1. Host and target setup 1h26m

1. About the instructor
2. Source code and course materials youtube
3. Host and target setup 1

4. Tool-chain download 2

https://releases.linaro.org/components/toolchain/binaries/7.5-2019.12/

5. Important Note
6. Installing gparted application
7. Tool-chain installation and PATH settings 3
8. Note for the students
9. Target preparation : Serial debug setup
10. Important documents
11. Understanding booting sequence of beaglebone black hardware 4

12. Preparing SD card for SD boot
13. Copying boot images to SD card 5
14. Booting BBB via SD card
15. Making SD boot default on BBB by erasing eMMC MBR

16. Updating Linux kernel image 7

pre-built image of the kernel : the kernel version is 4.4.62.

/home/scott/workspace/src/linux_bbb_4.14 3/26 4.14

```
  174  git init
  175  git clone https://github.com/beagleboard/linux.git linux_bbb_4.14
  178  cd linux_bbb_4.14/
  186  git checkout 4.14.108-ti-r130

  443  cd workspace/
  445  cd src
  447  cd linux_bbb_4.14/
```

kernel_image_update_5.10.pdf 5.10-rt

17. Linux kernel compilation

/home/scott/workspace/src/linux_bbb_4.14

18. Modules compilation

```
$ uname -r
  5.3.0-40.generic

$ make -C /lib/modules
  4.14.108/     5.3.0-28-generic/       5.3.0-40-generic/                          -> lecture


  scott@host:~/workspace/ldd/custom_drivers/001hello_world$ ll /lib/modules/
  4.14.108/     5.4.0-150-generic/      5.4.0-84-generic/                          -> 18.04 LTS


    Ubuntu release      Arch        Kernel Version
    ----------------------------------------------
    Ubuntu 20.04 LTS    64-bit x86  5.4 (GA)
    Ubuntu 18.04 LTS    64-bit x86  5.4 (HWE)   Hardware Enablement : the most recent versions of the Linux kernel.
    Ubuntu 18.04 LTS    64-bit x86  4.15 (GA)   General Availabity : the most stable kernel with an original LTS
    Ubuntu 16.04 LTS    64-bit x86  4.15 (HWE)


$ make -C /lib/modules/5.3.0-40-generic/build/ M=$PWD modules
$ ls
  main.c main.ko main.mod mdin.mod.c main.mod.o main.o Makefile modules.order Module.symvers
$ sudo insmod main.ko
$ dmesg
  [164.xxxx] main: loading out-of-tree module tains kernel.
  [164.xxxx] main: module verfification failed: signature and /or required key missing - tainting kernel
  [164.xxxx] Hello world
$ sudo rmmod main.ko
$ dmesg
  [288.xxxx] Good bye world
```

30. Testing of an LKM on target

```
$ sudo make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C /home/scott/workspace/src/linux_4.14/ M=$PWD modules     => 5.3.40 in
lecture
$ file main.ko
  main.ko: ELF 32-bit LSB relocatable, ARM, EABIS version 1 (SYSV), ....
$ modinfo main.ko
  filename:     /path/to/main.ko
  board:            Beaglebone black REV A5
  description:  a simple hello world kernel module
  author:           ME
  license:      GPL
  depends:
  name:         main
  vermagic:     4.14.108 SMP preemp mod_unload modversions ARMv7 p2v8
$
$ arm-linux-gnueabihf-objdump -h main.ko
  main.ko:      file format elf32-littlearm
```

```
  section:
  idx name  size        VMA     LMA     file off algn
  0 .note.
  1 .text
  2 .init.text
  3 .exit.text
  4 .ARM.extab.init.text
  5 .ARM.exidx.init.text
  6 .ARM.extab.exit.text
  7 .ARM.exidx.exit.text
  8 .modinfo
  9 .rodata.str1.4
  10 __version
  11 .data
  12 .gnu.linkonce.this_module
  13 .plt
  14 .init.plt
  15 .bss
  16 .comment
  17 .note.GNU-stack
  18 .ARM.attributes

# sudo insmod main.ko
  [ 200.35xxx] main: loading out-of-tree module taints kernel
  [ 200.35xxx] Hello world
# dmesg | tail
  [ 200.35xxx] main: loading out-of-tree module taints kernel
  [ 200.35xxx] Hello world
# sudo rmmod main.ko
  [ 288.35xxx] Good bye world


/usr # dmesg    <- my BBB

[   10.627696] g_ether gadget: Ethernet Gadget, version: Memorial Day 2008
[   10.639452] g_ether gadget: g_ether ready
[   10.923157] g_ether gadget: high-speed config #2: RNDIS
[  121.502878] random: crng init done
[  215.820416] main: loading out-of-tree module taints kernel.
[  215.826708] Hello world
[  226.387175] Good bye world
/usr #
```

31. Makefile 13

```
KDIR = /lib/modules/$(shelluname -r)/build/     ->       5.3.0-40-generic
```

https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt)

You have to add the Linux kernel module inside the Linux kernel source tree and let the Linux build system builds that.
If you want to list your kernel module selection in kernel menuconfig, then create and use a Kconfig file

. Kconfig file

```
scott@vbox:~/.../src/linux_bbb_4.14/drivers/char/my_c_dev $ ls
 Kconfig    main.c                                                    <= newly added

    menu “my custom modules”
        config CUSTOM_HELLOWORLD
            tristate “hello world module support”
            default m
    endmenu


scott@vbox:~/.../src/linux_bbb_4.14/drivers/char $ vi Kconfig              <= upper level Kconfig

    ...
  source "drivers/char/my_c_dev/Kconfig"

  endmenu




scott@vbox:~/.../src/linux_bbb_4.14/drivers/char/my_c_dev $ vi Makefile

    #obj-<config_item> += <module>.o                                       <= newly added

    obj-$(CONFIG_CUSTOM_HELLOWORLD) += main.o                              // $(CONFIG_CUSTOM_HELLOWORLD) if we don't know
    it's y or n or m.



scott@vbox:~/.../src/linux_bbb_4.14/drivers/char $ vi Makefile             <= upper level Makefile

    ...
    obj-y                          += my_c_dev/

    // To select a kernel module under that menu, we have config item. But, select our folder we don’t have any config item.
```

```
scott@vbox:~/.../src/linux_bbb_4.14 $ make ARCH=arm menuconfig

    Device Drivers --->
      Character devices --->
        fastbit custom modules --->
    <m>    helloworld module support    : dynamically linked. '*' means statically linked.


<help>

There is no help available for this option.
Symbol: CUSTOM_HELLOWORLD [=m]
Type  : tristate
Prompt: helloworld module support
    Location:
    -> Device Drivers
        -> Character devices
        -> fastbit custom modules
    Defined at drivers/char/my_c_dev/Kconfig:2




scott@vbox:~/.../src/linux_bbb_4.14 $ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules -j4

  scripts/kconfig/conf  --silentoldconfig Kconfig
    ...
    CC [M]  drivers/char/my_c_dev/main.o
    Building modules, stage 2.
    ..
    LD [M]  drivers/char/my_c_dev/main.ko



scott@vbox:~/.../src/linux_bbb_4.14/drivers/char/my_c_dev $ modinfo main.ko
    filename:       /home/scott/workspace/src/linux_bbb_4.14/drivers/char/my_c_dev/main.ko
    board:          Beaglebone black REV A5
    description:    A simple hello world kernel module
    author:         Kiran Nayak
    license:        GPL
    depends:
    intree:         Y                                               <= !!!!
    name:           main
    vermagic:       4.14.108+ SMP preempt mod_unload modversions ARMv7 p2v8
```

33. printk : to revisit

3. Character device and driver 2h5m

34. What is device driver ? 88

write(fd, 0xab);   // echo 0xAB > /dev/rtc

Udev : populates /dev w/ device files

devices files are under VFS and identified by major/minor numbers

35. A char driver, char device and char device number 15

Let’s create a device number.
Your driver has to ask the kernel to dynamically allocate the device number or numbers.
Basically, what you should be doing here is, you should be using kernel APIs and kernel utilities(shown in Figure 6) in order to a request various services from the kernel.

