EMBEDDED DEVICE DRIVERS

Linux Device Drivers on Beaglebone Black

LKM: Major, minor numbers

- The kernel represents char and block devices
    - As pairs of 2 numbers: ***<major>:<minor>***

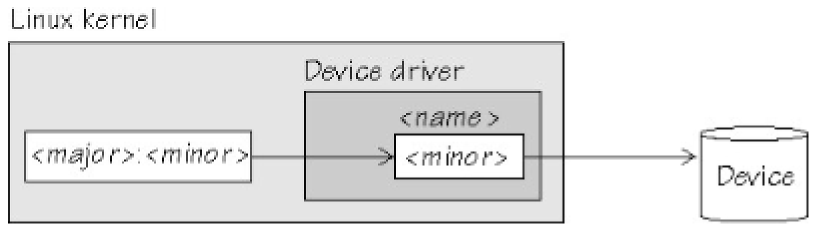
    - Major number
        - Identifies the driver associated with the device
            - Can be shared by multiple device drivers
            - Can be seen by ***cat /proc/devices***

    - Minor number
        - For each device sharing a major number
            - A minor number now identifies it for the device driver
        - The device driver uses it to distinguish individual devices

- Some major numbers reserved for particular device drivers
    - Others are allotted in a dynamic fashion

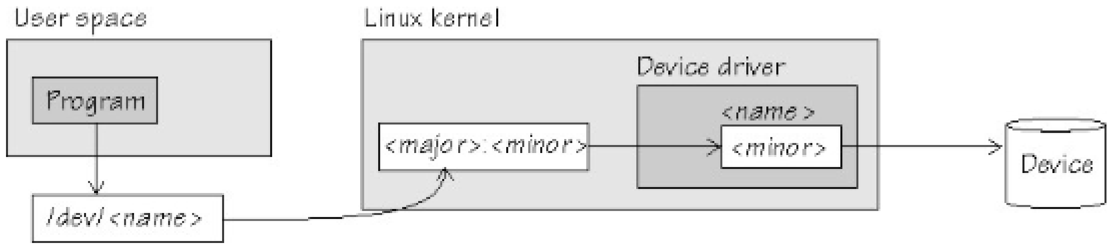
LKM: Major, minor usage

- Kernel allots *major:minor* to device
  - **major** for the device driver
  - **minor** for devices handled by the driver
    - Kernel does not bother about minor number



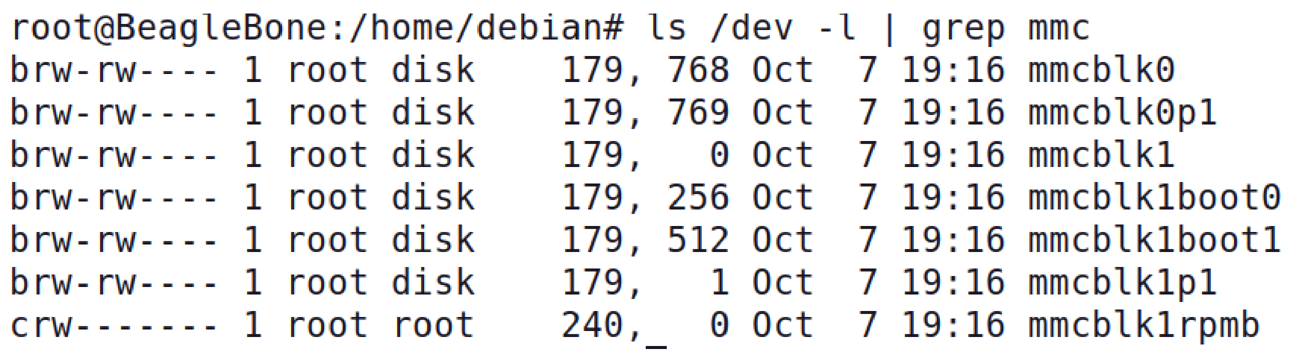
- Device driver creates a device name ***<name>***
  - As per a driver-specific naming scheme

- User space programs access the device
  - Via the ***/dev/<name>*** exposed by the driver
  - Called **device node**



