Guide for installing Litmus-RT on a Raspberry PI 3

Mercea Otniel Bogdan

Politehnica University of Timisoara

Prerequisites:

- Necessary hardware: a microSD card (min. 8 GB)
- microSD card reader
- A Linux host set up of kernel build
- Good knowledge of linux kernel compiling
- Raspberry PI3 ArchLinux credentials:
    - User: alarm -> password: alarm
    - User: root -> password: root

1. Install ArchLinux on Raspberry PI 3

a) Install the Linaro Toolchain

- Get the latest Linaro Toolchain for ARM64 (AARCH64) architecture running on a x86_64 host. The latest version can be found at: https://releases.linaro.org/components/toolchain/binaries/latest/aarch64-linux-gnu/

```
wget https://releases.linaro.org/components/toolchain/binaries/latest/aarch64-linux-gnu/gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu.tar.xz
```

- Extract the archive

```
tar xvvJf gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu.tar.xz
```

- Make sure the toolchain is added to your PATH environment variable. An example can be found below (assuming the toolchain was extracted in the home directory). It is recommanded to add this line to your .bashrc file in your home directory

```
export PATH=$PATH:$/home/user/gcc-linaro-7.2.1-2017.11-x86_64_aarch64-linux-gnu/bin
```

b) Install arch linux on Raspberry PI 3:

a) Create partition table

```
fdisk /dev/sdX
```

At the fdisk prompt,delete old partitions and create a new one:

- Type o. This will clear out any partitions on the drive.
- Type p to list partitions. There should be no partitions left.
- Type n, then p for primary, 1 for the first partition on the drive, press ENTER to accept the default first sector,

- then type +100M for the last sector.
- Type t, then c to set the first partition to type W95 FAT32 (LBA).
- Type n, then p for primary, 2 for the second partition on the drive, and then press ENTER twice to accept the default first and last sector.
- Write the partition table and exit by typing w.
- After the partition table has been written, exit fdisk and issue a `sync` command

The resulted partition table should look similar to this:

```
Disk /dev/sdd: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xab9c8d91
Device         Boot  Start      End  Sectors  Size Id Type
/dev/sdd1        2048   206847   204800  100M  c W95 FAT32 (LBA)
/dev/sdd2      206848 62333951 62127104 29.6G 83 Linux
```

b) Create the filesystems

- Create the filesystem for the boot partition

```
mkfs.vfat /dev/sdX1
```

- Create the filesystem for the root partition

```
mkfs.ext4 /dev/sdX2
```

- Issue a sync command to make sure the changes have been written to the microSD card

```
sync
```

- Check for write errors in kernel messages log (dmesg). If any write errors are present for your sdX driver, change to a new microSD card and repeat the process

c) Mount the filesystems on the microSD card

- Create a mount point. This example will use a mount point in the home directory:

```
mkdir sdmount
mkdir sdmount/boot
mkdir sdmount/root
```

- Mount the filesystems

```
mount /dev/sdX1 ~/sdmount/boot
mount /dev/sdX1 ~/sdmount/root
```

- Verify that the partition have been mounted properly and insure that they were mounted in read/write mode. If not, change to a new microSD card and repeat the process

d) Download and extract the root filesystem

```
wget http://os.archlinuxarm.org/os/ArchLinuxARM-rpi-3-latest.tar.gz
tar -xpf ArchLinuxARM-rpi-3-latest.tar.gz -C /~/sdmount/root
sync
```

e) Move boot files to the first partition:

```
mv ~/sdmount/root/boot/* ~/sdmount/boot
```

f) Unmount the two partitions:

```
unmount ~/sdmount/root
unmount ~/sdmount/boot
sync
```

g) Check for any errors regarding your microSD card in kernel message log (dmesg)

h) Test your new ArchLinux installation on a Raspberry PI3. (insert the microSD card and power on your Raspberry PI 3). The ArchLinux system should boot.

