**1. Sysfs comes first.**

* **Sysfs is a virtual filesystem that the Linux kernel creates in /sys to expose information about devices, drivers, and other kernel objects. It is mounted very early during the boot process, before user-space processes like udevd start.**

**2. Udev comes later.**

* **Udev is a user-space daemon (udevd) that listens for kernel events (via uevent) and dynamically manages device nodes in /dev. It relies on sysfs to get device attributes and apply rules for naming, permissions, and symbolic links.**

**Sequence of Events**

1. **The kernel detects a hardware device.**
2. **The kernel creates an entry in sysfs (/sys).**
3. **The kernel generates a uevent (kernel event).**
4. **Udev receives this event and queries sysfs for more details.**
5. **Udev applies rules to name the device, set permissions, and create /dev nodes.**

**Since sysfs is created by the kernel first and udev depends on it for device information, sysfs comes before udev in the boot process.**

**Sysfs (/sys) is a virtual filesystem that the Linux kernel creates to expose information about devices, drivers, and kernel subsystems. It is populated dynamically by the kernel based on hardware detection and driver initialization.**

**Steps in Sysfs Population**

**1. Kernel Initialization**

* **During boot, the kernel initializes core subsystems like PCI, USB, and block devices.**
* **It mounts sysfs (/sys) early in the boot process, usually via mount -t sysfs sysfs /sys.**

**2. Device Discovery by the Kernel**

* **The kernel detects hardware components using buses like PCI, USB, I2C, SPI, etc.**
* **When a new device is detected, the kernel creates a struct device in memory to represent it.**

**3. Kernel Populates Sysfs Entries**

* **Each device in the kernel’s internal data structures corresponds to an entry in sysfs.**
* **The kernel uses kobject structures and the driver model to create directories and files in /sys.**
* **Example: If a new PCI device is detected, the kernel adds /sys/bus/pci/devices/0000:00:1f.2/.**

**4. Attributes and Files Creation**

* **Each kobject can have attributes (files in /sys) that provide device properties.**
* **These attributes are exposed using sysfs\_create\_file() or sysfs\_create\_group().**

**5. Dynamic Updates**

* **Sysfs is dynamically updated when devices are added or removed.**
* **If a device is hot-plugged (like a USB drive), the kernel creates new sysfs entries.**
* **When a device is removed, its corresponding sysfs entry disappears.**

**Example: A PCI Device Entry in Sysfs**

**/sys/bus/pci/devices/0000:00:1f.2/**

**│── driver -> ../../../../bus/pci/drivers/ahci**

**│── resource**

**│── subsystem -> ../../../../bus/pci**

**│── uevent**

**│── vendor**

**│── device**

**│── class**

**Sysfs is populated by the kernel, using internal data structures like kobject and struct device. It is updated dynamically as hardware is detected, initialized, or removed. User-space tools like udev use sysfs to manage devices dynamically.**

**Sysfs is updated dynamically by the Linux kernel whenever devices, drivers, or other kernel objects are added or removed. This happens through kobjects, udev events, and hotplug mechanisms. Let's break down the internals step by step.**

**1. Core Components Involved in Sysfs Updates**

* 1. **Kobjects (struct kobject)**
* **A kobject represents a kernel object (e.g., a device, driver, or subsystem). Each kobject is associated with sysfs and is linked to a parent (hierarchical structure).The kernel uses kobject\_add() to add a new object to sysfs.**

**1.2 Kset (struct kset)**

* **A kset is a collection of kobjects. It represents groups like /sys/class, /sys/bus, etc.**

**1.3 Attributes (struct attribute)**

* **Sysfs exposes information via attributes (files in /sys). These are defined using sysfs\_create\_file() or sysfs\_create\_group().**

**2. How a New Device Gets Added to Sysfs**

**Step 1: Device Discovery**

* **When a device is detected (e.g., via PCI, USB, or another bus), the kernel calls device\_register().**
* **This creates a struct device and associates it with a bus (e.g., PCI, USB).**

