Linux for Embedded Systems

Lab 5 - Report

By Noman Noor (id: 302343)

Current Date: 22/06/2022 (dd/mm/yyyy)

PROBLEM DESCRIPTION

Task: Make the Lab RPi4 Music Player but using OpenWRT instead of Buildroot this time.

Setting up OpenWRT

After booting into the provided Recovery OS on the Lab Raspberry Pi 4, we change working directory to some directory on the initram/ramdisk file system to work on this. I chose "/root".

```
cd /root
```

Then we download the pre-compiled factory image of OpenWRT for Raspberry Pi 4:

```
wget
https://downloads.openwrt.org/releases/21.02.3/targets/bcm27xx/bcm2711/open
wrt-21.02.3-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz
```

However, the version of the wget on the provided Recovery OS didn't have https support (not even using the flag "--no-check-certificate"), so I had to download it on my lab PC and then wget it from there (like usual, using Python -m http.server on the lab PC).

Then, we decompress the archive as follows:

```
gzip -d openwrt-21.02.3-bcm27xx-bcm2711-rpi-4-ext4-factory.img.gz
```

Note that after this only the contents of the archive will remain, but the archive itself will be deleted.

Now, we create loop devices in order to only get the kernel image, device tree, cmdline.txt, and the rootfs image.

```
losetup -fP openwrt-21.02.3-bcm27xx-bcm2711-rpi-4-ext4- factory.img
```

Then we mount some partitions as follows:

```
mkdir /tmp/a ; mount /dev/loop0p1 /tmp/a
mkdir /tmp/d ; mount /dev/mmcblk0p1 /tmp/d
```

Next, we copy the kernel image, device tree, and cmdline.txt to the correct locations for our lab RPi boot setup.

```
cp /tmp/a/kernel8.img /tmp/d/user/Image
cp /tmp/a/bcm2711-rpi-4-b.dtb /tmp/d/user
cp /tmp/a/cmdline.txt /tmp/d/user
```

Then, using fdisk we delete the second partition of the SD Card and create new partitions as we desire. I wanted only one partition, so I made the second partition start from the end of the vfat boot partition to the end of the SD Card. However, you may want to reserve some space for any partitions you might want to create in the future.

Afterwards, we flash the small rootfs image from the OpenWRT image (which was for the whole device (our SD Card in this case)) with a suitable block size for SD Card reliability/health:

```
dd if=/dev/loop0p2 of=/dev/mmcblk0p2 bs=4096
```

NETWORK CONFIGURATION

Since OpenWRT is mostly built for networking devices like routers, its networking behavior may appear suspicious to the faculty network's security. As such, it may cause various problems. Therefore, we disconnect the RPi from the ethernet port and reboot in order to rework the configuration. To do so, we may just edit the configuration file (/etc/config/network) using a text editor, but I chose to use the following commands (note that we can't use the LuCI web interface of the OpenWRT system because it is not connected to the network right now for reasons described earlier in the paragraph):

```
uci delete network.lan
uci set network.lan=interface
uci set network.lan.proto='dhcp'
uci commit
```

Then, restart the network with the new settings:

```
/etc/init.d/network restart
```

I also precautionarily rebooted the whole device before connecting the ethernet cable again.

Connect your ethernet cable now, and you’re done.

PROGRAMS AND PACKAGES

Now, there are several ways to install new programs (e.g. from lab PC using the LuCI web interface on {RPI4_IP_ADDRESS} (the port was :80, if needs to be manually appended)), but we’ll be using OpenWRT’s package manager, opkg.

Note that, since at least some of the data of opkg (including package lists, etc.) are directly or indirectly (by symlinks or hardlinks) stored somewhere in the temporary directory /tmp, the following command needs to be used after every reboot:

```
opkg update
```

Then, we install packages using:

```
opkg install {package_name}
```

The packages I installed (I think I remember all of them, but no guarantees) had the following package names:

1. gpiod
2. alsa-utils
3. mpc
4. mpd (or “mpd-full”?)
5. python3 (or just “python”?)
6. python3-pip
7. python3-flask
8. python3-flask-login
9. python3-gpiod

(I’m not 100% certain because I don’t have access to a system with opkg right now and the OpenWRT website’s searchable package list is having problems at least as of the moment I’m writing this report.)

SETTING UP THE MUSICPLAYER

Before I get into changes I made, the archive that will be in the same directory as this report is sort of an overlay. You download it to the “/” directory using wget, and use tar -xzf {archive_filename} to extract it right there and act as an “overlay” by replacing existing configuration files, and add new directories and files (e.g. the music ,etc.) as needed.

The above procedure also adds the script “musicplayer” to “/etc/init.d/” but OpenWRT is different from buildroot in how init.d is handled. Instead of having the startup scripts “S##scriptname” in init.d and have them always start, OpenWRT handles this differently. I used the following resource to write the “musicplayer” script overlaid into /etc/init.d : [OpenWrt Wiki] Init Scripts.

Then we had to use the following command to actually enable it at start-up (can also be done from LuCI Web Interface):

```
/etc/init.d/musicplayer enable
```

This sets a symlink from the script to the /etc/rc.d/ directory with the name of the symlink being “S99musicplayer” in my case. The /etc/rc.d/ in OpenWRT is equivalent of /etc/init.d in Buildroot, because OpenWRT comes with support for enabling and disabling init scripts (as long as they are in init.d directory, which just serves as a “database” of sorts of every possible script).

Then reboot, and we’re done.

Note: there were other changes that had to be made to the way the program itself works because in OpenWRT /var is a symlink to /tmp (at least by default). Therefore, the files I was overlaying into /var on buildroot needed to be moved to a different directory. I chose the “/lol” directory and edited my programs and configurations accordingly.

Side note: I should have also added the command “amixer set PCM 100%” to init.d like I had for Buildroot. In fact, OpenWRT handles this better because no matter what device (e.g. headphones) is connected, PCM is the name used for volume control and as such the volume setting is global. Therefore, unlike in Raspberry Pi, where my script required the headphones to be plugged in at the start to set volume to 100% on the sound device itself (the volume buttons used by the “music player” will still work as they correspond to MPC’s volume setting) but here it would set it too 100% regardless of if the headphone was plugged in on startup or not.

: FINISHED :