Now, to create a device number, you just have to use a kernel API alloc_chrdev_region().
So, you have to use alloc_chrdev_region(). This creates a device number. And for the registration, you can use these APIs cdev_init() and cdev_add().

And after that, the driver should create a device files. For that, you can use these kernel APIs class_create() and device_create().
The creation things we are going to do in module initialization function.

Whenever you load a module, so these creation a services must be executed and your driver must be ready to accept a system calls from the user space program.

That’s why it makes sense to do this creation process in the module initialization function.

When you remove the module, it’s better you delete all those a resources what you requested from the kernel.
Otherwise, it will simply consume a resources of the kernel. That’s why, let’s say, if you use unregister_chrdev_region(), it will delete the device number which is allocated for your module, that it can be reused for some other module.

After that, you can use cdev_del() to delete the registration, that will free some memory. class_destroy and device_destroy will delete your device files.
That’s why, all these deletion things we are going to do in module clean-up function. Because these deletion things should be executed whenever you remove the module.

```
scott@host:~/workspace/ldd/custom_drivers/custom_drivers/001hello_world$ nm main.ko
0000000000000000 T cleanup_module
                 U __fentry__
0000000000000000 t helloworld_cleanup
0000000000000000 t helloworld_init
0000000000000000 T init_module
0000000000000000 r _note_6
                 U printk
0000000000000000 D __this_module                        <= !!!
000000000000004d r __UNIQUE_ID_author37
0000000000000000 r __UNIQUE_ID_board39
...
00000000000000ae r __UNIQUE_ID_vermagic36
0000000000000000 r ____versions
```

```
>> main.mod.c

#include <linux/build-salt.h>
#include <linux/module.h>
#include <linux/vermagic.h>
#include <linux/compiler.h>

BUILD_SALT;

MODULE_INFO(vermagic, VERMAGIC_STRING);
MODULE_INFO(name, KBUILD_MODNAME);

__visible struct module __this_module           <= !!!
__section(.gnu.linkonce.this_module) = {
        .name = KBUILD_MODNAME,
```

```
struct file_operations {
    struct module *owner;                       <- __this_module : to prevent a module from being unloaded while the
    structure in use
    loff_t (*llseek) (struct file *, loff_t, int);

}
```

```
cdev_init()         -> just initialization
```

39. Character device registration contd. 121

```
cdev_add()          -> real registration
```

40. Character driver file operation methods 18

Character Driver System Calls

What is a file object?
  Whenever a file is opened, a file object is created in the kernel space.
  The file object stores information about the interaction between an open file and a user process.

def_chr_fops

```
open                                                                                    user
-----------------------------------------------------------------------------------------------
  -> do_sys_open                    --> return 'fd' to user space (-1 means failure)   kernel
       -> do_filp_open              --> 'file' object allocatioin
            -> do_dentry_open       --> using default (dummy) fops, e.g. chrdev_open for open
                 -> chrdev_open     --> replacing default fops
                       -> pcd_open
```

do_filp_open : a file object is created

ref) understanding linux kernel 3ed, p524

When device file gets created
  1) create device file using udev
  2) inode object gets created in memory and inode's i_rdev field is initialized with device number(dev_t)
  3) inode object's i_fop field is set to dummy default file operations (def_chr_fops)

When user process executes open system call
  1) user invokes open system call on the device file
  2) file object gets created   (do_filp_open)
  3) inode's i_fop gets copied to file object's f_op (dummy default file operations of char device file)
  4) open function of dummy default file operations gets called (chrdev_open)
  5) inode object's i_cdev field is initialized with cdev which you added during cdev_add (lookup happens using inode-> i_rdev field)

6) inode->cdev->fops (this is a real file operations of your driver) gets copied to file->f_op
7) file->f_op->open method gets called (read open method of your driver)

```
static int chrdev_open(struct inode *inode, struct file *filp)
{
    const struct file_operations *fops;
    struct cdev *p;
    struct cdev *new = NULL;
    int ret = 0;
    spin_lock(&cdev_lock);
    p = inode->i_cdev;

    if (!p) {
        struct kobject *kobj;
        int idx;
        spin_unlock(&cdev_lock);
        kobj = kobj_lookup(cdev_map, inode->i_rdev, &idx);
        if (!kobj) return -ENXIO;
        new = container_of(kobj, struct cdev, kobj);
        spin_lock(&cdev_lock);
        p = inode->i_cdev;
        if (!p) {
            inode->i_cdev = p = new;
            list_add(&inode->i_devices, &p->list);
            new = NULL;
        } else if (!cdev_get(p))
            ret = -ENXIO;
    } else if (!cdev_get(p))
        ret = -ENXIO;

    spin_unlock(&cdev_lock);
    cdev_put(new);

    if (ret) return ret;
    ret = -ENXIO;
    fops = fops_get(p->ops);

    if (!fops) goto out_cdev_put;
    replace_fops(filp, fops);

    if (filp->f_op->open) {
        ret = filp->f_op->open(inode, filp);
        if (ret) goto out_cdev_put;
    }
    return 0;

out_cdev_put:
    cdev_put(p);
```

```
            return ret;
        }
```

41. Character driver file operation methods contd.

    Open Method   Close system call
    Release Method:
    Write Method:
    Llseek Method:

42. Implementing file operation methods

    Accessing File Operations Structure

     include/linux/fs.h

```
    struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
        ...
        int (*clone_file_range)(struct file *, loff_t, struct file *, loff_t, u64);
        ssize_t (*dedupe_file_range)(struct file *, u64, u64, struct file *, u64);
    } __randomize_layout;
```

    Implementing File Operation Functions
    Method Name Replacement
    Creating Method Variables
    Adding Return Values

43. File operations structure initialization

```
    CarBMW = {a, b, c, d};                      // C89
    CarBMW = {.A = a, .B = b, .C = c, .D = d};  // C99
```

