EMBEDDED DEVICE DRIVERS

Linux Device Drivers on Beaglebone Black

LKM: Kernel Timers

- The kernel keeps track of time
  - By means of timer interrupts
  - Generated at regular pre-defined intervals
  - Using the system time hardware

- Kernel timers are used:
  - To poll hardware at fixed intervals
    - When hardware cannot generate interrupts
  - To send data/messages at fixed intervals
  - To time out waiting for events

LKM: Kernel Timer API

- Regular timers are defined as structures in the kernel

  *struct **timer_list** {*

  …
  *unsigned long **expires**;*
  *void (***function**)(unsigned long);*
  *unsigned long **data**;*
  }

  - *expires: Contains expiration time in **jiffies***
  - *function: Callback function called on expiry*
  - *data: Data to pass to **function()***

- Creation

  *void **timer_setup**(struct timer_list *timer, void (*function)(unsigned long), unsigned long data);*

- Start / modification

  *int **mod_timer**(struct timer_list *timer, unsigned long expires);*

- Stop / deactivate

  *int **del_timer**(struct timer_list *timer);*

- Header file:

  # include <linux/timer.h>

LKM: Kernel timer callback

- The callback function
  - Called on timer expiry (one-shot / periodic)
  - Executes in interrupt context!
    - *Check using the **in_interrupt()** function*
  - Hence, we should not do any of these in the callback:
    - Go to sleep / relinquish the CPU
    - (Try to) acquire a mutex
    - Perform time-consuming tasks
    - Access user space virtual memory address(es)

LKM: HR Timer

- The Kernel Timer is based on jiffies
  - Defined when computers were not as fast as today
  - While it gives decent performance
    - It is not acceptable for devices
      - That want nanosecond level resolution

- Kernel v2.6.21 onwards
  - Introduced the High Resolution Timer (hrtimer)
    - With **nanosecond** level resolution (64-bit)

  - Available only when kernel is compiled with
    - ***CONFIG_HIGH_RES_TIMER=y***
      - *Check in /boot/config…*

  - To check if HR timer is supported:

    $ cat /proc/timer_list
    - resolution: 1 nsecs

LKM: HR Timer API (1/2)

- Header files:
  #include <linux/hrtimer.h>
  #include <linux/ktime.h>

- New datatype to store time in nanosecs: ***ktime_t***

- Convert from secs, nsecs to ktime_t:
  ktime_t ***ktime_set****(long secs, long nanosecs);*

- Define an HR Timer
  struct hrtimer *my_hrtimer;*

- Initialize HR timer
  void ***hrtimer_init****(struct hrtimer **timer**, clockid_t **clock_id**, enum hrtimer_mode **mode**);*

  - *timer: HR timer*
  - *clock_id: CLOCK_MONOTONIC / CLOCK_BOOTTIME / CLOCK_REALTIME / CLOCK_TAI*
  - *mode: HRTIMER_MODE_ABS / HRTIMER_MODE_REL*

LKM: HR Timer API (2/2)

- Write a timer **callback**
  - Prototype:
    *enum hrtimer_restart **my_timer_cb**(struct hrtimer *timer);*
  - Return HRTIMER_RESTART / HRTIMER_NO_RESTART

- Connect HR timer with its callback
  my_hrtimer.function = my_timer_cb;

- Start / Stop HR timer
  *int **hrtimer_start**(struct hrtimer ***timer**, ktime_t **time**, const enum hrtimer_mode **mode**);*
  *int **hrtimer_cancel**(struct hrtimer *timer);*

- Forward HR timer (for periodic behavior)
  *u64 **hrtimer_forward_now**(struct hrtimer *timer, ktime_t interval);*

LKM: Timers exercise

- Refer ***mod7*** directory
  - The file ***mod71.c*** contains driver code for kernel timer
    - Study kernel timer setup and callback definitions
    - Compile and load on BBB
    - Observe ***dmesg*** output; unload driver
  - The file ***mod72.c*** contains driver code for HR timer
    - Study HR timer setup and callback definitions
    - Compile and load on BBB
    - Observe ***dmesg*** output; unload driver
    - Compare timestamps for the 2 types of timers
  - The file ***mod73.c*** contains driver code for *time differencing*
    - This uses **msleep()** from ***<linux/delay.h>***
    - Compare ***msleep()*** precision vis-à-vis ***HR timer***

THANK YOU!