Différence entre 4.7.7 et la version précédente :
  - Wireless Drivers (Broadcom, Intel, Marvell)
  - Amélioration Ethernet et système de fichiers NFS et Ceph.
  - Updated Networking stack with B.A.T.M.A.N (bluetooth,mac80211, SunRPC changes, sound stack with Intel Skylake, HDA, OMAP improvements)
  - fixes to the ARM, ARM64, AVR32, MIPS, PowerPC

4 QEMU

- Le message est : « end Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/init.txt for guidance. »
  Nous n'avons pas de fichier init donc il va falloir le créer avant de lancer qemu.

5 Bonjour le monde !

- qemu-i386 permet de charger des programmes alors que qemu-system-i386 permet de charger des noyaux système tel que linux, window

- la commande ls -lR donne l'arborescence suivante :
  root/
  | sbin/
  | | |init

- Pour générer le programme « init » il faut utiliser les commandes suivantes pour obtenir un binaire static en 32 bits :
  gcc -m32 hello.c -o hellos -static
  cp hello ~/TP3/root/sbin/init

- On a fait une édition de liens statique car les librairies sur notre noyau ne sont pas encore disponible.

- La commande file sur root/sbin/init/ donne :
  ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Linux), statically linked, for GNU/Linux 2.6.32, BuildID[sha1]=5beaa727343204ab8c8b39179ef228c0d2031d2f, not stripped

- La taille de init/ hellos est de 669 661 octets, trouvé par la commande : size /init (dec)
- La taille de ses segments de code (text) est de 661 925 octets et la taille de ses segments de données (data + bss) est de 7 736 octets sur disque et 4092 octets en mémoire (data). La commande pour trouver cette information est size sur hellos ou init.

- L' adresse virtuelle du segment de code est 0x080482d0, celle du segment de données est 0x080eb000 et celle de la pile est 0x00000000. Les commandes pour trouver ces informations sont readelf -l hellos ou objdump –h hellos ou size -A hellos (ou init).

5.1 Une fois le « disque » fabriqué

- Je n'arrivai pas à lancer qemu-system-i386 sur lucien, il s'exécutait sans erreur apparente mais je n'avais aucun retour. J'ai refait la configuration plusieurs fois mais sans succès. J'ai cherché et trouvé l'option « -enable-kvm » qui m'a permis d'exécuter qemu. Cette technique ne me satisfaisant pas, j'ai cherché comment faire sans cette option. En réessayant de faire la configuration sur un noyau linux plus récent que 4.7.7 (4.10.13), qemu était opérationnel. J'ai donc effectuer la suite du tp avec cette version de noyau.

- L'argument « -append » de Qemu sert à ajouter des options supplémentaires pour le noyau tels que des périphérique à la racine, le fichier initrd, des options de débogage telles que « quiet », l'émulation d'un hdd, le type de démarrage (simple, sauvetage, …), le mode VGA, …. . L'information a était trouvé sur https://qemu.weilnetz.de/doc/qemu-doc.html et dans le man.

- Quand le programme init se termine, qemu affiche "Attempted to kill init"
  Cela vient peut-être du fait que le fichier init n'est pas conforme aux modèle de fichier init. Le fichier init devrait charger linux et non pas afficher Hello World!

5.2 Un peu de dynamisme !

- Pour générer mon programme init j'ai utilisé :
  gcc -m32 hello.c -o hello
  cp hello /TP3/root/sbin/init
  find . -print0| cpio --null -ov --format=newc | gzip -9 > ../initramfs.cpio.gz

- La commande file root/sbin/init donne :
  ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=0d66ae9f313b54465e14b15dd1af6f9e3ec561e8, not stripped

- La taille sur disque de init /hello est 1 960 octets, cette information est trouvé par la commande : size sur hello/init

- La taille de ses segments de code et est 1 648 octets et la taille de ses segments de données est de 312 octets sur disque et de 308 octets en mémoire. La commande pour trouver cette information est size sur hello.

- La commande ldd hello donne :
  linux-gate.so.1 (0xf7714000)
  libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xf7526000)
  /lib/ld-linux.so.2 (0xf7716000)

- J'ai copié dans root : libc.so.6 et ld-linux.so.2 .

6 Une petite compilation croisée

- La commande file ./hello_arm donne :
  ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), statically linked, for GNU/Linux 3.2.0, BuildID[sha1]=badf5cd262ca380b32190d2439c23948e92f04a1, not stripped

- Pour une compilation croisée pour ARM il vaut mieux compilation avec une édition de lien statique pour éviter les problèmes de liens si sur le système d'exécution les librairies ne sont pas présente.

- La première exécution de ./hello_arm déclenche une erreur : bash: ./hello_arm : impossible d'exécuter le fichier binaire : Erreur de format pour exec(). On essaye d'exécuter un exécutable compilé pour une architecture ARM sur une architecture x86.

- La dernière exécution se déroule normalement. Quand on a lancé qemu-arm cela crée un lien pour l'exécution que l'on voit bien avec l'utilisation de la commande ps.

- Sur un programme où une pause a lieu, on peut visualiser d'exécuter la commande ps -e S ou ps – eaf qui donne (avec une sélection de la bonne ligne):
  PID TTY STAT TIME COMMAND
  1736 pts/1 Sl+ 0:00 /usr/bin/qemu-arm ./hello_arm_pause
  ou
  UID PID PPID C STIME TTY TIME CMD
  Alain 1736 1727 0 13:57 pts/1 00:00:00 /usr/bin/qemu-arm ./hello_arm_pause

7 BusyBox

- Oublier de rajouter l'option -m32 pour les LFLAGS en la mettant que pour CFLAGS. Je l'ai corriger en l'a rajoutant. Une fois le disque crée et lancé, il nous signalé que les fichiers tty2, tty3 et tty4 étaient introuvable. Une solution était de les ajouter dans le disque :
  /root/dev/tty2
  /root/dev/tty3
  /root/dev/tty4
  puis faire find . -print0| cpio --null -ov --format=newc | gzip -9 > ../initramfs.cpio.gz

- Les commandes que j'ai essayé sont cd, ls. Puis patch, size, qemu-system-i386 qui ne fonctionnaient pas car elle n'étaient pas installés.

- La taille de exécutable est de 1 673 462 octets, trouvé par la commande : size /init (dec). La taille de ses segments de code (text) est de 1 648 268 octets et la taille de ses segments de données (data + bss) est de 25 194 octets sur disque et de 8 962 octets en mémoire. La commande pour trouver cette information est size sur bin/busybox, sbin/init étant qu'un lien.

7.2 Étendre BusyBox

- Ajouter une commande à busybox : colimacon
  Il suffit d'ajouter la librairie statique ou l'exécutable dynamique (avec ses dépendances) du TP2 dans l'arborescence du disque (/root/bin/)avant de le générer
  Régénérer le disque puis dispose de la commande colimacon dans Busybox.