*   44. Creating device files   -> dynamic device file creation   127~133

class_create and
device_create

45. Character driver cleanup function implementation

```
scott@host:~/workspace/.../002pseudo_char_driver$ sudo insmod pcd.ko

  [46707.310503] pcd_driver_init :Device number <major>:<minor> : 240:0
  [46707.310795] pcd_driver_init :Module init was successful

scott@host:~/workspace/.../002pseudo_char_driver$
scott@host:~/workspace/.../002pseudo_char_driver$ cd /sys/class/pc

  pcd_class/ pci_bus/   pci_epc/

scott@host:~/workspace/.../002pseudo_char_driver$ ls /sys/class/pcd_class

   pcd

scott@host:~/workspace/.../002pseudo_char_driver$ ls /sys/class/pcd_class/pcd

   dev  power  subsystem  uevent

scott@host:/sys/class/pcd_class/pcd$ cat dev

  240:0

scott@host:/sys/class/pcd_class/pcd$ cat uevent

  MAJOR=240
  MINOR=0
  DEVNAME=pcd

scott@host:/sys/class/pcd_class/pcd$ ll /dev/pcd

  crw------- 1 root root 240, 0 Apr 24 19:36 /dev/pcd

scott@host:/sys/class$ ls

                                          ^^^^^^^^^
  ata_device  devcoredump     firmware     iommu     pcd_class     pps           rtc           thermal       wakeup
  ata_link    devfreq         gpio         leds      pci_bus       printer       scsi_device   tpm           watchdog
  ata_port    devfreq-event   graphics     mdio_bus  pci_epc       ptp           scsi_disk     tpmrm
  backlight   dma             hidraw       mem       phy           pwm           scsi_generic  tty
  bdi         dmi             hwmon        misc      powercap      rapidio_port  scsi_host     vc
```

```
block       drm             i2c-adapter  mmc_host  power_supply  regulator     sound         vfio
bsg         drm_dp_aux_dev  i2c-dev      nd        ppdev         remoteproc    spi_master    virtio-ports
dax         extcon          input        net       ppp           rfkill        spi_slave     vtconsole
```

```
scott@host:/sys/class$ sudo rmmod pcd.ko
scott@host:/sys/class$ dmesg

  [46707.310503] pcd_driver_init :Device number <major>:<minor> : 240:0
  [46707.310795] pcd_driver_init :Module init was successful
  [47219.531427] pcd_driver_cleanup :module unloaded
```

4. Character driver file operation implementation 1h1m 134~

46. Understanding read method
47. Understanding error codes
48. Read method implementation

49. Understanding write method
50. Write method implementation

51. lseek method
52. lseek method implementation

53. Testing pseudo char driver

```
$ sudo -s
# insmod pcd.ko
# dmesg

  [54292.379629] pcd_driver_init :Device number <major>:<minor> : 240:0
  [54292.379692] pcd_driver_init :Module init was successful

# echo "Hello, welcome to the course" > /dev/pcd

  [54292.379629] pcd_driver_init :Device number <major>:<minor> : 240:0
  [54292.379692] pcd_driver_init :Module init was successful

  [54432.442779] pcd_open :open was successful                          -> echo issued an OPEN system cal.

  [54432.442787] pcd_write :write requested for 29 bytes                -> echo issued a WRITE system cal.
  [54432.442788] pcd_write :current file position = 0
  [54432.442788] pcd_write :Number of bytes successfully write = 29
  [54432.442789] pcd_write :Updated file position = 29
```

```
  [54432.442791] pcd_release :release was successful                        -> echo issued an RELEASE system cal.

# cat /dev/pcd
  Hello, welcome to the course

  [54679.429313] pcd_open :open was successful

  [54679.429319] pcd_read :read requested for 131072 bytes
  [54679.429319] pcd_read :current file position = 0
  [54679.429321] pcd_read :Number of bytes successfully read = 512
  [54679.429321] pcd_read :Updated file position = 512

  [54679.429324] pcd_read :read requested for 131072 bytes
  [54679.429325] pcd_read :current file position = 512
  [54679.429325] pcd_read :Number of bytes successfully read = 0            -> end of file
  [54679.429326] pcd_read :Updated file position = 512

  [54679.429331] pcd_release :release was successful


# cp test.txt /dev/pcd


    root@host:~/workspace/ldd/custom_drivers/custom_drivers/002pseudo_char_driver# cp pcd.c /dev/pcd
      cp: error writing '/dev/pcd': No space left on device

    root@host:~/workspace/ldd/custom_drivers/custom_drivers/002pseudo_char_driver# dmesg | tail - 20

      [55108.260918] pcd_open :open was successful

      [55108.260928] pcd_write :write requested for 4289 bytes
      [55108.260929] pcd_write :current file position = 0
      [55108.260929] pcd_write :Number of bytes successfully write = 512
      [55108.260930] pcd_write :Updated file position = 512

      [55108.260931] pcd_write :write requested for 3777 bytes              4289 - 512 = 3777
      [55108.260931] pcd_write :current file position = 512
      [55108.260932] pcd_write :Number of bytes successfully write = 0      -> end of memory
      [55108.260932] pcd_write :Updated file position = 512

      [55108.261188] pcd_release :release was successful


      root@host:~/workspace/ldd/custom_drivers/custom_drivers/002pseudo_char_driver# cp pcd.c /dev/pcd
      cp: error writing '/dev/pcd': Cannot allocate memory

      root@host:~/workspace/ldd/custom_drivers/custom_drivers/002pseudo_char_driver# dmesg | tail -10

      [55328.641690] pcd_driver_init :Module init was successful
```

```
[55338.920580] pcd_open :open was successful

[55338.920590] pcd_write :write requested for 4372 bytes
[55338.920590] pcd_write :current file position = 0
[55338.920591] pcd_write :Number of bytes successfully write = 512
[55338.920592] pcd_write :Updated file position = 512

[55338.920593] pcd_write :write requested for 3860 bytes
[55338.920593] pcd_write :current file position = 512
[55338.920593] pcd_write :--No space left on the device              <-   count == 0

[55338.920812] pcd_release :release was successful
```

54. Error handling

5. Char driver with multiple device nodes 1h49m 187~

55. pcd driver with multiple devices

56. Pcd driver with multiple devices code implementation part-1
57. Pcd driver with multiple devices code implementation part-2
58. Pcd driver with multiple devices code implementation part-3
59. Pcd driver with multiple devices code implementation part-4
60. Pcd driver with multiple devices code implementation part-5
61. Pcd driver with multiple devices code implementation part-6
62. Pcd driver with multiple devices code implementation part-7

