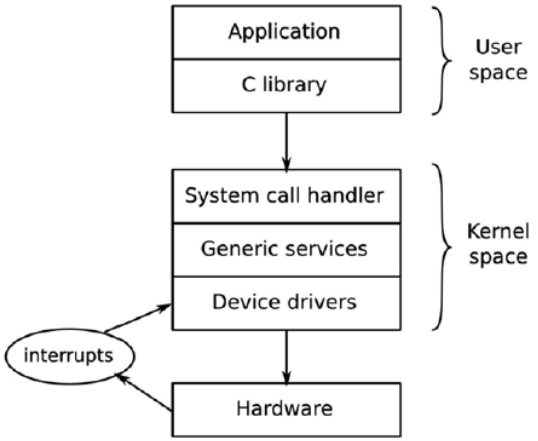
EMBEDDED DEVICE DRIVERS

Linux Device Drivers on Beaglebone Black

The role of the kernel

- Kernel roles
  - Initialize system, control hardware, handle interrupts
  - Operates in **kernel space**
- Application
  - Perform user-defined tasks
  - Operates in **user space / userland**
- Kernel space – User space
  - Bridged by
    - C library, System call interface



Kernel modules: What?

- Kernel modules are object codes that can be
  - Loaded / unloaded into the kernel
  - **On demand**
  - Hence called **‘loadable kernel modules’ (LKM)**
- They extend the kernel functionality
  - In runtime – without a reboot of the running kernel
- Methods to add functionality to kernel
  - Add code to kernel source tree and re-compile
  - Add compiled object code into a running kernel

Kernel modules: Types

- Loadable kernel modules serve various purposes
  - Device drivers
    - Object code for a specific piece of hardware
    - Kernel uses it to deal with that hardware
  - Filesystem drivers
    - These interpret the contents of a filesystem
      - Such as files, directories, etc. on SSD / HDD / USB stick, etc.
      - Examples: FAT, FAT32, NTFS, ext2/3/4, xfs, btrfs
  - System calls
    - Userspace programs use these to get services from kernel
    - Operations: open, read, write, close
    - Most system calls are built in – LKMs can extend functionality

Kernel modules: Advantages

- Kernel functionality can be enhanced
  - In runtime
  - On demand
  - By loading a kernel module
  - Enables ‘plug-and-play’ features
- No need to recompile and reboot kernel (during dev)
- Makes optimal use of computing resources
  - CPU, memory, hardware, etc.
  - By loading only when hardware is present
  - And unloading when hardware is removed

Kernel module vs. User program

- Runs in kernel space
  - Shares address space with kernel
- Has higher execution privileges
  - Since it is part of the kernel
- Does not execute sequentially
  - Registers itself with kernel for future use
- Uses different header files
  - No access to standard C library

- Runs in user space
  - Has private memory address space (provided by kernel)
- Runs at lower privilege levels
  - No access to hardware
- Usually executes sequentially
  - No concept of registration while running
- Uses C library / sys calls
  - With well-documented header files

Kernel module vs. Device driver

- Very little difference as such

- Kernel module is any compiled object code
  - That can be inserted into the running kernel
- Device driver is object code that drives a piece of hardware
  - So high chance for it to be a loadable kernel module

Linux Device Drivers: Types

- Linux kernel device classification:
  - Character
    - Devices that handle data byte-by-byte
      - Examples: keyboard, mouse, serial ports
    - Handled by **character device drivers**
      - Create a char device file (in **/dev**)
      - Open, read, write, close calls on the char file so created
  - Block
    - Devices that handle data in blocks / chunks
      - Examples: Disks (HDD, SSD, USB, CDROM)
    - Handled by **block device drivers**
      - Create a block device file (in **/dev**)
      - Open, read, write, close calls on the block file so created
  - Network
    - Devices that send / receive data on a ‘network’ in ‘packets’
    - Handled by Linux **network subsystem**

THANK YOU!