**Step 2: Creating a Kobject**

* **The kernel calls:**

**kobject\_init(&dev->kobj, &device\_ktype);**

**kobject\_add(&dev->kobj, parent\_kobj, name);**

* **This adds a new directory in /sys/bus/<bus\_name>/devices/.**

**Step 3: Creating Sysfs Files**

* **The device’s attributes are created via:**

**sysfs\_create\_file(&dev->kobj, &attr);**

* **This results in files like /sys/devices/pci0000:00/0000:00:1f.2/vendor.**

**Step 4: Notifying Userspace (Udev)**

* **The kernel sends a uevent using kobject\_uevent(), which informs udev about the new device.**
* **Example event:**

**add@/devices/pci0000:00/0000:00:1f.2**

* **Udev reads sysfs attributes and applies rules to manage /dev nodes.**

**3. How Sysfs Handles Device Removal**

**Step 1: Kernel Removes Device**

* **When a device is removed (e.g., USB unplugged), the kernel calls:**

**device\_unregister(dev);**

* **This removes the struct device from the kernel’s internal lists.**

**Step 2: Removing Sysfs Entries**

* **The kernel calls:**

**sysfs\_remove\_file(&dev->kobj, &attr);**

**kobject\_del(&dev->kobj);**

* **This deletes the corresponding /sys directory.**

**Step 3: Notifying Userspace**

* **The kernel generates a remove uevent:**

**remove@/devices/pci0000:00/0000:00:1f.2**

* **Udev processes the event and removes related /dev nodes.**

**4. Hotplug Mechanism and Dynamic Updates**

* **Hotplugging works via kobject\_uevent(), which triggers user-space scripts.**
* **Dynamic sysfs updates are handled when:** 
  + **A new device is added (creates sysfs entries).**
  + **A device driver is loaded (links /sys/class/<driver>).**
  + **A device is removed (removes sysfs entries).**

**5. Example: Adding a USB Device**

**1. Device Detected**

* **The kernel detects a new USB device and creates /sys/bus/usb/devices/1-1/.**

**2. Sysfs Files Created**

* **Attributes like /sys/bus/usb/devices/1-1/idVendor appear.**

**3. Udev Processes Event**

* **Udev applies rules and creates /dev/bus/usb/001/001.**

**4. Device Removed**

* **The kernel calls device\_unregister(), deleting /sys/bus/usb/devices/1-1/.**

**In the Linux kernel, device\_register() and device\_unregister() are used to add and remove devices dynamically. These functions are typically called by bus drivers, platform drivers, and subsystem code when devices are detected or removed.**

**1. Understanding device\_register()**

**device\_register() is called to register a device with the Linux device model. It creates a struct device entry in /sys/devices/ and notifies user-space via udev events.**

**Function Call Flow**

**int device\_register(struct device \*dev) {**

**dev->kobj.kset = devices\_kset;**

**return device\_add(dev);**

**}**

**This internally calls:**

1. **device\_initialize(dev)**
   * **Initializes the device’s kobject.**
   * **Sets up parent-child relationships.**
2. **kobject\_add(&dev->kobj, parent, name)**
   * **Creates a sysfs entry (/sys/devices/...).**
3. **bus\_add\_device(dev)**
   * **Adds the device to the associated bus (PCI, USB, etc.).**
   * **Calls bus->match(dev, drv) to find a driver.**
4. **device\_add\_attrs(dev)**
   * **Creates sysfs attributes (like idVendor, idProduct for USB).**
5. **kobject\_uevent(&dev->kobj, KOBJ\_ADD)**
   * **Sends a "device added" event to user-space (udevd listens to this).**

**2. Where device\_register() Gets Called**

**Case 1: Platform Drivers (e.g., SoC Devices)**

* **In platform drivers (platform\_driver\_probe()):**

**struct platform\_device \*pdev = platform\_device\_alloc("my\_device", -1);**

**device\_register(&pdev->dev);**

**Case 2: PCI Drivers**

* **When a PCI device is detected, the kernel PCI subsystem calls:**

**pci\_device\_probe() -> device\_register(&pci\_dev->dev);**

**Case 3: USB Drivers**

* **When a new USB device is connected:**

**usb\_new\_device() -> device\_register(&usb\_dev->dev);**

**3. Understanding device\_unregister()**

**device\_unregister() is used to remove a device from the kernel when it is no longer needed (e.g., USB unplugged).**

**Function Call Flow**

**void device\_unregister(struct device \*dev) {**

**device\_del(dev);**

**put\_device(dev);**

**}**

**This internally calls:**

1. **device\_remove\_file()**
   * **Removes sysfs attributes.**
2. **bus\_remove\_device(dev)**
   * **Removes the device from its associated bus.**
3. **kobject\_del(&dev->kobj)**
   * **Deletes the sysfs entry (/sys/devices/...).**
4. **kobject\_uevent(&dev->kobj, KOBJ\_REMOVE)**
   * **Sends a "device removed" event to user-space.**