63. Pcd driver with multiple devices testing

```
$ sudo insmod pcd_n.ko

  [ 3708.848439] pcd_driver_init : Device number <major>:<minor> = 240:0
  [ 3708.848484] pcd_driver_init : Device number <major>:<minor> = 240:1
  [ 3708.848499] pcd_driver_init : Device number <major>:<minor> = 240:2
  [ 3708.848514] pcd_driver_init : Device number <major>:<minor> = 240:3
  [ 3708.848535] pcd_driver_init : Module init was successful


$ echo hello > /dev/pcdev-1
```

```
bash: /dev/pcdev-1: Permission denied

[ 3708.848535] pcd_driver_init : Module init was successful
[ 3708.8485XX] pcd_open : minor access = 0
[ 3708.8485XX] pcd_open : open was unsuccessful


$ strace dd if=pcd_n.c of=/dev/pcdev-1

execve("/bin/dd", ["dd", "if=pcd_n.c", "of=/dev/pcdev-1"], 0x7ffd37eb4840 /* 53 vars */) = 0
brk(NULL)                               = 0x55e06a28a000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=79287, ...}) = 0
mmap(NULL, 79287, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f18eb28a000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f18eb288000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f18eac84000
mprotect(0x7f18eae6b000, 2097152, PROT_NONE) = 0
mmap(0x7f18eb06b000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f18eb06b000
mmap(0x7f18eb071000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f18eb071000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f18eb289540) = 0
mprotect(0x7f18eb06b000, 16384, PROT_READ) = 0
mprotect(0x55e0692dd000, 4096, PROT_READ) = 0
mprotect(0x7f18eb29e000, 4096, PROT_READ) = 0
munmap(0x7f18eb28a000, 79287)           = 0
rt_sigaction(SIGINT, NULL, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigaction(SIGUSR1, {sa_handler=0x55e0690d00e0, sa_mask=[INT USR1], sa_flags=SA_RESTORER, sa_restorer=0x7f18eacc2f10}, NULL,
8) = 0
rt_sigaction(SIGINT, {sa_handler=0x55e0690d00d0, sa_mask=[INT USR1], sa_flags=SA_RESTORER|SA_NODEFER|SA_RESETHAND,
sa_restorer=0x7f18eacc2f10}, NULL, 8) = 0
brk(NULL)                               = 0x55e06a28a000
brk(0x55e06a2ab000)                     = 0x55e06a2ab000
openat(AT_FDCWD, "/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=3004224, ...}) = 0
mmap(NULL, 3004224, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f18ea9a6000
close(3)                                = 0
openat(AT_FDCWD, "pcd_n.c", O_RDONLY)   = 3
dup2(3, 0)                              = 0
close(3)                                = 0
lseek(0, 0, SEEK_CUR)                   = 0
openat(AT_FDCWD, "/dev/pcdev-1", O_WRONLY|O_CREAT|O_TRUNC, 0666) = -1 EACCES (Permission denied)
openat(AT_FDCWD, "/usr/share/locale/locale.alias", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2995, ...}) = 0
```

```
  read(3, "# Locale name alias data base.\n#"..., 4096) = 2995
  read(3, "", 4096)                       = 0
  close(3)                                = 0
  openat(AT_FDCWD, "/usr/share/locale/en_US.UTF-8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en_US.utf8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en_US/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en.UTF-8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en.utf8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en_US.UTF-8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or
  directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en_US.utf8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or
  directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en_US/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en.UTF-8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or
  directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en.utf8/LC_MESSAGES/coreutils.mo", O_RDONLY) = -1 ENOENT (No such file or
  directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en/LC_MESSAGES/coreutils.mo", O_RDONLY) = 3
  fstat(3, {st_mode=S_IFREG|0644, st_size=578, ...}) = 0
  mmap(NULL, 578, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f18eb29d000
  close(3)                                = 0
  write(2, "dd: ", 4dd: )                     = 4
  write(2, "failed to open '/dev/pcdev-1'", 29failed to open '/dev/pcdev-1') = 29
  openat(AT_FDCWD, "/usr/share/locale/en_US.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en_US.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en_US/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale/en/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en_US.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en_US.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en_US/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en.UTF-8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en.utf8/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  openat(AT_FDCWD, "/usr/share/locale-langpack/en/LC_MESSAGES/libc.mo", O_RDONLY) = -1 ENOENT (No such file or directory)
  write(2, ": Permission denied", 19: Permission denied)     = 19
  write(2, "\n", 1
  )                       = 1
  close(2)                                = 0
  exit_group(1)                           = ?
  +++ exited with 1 +++

scott@host:~/workspace/ldd/ex/003_pseudo_char_driver_multiple$ sudo dd if=pcd_n.c of=/dev/pcdev-2

  dd: writing to '/dev/pcdev-2': Cannot allocate memory
  2+0 records in
  1+0 records out
  512 bytes copied, 0.000306726 s, 1.7 MB/s
```

```
scott@host:~/workspace/ldd/ex/003_pseudo_char_driver_multiple$ dmesg | tail
  [ 4143.773984] pcd_open : minor access = 1
  [ 4143.773985] pcd_open : open was successful

  [ 4143.773999] pcd_write : Write requested for 512 bytes
  [ 4143.774000] pcd_write : Current file position = 0
  [ 4143.774001] pcd_write : Number of bytes successfully written = 512
  [ 4143.774002] pcd_write : Updated file position = 512

  [ 4143.774004] pcd_write : Write requested for 512 bytes
  [ 4143.774005] pcd_write : Current file position = 512
  [ 4143.774006] pcd_write : No space left on the device

  [ 4143.774289] pcd_release : release was successful



$ sudo dd if=pcd_n.c of=/dev/pcdev-2 count=1

  1+0 records in
  1+0 records out
  512 bytes copied, 0.00010661 s, 4.8 MB/s

$ dmesg | tail -15

  [ 4336.492774] pcd_open : minor access = 1
  [ 4336.492775] pcd_open : open was successful

  [ 4336.492784] pcd_write : Write requested for 512 bytes
  [ 4336.492785] pcd_write : Current file position = 0
  [ 4336.492785] pcd_write : Number of bytes successfully written = 512
  [ 4336.492786] pcd_write : Updated file position = 512

  [ 4336.492788] pcd_release : release was successful

scott@host:~/workspace/ldd/ex/003_pseudo_char_driver_multiple$ sudo dd if=pcd_n.c of=/dev/pcdev-3 count=1 bs=100

  1+0 records in
  1+0 records out
  100 bytes copied, 0.000108374 s, 923 kB/s


scott@host:~/workspace/ldd/ex/003_pseudo_char_driver_multiple$ sudo cat /dev/pcdev-3

  #include<linux/module.h>
  #include<linux/fs.h>
  #include<linux/cdev.h>
  #include<linux/device.h>
  #incluscott@host:~/workspace/ldd/ex/003_pseudo_char_driver_multiple$
```

```
scott@host:~/workspace/ldd/ex/003_pseudo_char_driver_multiple$ dmesg | tail -15

[ 4628.650774] pcd_open : minor access = 2
[ 4628.650775] pcd_open : open was successful

[ 4628.650781] pcd_read : Read requested for 131072 bytes
[ 4628.650782] pcd_read : Current file position = 0
[ 4628.650783] pcd_read : Number of bytes successfully read = 1024
[ 4628.650784] pcd_read : Updated file position = 1024

[ 4628.650794] pcd_read : Read requested for 131072 bytes
[ 4628.650794] pcd_read : Current file position = 1024
[ 4628.650795] pcd_read : Number of bytes successfully read = 0
[ 4628.650795] pcd_read : Updated file position = 1024

[ 4628.650801] pcd_release : release was successful
```

64. Pcd driver with multiple devices testing contd

65. Pcd driver with multiple devices lseek implementation

66. Container of discussion

```
#define platform_driver_register(drv)   __platform_driver_register(drv, THIS_MODULE)
extern int __platform_driver_register(struct platform_driver *, struct module *);
```

Platform driver structure : struct platform_driver

Registering a platform device : int platform_device_register(struct platform_device *pdev);

Platform device structure : struct platform_device

probe(),	Called when matched platform device is found
remove(),	
shutdown(),	Called at shut-down time to quiesce the device
suspend(),	Called to put the device to sleep mode. Usually to a low power state
resume(), ...	Called to bring a device from sleep mode

Platform device – driver matching 234

- "matching" mechanism of the bus core

The Linux platform core implementation maintains platform device and driver lists.
Whenever you add a new platform device or driver, this list gets updated and matching mechanism triggers.

Every bus type has its match function, where the device and driver list will be scanned.

Points to remember 239

- Whenever a new device or a new driver is added, the matching function of the platform bus runs, and if it finds a matching platform device for a platform driver, the probe function of the matched driver will get called. Inside the probe function, the driver configures the detected device.
- Details of the matched platform device will be passed to the probe function of the matched driver so that driver can extract the platform data and configure it.

Probe function of the platform driver 241

- Probe function must be implemented by the platform driver and should be registered during platform_driver_register().
- When the bus matching function detects the matching device and driver, probe function of the driver gets called with detected platform device as an input argument
- Note that probe() should in general, verify that the specified device hardware actually exists.
  Sometimes platform setup code can't be sure.
  The probing can use device resources, including clocks, and device platform_data.
- The the probe function is responsible for
- Device detection and initialization
- Allocation of memories for various data structures,
- Mapping i/o memory
- Registering interrupt handlers

- Registering device to kernel framework, user level access point creations, etc
- The probe may return 0(Success) or error code. If probe function returns a non-zero value, meaning probing of a device has failed.

