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-requisite
  - 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 Development (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

- 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).

        - Disable Cryptographic API -> Certificates for Signature Checking -> Additional X.509 keys for default system keyring = (keep blank)
- 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

Linux kernel signing

- If your computer secure boot is ON.
- https://gloveboxes.github.io/Ubuntu-for-Azure-Developers/docs/signing-kernel-for-secure-boot.html

Kernel module programming in Distro kernel

- sudo apt install linux-headers-`uname -r`
    - Download kernel headers into /usr/src directory.
    - In /lib/modules/x.y.z-wwww, a "build" symlink points to /usr/src/linux-headers-x.y.z-wwww

printk()

- Print error messages in kernel log.
- printk(KERN_INFO "a log message.\n"); --> Log buffer --> Kernel log file
    - Log buffer is flushed into log file when "\n" is encoutered or buffer is full.
- These messages can be monitored using "dmesg" command.
- cmd> sudo dmesg | tail

Mechanism vs Policy

- Device drivers should be policy-free.

- Device drivers should have Mechanism to operate the device.
- User space application should define/control the policy -- How to operate the device.

Module stacking

- One module is calling functions/variables from another module.
- Refer slides

modprobe tool

- Used to insert or remove modules in kernel.
- The module must be in module-tree i.e. /lib/modules/kernel-version and updated into module dependency database.
- To add modules into module-tree -- cp command.

```
sudo cp *.ko /lib/modules/uname -r/kernel
```

- To update in module dependency database (modules.dep) -- depmod command.

```
sudo depmod
```

- To load module in kernel. Dependency modules are auto-loaded (if not already loaded).

```
sudo modprobe module-name
```

- To unload module from kernel.

```
sudo modprobe -r module-name
```

Load while booting

- Copy module .ko into /lib/modules/kernel-version and update dependency database.
- Make its entry into /etc/modules.

Don't Load while booting

- Make its entry into /etc/modprobe.d/blacklist.conf.

Kernel data structures

- Predefined data structures used in Linux kernel
  - Linked list
  - Circular Queue (FIFO)
  - Red & Black Tree
  - Hash table/map

Circular Queue (FIFO)

- Refer slides
  - struct kfifo

Red & Black Tree

- Self-balancing binary search tree.
- RBTree is more efficient than AVL tree.
- Linux implementation is called as rbtree.
  - struct rb_root
  - struct rb_node
- Used in CFS scheduler (Completely Fair Scheduler).

Hash Table

- Associative data structure (key-value)
- Linux kernel hashtable is designed for mapping an id (number) to some pointer.
  - struct idr

Signing kernel module for Secure Boot

- cmd> touch ~/.rnd
- cmd> vim ~/.openssl.cnf

```
HOME                    = .
RANDFILE                = $ENV::HOME/.rnd
[ req ]
distinguished_name      = req_distinguished_name
x509_extensions         = v3
string_mask             = utf8only
prompt                  = no

[ req_distinguished_name ]
countryName             = IN
stateOrProvinceName     = MH
localityName            = Pune
0.organizationName      = Sunbeam
commonName              = Module Sign
emailAddress            = nilesh@sunbeaminfo.com

[ v3 ]
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer
basicConstraints        = critical,CA:FALSE
extendedKeyUsage        = codeSigning,1.3.6.1.4.1.311.10.3.6,1.3.6.1.4.1.2312.16.1.2
nsComment               = "OpenSSL Generated Certificate"
```

- cmd> cd ~
- cmd> openssl req -config ./openssl.cnf -new -x509 -newkey rsa:2048 -nodes -days 36500 -outform DER -keyout "MOK.priv" -out "MOK.der"
- cmd> sudo mokutil --import MOK.der
- cmd> kmodsign sha512 MOK.priv MOK.der module.ko

Assignments

1. Implement Module Programming codes in Gokhale sir's notes (Hello Module, Split Module, Module Param, Export and Import). -- Practice

2. Implement Pseudo device driver code from Gokhale sir's notes. -- Practice
3. Implement a kernel module, that display all kernel modules (in kernel log).
4. Implement a kernel module, that display all processes starting from current process (in kernel log).