EMBEDDED DEVICE DRIVERS

Linux Device Drivers on Beaglebone Black

LKM: Userspace «» kernelspace

- Methods for interaction between
  - User space and kernel space
  - We already accessed module params thus:
    - ***/sys/module/<module_name>/parameters/<param_name>***
      - *Example: /sys/module/**mod21**/parameters/**myint***
- sysfs (system file-system)
  - Enables interaction between user space and kernel
- Other methods
  - IOCTL
  - procfs
  - debugfs
  - configfs

LKM: sysfs: What?

- Sysfs is a virtual filesytem
  - Exported by the Linux kernel
    - Set of directories and files in a hierarchy
  - Can be accessed from user-space
    - To access / control underlying kernel module / driver
- Always mounted on /sys
- Variety of such “filesystems”
  - To export information from kernel space to user space
    - ***sysfs***: Devices and system specific info
    - ***procfs***: Process and entity specific info
    - ***debugfs***: Debugging specific info
    - **configfs**: Configuration specific info

LKM: Kernel objects

- Kernel object (kobject)
  - Glue that binds the sysfs and the kernel
  - Defined in <linux/kobject.h>

```
struct kobject {
        char *k_name;                   // Object name
        …
        struct kref kref;               // Ref count
        …
        struct kobject *parent;         // Parent object
        …
        struct kobj_type *ktype;        // Ktype of the object
        …
        };
```

LKM: sysfs entry creation (1/2)

- Create a directory entry in /sys
  - Creation API:

    *struct kobject **kobject_create_and_add**(const char *name, struct kobject *parent);*

    - Parent options – create our directory in…
      - *kernel_obj*: /sys/kernel/…
      - *firmware_obj*: /sys/firmware/…
      - *fs_obj*: /sys/fs/…
      - *NULL*: /sys/…

  - Destruction API:

    *void **kobject_put**(struct kobject *);*

LKM: sysfs entry creation (2/2)

- Create a file in our sysfs' directory
  - Creation API:

    *int **sysfs_create_file**(struct kobject*kobj, const struct attribute *attr);*
  - Destruction API:

    *void **sysfs_remove_file**(struct kobject *kobj, const struct attribute *attr);*

- Attribute argument for file creation/deletion
  - Kobject attribute
    - Represented as regular file in sysfs
    - Typical functions used:
      - *show() – for a read*
      - *store() – for a write*

```
struct kobj_attribute {
        struct attribute attr;
        ssize_t (*show)(struct kobject *kobj, struct kobj_attribute *attr, char *buf);
        ssize_t (*store)(struct kobject *kobj, struct kobj_attribute *attr, const char *buf);
        };
```

    - ***__ATTR(name, permissions, show_ptr, store_ptr);***

LKM: sysfs Exercise

- Refer the mod10 directory
  - The file mod10.c contains the module code
    - We create a sysfs directory in /sys/kernel
      - ***/sys/kernel/my_sysfs***
    - We then create an int variable
      - ***/sys/kernel/my_sysfs/sysfs_int***
    - We also define show and store functions
      - And register them through __ATTR
    - Compile and load the module on BBB
    - Observe the sysfs entries so created
      - Perform a read and a write
    - Unload the module

THANK YOU!