EMBEDDED DEVICE DRIVERS

Linux Device Drivers on Beaglebone Black

LKM: Multi-file Modules

- So far, we have created modules
  - With code spanning a single .c file
- What if we want to create a complicated module
  - Whose code cannot fit into a single .c file?

- Multi-file modules
  - Change in the Makefile
    - Inform the Kernel Build machinery
      - That we have multiple .c files
      - Contributing to a single module (.ko)

LKM: Multi-file mod exercise

- Refer ***mod4*** directory
  - The files ***mod41.c*** and ***mod42.c*** contain the src code
    - Notice how code is split between the 2 .c files
    - Also note the change in the ***Makefile***
      - We create a single .ko file: ***mod4.ko***
    - Compile and load the single ***mod4.ko*** on to the BBB
    - Record your observations from ***dmesg***

C: Symbols

- What is a symbol?
  - A variable or function
  - A name representing some space in system memory
    - Which could be data / instructions (code)

- In all programming languages
  - Symbols are ‘labels’ attached to data/code
  - They need to be defined
    - Exactly once – *space is allocated here*
  - They can then be declared / referenced
    - Any number of times – *no space allocation, just a reference*

C: Sharing Symbols (Userspace)

- In C (user space)
  - We share symbols between files in a project
    - By creating them as ***‘global’***
      - In the symbol creating file
    - And referring to them as ***‘extern’***
      - In the symbol accessing file

- In C
  - Symbols are global – *no ‘global’ keyword*
    - If initiated /defined outside any function
  - Symbols are local
    - If initiated / defined inside functions
    - Or defined as file-local by keyword ‘static’

- How does this carry over to kernel modules?

LKM: EXPORT_SYMBOL*

- Till kernel v2.4
    - All non-static symbols were exported
    - To the global kernel space
    - By default

- Kernel v2.6 onwards
    - Only symbols with
        - EXPORT_SYMBOL() / EXPORT_SYMBOL_GPL() macros
    - Are exported to the global kernel space

    - EXPORT_SYMBOL()
        - Exports to all kinds of modules (GPL as well as non-GPL)
    - EXPORT_SYMBOL_GPL()
        - Only exports to modules which are GPL

LKM: Using EXPORT_SYMBOL

- Exporting module
  - Define the symbol (**my_sym**)
    - Variable / function
  - Append this after the definition
    - **EXPORT_SYMBOL(my_sym)** or **EXPORT_SYMBOL_GPL(my_sm)**

- Accessing module
  - Declare the symbol using ***extern***
  - Use / access the symbol

LKM: Symbol share exercise

- Refer ***mod5*** directory
  - The files ***mod51.c*** and ***mod52.c*** contain source code
    - For 2 drivers!
    - Module ***mod51*** exports
      - A char string ***(mod51_string)***
      - A function ***(mod51_func())***
    - Module ***mod52*** uses both these exported symbols
      - By declaring them ***extern*** before usage
    - Observe the Makefile change for creating 2 drivers
      - We get 2 .ko outputs: mod51.ko, mod52.ko
  - Compile and load the modules on BBB
    - Observe the ***Module.symvers*** file

  - Maintain the loading sequence: mod51 followed by mod52
    - Observe ***lsmod*** output
    - Observe the ***dmesg*** output
    - Try unloading mod51 – what happens?
  - Unload both modules; load them in reverse order – what happens?

THANK YOU!