CSEE 4840
Embedded System Design
Tutorial: Configuring Linux: Kernels and All That

Stephen A. Edwards
Columbia University

2015

The Linux system on our SoCKit boards consists of three pieces:

1. The Linux Kernel

   This is the big C program that actually runs on the ARM processors. It's responsible for running multiple processes, managing resources such as memory, communicating with hardware, providing filesystems, and so forth.

   The Linux kernel is due to Linus Torvalds and many others; the various versions can be downloaded from `http://www.kernel.org`. Versions are numbered, such as "3.8."

   In our environment, the kernel resides in file called "uImage."

2. The Root Filesystem

   This is the root directory "/" and all the files under it, including all the programs under /bin; all the libraries in /lib; configuration files in /etc; /root, the home directory for the superuser; the device files under /dev; and a few others.

   The contents of the root filesystem typically starts from a distribution, such as Debian, Ubuntu, or Linaro.

3. A Bootloader

   On the SoCKit board, there are actually two: a first-stage bootloader responsible mostly for configuring the DDR memory and loading the rather elaborate second-stage bootloader "u-boot" that has a command-line interface, the ability to load the Linux kernel into memory from a FAT filesystem on an SD card or over the network.

1 The Configuration for Lab 2

1.1 *The Linux Kernel*

I used the modified Linux kernel (with support for the Altera framebuffer) prepared for the RocketBoards Linaro example.[1]

This requires the ARM cross compiler that comes with the Altera tools. The easiest way to ensure this is in your path is to run */opt/altera/quartus-13.1/embedded/embedded_command_shell.sh*.

```
git clone https://github.com/altcrauer/linux.git
cd linux
git checkout -b neek_soc_38 origin/arrow_sockit_vga
export ARCH=arm
export CROSS_COMPILE=arm-none-eabi-
export LOADADDR=0x8000
make socfpga_defconfig
make menuconfig
make uImage
make socfpga_cyclone5.dtb
```

This creates the Linux kernel image *arch/arm/boot/uImage* and the device tree blob *arch/arm/boot/dts/socfpga_cyclone5.dtb*.

For lab 2, I disabled the virtual terminal, which normally uses the framebuffer to display text as it is booting and running. To do this, after *make menuconfig*, I disabled Device Drivers → Character devices → Virtual terminal.

1.2 *The Root Filesystem*

I used a Linaro root filesystem I selected from http://www.linaro.org.

```
wget http://releases.linaro.org/14.01/ubuntu/saucy-images/nano/\
  linaro-saucy-nano-20140126-627.tar.gz
tar zxf linaro-saucy-nano-20140126-627.tar.gz
```

This produces a root filesystem in a directory called “binary.” For the full effect, the “tar” command needs to be run as root to get all the file permissions right. Also, only root can create the device nodes in the /dev directory.

[1] http://www.rocketboards.org/foswiki/Projects/SoCKitLinaroLinuxDesktop

2 The Configuration for Lab 3

2.1 The Linux Kernel

```
git clone git://git.rocketboards.org/linux-socfpga.git
cd linux-socfpga
git checkout -b lab3-kernel-3.8 origin/socfpga-3.8
export ARCH=arm
export CROSS_COMPILE=arm-none-eabi-
export LOADADDR=0x8000
make socfpga_defconfig
make uImage
make socfpga_cyclone5.dtb
```

2.2 The Root Filesystem

```
wget http://releases.linaro.org/13.11/ubuntu/raring-images/\
  nano/linaro-raring-nano-20131124-562.tar.gz
tar zxf linaro-raring-nano-20131124-562.tar.gz
```

2.3 More Configuration

After booting the board to the Linux command prompt, the network needs be configured:

```
route add default gw 192.168.1.1
echo "nameserver 128.59.1.3
nameserver 128.59.1.4" > /etc/resolv.conf
```

This should enable *apt-get* to install tools such as *make* and the C compiler:

```
apt-get update
apt-get install -y make gcc
```