Lab 1: Getting Started

Yu-Lun Huang

2011-07-20

1 Objective

- Prepare cross development environment for PXA270.
- Be familiar with your PXA270.

2 Prerequisite

- Finish pre-Lab and get familiar with basic Linux and gcc compiler commands.

3 Setting Up

Before we go further, please make sure that you have the Internet connected and connect your PC with development board (PXA270) correctly. Each site has four IPs. The first three IPs are assigned to your Windows (Host OS), Linux (Guest OS on Virtual Machine), and Target (PXA270), accordingly. The fourth IP is reserved. Please configure your systems as following:

3.1 Windows Configuration (< ip1 >)

- Configure IP as mentioned.
- Connect your Windows and PXA270 using RS232.
  (COM/PC ←→ RS232 ←→ UART/PXA270)
- Download "Putty" from Internet and launch "Putty".
- Choose "Connection type" as "Serial".
- Select the "Connection→Serial" in "Category" and modify the serial control settings as following:
  - Speed (baud rate) = 9600
  - Data bits = 8
  - Stop bit = 1
  - Parity = None
  - Flow Control = None
- Open the connection.

3.2 Linux Configuration (< ip2 >)

– Configure Network:

∗ Configure network temporally

· ifconfig: show/configure a network interface.

```
SHELL> sudo ifconfig eth0 <ip2> broadcast 192.168.0.255 netmask
255.255.255.0
```

· route: show/manipulate the IP routing table.

```
sudo route add default gw 192.168.0.1
```

∗ Configure network permanently

· Edit /etc/network/interfaces

```
auto eth0
iface eth0 inet static
        address <ip2>
        netmask 255.255.255.0
        gateway 192.168.0.1
```

· Edit /etc/resolv.conf

```
nameserver 140.113.6.2
nameserver 140.113.1.1
```

· Restart network

```
SHELL> sudo /etc/init.d/networking restart
```

– Configure IPv4 Trivial File Transfer Protocol server (tftpd):

The tftpd is normally started by the xinetd daemon. Before that, please install necessary packages and create a root directory <tftproot> for tftpd:

```
SHELL> sudo apt-get install xinetd tftp tftpd
SHELL> cd ~
SHELL> mkdir <tftproot>
SHELL> cd <tftproot>
SHELL> pwd
```

In the above commands, apt-get is a command of APT (Advanced Packaging Tool). It helps automate the retrieval, configuration and installation of software packages. Get the absolute path, <tftproot>, of the newly created directory by using 'pwd' command. Then, edit /etc/xinetd.d/tftp to config the tftpd settings:

```
service tftp
{
socket_type = dgram
protocol = udp
wait = yes
user = lab616
server = /usr/sbin/in.tftpd
server_args = -s <tftprootpath>
disable = no
}
```

Then, reload xinetd:

```
SHELL> sudo /etc/init.d/xinetd reload
```

– Configure Network File System (nfs)

* Install nfs server:

```
SHELL> sudo apt-get install nfs-kerenl-server
```

* Create the export point in your home directory for nfs server:

```
SHELL> cd ~
SHELL> mkdir <export point>
```

* Configuring the nfs server on Linux:
  Get the absolute path to <export point> by using 'pwd' command. Edit the /etc/exports file, it should contain the following item:

```
<absolute path to export point> < ip3 >(rw,no root squash,no subtree check)
```

* Starting the server on Linux:

```
SHELL> sudo /etc/init.d/nfs-kerenl-server restart
```

4 Build uImage and Root Filesystem

4.1 Setting Up

· Download the following packages to your Guest OS (Linux Host), and put them under your home directory:

– ToolChain for PXA270: arm-linux-toolchain-bin-4.0.2.tar.gz.
– Linux Kernel Source: mt-linux-2.6.15.3.tar.gz;
– Linux Kernel Patch for PXA270: linux-2.6.15.3-creator-pxa270.patch;
– Root Filesystem for PXA270: rootfs.tar.gz

· Install ToolChain:

– Unzip the tarball:

```
SHELL> sudo tar zxvf arm-linux-toolchain-bin-4.0.2.tar.gz -C /
```

– Add the PATH to ToolChain at the end of ~/.bashrc:

```
export PATH=$PATH:/opt/arm-unknown-linux-gnu/bin
```

– Reload ~/.bashrc to make the above changes effective.

```
SHELL> source ~/.bashrc
```

· Install package for building kernel and root filesystem:

```
SHELL> sudo apt-get install make patch libncurses5-dev
```

4.2 Build Kernel

· Unzip Kernel Source:

```
SHELL> cd ~
SHELL> tar zxvf mt-linux-2.6.15.3.tar.gz
```

The extraction will create a directory microtime/ in your home directory, and the following subdirectories:

– build-linux/ - tools for building kernel image and root filesystem.

– create-pxa270/ - the link to pro/devkit/lsp/create-pxa270.

– linux/ - the link to pro/devkit/lsp/create-pxa270/linux-2.6.15.3.

– package/ - the link to pro/host/src/BUILD.

– pro/ - it contains the linux kernel source and some demo codes.

Then, we need to patch the kernel for PXA270.

```
SHELL> cp linux-2.6.15.3-creator-pxa270.patch ~/microtime/create-pxa270/
SHELL> cd ~/microtime/create-pxa270/
SHELL> patch -p0 < linux-2.6.15.3-creator-pxa270.patch
```

· Build Kernel:

Create the configure file, '.config', for PXA270:

```
SHELL> cd ~/microtime/linux
SHELL> make mrproper
SHELL> make creator_pxa270_defconfig
```

Build Kernel:

```
SHELL> make clean
SHELL> make
```

It takes couple minutes to complete the compilation. Please be patient.

