General Information

- Board Name: Allwinner Nezha
- More Information: https://linux-sunxi.org/Allwinner_Nezha

Building Kernel Modules (on device)

- No kernel sources available via apt
- (At least similar) Kernel sources available at https://github.com/Tina-Linux/tina-d1x-linux-5.4
- Prepare kernel sources accordingly (run commands inside cloned root kernel directory):

  ```
  make mrproper
  zcat /proc/config.gz > .config
  make LOCALVERSION= oldconfig
  # If questions appear, just hit enter until it's silent.
  make LOCALVERSION= modules_prepare
  ```

  The `LOCALVERSION=` variable is required to avoid the annoying `+` suffix in the `vermagic` of the module that we build afterwards. (See https://stackoverflow.com/a/21078699 and https://linux.die.net/lkmpg/x380.html)

  Probably, a less hacky way would be recompiling and using the whole kernel. But since the partitions of the vendor image seem to be *interesting*, probably, we would also need to rebuild the whole filesystem...

- Navigate to the directory of the kernel module that you want to compile and run the following command:

  ```
  make -C /<path>/<to>/tina-d1x-linux-5.4 M=$PWD LOCALVERSION= modules
  ```

- Be happy. If errors occur when loading the module, use `modinfo` and `uname -a` to verify that the `vermagic` values match.

Building Kernel Modules (cross compiler)

Host Information

```
fir3@fir3:~/michael/PTEditor-RISCV/module$ uname -a
Linux fir3 4.19.0-22-amd64 #1 SMP Debian 4.19.260-1 (2022-09-29) x86_64
GNU/Linux
fir3@fir3:~/michael/PTEditor-RISCV/module$ cat /etc/debian_version
10.13
```

Building the cross compiler

- Debian provides `riscv64-linux-gnu-gcc` in the APT upstream. At the time of writing this, we cannot use that for compiling kernel modules for our board, because the *v*-support (RISC V v-suffix = vector instructions) is missing in the GCC, but our board chip needs that.

  ```
  root@sipeed:~# cat /proc/cpuinfo
  processor       : 0
  hart            : 0
  isa             : rv64imafdcvu
  mmu             : sv39
  ```

- Hence, we need to build a cross compiler that provides this support. After some googling, I found this SO answer: https://stackoverflow.com/a/71730953. Note that initially I skipped reading the answer and did not change any branches. Since everything worked out, I expect that they answer is outdated.

  A the time of cloning, HEAD was `29d02b75fb6c0b664af56011d8292d1e71c96913`.

- *If you are working on a remote, e.g. via SSH, use* `screen` *for this step to avoid connection loss leading to premature build termination. On my virtualized Debian, this step took about 1.5h.*

  Follow Prerequisites and Installation (Linux). I installed stuff to `/opt/riscv`, like suggested in the `README.md`.

Compiling kernel modules

- See *Building Kernel Modules (on device)* for the adjusted kernel git repo.

- Prepare kernel sources accordingly.

  ```
  # PWD = Kernel repo root directory
  make mrproper

  # Somehow copy the kernel configuration from the board to the this
  directory, e.g.
  # via SFTP:
  #
  # curl -s sftp://root:licheepi@localhost:22222/proc/config.gz | zcat -c >
  .config

  make ARCH=riscv CROSS_COMPILE=/opt/riscv/bin/riscv64-unknown-linux-gnu-
  LOCALVERSION= oldconfig

  make ARCH=riscv CROSS_COMPILE=/opt/riscv/bin/riscv64-unknown-linux-gnu-
  LOCALVERSION= modules_prepare
  ```

  See *Building Kernel Modules (on device)* for more explanation or information.

- Navigate to the directory of the kernel module that you want to compile and run the following command:

  ```
  make -C /<path>/<to>/tina-d1x-linux-5.4 M=$PWD ARCH=riscv
  CROSS_COMPILE=/opt/riscv/bin/riscv64-unknown-linux-gnu- modules
  ```

- The end.