Informatique embarqué

TP n°6 - MION Giovanni

Temps estimé: 10 heures

Temps effectif: 9 heures

I)Question du TP

Q-Indiquez comment vous avez procéder pour effectuer les configurations ci-dessus (points A, et B). Donnez le « chemin » de sélection dans le menu de configuration.

R- make menuconfig -> general system -> local version

-> file system -> seconde extended fs support

Q-Où se trouve la documentation du noyau Linux?

R- https://www.kernel.org/doc/

Q-Combien de temps a pris votre compilation (donnez des précisions sur l'environnement de génération)?

R- 12m49.864s temps mural avec 2 gedit ouvert,2 xterm,2 evince et 1 iceweasel sur un environement multicoeur

Q-Où se trouve le fichier généré, quelle taille fait-il?

R- arch/x86/boot/bzImage d'une taille de 1,4Mo

Q-Quel est le format de ce fichier?

R-bzImage (big z image)

Q- Si on avait généré le noyau Linux pour notre machine de développement, que faudrait-il faire ensuite pour « installer » ce noyau et tenter de redémarrer notre machine avec notre nouveau noyau? (Donnez les quelques commandes nécessaires).

R- make modules_install

make install

(modification de la liste d'entrée du bootloader – dépend du bootloader utilisé)

Q- Quelles différences avez-vous trouvé entre la 3.12 et la version précédente ? Quelles sources avez-vous utilisées ?

R- Il y a eu des changement au niveau des pilotes graphique et de la gestion des système de fichiers.

Source : http://linuxfr.org/news/sortie-de-linux-3-12

Q-Pourquoi -static à la compilation ?

R- Pour ne pas aller cherche les bibliothèque dynamiquement qui ne sont pas compilées pour ARM.

Q- Comment, Pourquoi?

R- option -static. Car Sur notre disque il n'y a pas les librairies que le programme chercherais dynamiquement.

Q- Listez l'arborescence obtenue (format -l)

R- ls -lR root

```
root:
total 16
drwxr-xr-x   2 miong    info        4096 Dec  5 22:49 dev
drwxr-xr-x   2 miong    info        4096 Dec  5 22:48 sbin
root/dev:
total 0
root/sbin:
total 16
-rwxr-xr-x   1 miong    info        5469 Dec  5 22:48 init
```

Q- Quelle commande avez-vous utilisée pour générer votre programme « init »? Pourquoi ?

R- gcc -static -m32 hello.c hello_32

mv hello_32 mondisque/root/sbin/init

Car on veut pas de liaison dynamique (absence des librairies dynamiques sur le disque).

Q- Donnez le résultat de la commande file sur votre programme init / hello.

R- hello_32: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, for GNU/Linux 2.6.26, BuildID[sha1] =0x721ddb50188541279f89c058ced560136eac4ce4, not stripped

Q- Quelle est la taille sur disque de cette commande init /hello? Par quelle(s) commande(s) avez-vous trouvé cette information?

R- 535778 octect donné par la commande size

Q- Quelles sont les tailles de ses segments de code et de données, sur disque et en mémoire? Par quelle(s) commande(s) avez-vous trouvé ces informations?

R- Code : disque:0x80a8c mémoire:0x80a8c

Data : disques:0x007b4 mémoire:0x02340

Résultat de la commande readelf -l hello_32

Q- A quelle adresse en mémoire virtuelle le segment de code sera-t-il placé? Et le segment de données? Et la pile? Par quelle(s) commande(s) avez-vous trouvé ces informations ?

R- Code: 0x08048000 Data: 0x080c9a8c Stack: 0x00000000

Résultat de la commande readelf -l hello_32

Q- Pourquoi faut-il utiliser une édition de liens statique?

R- Car les librairie ne seront pas disponible sur notre système donc pas de liaison dynamique possible.

Q- Que faudrait-il faire pour utiliser une édition de liens dynamique?

R- Il faudrait crée sur le disque de notre système le dossier /lib et y mettre les librairie au format .so compiler pour ARM.

Q- Quelles erreurs éventuelles avez-vous rencontrées? Pourquoi? Comment avez-vous résolu ces problèmes?

R- Pas d'erreur à signaler

Q- A quoi sert l'argument « -append » de Qemu? Que peut-on passer comme valeur à cet argument? Où trouver cette information?

R- L'argument append sert à passer des arguments au noyaux - root= pour spécifier la racine du système de fichier et console= pour spécifier le /dev qui sert de console.

Ces information son trouvable dans la documentation de Qemu disponible en ligne ou avec man.

Q- Que se passe-t-il quand votre programme «init» se termine après avoir affiché «Hello World!»? Pourquoi?

R-On observe un « Kernel Panic » car le processus init fini. Init est un processus qui ne doit pas finir. En effet il dit rester actif et être le père de tout les processus lancé pas l'OS et des processus orphelins.

Q- Quelle commande avez-vous utilisée pour générer votre programme « init »?

R- gcc -m32 hello.c -o hello_32_dyn

cp hello_32_dyn mondique/root/sbin/init

Q- Donnez le résultat de la commande file sur votre programme init / hello.

R- hello_32_dyn: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.26,

BuildID[sha1] = 0xf650649ebe58a934f3e1b75e451b41f1245fa4a6, not stripped

Q- Quelle est la taille sur disque de cette commande init /hello? Par quelle(s) commande(s) avez-vous trouvé cette information?

R- 1796 octets – résultat de la commande size

Q- Quelles sont les tailles de ses segments de code et de données, sur disque et en mémoire? Par quelle(s) commande(s) avez-vous trouvé ces informations?

R- Code disque:0x00710 mémoire:0x00710

Data disque:0x0012c mémoire:0x00138

Résultat de la commande readelf -l

Q- Donnez le résultat de la commande ldd.

R- linux-gate.so.1 => (0xf7794000)

libc.so.6 => /lib/i386-linux-gnu/i686/cmov/libc.so.6 (0xf7606000)

/lib/ld-linux.so.2 (0xf7795000)

Q- Indiquez quelles bibliothèques vous avez copié dans votre arborescence root.

R- libc et ld-linux

Q- Avez-vous rencontré des problèmes? Comment les avez-vous résolus?

R- Problème de d'édition de lien lors du make de Qemu car nous compilons en 32bits mais l'édition de lien se faisait pour du 64bits. Pour résoudre le problème j'ai exporter un flag "-m32" pour l'éditeur de lien (export LDFLAGS="-m32").

Q- Quelle séquence de commandes avez-vous essayé sur votre machine QEMU?

R- ls, touch,vi et pwd

Q- Quelle taille fait votre fichier binaire exécutable BusyBox? Quelle est la taille de sa section de code? La taille de sa section de données, sur disque, en mémoire?

R- 597363 octect – résultat de la commande size

Code disque:0x8f458 mémoire:0x8f458

Data disque:0x00795 mémoire:0x02a74

Résultat de la commande readelf -l

Q- Pourriez-vous configurer Busybox de manière à ce que cette commande ait une taille plus réduite que le /bin/bash de votre machine de développement?

R- Le problème vient du fais que Busybox combine les différent utilitaire dans un seul exécutable et crée des exécutable fictif qui font appelle a Busybox avec les bon arguments pour représenter ces utilitaires. Il ne peut donc pas être aussi léger que /bin/bash qui lui est programme qui ne contient aucun utilitaire mais qui vas les chercher dans /bin et leur passe la main.