Modelo de dispositivos y drivers en linux

Mg. Ing. Pablo Slavkin
Mg. Ing. Hanes N. Sciarrone
MSE - 2024

Implementación de Manejadores de Dispositivos

Device and driver model

- Introducción.
- Bus Core drivers y Controladores de Bus.
- Device drivers.
- Introducción al DeviceTree.

Introducción

Introducción

- El modelo unificado de dispositivos fue presentado en la versión 2.6 de linux.
- Finalidad: presentar un mecanismo sencillo para representar dispositivos y describir su topología.
- El modelo de dispositivos y drivers es una manera universal de organizar dispositivos.
- Dispositivos y drivers se organizan en distintos buses.

Introducción

- Ventajas del modelo de drivers y dispositivos unificado:
  - Minimiza el duplicado de código.
  - Organización clara y limpia del código, separando drivers de dispositivos, descripción de hardware, etc.
  - Capacidad de determinar todos los dispositivos en el sistema, estado de los mismos, a que bus están conectados y que driver es responsable.
  - Capacidad de linkear drivers con dispositivos y viceversa.

Introducción

- Ventajas del modelo de drivers y dispositivos unificado (cont):
  - Categorización de los dispositivos por su tipo (clase).
  - Evita tener que saber la topología física de los dispositivos.
  - Administración del ciclo de vida de los objetos.
  - Administración de energía, especialmente en el orden que deben ser apagados los dispositivos.
  - Comunicación al espacio usuario a traves del filesystem virtual **sysfs**.

Introducción

- Glosario:
  - **Device:** Objeto físico o virtual que se conecta a un bus.
  - **Driver:** Entidad de software que puede sondear y ser vinculada a dispositivos (devices).
    El driver puede efectuar ciertas funciones de gestión sobre los dispositivos.
  - **Bus:** un dispositivo que sirve de punto de conexión para otros dispositivos.

Introducción

- El modelo de dispositivos está organizado alrededor de tres estructuras de datos principales:
  - **struct bus_type:** representa un tipo de bus (PCI, I2C, USB).
  - **struct device_driver:** representa un driver capaz de manipular ciertos dispositivos en un bus determinado.
  - **struct device:** representa un dispositivo conectado a un bus.
- Conocer la manera en que se organizan los dispositivos hará más sencilla la tarea de crear un driver.

Bus Core drivers y Controladores de Bus

Bus Core drivers y Controladores de Bus

- Por cada bus soportado por el kernel, existe un driver de core genérico asociado.
- Recordar que un bus es un canal entre el procesador y uno o más dispositivos (esquema ordenador).
- Para todo propósito del modelo de dispositivos, todos los dispositivos están conectados a un bus.
- El mencionado bus no necesariamente es un bus físico, puede ser virtual.

Bus Core drivers y Controladores de Bus

- El core driver de bus asigna una estructura **bus_type** y la registra según la lista de tipo de buses del kernel.
- **struct bus_type** está definido en el archivo **include/linux/device.h**
- Como se mencionó antes, representa un tipo de bus (I2C, USB, etc).
- Un ejemplo de instanciación de esta estructura es el core bus platform, definido en **drivers/base/platform.c**

Bus Core drivers y Controladores de Bus

- Uno de los campos de la estructura **bus_type** es un puntero a **struct subsys_private**, definido en **drivers/base/base.h**
- El miembro **klist_devices** de esta estructura **subsys_private** es una lista de los dispositivos del sistema asociados a un bus.
- Esta lista se actualiza al llamar a la función **device_register()**.
- Esta función se invoca al escanear el bus para detectar nuevos dispositivos durante la inicialización del sistema o en hotplug.

Bus Core drivers y Controladores de Bus

- El miembro **klist_drivers** de **subsys_private** es una lista de los drivers que pueden manejar dispositivos en ese bus.
- Esta lista se actualiza al llamar a la función **driver_register()**.
- Esta función se invoca al cuando un driver se inicializa (ejemplo un módulo que se carga).
- **NOTA**: Aquí diferenciamos que *no todos los módulos son drivers*.

Bus Core drivers y Controladores de Bus

- El proceso de registrado para dispositivos que son detectables (discoverable) es el siguiente:
  - El driver controlador de bus detecta el dispositivo e invoca a **device_register()**.
  - En este caso el miembro ancestro (parent) de la estructura **device** es apuntado al controlador de bus (quien lo invoca).
  - La lista de drivers asociados al bus es iterada para encontrar uno que pueda manejar el nuevo dispositivo (funcion **match**).
  - Al encontrar una coincidencia, el miembro **driver** de la estructura **device** es apuntada a dicho device driver.