Remove function of the platform driver 242

- Remove function gets called when a platform device is removed from the kernel to unbind a device from the driver or when the kernel no longer uses the platform device
- Remove function is responsible for
- Unregistering the device from the kernel framework
- Free memory if allocated on behalf of a device
- Shutdown/De-initialize the devic

https://www.kernel.org/doc/Documentation/driver-model/platform.txt

70. Platform driver code exercise 245

essential functions

```
int pcd_platform_driver_remove(struct platform_device *pdev)
int pcd_platform_driver_probe(struct platform_device *pdev)

static int __init pcd_platform_driver_init(void)
{
        platform_driver_register();
        return 0;
}

static void __exit pcd_platform_driver_exit(void)
{
    platform_driver_unregister();
}
```

Code exercise: 248

Implementation of pseudo character driver as platform driver

- Repeat the exercise pseudo character driver with multiple devices as a platform driver.
- The driver should support multiple pseudo character devices(pcdevs) as platform devices
- Create device files to represent platform devices
- The driver must give open, release, read, write, lseek methods to deal with the devices

kernel module 1 : platform driver 250
kernel module 2 : platform device setup

Platform device setup 252

1. Create 2 platform devices and initialize them with required information
   - Name of a platform device
   - Platform data
   - Id of the device
   - Release function for the device
2. Register platform devices with the Linux kernel

71. Platform device setup code implementation

```
struct pcdev_private_data       /*Device private data structure */
{
    struct pcdev_platform_data pdata;
    char *buffer;
    dev_t dev_num;
    struct cdev cdev;
};

struct pcdrv_private_data       /*Driver private data structure */
{
    int total_devices;
    dev_t device_num_base;
    struct class *class_pcd;
    struct device *device_pcd;
};
```

struct platform_device 347

```
{
    const char *name;           <= used to match
    int id;                     <= for multiple instances
    struct device dev;                                      311
}
```

72. Platform device setup code implementation contd. : setup device(module 2)

Platform device release function 251
- Callback to free the device after all references have gone away.
- This should be set by the allocator of the device

```
platform_add_devices(platform_pcdevs,ARRAY_SIZE(platform_pcdevs) );
platform_device_unregister(&platform_pcdev_1);
```

```
$ sudo insmod pcd_device_setup.ko
```

```
$ ls /sys/devices/platform/
  eisa.0                  pcdev-A1x.0       platform-framebuffer.0      uevent
  'Fixed MDIO bus.0'      pcdev-B1x.1       power
  i8042                   pcdev-C1x.2       reg-dummy
  intel_rapl_msr.0        pcdev-D1x.3       rtc_cmos
  kgdboc                  pcspkr            serial8250

$ dmesg | tail -30
  [18617.022770] pcdev_platform_init : Device setup module loaded

$ sudo rmmod pcd_device_setup.ko
$ dmesg | tail -30
  [18861.674563] pcdev_release : Device released
  [18861.674825] pcdev_release : Device released
  [18861.674869] pcdev_release : Device released
  [18861.674880] pcdev_release : Device released
  [18861.674881] pcdev_platform_exit : Device setup module unloaded

$ ls /sys/devices/platform/
  eisa.0                pcspkr                      serial8250
  'Fixed MDIO bus.0'    platform-framebuffer.0      ueven
  i8042                 power
  intel_rapl_msr.0      reg-dummy
  kgdboc                rtc_cmos
```

73. Platform driver code implementation part-1 : driver(module 1)

```
struct device_driver        in linux/device.h
platform_driver_register    in platform_device.h
```

74. Platform driver code implementation part-2

```
# insmod pcd_device_setup.ko
  [18861.674881] pcdev_platform_init : Device setup module loaded

# insmod pcd_platform_driver.ko
  [18861.674881] pcdev_platform_init : Device setup module loaded
  [18861.674881] pcd_platform_driver_probe : A device is detected
  [18861.674881] pcd_platform_driver_init : pcd platform driver loaded

# rmmod pcd_platform_driver.ko
  [18861.674881] pcdev_platform_init : Device setup module loaded
  [18861.674881] pcd_platform_driver_probe : A device is detected
  [18861.674881] pcd_platform_driver_init : pcd platform driver loaded
  [18861.674881] pcd_platform_driver_remove : A device is removed
  [18861.674881] pcd_platform_driver_cleanup : pcd platform driver unloaded
```

```
# rmmod pcd_device_setup
  [18861.674881] pcdev_platform_init : Device setup module loaded
  [18861.674881] pcd_platform_driver_probe : A device is detected
  [18861.674881] pcd_platform_driver_init : pcd platform driver loaded
  [18861.674881] pcd_platform_driver_remove : A device is removed
  [18861.674881] pcd_platform_driver_cleanup : pcd platform driver unloaded

  [18861.674881] pcdev_release : Device released
  [18861.674881] pcdev_release : Device released
  [18861.674881] pcdev_platform_exit : Device setup module unloaded
```

or

```
# insmod pcd_platform_driver.ko
  [18861.674881] pcd_platform_driver_init : pcd platform driver loaded

# insmod pcd_device_setup.ko
  [18861.674881] pcd_platform_driver_init : pcd platform driver loaded
  [18861.674881] pcd_platform_driver_probe : A device is detected
  [18861.674881] pcdev_platform_init : Device setup module loaded

# rmmod pcd_device_setup
  [18861.674881] pcd_platform_driver_init : pcd platform driver loaded
  [18861.674881] pcd_platform_driver_probe : A device is detected
  [18861.674881] pcdev_platform_init : Device setup module loaded
  [18861.674881] pcd_platform_driver_remove : A device is removed
  [18861.674881] pcdev_release : Device released
  [18861.674881] pcdev_release : Device released
  [18861.674881] pcdev_platform_exit : Device setup module unloaded
```

75. Platform driver code implementation part-3

76. Platform driver code implementation part-4

```
    pdata = pdev->dev.platform_data;
or  pdata = (struct pcdev_platform_data*)dev_get_platdata(&pdev->dev);
```

Kernel memory allocation APIs 325

- kmalloc ()
- kfree ()

```
void* kmalloc( size_t size, gfp_t flags);//include/linux/slab.h 326
```

Used to allocate memory in kernel space by drivers and kernel functions
Memory obtained are physically(RAM) contiguous

gfp_t : get free pages

kzalloc - allocate memory. The memory is set to zero.

