**ECGR 6181/8181 - Lab 3**

**Objective:** Creating a minimalist embedded Linux distribution

**Outcomes:**

After this lab, you will be able to

* Build a Linux kernel image for ARM Versatile board
* Build a root file system with Busybox
* Use Buildroot to build an embedded Linux file system, and test application

**Build Kernel Image**

Create a new Lab3 directory

$ mkdir Lab3

$ cd Lab3

Download the latest stable version of the Linux kernel from www.kernel.org.

$ tar xvf <linux-x.x.x>.gz

Install flex and bison

$ sudo apt-get install flex

$ sudo apt-get install bison

$ cd linux-x.x.x

$ make ARCH=arm versatile\_defconfig

Install ncurses. Used to create text based user interfaces

$ sudo apt-get install libncurses5-dev libncursesw5-dev

Install Linux cross compiler toolchain (Ubuntu/Linaro)

$ sudo apt-get install gcc-arm-linux-gnueabi

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- versatile\_defconfig

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi-

This will start the building of the kernel using the ARM cross compiler (will take some time); the build will create, among other binaries, a compressed kernel in a file called zImage located in

“arch/arm/boot”

Let’s try out the brand new kernel -

$ qemu-system-arm -M versatilepb -kernel arch/arm/boot/zImage -dtb arch/arm/boot/dts/arm/versatile-pb.dtb -serial stdio -append "serial=ttyAMA0"

Kernel panics with message - Unable to mount root fs (file system). Ctrl-Alt-g to get the mouse back from QEMU. Machine - quit on the menu screen.

**Build rootfile system with Busybox**

Download Busybox source from http://www.busybox.net. Version 1.36.1 worked for me.

$ cd ..

$ wget http://busybox.net/downloads/busybox-1.36.1.tar.bz2

BusyBox combines tiny versions of many common UNIX utilities into a single small

executable. BusyBox has been written with size-optimization and limited resources in

mind and is easily customizable for embedded systems.

$ tar xjf busybox-1.36.1-tar.bz2

$ cd busybox-1.36.1

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- defconfig

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- menuconfig

Check the option to build Busybox as a static executable (no shared libs) under Settings. Exit and save.

Now compile.

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi-

$ make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabi- install

We need the following for our minimalist Linux system

init script – Kernel needs to run something as the first process in the system.

Busybox – it will contain basic shell and utilities (like cd, cp, ls, echo etc)

$ cd ..

$ mkdir rootfs

$ cd rootfs

Create a shell script file called init with the following contents

#!/bin/sh

mount -t proc none /proc

mount -t sysfs none /sys

mknod -m 660 /dev/mem c 1 1

echo -e "\nHello!\n"

exec /bin/sh

Make it executable

$ cd ..

$ chmod +x rootfs/init

Copy Busybox utilities

$ cp -av busybox-1.36.1/\_install/\* rootfs/

Create a standard directory layout

$ mkdir -pv rootfs/{bin,sbin,etc,proc,sys,usr/{bin,sbin}}

Create compressed filesystem images, often for use in embedded systems or as an initial RAM filesystem (initramfs) for Linux kernels.

$ cd rootfs

$ find . -print0 | cpio --null -ov --format=newc | gzip -9 > ../rootfs.cpio.gz

$ cd ..

Explanation -

The command creates a compressed CPIO (Copy In, Copy Out) archive of the current directory and its subdirectories, then saves it as `rootfs.cpio.gz` in the parent directory.

Here's a breakdown of each part:

- `find . -print0`: Searches the current directory (`.`) and its subdirectories, printing the pathnames separated by null bytes (`-print0`).

- `|`: Pipes the output of the previous command as input to the next command.

- `cpio --null -ov --format=newc`: Takes the pathnames from `find`, and packs them into a CPIO archive. Flags are:

- `--null`: Reads null-terminated filenames (matching `-print0` from `find`).

- `-o`: Creates a new archive (output).

- `-v`: Verbose; print files as they are added.

- `--format=newc`: Use the `newc` archive format, commonly used for initramfs.

- `|`: Again, pipes the output to the next command.

- `gzip -9`: Compresses the archive using Gzip with maximum compression (`-9`).

- `> ../rootfs.cpio.gz`: Redirects the compressed archive to `rootfs.cpio.gz` in the parent directory (`..`).

Boot the kernel with QEMU

$ qemu-system-arm -M versatilepb -kernel linux-x.x.x/arch/arm/boot/zImage -dtb linux-x.x.x/arch/arm/boot/dts/arm/versatile-pb.dtb -initrd rootfs.cpio.gz -serial stdio -append "root=/dev/mem serial=ttyAMA0"

You should see the # shell prompt! Try out a few Linux commands.

Verify we are on an ARM926EJ-S CPU

# cat proc/cpuinfo

Congratulations, you have successfully booted into your own embedded Linux distribution!

**To do (optional) -**

Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation. Follow the instructions in this video tutorial from Embedded Craft that demonstrates how to use buildroot to boot Linux kernel on VersatilePB with an ext2 file system, and then run a “Hello World” application. Please document your steps for future reference.

<https://www.youtube.com/watch?v=oy5PtFhVk5E>