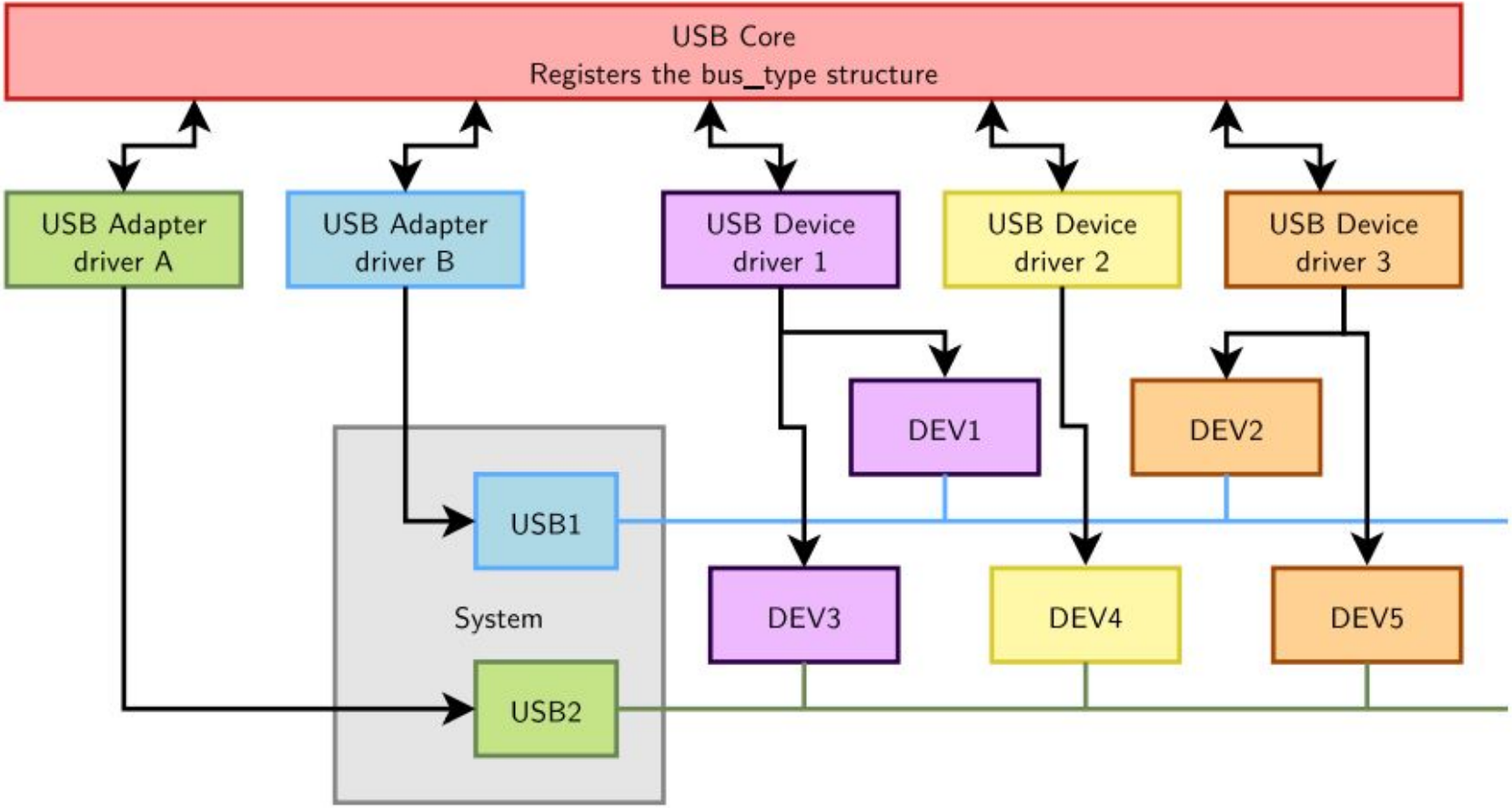
Bus Core drivers y Controladores de Bus

- Cuando se inserta un módulo en el kernel, y este invoca **driver_register():**
  - Se itera la lista de dispositivos asociados al bus.
  - Se utiliza la función **match()** para determinar si algún dispositivo puede ser manejado por este driver.
  - Al encontrar una coincidencia, el dispositivo se asocia con el device driver.
  - Se invoca la función **probe()** del driver, lo que recibe el nombre de *binding*.