77. Platform driver code implementation part-5
78. Platform driver code implementation part-6

to free allocated memory by kalloc()

```
struct platform_device {
    ...
    struct device   dev;
    ...
}

struct device {
    ...
    void *platform_data;        <=
    void *driver_data;          <=
    ...
}

// save the device private data pointer in platform device structure
pdev->dev.driver_data = dev_data;               or
dev_set_drvdata(*pdev->dev, dev_data);

==>

struct pcdev_private_data *dev_data = dev_get_drvdata(&pdev->dev);
kfree(dev_data->buffer);
kfree(dev_data);
```

79. Testing platform driver

```
# insmod pcd_device_setup.ko
# insmod pcd_platform_driver.ko
# lsmod                                 -> list all the modules
Module                  Size    Used by
pcd_platform_driver     16384   0       <= !!
pcd_device_setup        16384   0
...

# rmmod pcd_device_setup
#
# rmmod pcd_platform_driver
```

```
    # insmod pcd_platform_driver.ko
    # insmod pcd_device_setup.ko
    #
    # rmmod pcd_platform_driver


80. Linux device resource managed functions

                                                                    336~340

    devm_kzalloc()      <=>     kalloc & kfree      <= based on the existence of struct device
    devm_gpio_get()     <=>     gpio_get()/put()
    devm_request_irq()  <=>     request_irq()/free_irq()


    www.kernel.org/doc/Documentation/driver-model/devres.txt

        MFD, MUX, PCI, PHY, MEM, PWM, MDIO, IOMAP, INPUT, IRQ, IO region, IIC, CLOCK, DRM, GPIO, ...

*   81. Using device resource managed kernel functions

    kxxxx() -> devm_kxxxx()


*   82. Testing with more platform devices

    int platform_add_devices(struct platform_device **devs, int num) {
        for (int i = 0; i < num; i++) {
            platform_device_register(devs[i]);
        }
    }


    kernel crash w/ same ids

        struct platform_device platform_pcdev_1 = {
            .name = "pcdev-A1x",
            .id = 0,
        };


        struct platform_device platform_pcdev_2 = {
            .name = "pcdev-A1x",
            .id = 0,                                    =>  1 !!!
        };


        rmmod -> "Killed"   => need to reboot !!!
```

83. Fixing kernel crash

```
# ls -l /dev/pcdev- [tab][tab]
```

* 84. Platform device driver matching using platform device ids 254 ~ 256

device_id

```
static int platform_match(struct device *dev, struct device_driver *drv)    // where matching happens
{
    struct platform_device *pdev = to_platform_device(dev);
    struct platform_driver *pdrv = to_platform_driver(drv);

    of_driver_match_device(dev, drv)                // open firmware driver, ie device tree matching

    acpi_driver_match_device(dev, drv)              //

    return platform_match_id(pdrv->id_table, pdev); // id_table

    return (strcmp(pdev->name,drv->name) == 0);
}

static const struct platform_device_id *platform_match_id(const struct platform_device_id *id, struct platform_device *pdev)
{
    while (id->name[0]) {
        if (strcmp(pdev->name, id->name) == 0) {
            pdev->id_entry = id;                        <= updated to platform_device->id_entry
            return id;
        }
        id++;
    }
    return NULL;
}
```

if init and exit function consists of platform_driver_register and platform_driver_unregister only,

```
  =>  #define module_platform_driver(__platform_driver) \
          module_driver(__platform_driver, platform_driver_register, \
          platform_driver_unregister)
```

* 85. Fixing error handling in probe function

really_probe()

-> drv->probe()

7. Device tree 1h26m 258

86. Introduction to device tree

What is device tree? 260 ~ 262

- The "Open Firmware Device Tree", or simply Device Tree (DT),
  is a data exchange format used for exchanging hardware description data with the software or OS.

- More specifically, it is a description of hardware that is readable by an operating system
  so that the operating system doesn't need to hard code details of the machine.

- In short, it is a new and recommended way to describe non-discoverable devices(platform devices)
  to the Linux kernel, which was previously hardcoded into kernel source files.

Source : Documentation/devicetree/usage-model.txt

Device tree

- An operating system uses the Device Tree to discover the topology of the hardware at runtime,
  and thereby support a majority of available hardware without hardcoded information
  (assuming drivers were available for all devices)

- The most important thing to understand is that the DT is simply a data structure
  that describes the hardware. There is nothing magical about it,
  and it does not magically make all hardware configuration problems go away

- DT provides a language for decoupling the hardware configuration from the device driver
  and board support files of the Linux kernel (or any other operating system for that matter).

- Using it allows device drivers to become data-driven. To make setup decisions
  based on data passed into the kernel instead of on permachine hardcoded selections.

- Ideally, a data-driven platform setup should result in less code duplication
  and make it easier to support a wide range of hardware with a single kernel image.

Why DT is used ?

Linux uses DT for,
- Platform identification : identify the board or machine on which the kernel runs
- Device population:
- The kernel parses the device tree data and generates the required software data structure, which will be used by the kernel code.

Ideally, the device tree is independent of any os; when you change the OS, you can still use the same device tree file to describe the hardware to the new OS. That is, the device tree makes “adding of device information “ independent of OS

More reading
- https://elinux.org/Device_Tree_What_It_Is
- https://www.kernel.org/doc/Documentation/devicetree/usagemodel.txt

87. Writing device tree 265

Device tree specification
- You can get the full specification here
- https://www.devicetree.org/

Writing device tree

- The device tree supports a hierarchical way of writing hardware description at the soc level, common board level, and board-specific level. Most of the time, writing a new device tree is not difficult, and you can reuse most of the common hardware information from the device tree file of the reference board.

- For example, when you design a new board, which is slightly different from another reference board, then you can reuse the device tree file of the reference board and only add that information which is new in your custom board

Describing hardware hierarchy
- It comes at various level because the board has many device blocks
- SOC
- SOC has an on-chip processor and on-chip peripherals
- The board also has various peripherals onboard, like sensors, LEDs, buttons, joysticks, external memories, touchscreen, etc

Modular appoach

mod1 : SOC specific device tree file Board specific device tree file
(This DT file is used as an include file and can be used with another board which is based on same SOC)

mod2 : Board specific device tree file

linux/arch/arm/boot/dts/am335x-evm.dts => #include "am33xx.dtsi" (soc level device file)

am335x-boneblack.dts 272, 278

```
#include "am33xx.dtsi"
#include "am335x-bone-common.dtsi"
#include "am335x-boneblack-common.dtsi"
```

88. Device tree structure 19

Overview of device tree structure

- ✓ Device tree is a collection of device nodes
- ✓ A ‘device node’ or simply called ‘a node’ represents a device.
  Nodes are organized in some systematic way inside the device tree file.
- ✓ They also have parent and child relationship, and every device tree must have one root node
- ✓ A node explains itself, that is, reveals its data and resources using its “properties.”

Root node

- The device tree has a single root node of which all other device nodes are descendants.
  The full path to the root node is /.
- All device trees shall have a root node, and the following nodes shall be present at the root of all device trees:
  - One /CPUs node
  - At least one /memory node

Chapter 3 :DEVICE NODE REQUIREMENTS Devicetree Specification Release v0.3a

How to write a device tree ?

- Remember that you most probably be writing device tree addons or overlays
  for your board-related changes but not for entire soc.
- The soc specific device tree will be given by the vendor in the form of device tree
  inclusion file (.dtsi) and you just need to include that in your board-level device tree
- Follow modulatory approach while writing device tree

am335x-boneblack.dts 272, 278

```
/ {
     model = "TI AM335x BeagleBone Black";          <= added
     compatible = "ti, am335x-bone-black", ... ;    <= override that of included files
```

```
  };                    <= root level node (children of root)
};                      <= root


 &sgx {                 <= reference
    status = "okay";
 };


am33xx.dtsi

    / {
         compatible = "ti, am33xx";                 <= overridden

         sgx: sgx@56000000 {
            compatible = "    ";
            ti, hwmods = "gfx";

         };
      };
    };


#include "am335x-bone-common.dtsi"
#include "am335x-boneblack-common.dtsi"


Never edit top level files => it will be overridden later.
```


Device tree writing syntax 279

- Node name
- Node Label
- Standard and non-standard property names
- Different data type representation (u32, byte, byte stream, string, stream of strings, Boolean, etc)
- SoC node and children

Node name

- Refer device tree specification Release v0.3 from devicetree.org

https://github.com/devicetree-org/devicetree-specification/releases/tag/v0.4

The unit-address component of the name is specific to the bus type on which the node sits.
It consists of one or more ASCII characters from the set of characters in Table 2.1.
The unit-address must match the first address specified in the reg property of the node.

