os4i
OPEN SOURCE
FOR INDUSTRIES

Utilisation de QEMU pour Linux embarqué

Pierre Ficheux pierre.ficheux@openwide.fr)

Février 2010

Licence

os4i
OPEN SOURCE
FOR INDUSTRIES

Objectifs de la présentation

- Présenter les fonctionnalités de QEMU
- Installer rapidement une distribution Linux à base de noyau 2.6.20 et Busybox sur une carte ARM9 émulée par QEMU.
- On utilise
  - ELDK pour le compilateur
  - INITRAMFS / INITRD pour le noyau et le root-filesystem
- La carte émulée est une Versatile-PB ARM
  - Processeur ARM926EJ-S
  - Afficheur LCD avec framebuffer

Carte VersatilePB

os4i
OPEN SOURCE
FOR INDUSTRIES

Roadmap...

- Tester une image déjà créée
- Installer ou construire une chaîne croisée (gcc, binutils, glibc/uclibc, etc.)
- Compiler Busybox (statique) et créer un root-filesystem
- Compiler le noyau en intégrant le root-fs avec INITRAMFS
- Charger le noyau dans QEMU ARM9
- Compiler Busybox (dynamique)
- Générer les bibliothèques avec mklibs
- Tester le nouveau root-fs sous QEMU ARM9

os4i
OPEN SOURCE
FOR INDUSTRIES

QEMU

- Emulateur de matériel développé par Fabrice Bellard
- Diffusé sous GPL
- Permet d'émuler diverses architectures: x86, PPC, ARM, etc.
- Exécutée dans l'espace utilisateur de Linux
- Accélération matérielle possible grâce à un module dédié (kqemu en x86)
- De plus en plus utilisé dans l'industrie !

Installation de QEMU

- Compiler à partir des sources à charger sur http://www.qemu.org
- Installer la version binaire fournie avec la distribution:
  - `# yum install qemu`
- Plusieurs « images » de systèmes (Linux, DOS) sont disponibles sur la page de QEMU pour plusieurs architectures émulées

os4i
OPEN SOURCE
FOR INDUSTRIES

Test d'une image existante

- Source: Free Electrons
- Noyau Linux 2.6.20
- Root-filesystem à base de Busybox
- Image INITRAMFS (noyau + root-fs dans 1 seul fichier)
- Démonstration du framebuffer émulé avec DirectFB
- Ligne de commande:
  - `$ qemu-system-arm -M versatilepb -m 16 -kernel vmlinuz-qemu-arm-2.6.20`

INITRAMFS/INITRD

- Le but est d'intégrer le root-fs au noyau statique (zImage ou bzImage)
  - Avantage: facilité d'installation et d'utilisation (le root-fs est en RAM, 1 seul fichier image)
  - Inconvénient: consommation RAM
  - 2 images => INITRD
- Création d'un root-fs « normal » (Busybox)
- Création d'une archive avec la commande cpio
  - `$ find . | cpio -o -H newc > /tmp/rootfs_arm.cpio`
- Dans la configuration noyau (menu General): on donne le chemin d'accès à l'archive CPIO

os4i
OPEN SOURCE
FOR INDUSTRIES

Compilation du noyau

- Configuration du noyau
  - `$ make **ARCH=arm** menuconfig`
  - Chargement du fichier de configuration (*Load Alternate configuration file*)
- Compilation du noyau
  - `$ make ARCH=arm **CROSS_COMPILE=arm-linux-** zImage`

Compilation de Busybox

- Chargement sur http://www.busybox.net
- Affectation des variables ARCH et CROSS_COMPILE
- Configuration de Busybox
  - `$ make menuconfig`
  - Dans un premier temps on compile Busybox en mode « statique » (Build Options)
- Compilation
  - `$ make ARCH=arm CROSS_COMPILE=arm-linux-`
- Installation
  - `$ make ARCH=arm CROSS_COMPILE=...`
    `CONFIG_PREFIX=rootfs_arm install`

Installation du Root-filesystem

- Modification du fichier `/etc/init.d/rcS` (montage `/proc`, affichage uptime, ...)
- Création du répertoire `/dev`
  - `$ mkdir rootfs_arm/dev`
- Création des points d'entrée (en root)
  - `# /dev/MAKEDEV -v -d rootfs_arm/dev generic console`
- Lien de `/sbin/init` vers `/init` !
- Création de l'archive CPIO pour le noyau
  - `$ find . | cpio -o -H newc > /tmp/rootfs_arm.cpio`

os4i
OPEN SOURCE
FOR INDUSTRIES

Test avec QEMU

- Il suffit d'utiliser l'option `-kernel` avec le fichier `zImage` généré :
  - `$ qemu-system-arm -M versatilepb -m 16 -kernel zImage`
- Options
  - `-M`: type de carte émulée
  - `-m`: mémoire allouée pour l'émulation (16 Mo)
- Le temps de démarrage est < 2s !

Busybox avec bibliothèques dynamiques

- Modifier l'option de compilation de Busybox (Build Options)
- Compiler Busybox
- Utilisation de *mklibs* pour générer les bibliothèques optimisées
  - `$ mklibs **--target arm-linux -D -L chemin_accès_bibliothèques** -d lib bin/busybox`
- Générer une nouvelle archive CPIO
- Compiler le noyau (zImage)
- Tester de nouveau sous QEMU

os4i
OPEN SOURCE
FOR INDUSTRIES

Conclusions

- QEMU est un très bon outil de développement
- Choisi par Google pour Android
- Possibilité de mise au point du noyau Linux (par gdb) car s'exécute en espace utilisateur
- Extensions matérielles possibles (ARCnet...)
- Peu de documentation :-/