- In some cases a new generation of the modules.dep and map files may be required (if for example networking does not start…). In this case execute the `depmod` command on your RPI3 system with rool privileges and reboot

```
[root@alarm ~]# depmod
[root@alarm ~]# reboot
```

i) Upgrade your system to the latest version and reboot

```
[root@alarm ~]# pacman -Su
```

2. Compile and Install the Linux Kernel with the LITMUS^RT patch. This tutorial will use the LITMUS^RT extensions on top of the Linux Kernel version 4.9.30

a) Get the Linux Kernel:

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
cd linux-stable
git checkout v4.9.30
```

b) Apply the LITMUS^RT patch

```
git remote add litmus https://github.com/LITMUS-RT/litmus-rt.git
git fetch litmus
git remote -v
git cherry-pick v4.9.30..litmus/linux-4.9-litmus
```

c) Copy the .config file in the linux-stable folder (the root of the the kernel) from the Raspberry PI 3. A default .config file is present in the archive

- In order to extract the .config from the newly running system of the RPI3 ArchLinux system, copy the /proc/config.gz file from RPI to your Linux system (maybe via scp) used to compile the kernel. Obtain the .config file by extracting it to the linux-stable directory. Example:

```
zcat config.gz > ~/linux-stable/.config
```

- Insure that CONFIG_MODULES=y.

```
cat ~/linux-stable/.config | grep "CONFIG_MODULES"
```

d) Configure the kernel using menuconfig – in the linux-stable directory

```
cd ~/linux-stable
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- menuconfig
```

- To configure the kernel, the following steps are necessary:
  I. Add a recognizable local version, such as –litmus (or whatever you want). It is used to show that you are running the compiled kernel. This will be necessary after we have the running kernel on the board in order to insure what kernel is currently running. This can be entered under **General setup->Local version** - append to kernel release.
  II. Enable in-kernel preemptions. This can be set under **Kernel features->Preemption model**. Choose **Preemptible Kernel (Low-Latency Desktop)**.
  III. Disable group scheduling. First, disable the **Automatic process group scheduling** option under **General setup**. Second, under **General setup->Control group support->CPU controller**, disable **Group scheduling for SCHED_OTHER**
  IV. Disable frequency scaling and power management options that affect timer frequency. Under **General setup->Timers subsystem->Timer tick handling**, set the option to **constant rate, no dynticks**. Under **Power management** options, make sure that **Suspend to RAM and standby, Hibernation and Opportunistic sleep** are disabled. Under **CPU Power Management->CPU Frequency scaling**, disable **CPU Frequency scaling**.
  V. When planning to do development, enable tracing in LITMUS^RT. Under **LITMUS^RT->Tracing**, enable **TRACE() debugging.** Note that this is a high-overhead debug tracing interface that must not be enabled for any benchmarks or production use of the system

e) Compile the kernel. The **–j** parameter instructs the make system how many CPUs to use for compiling the kernel. Adjust this parameter according to your host Linux system's hardware capabilities

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- -j 4
```

f) Insert the microSD card into the Linux host system and mount the filesystems according to step 1.c)

```
mount /dev/sdX1 ~/sdmount/boot
mount /dev/sdX1 ~/sdmount/root
```

g) After compilation copy Image and bcm2837-rpi-3-b.dtb from the linux folder to the mounted SD card

```
cp ~/linux-stable/arch/arm64/boot/dts/broadcom/bcm2837-rpi-3-b.dtb
~/sdmount/boot/dtbs/broadcom
```

h) Copy the newly compiles linux image

```
cp ~/linux-stable/arch/arm64/boot/Image ~/sdmount/boot/
```

i) Install kernel modules:

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- INSTALL_MOD_PATH=~mak/sdmount/root
modules_install
```

j) Invoke a force sync and unmount the filesystems

```
sync
unmount ~/sdmount/boot
unmount ~/sdmount/root
```

k) Boot the new kernel. After login check if the kernel version matches along with the local version suffix entered in step 2.d)

NOTE: There are some situations where after these steps the system does not start. The systems boots the kernel but it reports that it cannot mount the root partition of the sdcard. If this situation occurs, the only solution is to start this guide all over again.

l) After boot, on your RPI3 system, using root privileges, run depmod Invoke a force sync and after reboot.