If the node has no reg property, the @unit-address must be omitted and the node-name alone
differentiates the node from other nodes at the same level in the tree.
The binding for a particular bus may specify additional, more specific requirements
for the format of reg and the unit-address.

ex. 281

```
    i2c0: i2c@44e0b000 {
        compatible = "ti, omap4-i2c";
        #address-cells = <1>;
        #size-cells = <0>;
        ti, hwmods = "i2c1";
        reg = <0x44e0b000 0x1000>;      <= AM335x TRM ch2 memory map
        interrupts = <70>;
        status = "disabled";
    };

    i2c1: i2c@44e2a000 {
        compatible = "ti, omap4-i2c";
        #address-cells = <1>;
        #size-cells = <0>;
        ti, hwmods = "i2c2";
        reg = <0x4802a000 0x1000>;
        interrupts = <71>;
        status = "disabled";
    };

    i2c2: i2c@48060000 {

        ...
```

90. Device tree parent and child node 282 ~ 283

```
    &i2c0 {
        status = "okay";

        tps: tps@24 {
            reg = <0x24>;   // i2c address
        };

        baseboard_eeprom: baseboard_eeprom@50 {
            reg = <0x50>

            baseboard_data: baseboard_data@0 {
                reg = <0 0x100>;
            };
        };
    };
```

```
label,
gpios,                      : standard property
default-state

linux,default-trigger       : custom property
```

Different types of properties

- Standard property
- Custom property (non-standard)
- Standard properties are those which is explained by the specification and the device-driver binding documentation
- Custom properties are specific to a particular vendor or organization which is not documented by the specification.
- That is why, when you use custom property, always begin with your organization name.

Root compatible property of BBB

```
/ {
    model = "TI AM335x BeagleBone Black";
    compatible = "ti,am335x-bone-black", "ti,am335x-bone", "ti,am33xx";
};
                Sorted string list from most compatible to least.

; Root compatible property is used for machine identification
```

Uses of compatible property

1. Machine identification and initialization
2. Match and load the appropriate driver for the device

```
    &i2c0 {
        status = "okay";                            !!!!

        tps: tps@24 {
            reg = <0x24>;   // i2c address
        };

        baseboard_eeprom: baseboard_eeprom@50 {
            compatible = "atmel,24c256";    <= match found in linux/drivers/misc/eeprom/at24.c
            reg = <0x50>

            #address-cells = <1>;
            #size-cells = <1>;
```

```
            baseboard_data: baseboard_data@0 {
                reg = <0 0x100>;
            };
        };
    };

    i2c0: i2c@44e0b000 {
        compatible = "ti, omap4-i2c";               <= matched !
        #address-cells = <1>;
        #size-cells = <0>;
        ti, hwmods = "i2c1";
        reg = <0x44e0b000 0x1000>;
        interrupts = <70>;
        status = "disabled";
    };


    in linux/drivers/i2c/buses/i2c-omap.c                          -> drivers

    static struct platform_driver omap_i2c_driver = {
        .proble = omap_i2c_probe,
        .remove = omap_i2c_remove,
        .driver = {
            .name = "omap_i2c",
            .pm = &omap_i2c_pm_ops,
            .of_match_table = of_match_ptr(omap_i2c_of_match),
        },
    }

    static const struct of_device_id omap_i2c_of_match[] = {
        {
            .compatible = "ti, omap4-i2c",          <= matched !
            .data = &omap4_pdata,
        },
        {
            .compatible = "ti, omap3-i2c",
            .data = &omap3_pdata,
        },
        ...
    }
```


Device tree bindings

- How do you know which property name and value pair should be used to describe a node in the device tree?
- Device tree binding document. The driver writer must document these details
- The properties that are necessary to describe a device in the device tree depends on the requirements of the Linux driver

for that device
• When all the required properties are provided, the driver of charge can detect the device from the device tree and configure it.

linux/Documentation/devicetree/bindings/i2c/i2c-omap.txt

```
Required properties:
- compatible : Must be
    "ti,omap2420-i2c" for OMAP2420 SoCs
    "ti,omap2430-i2c" for OMAP2430 SoCs
    "ti,omap3-i2c" for OMAP3 SoCs
    "ti,omap4-i2c" for OMAP4+ SoCs
    "ti,om654-i2c", "ti,omap4-i2c" for AM654 SoCs
    "ti,j721e-i2c", "ti,omap4-i2c" for J721E SoCs

Recommended properties:

Optional properties:

ex...
```

- Compatible strings and properties are first defined by the client program (OS , drivers) then shared with DT writer

Device tree bindings- points to remember

- Case 1 : When the driver is already available in the Linux kernel for the device ‘x,’
  but you just need to add device ‘x’ entry in the device tree
  then you must consult ‘x’ drivers binding document
  which guides you through creating device tree node for device ‘x.’

- Case 2 : When the driver is not available for the device ‘x,’
  then you should write you own driver, you should decide what properties to use
  (could be a combination of standard and non-standard property),
  you should then provide the device tree binding document
  describing what are all the properties and compatible strings a device tree write must include.

ex. lm75

linux/Documentation/devicetree/bindings/hwmon/lm75.txt

```
sensor@48 {
   compatible = "st,stlm75";
   reg = <0x48>;
}
```

linux/drivers/hwmon/lm75.c

```
static const struct of_device_id lm75_of_match[] = {
   .compatible = "ti, tmpXXX";
   .data = (void *)tmpXXX
};
```

ex. mpu 6050

linux/Documentation/devicetree/bindings/iio/imu/inv_mpu6050.txt

```
mpu6050@68 {
   compatible = "invensense,mpu6050";
   reg = <0x68>;
   ...
}
```

linux/drivers/hwmon/lm75.c

```
static const struct of_device_id lm75_of_match[] = {
   .compatible = "ti, tmpXXX";
   .data = (void *)tmpXXX
};
```

https://kernel.org/doc/Documentation/devicetree/bindings/i2c/i2c-omap.txt

Linux conventions to write device tree

hex constants are lower case
- use "0x" instead of "0X"
- use a..f instead of A..F, eg 0xf instead of 0xF

node names
- should begin with a character in the range 'a' to 'z', 'A' to 'Z'
- unit-address does not have a leading "0x" (the number is assumed to be hexadecimal)
- unit-address does not have leading zeros
- use dash "-" instead of underscore "_"

label names
- should begin with a character in the range 'a' to 'z', 'A' to 'Z'
- should be lowercase
- use underscore "_" instead of dash "-"

property names
- should be lower case
- should begin with a character in the range 'a' to 'z'
- use dash "-" instead of underscore "_"

https://elinux.org/Device_Tree_Linux#Linux_vs_ePAPR_Version_1.1

94. pcd device tree version

```
# git branch -a
 *4.14
  remote/origin/5.4
  remote/origin/HEAD -> origin/4.14

# git stash && git checkout 5.4

# git branch
  4.14
  4.9
 *5.4

# git stash apply

# git clone https://github.com/beagleboard/linux.git linux_bbb_5.4
```

95. Switching to Linux kernel version 5.4

```
# git checkout 5.4
```

96. Updating Linux kernel image of 5.4

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bb.org_defconfig
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- uImage dtbs LOADADDR=0x80008000 -j4
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 modules
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install


$ cd /home/kiran/workspace/ldd/src/linux_bbb_5.4
$ cd arch/arm/boot
$ cp uImage /media/kiran/BOOT/
$ cd dts
$ cp am335x-boneblack.dtb /media/kiran/BOOT/
$ cd /ib/modules/
$ sudo cp -a 5.4.47/ /media/kiran/ROOTFS/lib/modules/
$ sync