LKM: Major, minor example

- Example on BBB for the MMC block device



```
root@BeagleBone:/home/debian# ls /dev -l | grep mmc
brw-rw---- 1 root disk    179, 768 Oct  7 19:16 mmcblk0
brw-rw---- 1 root disk    179, 769 Oct  7 19:16 mmcblk0p1
brw-rw---- 1 root disk    179,   0 Oct  7 19:16 mmcblk1
brw-rw---- 1 root disk    179, 256 Oct  7 19:16 mmcblk1boot0
brw-rw---- 1 root disk    179, 512 Oct  7 19:16 mmcblk1boot1
brw-rw---- 1 root disk    179,   1 Oct  7 19:16 mmcblk1p1
crw------- 1 root root    240,_  0 Oct  7 19:16 mmcblk1rpmb
```

- Here, 179 is the major number for ***mmcblk***
  - The block device driver then creates names for entities
    - Disks: *mmcblk0, mmcblk1*
    - Partitions: *mmcblk0p1, mmcblk1p1, mmcblk1boot0, mmcblk1boot1, etc.*
  - Minor numbers assigned to all these
    - Used by block device driver (179) only

LKM: Major, minor allocation

- ***dev_t*** datatype holds the *major:minor* pair
    - 32-bit number, defined in ***<linux/types.h>***

- Macro for creating ***dev_t*** from ***major:minor***

  dev_t dev = ***MKDEV****(int major, int minor);*
- Macros for getting the ***major:minor*** from ***dev_t***

  int major = ***MAJOR****(dev_t dev);*

  int minor = ***MINOR****(dev_t dev);*

- Major, minor number pairs can be allocated
    - Statically
        - This method assigns the *major:minor* if it is available
        - Number needs to be known in advance
    - Dynamically
        - Kernel assigns *major:minor* from available pool at runtime

- Header file:

  #include <linux/fs.h>

LKM: Major:minor (static)

- API for obtaining range of major numbers

  *int **register_chrdev_region**(dev_t first, unsigned int count, char *name)*

  - *first*: Starting device number of the range (***dev_t*** variable)
  - *count*: No of contiguous numbers desired
  - *name*: Device name associated with this range
    - Will appear in ***/proc/devices*** and ***sysfs***

  Return value:

    - 0: If successful
    - <0: If not, no range created

LKM: Major:minor (dynamic)

- API for obtaining range of major numbers

  *int **alloc_chrdev_region**(dev_t *dev, unsigned int firstminor, unsigned int count, char *name)*

  - *dev*: Output parameter, holds first number in allotted range
  - *firstminor*: Stating minor number, usually 0
  - *count*: No of contiguous numbers desired
  - *name*: Device name associated with this range
    - Will appear in ***/proc/devices*** and ***sysfs***

  Return value:
    - 0: If successful
    - <0: If not, no range created

LKM: Major:minor static/dynamic

- Static allocation
  - One knows in advance which *major:minor* to allot/use
  - Assumes that the desired range is always free
  - Device nodes can be created in advance

- Dynamic allocation
  - More practical approach, since kernel allocates it based on free pool availability
  - Avoids conflicts with other devices, since kernel handles overlapping requests
  - Device nodes cannot be created in advance
    - *Read /proc/devices to create it*

LKM: Major:minor Unregister

- When the allotted / allocated major:minor range is not in use, it should be freed

  void ***unregister_chrdev_region****(dev_t first, unsigned int count)*

  - *first*: dev_t variable representing the range
  - *count*: No of major:minors obtained

- Usually called in the cleanup / exit part of the module / device driver

LKM: Exercises

- Refer ***mod3*** directory
  - Static allotment
    - Refer ***mod31.c***
    - Compile the module, transfer it to BBB and load it
    - It seeks static allotment for MAJOR 202 for ***cdac_edd*** device
    - Observe the output of **dmesg**
    - Also *cat /proc/devices* | *grep cdac_edd*
    - Unload the module
  - Dynamic allocation
    - Refer ***mod32.c***
    - Compile the module, transfer it to BBB and load it
    - It seeks dynamic allotment for ***cdac_edd*** device
    - Observe the output of **dmesg**
    - Also *cat /proc/devices* | *grep cdac_edd*
    - Unload the module
  - Try loading both mod31 and mod32 together

THANK YOU!