ECEN 5013

Assignment 3: Manual Kernel and Root Filesystem Build

Github Classroom Link

https://classroom.github.com/a/nmp_D6e2

Github Classroom Start Instructions

See Github Classroom Assignment Start Instructions to setup your submission repository based on Assignment 2 and the aesd-assignments repository.

Suggested Reading:

1. Lecture 6
2. Mastering Embedded Linux Programming Chapters 4 and 5.

Implementation:

1. Modify your tester.sh script to remove the cross compile and make step.

Add a BASH script "**manual_linux.sh**" to your repository which uses the MELP crosstool-ng toolchain to build a barebones kernel and rootfs and boots using QEMU. **Your manual_linux.sh script** should do the following:

2. Take a single argument **outdir** which is the location on the filesystem where the output files should be placed.
    a. If not specified, your script should use /tmp/ecen5013 as **outdir**
3. Create directory **outdir** if it doesn't exist. Fail if the directory could not be created.
4. Build a kernel image using instructions in MELP Chapter 4.
    a. Use git to clone the linux-stable kernel source tree if it doesn't exist in **outdir**. Checkout v5.1.10
    b. Use versatile_defconfig to create the initial .config file.
    c. Make zImage, modules, and dtbs with "Building a kernel for QEMU" commands in MELP chapter 4
    d. Copy resulting files generated in step 3.c to **outdir.**
5. Your script should build a root filesystem in **outdir/**rootfs as described in MELP chapter 5, performing all of these operations in an automated fashion
    a. Setup the staging directory tree.

b. Checkout and build busybox version 1_31_stable instead of the one mentioned in MELP chapter 5..
    i. Use CROSS_COMPILE=arm-unknown-linux-gnueabi-
    ii. Use CONFIG_PREFIX to install to outdir/rootfs
c. Copy in the necessary libraries as described in “Libraries for the root filesystem”
d. Add minimal device nodes for /dev/null and /dev/console
e. Cross compile your writer application from Assignment 2 and place in the outdir/rootfs/home directory
f. Copy your finder.sh and (modified as described in step 1 above) tester.sh scripts from Assignment 2 into the **outdir/**rootfs/home directory
g. Create a standalone initramfs and **outdir**/initramfs.cpio.gz file based on the contents of the staging directory tree.
h. Start qemu-system-arm using the kernel, dtb, rootfs, and initramfs files in **outdir**

Validation:

1. Your script should completely build or rebuild all components and start QEMU in a new directory/existing directory **outdir** with the installed kernel. It should not require or use interactive content from the user beyond the **outdir** command line argument when run the first time on a new outdir.
2. You should be able to cd to the /home directory and run ./tester.sh from your QEMU console prompt, getting a success response.
3. Your writer application should run successfully (after being cross compiled successfully) inside QEMU.
4. You should submit your modified manual_linux.sh script and any necessary scripts called from it in your assignment repository.

Note:

- You may see an exception message on boot with content like this:

```
WARNING: CPU: 0 PID: 1 at drivers/gpu/drm/drm_atomic_helper.c:1429
drm_atomic_helper_wait_for_vblanks.part.1+0x250/0x268 [CRTC:31:crtc-0] vblank
wait timed out Modules linked in: CPU: 0 PID: 1 Comm: swapper Tainted: G W
5.1.10 #1 Hardware name: ARM-Versatile
…
Exception stack(0xcf821fb0 to 0xcf821ff8) 1fa0: 00000000 00000000 00000000
00000000 1fc0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 1fe0: 00000000 00000000 00000000 00000000 00000013 00000000
```

  Please ignore this error message on the current assignment. We’ll address in future assignments.