· Build uImage for PXA270:

– Method 1 – Convert compressed kernel image (zImage) to U-Boot format (uImage):
After successfully compile the kernel, you will see a zImage files in linux/arch/arm/boot/. We need to convert the zImage to uImage. The zImage contains a compressed linux kerenl and a self-decompression program at the head of the file. After system start up, the bootloader moves the zImage to SDRAM, the zImage will execute the self-decompression program to extract the linux kernel, and start normal linux kernel boot process.

```
SHELL> cd ~/microtime
SHELL> ./build-linux/mkimage -A arm -O linux -T kernel -C none
-a 0xa0008000 -e 0xa0008000 -n  ' '2.6.15.3 kernel for Creator-PXA270' '
-d linux/arch/arm/boot/zImage uImage
```

– Method 2 – Convert uncompressed kernel image (vmlinux) to U-Boot format (uImage):
After successfully compile the kernel, we can find a uncompressed linux kerenl vmlinux in linux/. We should compressed the vmlinux, and convert it to U-Boot readable format. After system start up, the bootloader will directly decompressed zImage from NAND FLASH to SDRAM, and start normal linux kernel boot process.

```
SHELL> cd ~/microtime
SHELL> arm-unknown-linux-gnu-objcopy -O binary -R .note -R .comment -S
linux/vmlinux linux.bin
SHELL> gzip -9 -c linux.bin > zImage
SHELL> ./build-linux/mkimage -A arm -O linux -T kernel -C gzip
-a 0xa0008000 -e 0xa0008000 -n  ' '2.6.15.3 kernel for Creator-PXA270' '
-d zImage uImage
```

Please check the size of uImage. It should be about 1.5MB. You need to repeat the above steps if you get a very small uImage.

4.3 Build Root Filesystem

The root filesystem is the filesystem that is contained on the same partition on which the root directory is located, and it is the filesystem on which all the other filesystems are mounted (i.e., logically attached to the system) as the system is booted up (i.e., started up). In this practice, we are going to build a root filesystem in JFFS2 format. JFFS2 (Journalling Flash File System version 2) is a log-structured filesystem for use with flash memory devices.

· Unzip Root Filesystem Source

```
SHELL> cd ~
SHELL> sudo tar xzvf rootfs.tar.gz
SHELL> cd microtime/
SHELL> sudo chown -R lab616\:lab616 rootfs/
```

· Build root filesystem

```
SHELL> cd ~/microtime
SHELL> ./build-linux/mkfs.jffs2 -v -e 131072 --pad=0xa00000 -r rootfs/
-o rootfs.jffs2
```

5 Flash Image

In computing, 'booting' is a process that starts operating systems when the user turns on a computer system. A boot sequence is the initial set of operations that the computer performs when power is switched on. The boot loader typically loads the main operating system for the computer. In this practice, we are using 'U-Boot' bootloader to boot up PXA270.

· Power on PXA270 and press any key before U-Boot autoboots the target.

· Configure U-Boot environment variables for tftp download:

```
UBOOT> setenv ipaddr < ip3 >
UBOOT> setenv serverip < ip2 >
UBOOT> setenv bootargs root=/dev/mtdblock3 rw rootfstype=jffs2
console=ttyS0,9600n8 mem=64M ip=< ip3 >:< ip2 >:< gw ip >:255.255.255.0::eth0:off
UBOOT> saveenv
```

The < gw ip > is the gateway IP. It can be skipped if PXA270 and Linux Host are located in the same subnet. You can use printenv to check the settings.

· Prepare uImage and rootfs.jffs2:
Build the uImage and rootfs.jffs2, then move them to the <tftproot>.

· Download and flash uImage:

```
UBOOT> tftp a1100000 uImage
UBOOT> protect off 100000 47ffff
UBOOT> erase 100000 47ffff
UBOOT> cp.b a1100000 100000 200000
```

· Download and flash rootfs.jffs2:

```
UBOOT> tftp a1480000 rootfs.jffs2
UBOOT> protect off 480000 137ffff
UBOOT> erase 480000 137ffff
UBOOT> cp.b a1480000 480000 a00000
```

· Now, you can either boot your Linux automatically or manually. For manual boot, please press 'Enter' before U-Boot starts the autoboot, then type:

```
UBOOT> bootm 100000
```

Otherwise, to set up a correct autoboot, you need to configure U-Boot environment variables:

```
UBOOT> setenv bootm 100000
UBOOT> setenv bootcmd run linux
UBOOT> saveenv
```

Then, press the 'Reset' bottom on PXA270 to enable the autoboot.

6 Development Environment on PXA270

6.1 Set up NFS on PXA270

· Mount to the nfs server (nfsd) on Linux:

```
> portmap&
> mount < ip2 >:<absolute path to export point> /mnt
```

6.2 Cross-Compile

A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is run. To compile a program running on your PXA270, you need a cross compiler that can generate ARM9 object codes. So, please follow the instructions to compile your program for PXA270:

· Prepare your source code. Here we use hello.c as an example.

· Use cross-compiler to build up executable file for PXA270:

```
SHELL> arm-unknown-linux-gnu-gcc -o hello hello.c
-L /opt/arm-unknown-linux-gnu/arm-unknown-linux-gnu/lib/
-I /opt/arm-unknown-linux-gnu/arm-unknown-linux-gnu/include/
```

6.3 Execute Program on PXA270

· Move or copy the executable file to <export point>:

```
SHELL> mv hello <absolute path to export point>
```

```
SHELL> cp hello <absolute path to export point>
```

· Execute program under NFS mount point on PXA270:

```
> cd /mnt
> ./hello
```

Lab 1 Assignments

1. Write a C program to print your student id and name.
2. Cross compile and download the program to your PXA270 via NFS. Execute the program.
3. (Optional) Put your program to your rootfs and flash the new rootfs with your PXA270 image.