Bus Core drivers y Controladores de Bus

- Existen dos ocasiones donde un driver intenta hacer un bind de dispositivo:
  a. Cuando el driver se registra (si el dispositivo ya existe).
  b. Cuando se crea el dispositivo (si el driver ya está registrado).
- En resumen, el driver de bus registra un bus en el sistema y:
  a. Permite que se registren drivers controladores de bus (detectan dispositivos y configuran sus recursos).
  b. Permite el registro de device drivers.
  c. Asocia dispositivos (devices) y drivers.

Bus Core drivers y Controladores de Bus



Device drivers

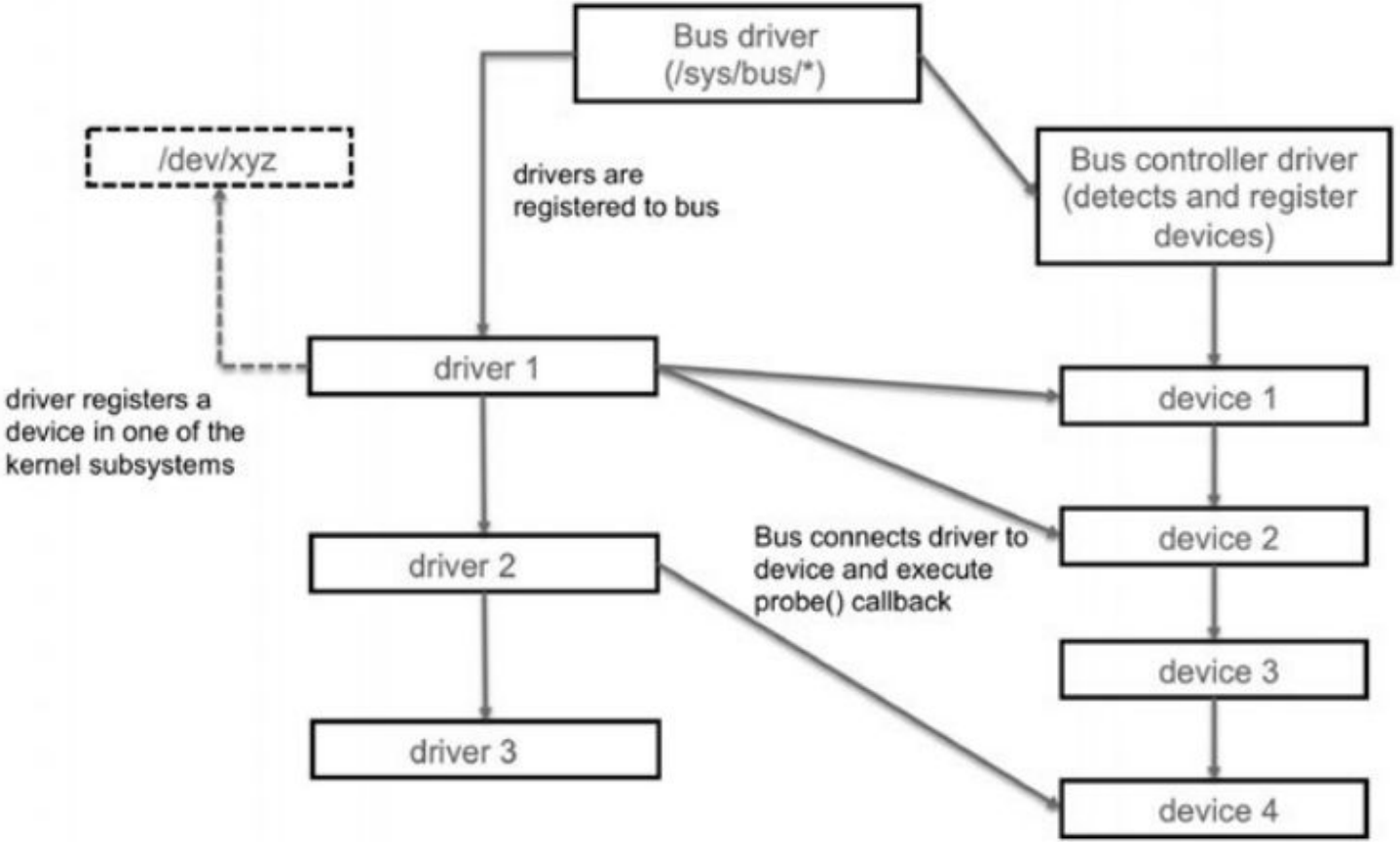
Device Drivers

- Como se menciona en slides anteriores, cada device driver se registra con el bus core driver utilizando **driver_register().**
- Luego el device model core trata de hacer un bind con un dispositivo.
- Al detectarse un device que puede ser manejado por este driver, se invoca al miembro **probe()** del driver.
- La información de configuración se obtiene desde el Device Tree.

