el32-buildroot

Buildroot fork for EL32 Embedded Linux course.

This repository contains support for building an Embedded Linux distribution (kernel, rootfilesystem) and a crosscompilation toolchain.

Setup

Install required tools:

```
$ sudo apt-get install -y build-essential libncurses5-dev whois git subversion parted kpartx extlinux q
```

Clone this repository:

```
git clone --depth 1 https://github.com/fhict/el32-buildroot.git buildroot
```

Soekris NET4801

Build an Embedded Linux distribution and crosscompilation toolchain for the Soekris NET4801 board.:

```
$ cd buildroot
$ make soekris_net4801_defconfig
$ make
```

Now you will have to wait quite a while for it to complete. If everything went well a kernel (`bzImage`), a root filesystem (`rootfs`), and a bootable disk image (`soekris.img`) will be available in the directory `output/images/` .

The toolchain generated by Buildroot is located by default in `output/host/` . The simplest way to use it is to add `output/host/usr/bin/` to your PATH environment variable (`export PATH=$PATH:$HOME/buildroot/output/host/usr/bin`). To make this permament enter it in your .bashrc file (`echo 'export PATH=$PATH:$HOME/buildroot/output/host/usr/bin' >> ~/.bashrc`). You can then use the crosscompilers simply as `i586-linux-gcc` (C compiler) and `i586-linux-g++` (C++ compiler) without a path prefix.

The disk image should be written on a CompactFlash (CF) card and inserted in the Soekris board. Writing the image on the CF card can be done as follows:

```
$ sudo umount /dev/sdc*
$ sudo dd if=output/images/soekris.img of=/dev/sdc bs=1M
$ sudo sync
```

Above command assumes the CF card reader is connected as /dev/sdc. Check the output of the command `dmesg | tail` directly after inserting a CF card to see if this is the case. If not change the device name to comply with your configuration.

To test the kernel and rootfs with QEMU:

```
$ qemu-system-i386 -kernel output/images/bzImage -append "root=/dev/sda panic=1 console=ttyS0" -no-rebo
```

This command will boot a virtualized Soekris. In fact it is running the Soekris root filesystem on a virtualized PC.

After booting the real or the virtual Soekris you should be able to login as `root` with password `root` .

Raspberry Pi

Build an Embedded Linux distribution and crosscompilation toolchain for the Raspberri Pi.:

```
$ cd buildroot
$ make raspberrypi_defconfig
$ make
```

Now you will have to wait quite a while for it to complete. If everything went well a kernel (`zImage`), a root filesystem (`rootfs`), and a bootable disk image (`rpi.img`) will be available in the directory `output/images/` .

The toolchain generated by Buildroot is located by default in `output/host/` . The simplest way to use it is to add `output/host/usr/bin/` to your PATH environment variable (`export PATH=$PATH:$HOME/buildroot/output/host/usr/bin`). To make this permament enter it in your .bashrc file (`echo 'export PATH=$PATH:$HOME/buildroot/output/host/usr/bin' >> ~/.bashrc`). You can then use the crosscompilers simply as `arm-linux-gcc` (C compiler) and `arm-linux-g++` (C++ compiler) without a path prefix.

The disk image should be written on a SD card and inserted in the Raspberry Pi. Writing the image on the SD card can be done as follows:

```
$ sudo umount /dev/mmcblk0*
$ sudo dd if=output/images/rpi.img of=/dev/mmcblk0 bs=4M
$ sudo sync
```

Above command assumes the SD card reader is connected as /dev/mmcblk0. Check the output of the command `dmesg | tail` directly after inserting a SD card to see if this is the case. If not change the device name to comply with your configuration.

To test the kernel and rootfs with QEMU first download a suitable qemu compatible kernel:

```
$ wget http://xecdesign.com/downloads/linux-qemu/kernel-qemu
```

Then start QEMU with your new rootfilesystem:

```
$ qemu-system-arm -M versatilepb -cpu arm1176 -m 256 -kernel kernel-qemu -append "root=/dev/sda panic=1
```

This command will boot a virtualized Raspberry Pi. In fact it is running the Rapberry Pi root filesystem on a virtualized VersatilePB board.

After booting the real or the virtual Raspberry Pi you should be able to login as `root` with password `root` .