**4. Where device\_unregister() Gets Called**

**Case 1: Platform Drivers**

* **When a platform device is removed:**

**platform\_device\_unregister(pdev);**

**Case 2: PCI Drivers**

* **When a PCI device is removed:**

**pci\_stop\_device(pci\_dev);**

**device\_unregister(&pci\_dev->dev);**

**Case 3: USB Drivers**

* **When a USB device is disconnected:**

**usb\_disconnect() -> device\_unregister(&usb\_dev->dev);**

**5. Example: USB Device Insertion and Removal**

**Insertion Flow**

1. **User plugs in a USB device.**
2. **Kernel detects the device and calls:**

**usb\_new\_device() -> device\_register(&usb\_dev->dev);**

1. **A new sysfs entry appears in /sys/bus/usb/devices/.**
2. **Udev creates /dev/bus/usb/....**

**Removal Flow**

1. **User removes the USB device.**
2. **Kernel calls:**

**usb\_disconnect() -> device\_unregister(&usb\_dev->dev);**

1. **The sysfs entry is removed.**
2. **Udev removes the /dev node.**

**Conclusion**

* **device\_register() is called when a new device is detected (via PCI, USB, or platform bus).**
* **device\_unregister() is called when a device is removed.**
* **These functions are used inside bus drivers and platform drivers to manage device lifecycle.**
* **Sysfs updates dynamically based on these calls.**

**When a hotplug device (e.g., USB, PCI, or hot-swappable storage) is inserted, the Linux kernel dynamically detects the device, registers it, and creates sysfs entries. Let's go through the call sequence of how device\_register() and device\_unregister() are invoked during hotplug events.**

**1. High-Level Sequence of Events (USB Example)**

1. **Hardware Event: A USB device is physically plugged in.**
2. **Interrupt Handling: The host controller (e.g., xhci\_hcd for USB 3.0) triggers an interrupt.**
3. **Device Enumeration:** 
   * **The kernel queries the device via the USB bus.**
   * **Reads vendor ID, product ID, and class information.**
4. **Device Registration:** 
   * **Kernel calls device\_register() to create a struct device.**
   * **This creates sysfs entries under /sys/bus/usb/devices/.**
5. **Userspace Notification:** 
   * **The kernel sends a uevent to udev, which creates /dev nodes.**

**When a device is removed, the same process is reversed using device\_unregister().**

**2. Kernel Call Sequence for Device Hotplugging (USB Example)**

**Step 1: Hardware Event and Interrupt Handling**

**When a USB device is plugged in:**

* **The host controller detects it and generates an interrupt.**
* **The corresponding Interrupt Service Routine (ISR) is called:**

**xhci\_irq() // Handles USB interrupt**

* **This triggers USB enumeration.**

**Step 2: Device Enumeration (USB Stack)**

**The kernel queries the device through the USB stack:**

1. **USB Core Handles Device Addition**

**hub\_event() // Detects new device on the USB hub**

**usb\_add\_device() // Adds new device to USB core**

1. **Creates a USB Device Structure (struct usb\_device)**

**usb\_alloc\_dev()**

**Step 3: Device Registration (device\_register())**

**Once the USB device is enumerated, it is registered as a device:**

1. **The kernel calls:**

**usb\_new\_device() -> device\_register(&usb\_dev->dev);**

1. **This calls:**

**device\_initialize() // Initializes kobject**

**device\_add() // Creates sysfs entries**

1. **A new sysfs entry appears under:**

**/sys/bus/usb/devices/1-1/**

1. **The kernel sends a uevent to notify user-space:**

**kobject\_uevent(&usb\_dev->dev.kobj, KOBJ\_ADD);**

**Step 4: Udev Handles Device in Userspace**

* **Udevd (/sbin/udevd) receives the event and reads sysfs attributes.**
* **Based on udev rules (/etc/udev/rules.d/), it may:** 
  + **Create a /dev node (e.g., /dev/sdb for USB storage).**
  + **Set permissions or symlinks (/dev/usb/mydevice).**

