Embedded Linux Device Drivers

Module

- Contents
    - Linux Kernel compilation
    - Embedded Linux
    - Linux kernel module programming
    - Linux device driver programming
        - Pseudo char device driver
        - Platform device drivers
        - Kernel data structures
        - Interrupt handling
        - Synchronization & waiting queue
        - Kernel memory management
        - IO Port access
        - Linux driver model
        - Driver debugging techniques
        - Time management
        - USB device drivers
        - GPIO, SPI & I2C Device drivers
- Evaluation
    - Theory exam: 40 marks MCQ -- CCEE
    - Lab exam: 40 marks
    - Internal exam: 20 marks
- Setup
    - Most of device drivers can be implemented on one of the following
        - Native Ubuntu Linux
        - VM - Ubuntu Linux
        - Beaglebone Black/Raspberri Pi -- GPIO, I2C & SPI device drivers
- Pre-reqisite
    - Embedded Operating System -- File system architecture (VFS)
    - Micro-controller programming -- GPIO, UART, I2C, SPI
    - C Programming -- Function pointers, Structures (member offset)
- Books
    - Linux Device Driver (2.6.10)
    - Professional Linux Kernel Architecture (Device Drivers, VFS, Module programming) (2.6.x)
    - Linux Kernel Develoment (Linux Internals - 2.6.34)
    - Linux Kernel In Nutshell (Kernel compilation - Ch 1 to 3)

- Building Embedded Linux (Kernel compilation for Embedded devices)

Linux Kernel

- Linux kernel has static and dynamic components.
- Static components are
  - Scheduler
  - Process management
  - Memory management
  - IO subsystem (core)
  - System calls
- Dynamic components are
  - File systems (like ext3, ext4, FAT)
  - Device drivers
- Static components are compiled into the kernel binary image.
  - They are kernel components.
  - The kernel image is /boot/vmlinuz.
- Dynamic components are compiled into kernel objects (*.ko files).
  - They are non-kernel components.
  - They are located in /lib/modules/kernel-version.
- Static components --> Linux kernel image (vmlinuz)
  - Scheduler, Process mgmt, Memory mgmt, System Calls, ...
  - terminal> make bzImage
    - Compile all static components and create a monolithic kernel image i.e. vmlinux (ksrc/arch/x86/boot/)
    - Compress vmlinux image into vmlinuz (ksrc/arch/x86/boot/) -- so that the file can be loaded quickly into RAM while booting.
    - Once vmlinuz image is loaded at runtime, it is extracted (self-extracted) and further execution continues.
  - terminal> sudo make install
    - Copy binary compressed kernel image (vmlinuz) into /boot directory.
    - Update the grub to include the new kernel entry (into /boot/grub/grub.cfg)
- Dynamic components --> Linux kernel modules (*.ko)
  - File system managers, Device drivers, ...
  - terminal> make modules
    - Compile all dynamic components and create .ko files in respective directories (mainly drivers, fs).
  - terminal> sudo make modules_install
    - Create new directory under /lib/modules/ for the kernel version.

    - Copy all *.ko files into that directory /lib/modules/kernel-version/

Linux Kernel compilation (for PC)

- Necessary tools/packages should be installed on your system.
  - n-curses, gcc, binutils, etc.
  - terminal> sudo apt-get install build-essential libncurses-dev bison flex libssl-dev libelf-dev
- Download appropriate kernel version from www.kernel.org.
- Extract Linux kernel.
  - terminal> tar xvf filepath
    - x: extract the compressed file
    - v: verbose -- display names of all files extracted
    - z or j or p (optional): z - gnu zip (.tar.gz), j - binary zip (.tar.bz2), p - extended zip (.tar.xz)
    - f: file -- path of compressed file
- configure/customize the kernel.
  - For configuration one must have detailed knowledge of underlying hardware.
  - option 1: make defconfig
    - Use default config for given arch.
    - Compiles minimal kernel (and may not include many driers of the peripherals).
    - This kernel may or may not boot on target system.
  - option 2: make config
    - show many questions and user should answer each question (depending on requirement).
  - option 3: make menuconfig OR make gconfig OR make xconfig
    - gconfig -- GTK based graphics (GNOME)
    - xconfig -- Graphical
    - menuconfig -- text based graphics (n-curses)
    - Select config values
    - Select components to be compiled
      - n -- Do not compile [ ]
      - y -- Compile as static component [*]
      - m -- Compile as dynamic component (module) [M]
    - option 4: use existing/well-known config -- follow this
      - Copy known config (usually /boot/config-x.y.z) into kernel source tree as ".config" file.
        - e.g. cp /boot/config-4.15.0-33-generic .config
      - Then: make menuconfig -- change local version and other config (if required).
- compile kernel image (monolithic -- static components)
  - make bzImage
- compile kernel module (dynamic components)
  - make modules
- install/copy kernel modules into /lib/modules/
  - sudo make modules_install
- install/copy kernel image into /boot and make entry into bootloader.
  - sudo make install
- reboot and boot into new kernel