Intro to Linux Drivers

Kernel coding is different!

Can be hard to understand different syntax, functions, advanced C code in kernel!
Topics

1) What is the “Hello world” of Linux drivers?
2) How can we build a driver?
3) No printf()?!! What can we do?
Hello World -- the Driver
Building a Linux Driver
What’s in a Driver?

- printk():
  ```c
  printk(KERN_INFO "Hello world!\n");
  ```

- `module_init()` & `module_exit()` macros: tell kernel our functions to:
  ```c
  static int __init testdriver_init(void) {
    // Driver’s initialization code when loaded
  }
  static void __exit testdriver_exit(void) {
    // Driver’s cleanup code when unloaded
  }

  // Macros telling kernel which functions to run
  module_init(testdriver_init);
  module_exit(testdriver_exit);
  ```
What’s in a Driver?

- What are `__init` and `__exit`?

  ```c
  static int __init testdriver_init(void) {...}
  static void __exit testdriver_exit(void){...}
  ```

  ```c
  module_init(testdriver_init);
  module_exit(testdriver_exit);
  ```

  `__init`: startup only; freed when kernel booted.

  `__exit`: function not needed if modules built into kernel.

- `MODULE_XYZ()`: Macros defining module info

  ```c
  // Information about this module:
  MODULE_AUTHOR("Dr. Evil");
  MODULE_DESCRIPTION("A simple test driver");
  MODULE_LICENSE("GPL");       // Important to leave as GPL.
  ```
Driver Build Demo

(in my directory 12-TestDriver/)

• To Show
  - testdriver.c
  - Makefile
    1) invokes the kernel’s Makefile
    2) kernel re-executes our Makefile
    3) deploy .ko file to NFS public directory
Working with .ko files
Commands

- Commands for working with drivers (.ko files)
  - List loaded modules
    - ..
  - Load module
    - ..
  - Unload module
    - ..
  - View module info
    - ..
  - View strings
    - ..
Demo

- Load drv on target:
  # lsmod
  Columns are: Module, Size, # Used by (and those modules)
  # dmesg

  # insmod daDriver.ko

  # lsmod
  # dmesg

- Remove on target;
  # rmmod daDriver.ko
  # lsmod
  # dmesg

- View driver info
  [ (on host/target) {shows dependencies, vermagic, params}):
  # modinfo daDriver.ko
  # uname -r
  # strings daDriver.ko
printk()

- printk(): kernel's printf; view with dmesg
  - printk(KERN_INFO "Hello %d %s!\n", 1, "world");
  ..

- Log levels in KERNEL/include/linux/kern_levels.h
  - KERN_EMERG ("0") to KERN_DEBUG ("7")
  - Usually use..

- Important messages shown on serial port
  - Set threshold
    # echo 7 > /proc/sys/kernel/printk
  - View threshold
    # cat /proc/sys/kernel/printk
    First number is the console log level.
printk() - cont.

• UBoot Aside
  – You can set log level via Linux’s cmdline from UBoot
    => set bootargs ${bootargs} loglevel=3

• Demo
  – Open serial terminal (shows some messages)
  – View demo_printk.ko
    # insmod demo_printk.ko
Summary

• printk(): Kernel’s printf() to dmesg
  – Uses log levels KERN_EMERG to KERN_DEBUG

• module_init() and module_exit() set entry/exit points for driver

• .ko Commands
  – lsmod, insmod, rmmod, modinfo, strings