$ uname -r
5.4.47

$ ifconfig  -> reboot if usb0 & usb1 is not seen.
```

8. device tree nodes and platform driver 1h16m

97. Device tree nodes for pcd driver

```
workspace/ldd/src/linux_bbb_5.4/arch/arm/boot/dts# vi am335x-boneblack.dts
#include "am335x-boneblack-lddcrs.dtsi"

workspace/ldd/src/linux_bbb_5.4/arch/arm/boot/dts# vi am335x-boneblack-lddcourse.dtsi
/ {
    pcdev-1 {
        org,size = <512>
        org,device-serial-num = “PCDEV1ABC123”
        org,perm = <0x11>
    };
    pcdev-2 {
        org,size = <1024>
        org,device-serial-num = “PCDEV2ABC123”
        org,perm = <0x11>
    };
    pcdev-3 {
        org,size = <256>
        org,device-serial-num = “PCDEV3ABC123”
        org,perm = <0x11>
    };
    pcdev-1 {
        org,size = <1024>
        org,device-serial-num = “PCDEV4ABC123”
        org,perm = <0x11>
    };
};

-> device tree compiler to get dtb
```

98. Pcd platform driver DT coding part-1

```
Platform_device_id can be used only when device regiter function is called manually.
=> another list of device is needed inside device tree.


    static int platform_match(struct device *dev, struct device_driver *drv)
    {

        of_driver_match_device(dev, drv)                // open firmware driver, ie device tree matching

        // Refer to 84 for details.

    }
```

```
static inline int of of_driver_match_device(struct device *dev, const struct device_driver *drv)
{
    return of_match_device(drv->of_match_table, dev) != NULL;
}                                ^^^^^^^^^^^^^^^


struct of_device_id {
    char name[32];
    char type[32];
    char compatible[128];
    const void *data;
};

const struct of_device_id *of_match_device(const struct of_device_id *matches, const struct device *dev)
{
    if ((!matches) || (!dev->of_node))  return NULL;
    return of_match_node(matchds, dev->of_node);
}


struct device_node *of_node;
struct device_node {
    const char *name;

    struct property *properties;
    struct device_node *parent;
    struct device_node *child;
    struct device_node *sibling;
    ...
};
```

99. Testing device tree changes on board

    >> on VBox

```
    / {
        pcdev1: pcdev-1 {
            compatible = "pcdev-E1x","pcdev-A1x";
            org,size = <512>;
            org,device-serial-num = "PCDEV1ABC123";
            org,perm = <0x11>;
        };
    };

    # make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x-boneblack.dtb
    # scp arch/arm/boot/dts/am335x-boneblack.dtb debian@192.168.7.2:/home/debian/drivers

    or in Makefile,
```

```
        copy-dtb:
            scp     ~/workspace/ldd/source/linux_bbb_5.4/arch/arm/boot/dts/am335x-boneblack.dtb
            debian@192.168.7.2:/home/debian/drivers

        copy-drv:
            scp *.ko  debian@192.168.7.2:/home/debian/drivers


    >> on beaglebone

        # mount /dev/mmcblk0p1 /mnt/
        # cd /mnt/
        # ls
          am335x-boneblack_4.14.dtb     MLO     uImage      uEnv.txt
        # cp /home/debian/drivers/am335x-boneblack.dtb .
        # sync
        # reboot

        # ls /sys/devices/platform/
          ...       pcdev-1     pcdev-2     pcdev-3     pcdev-4     ...
        # ls /sys/devices/platform/pcdev-1
          driver_override       modalias    of_node     power       subsystem       uevent
        # ls /sys/devices/platform/pcdev-1/of_node
          compatible    name    org,device-serial-num   org,perm    org,size
        # cat /sys/devices/platform/pcdev-1/of_node/compatible
          pcdev-E1xpcdev-A1x


        $ sudo insmod pcd_platform_driver_dt.ko

        $ dmesg
        [ 168.xx]  pcd_platform_driver_dt: loading out-of-tree module taint kernel.
        [ 168.xx]  pcd_platform_driver_probe : A device is detected
        [ 168.xx]  pcd_platform_driver_probe : A device is detected
        [ 168.xx]  pcd_platform_driver_probe : A device is detected
        [ 168.xx]  pcd_platform_driver_probe : A device is detected
        [ 168.xx]  pcd_platform_driver_init : pcd platform driver loaded

        $ sudo rmmod pcd_platform_driver_dt
        [ 168.xx]  pcd_platform_driver_remove : A device is remove
        [ 168.xx]  pcd_platform_driver_remove : A device is remove
        [ 168.xx]  pcd_platform_driver_remove : A device is remove
        [ 168.xx]  pcd_platform_driver_remove : A device is remove
        [ 168.xx]  pcd_platform_driver_cleanup : pcd platform driver unloaded


100. Pcd platform driver DT coding part-2                          304~

    struct platform_device {
        struct device               dev;
```

```
        {
            struct device_node      *of_node;   // represents an associated device tree node
            struct fwnode_handle    *fwnode;    // fw device node
        };
    };
```

linux/include/of.h

```
    of_property_read_string(const struct device_node *np,
                            const char *propname,
                            const char **out_string);
```

pr_info(...) vs dev_info(dev, ...) -> can't use in _init & _exit where dev is not available)

101. Pcd platform driver DT coding part-3

102. Pcd platform driver DT coding part-4

9 device tree overlays 54m 317

103. Introduction to device tree overlays
104. Device tree overlays exercise
105. Updating u-boot
106. Updating u-boot contd.
107. Applying overlays using u-boot commands
108. Applying overlays and testing using u-boot uEnv.txt file

10 Linux device driver model 1h32m 341

109. Linux device model
110. kobjects
111. kobject type and kset
112. sysfs and kobject attributes
113. Creating sysfs attributes
114. pcd_sysfs_attributes coding part 1

115. pcd_sysfs_attributes coding part 2
116. pcd sysfs kernel module testing
117. show and store methods of the sysfs attributes
118. pcd_sysfs_attributes coding part 3
119. Attribute grouping

11 Linux GPIO subsystem 32m ???

120. Introduction
121. GPIOs of BBB
122. Pad configuration register
123. Linux GPIO subsystem
124. Consumer accessing GPIO pins

12 GPIO sysfs driver implementation 2h

125. Exercise GPIO Sysfs driver implementation
126. GPIO Sysfs driver implementation part-1
127. GPIO Sysfs driver implementation part-2
128. GPIO Sysfs driver implementation part-3
129. GPIO Sysfs driver implementation part-4
130. GPIO Sysfs driver implementation part-5
131. GPIO Sysfs driver implementation part-6
132. GPIO Sysfs driver implementation part-7
133. GPIO Sysfs driver implementation part-8
134. Gpio syfs driver testing

13 pin control subsystem of Linux 2h2m

135. Pin control subsystem and pin controller
136. Writing pin configuration node
137. LCD exercise
138. Significance of LCD application
139. Adding pin configuration node for gpios
140. Exploring LCD code
141. Sending command to LCD
142. Creating LCD command code
143. Testing LCD application over gpio sysfs
144. Assignment : Implementing LCD platform driver

14 Linux synchronization services 1h35m

145. Avoiding race conditions
146. Linux locking services
147. Spinlock Vs Mutex
148. Linux spinlock functions
149. Mutex
150. Using locking functions in the code
151. BONUS LECTURE