Booting an Arm-powered Embedded Linux system

The booting process starts with a boot loader.

Boot loader process:

There are usually two boot loaders running before the Linux kernel starts running:

1. The first-stage boot loader, which is hard-coded into the processor. At power-up, the processor goes to a certain memory address, reads the content of the address, and performs a jump to the address.
2. The second-stage boot loader, which is a program (e.g., U-Boot) that can load the operating system (i.e., the Linux Kernel.) After the kernel is loaded, the boot loader is discarded.

Boot loaders are also act as an interface to flash the system (i.e., writing data to the flash memory of the system.)

Linux kernel process, which can be separated into two subprocesses:

1. the Linux boot-up process that initializes hardware by performing:
   1. processor-level configuration, such as configurating the memory management unit (MMU), enabling the processor’s cache, and populating the processor’s data structure (i.e., in /proc/cpuinfo).
   2. board-level configuration: initialized peripherals
2. the kernel startup process that
   1. starts kernel’s threads and process management
   2. parses the command line
   3. runs the main kernel process
   4. mounts the root file system
   5. executes an init program (i.e., /sbin/init, /etc/init, /bin/init, and /bin.sh)

The init process