**Example udev rule:**

**ACTION=="add", SUBSYSTEM=="usb", ATTR{idVendor}=="1d6b", ATTR{idProduct}=="0002", SYMLINK+="my\_usb\_device"**

**3. Call Sequence for Device Removal (device\_unregister())**

**When the USB device is unplugged, the kernel reverses the process.**

**Step 1: Hardware Event (Device Removal)**

* **The USB host controller detects the disconnection and generates an interrupt.**
* **The kernel calls:**

**hub\_event() // Detects device removal**

**usb\_disconnect() // Starts cleanup process**

**Step 2: Device Unregistration**

1. **The kernel removes the device:**

**device\_unregister(&usb\_dev->dev);**

1. **Internally, this calls:**

**device\_del() // Removes sysfs entry**

**kobject\_uevent(&usb\_dev->dev.kobj, KOBJ\_REMOVE); // Notifies udev**

**Step 3: Udev Cleans Up Userspace**

* **Udevd receives the KOBJ\_REMOVE event.**
* **It removes the corresponding /dev entry.**

**4. Call Flow for PCI Hotplug (Example)**

**PCI devices (e.g., hot-swappable NVMe SSDs, Thunderbolt devices) follow a similar flow:**

**Device Insertion (PCI)**

1. **Interrupt Handling**

**pci\_scan\_single\_device() // Scans for new PCI device**

**pci\_bus\_add\_device() // Adds PCI device**

**device\_register(&pci\_dev->dev);**

1. **Sysfs Entry is Created**

**/sys/bus/pci/devices/0000:00:1f.2/**

1. **Udev Handles the Event**
   * **Creates /dev/nvme0n1 for an NVMe SSD.**

**Device Removal (PCI)**

1. **Kernel detects removal:**

**pci\_remove\_device()**

**device\_unregister(&pci\_dev->dev);**

1. **Sysfs Entry is Deleted**

**/sys/bus/pci/devices/0000:00:1f.2/ (removed)**

1. **Udev Cleans Up /dev/nvme0n1**

**5. Summary: Key Function Calls for Hotplugging**

**Device Insertion (Hotplug)**

|  |  |  |
| --- | --- | --- |
| **Step** | **Function Call** | **Description** |
| **1** | **xhci\_irq() (USB) / pci\_scan\_single\_device() (PCI)** | **Handles hardware detection** |
| **2** | **hub\_event() (USB) / pci\_bus\_add\_device() (PCI)** | **Enumerates device** |
| **3** | **usb\_new\_device() / pci\_device\_probe()** | **Registers device** |
| **4** | **device\_register(&dev->dev);** | **Registers with sysfs** |
| **5** | **kobject\_uevent(KOBJ\_ADD)** | **Sends event to user-space** |
| **6** | **udevd processes /sys** | **Creates /dev entry** |

**Device Removal (Hot-unplug)**

|  |  |  |
| --- | --- | --- |
| **Step** | **Function Call** | **Description** |
| **1** | **usb\_disconnect() / pci\_remove\_device()** | **Detects device removal** |
| **2** | **device\_unregister(&dev->dev);** | **Removes device from kernel** |
| **3** | **kobject\_uevent(KOBJ\_REMOVE)** | **Notifies user-space** |
| **4** | **udevd removes /dev node** | **Cleans up userspace** |

**Hotplugging allows devices (e.g., USB, PCIe, NVMe, Thunderbolt) to be dynamically added and removed. The Linux kernel detects these events, updates the sysfs filesystem, and notifies udev to manage /dev nodes.**

**1. High-Level Summary of Hotplug Flow**

1. **Device Insertion (Plugging in a USB, PCIe, etc.)**
   * **Hardware generates an interrupt.**
   * **The kernel enumerates the device (e.g., for USB: reads vendor & product IDs).**
   * **device\_register() creates a sysfs entry.**
   * **The kernel sends a uevent (KOBJ\_ADD) to udev.**
   * **Udevd reads sysfs attributes and creates a /dev node.**
2. **Device Removal (Unplugging)**
   * **Hardware detects removal, generating an interrupt.**
   * **The kernel calls device\_unregister() to remove the device.**
   * **Sysfs entry is deleted.**
   * **Udevd removes the /dev node.**