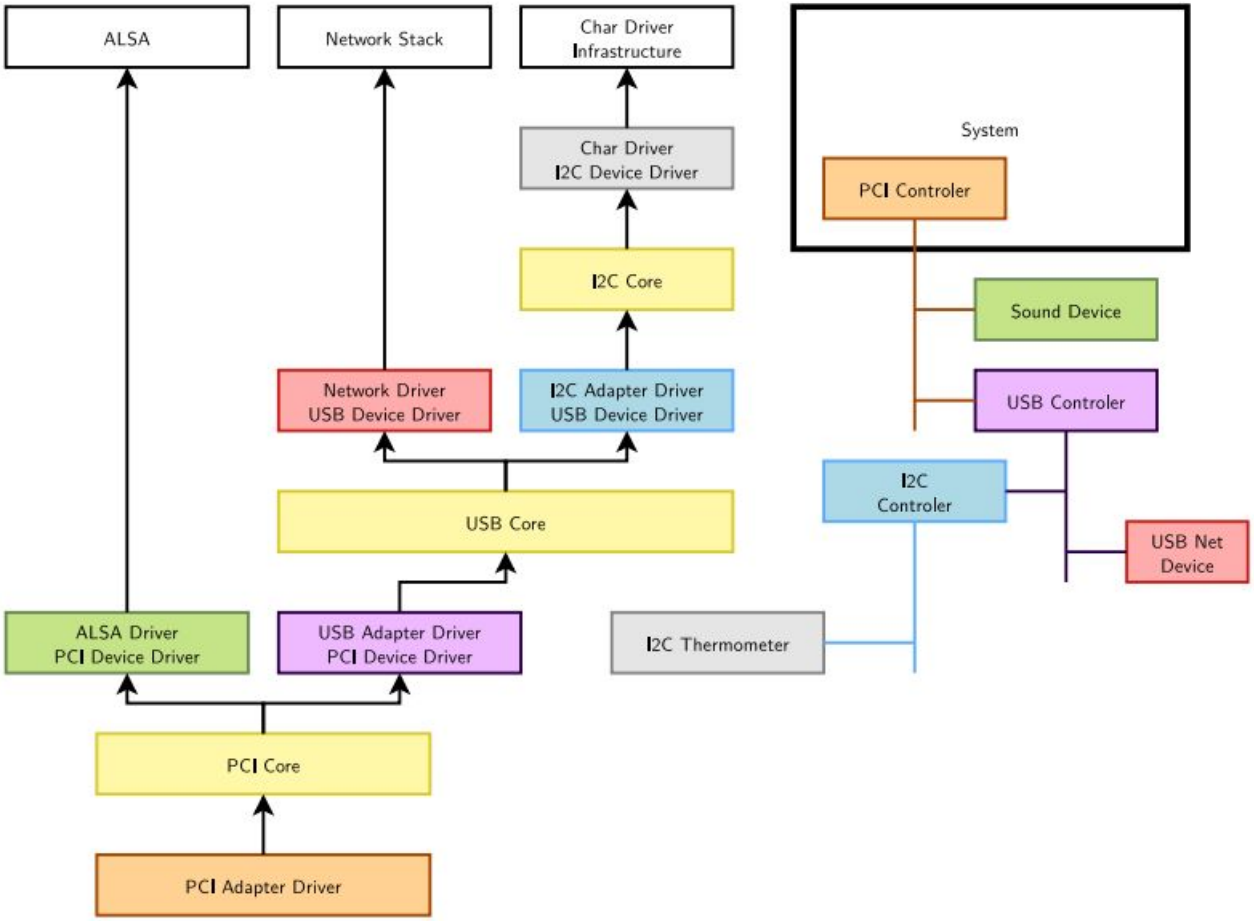
Device Drivers

- Cada driver es responsable de instanciar y registrar una instancia de la estructura **device_driver**.
- Esta estructura se encuentra definida en **include/linux/device.h**
- El miembro **bus** es un puntero del tipo **struct bus_type** donde el device driver está registrado.
- El miembro **probe** es una función callback, que se invoca por cada dispositivo detectado y soportado por el driver.
- El miembro **remove** es una función callback, que se invoca cuando el dispositivo se remueve o se remueve el driver.

Device Drivers



Device Drivers



Gracias.