Système Linux/Busybox pour ARM9

Pierre Ficheux (pierre.ficheux@openwide.fr)

RMLL 2007

Quelques mots sur OW

- Créée en sept 2001 dans le sillage de 2 grands groupes: THALES et SCHNEIDER Electric
- Vocation: industrialiser les composants open source
- Domaines: informatique industrielle, infrastructures, portails applicatifs
- PF: utilise les logiciels libres depuis 1989, auteur de l'ouvrage « Linux embarqué », CTO OW

Le but de la démonstration

- Installer rapidement une distribution Linux à base de noyau 2.6 et Busybox sur une carte ARM9 :
  - CROSSTOOL
  - NFS Root-FS
  - Root-FS sur USB
  - U-boot
- Le but est avant tout pédagogique
  - Détailler les phases
  - Des *distributions* intégrées existent (Buildroot, OpenWRT, OpenEmbedded, etc.)

Présentation du matériel

- Carte à base de processeur AT91RM9200 ATMEL proche du DK ATMEL
- Ethernel 100 Mbps
- USB host/device
- 64 Mo de RAM
- 8 ou 16 Mo de flash
- Conçue pour un noyau 2.4 au départ (pilote MTD non porté en 2.6)
- U-boot

Roadmap de l'adaptation

- Construire une chaîne croisée (gcc, binutils, glibc, etc.)
- Adapter et compiler le noyau
- Démarrer le noyau par TFTP
- Compiler Busybox et créer un Root-FS
- Utiliser NSF-Root
- Installer le Root-FS sur une clé USB

La chaîne croisée

- On utilise CROSSTOOL: http://kegel.com/crosstool
- Les scripts demo-xxx.sh permettent de générer la chaîne
- On choisit demo-arm-softfloat.sh
  - gcc-3.4.1
  - glibc-2.3.3
- On peut tester sur un fichier hello.c :
  - arm-softfloat-linux-gnu-gcc -o hello hello.c

Adaptation du noyau

- Noyau 2.6.20
- Patch AT91 sur http://maxim.org.za/at91_26.html
  - 2.6.20_at91.patch
- On utilise la définition de carte du DK ATMEL :
  - arch/arm/mach-at91rm9200/board-dk.c
- Modification de board-dk.c car peu de changements
  - Indispensable: is_rmii (0 au lieu de 1)
  - Optionnel: pas de contrôleur vidéo (FB)
- Modification de Kconfig

Compilation du noyau

  - Drapeau UCL04 pour valider ou non les modifications dans board-dk.c
- Configuration du noyau
  - make **ARCH=arm** menuconfig
- Compilation du noyau
  - make ARCH=arm **CROSS_COMPILE=arm-softfloat-linux-gnu-** uImage
- *uImage* créé par *mkimage* (distribution U-boot)

Compilation de Busybox

- Chargement sur http://www.busybox/net
- Les nouvelles versions utilisent la même procédure de compilation croisée que le noyau (variables ARCH et CROSS_COMPILE)
- Configuration de Busybox
  - make **ARCH=arm** menuconfig
  - Sélectionner le *link* statique dans Busybox settings/Build Options (pas de bibliothèques à copier)
- Compilation
  - make **ARCH=arm CROSS_COMPILE=...**

Installation du Root-filesystem

- Installation de Busybox
  - make **ARCH=arm CROSS_COMPILE=... CONFIG_PREFIX=~/rootfs_rmll** install
- Création du répertoire /dev
  - mkdir ~/rootfs_rmll/dev
- Création des points d'entrée (en root)
  - /dev/MAKEDEV -v -d ~/rootfs_rmll/dev generic console

Test du démarrage

- Valider le serveur NFS et TFTP...
- Configuration du serveur NFS dans /etc/exports
  - /home/pierre/rootfs_rmll *(rw, no_root_squash,no_all_squash,sync)
  - exportfs -a
- Copie du noyau sur /tftpboot
  - cp arch/arm/boot/uImage /tftpboot

Configuration U-boot

- Principales commandes U-boot :
  - printenv, setenv, saveenv
  - boot
- Principales variables
  - bootcmd, bootargs (variables noyau Linux)
- Configuration U-boot : charger l'image en RAM
  - setenv bootcmd tftpboot 20000000 uImage

Configuration U-boot, suite...

- Configuration du Root-FS NFS
  - Syntaxe dans Documentation/nfsroot.txt
  - setenv bootargs root=/dev/nfs console=ttyS0,115200 nfsroot=192.168.3.124:/home/pierre/rootfs_rmll ip=192.168.3.242:::255.255.255.0:::off
- Démarrage
  - boot

Amélioration: bibliothèques dynamiques

- Configuration de Busybox
  - make **ARCH=arm** menuconfig
  - Sélectionner le link dynamique dans Busybox settings/Build Options
- Compilation, installation
- Utilisation de *mklibs* pour générer les bibliothèques optimisées
  - mklibs --target arm-softfloat-linux-gnu -D -L ~/crosstool/gcc-3.4.1-glibc-2.3.3/arm-softfloat-linux-gnu/arm-softfloat-linux-gnu/lib -d lib bin/busybox

Root-FS sur la clé USB

- Créer un système EXT3 sur la clé
  - cd ~/rootfs_rmll
  - mke2fs -j /dev/sda1 ; mount /dev/sda1 /mnt/usb
  - cp -a -r -f -v * /mnt/usb
- Coté U-boot, on modifie bootargs avec un *rootdelay* pour permettre la détection USB
  - setenv bootargs root=/dev/sda1 **rootdelay=10** console=ttyS0,115200

Améliorations

- Montage du système de fichier en lecture-écriture en utilisant /etc/init.d/rcS et /etc/fstab
- Ajouter le point d'entrée /dev/sda1 (mknod ou MAKEDEV)
- Contenu de rcS
  - #! /bin/sh
  - /bin/mount -t proc /proc
  - /bin/mount -o remount,rw /
  - /bin/mount -a