**2. Detailed Function Call Flow**

**A. Device Insertion (Hotplug)**

**Step 1: Hardware Event & Interrupt Handling**

* **A device is physically inserted.**
* **The device controller (USB host, PCIe controller, etc.) detects the change and generates an interrupt.**
* **The kernel calls the appropriate Interrupt Service Routine (ISR):** 
  + **USB: xhci\_irq(), ehci\_irq()**
  + **PCIe: pciehp\_isr()**
  + **NVMe: nvme\_irq()**

**Step 2: Device Enumeration**

* **The kernel scans for newly connected devices:** 
  + **USB: hub\_event() → usb\_add\_device()**
  + **PCIe: pci\_scan\_single\_device() → pci\_bus\_add\_device()**
  + **NVMe: nvme\_probe()**
* **It queries the device vendor ID, product ID, and class information.**

**Step 3: Device Registration (device\_register())**

* **A struct device is created:**

**device\_register(&dev->dev);**

* **Internally, it calls:**

**device\_initialize(dev); // Initializes kobject**

**device\_add(dev); // Adds to sysfs**

* **Sysfs entry appears:**

**/sys/bus/usb/devices/1-1/**

**/sys/bus/pci/devices/0000:00:1f.2/**

**Step 4: Kernel Sends a Uevent to Userspace**

* **The kernel generates a uevent to notify userspace:**

**kobject\_uevent(&dev->kobj, KOBJ\_ADD);**

* **udevd listens for the event.**

**Step 5: Udev Creates /dev Node**

* **Udevd processes sysfs attributes and applies rules.**
* **It creates a device node, e.g.:**

**/dev/sdb (USB storage)**

**/dev/nvme0n1 (NVMe SSD)**

**B. Device Removal (Hot-Unplug)**

**Step 1: Hardware Detects Device Removal**

* **The device controller detects removal and generates an interrupt.**
* **The kernel calls:** 
  + **USB: usb\_disconnect()**
  + **PCIe: pci\_remove\_device()**
  + **NVMe: nvme\_remove()**

**Step 2: Device Unregistration (device\_unregister())**

* **The kernel removes the device:**

**device\_unregister(&dev->dev);**

* **Internally calls:**

**device\_del(dev); // Removes from sysfs**

**kobject\_uevent(&dev->kobj, KOBJ\_REMOVE);**

* **Sysfs entry is deleted:**

**/sys/bus/usb/devices/1-1/ (removed)**

**Step 3: Udev Cleans Up Userspace**

* **Udevd receives KOBJ\_REMOVE event.**
* **It removes /dev entries, e.g.:**

**rm /dev/sdb**

**3. Call Flow Table for Hotplug Devices**

|  |  |  |
| --- | --- | --- |
| **Stage** | **USB Function Calls** | **PCIe Function Calls** |
| **Device Inserted** | **xhci\_irq()** | **pciehp\_isr()** |
| **Device Scanned** | **hub\_event() → usb\_add\_device()** | **pci\_scan\_single\_device()** |
| **Device Registered** | **usb\_new\_device() → device\_register()** | **pci\_device\_probe() → device\_register()** |
| **Sysfs Updated** | **/sys/bus/usb/devices/1-1/** | **/sys/bus/pci/devices/0000:00:1f.2/** |
| **Uevent Sent** | **kobject\_uevent(KOBJ\_ADD)** | **kobject\_uevent(KOBJ\_ADD)** |
| **Userspace Creates /dev Node** | **udevd → /dev/sdb** | **udevd → /dev/nvme0n1** |
| **Device Unplugged** | **usb\_disconnect() → device\_unregister()** | **pci\_remove\_device() → device\_unregister()** |
| **Sysfs Entry Removed** | **/sys/bus/usb/devices/1-1/ (deleted)** | **/sys/bus/pci/devices/0000:00:1f.2/ (deleted)** |
| **Userspace Cleans Up** | **udevd → rm /dev/sdb** | **udevd → rm /dev/nvme0n1** |

**Conclusion:**

1. **Device Detection: The kernel detects hotplug devices using hardware interrupts.**
2. **Enumeration & Registration: The kernel probes the device, assigns it an ID, and registers it using device\_register().**
3. **Sysfs & Udev: sysfs updates dynamically, and udev manages /dev entries.**
4. **Device Removal: The kernel unregisters devices with device\_unregister(), and udev removes /dev nodes.**