Device links

By default, the driver core only enforces dependencies between devices that are borne out of a parent/child relationship within the device hierarchy: When suspending, resuming or shutting down the system, devices are ordered based on this relationship, i.e. children are always suspended before their parent, and the parent is always resumed before its children.

Sometimes there is a need to represent device dependencies beyond the mere parent/child relationship, e.g. between siblings, and have the driver core automatically take care of them.

Secondly, the driver core by default does not enforce any driver presence dependencies, i.e. that one device must be bound to a driver before another one can probe or function correctly.

Often these two dependency types come together, so a device depends on another one both with regards to driver presence *and* with regards to suspend/resume and shutdown ordering.

Device links allow representation of such dependencies in the driver core.

In its standard or *managed* form, a device link combines *both* dependency types: It guarantees correct suspend/resume and shutdown ordering between a "supplier" device and its "consumer" devices, and it guarantees driver presence on the supplier. The consumer devices are not probed before the supplier is bound to a driver, and they're unbound before the supplier is unbound.

When driver presence on the supplier is irrelevant and only correct suspend/resume and shutdown ordering is needed, the device link may simply be set up with the `DL_FLAG_STATELESS` flag. In other words, enforcing driver presence on the supplier is optional.

Another optional feature is runtime PM integration: By setting the `DL_FLAG_PM_RUNTIME` flag on addition of the device link, the PM core is instructed to runtime resume the supplier and keep it active whenever and for as long as the consumer is runtime resumed.

Usage

The earliest point in time when device links can be added is after :c:func:`device_add()` has been called for the supplier and :c:func:`device_initialize()` has been called for the consumer.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 47); *backlink***
>
> Unknown interpreted text role "c:func".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 47); *backlink***
>
> Unknown interpreted text role "c:func".

It is legal to add them later, but care must be taken that the system remains in a consistent state: E.g. a device link cannot be added in the midst of a suspend/resume transition, so either commencement of such a transition needs to be prevented with :c:func:`lock_system_sleep()`, or the device link needs to be added from a function which is guaranteed not to run in parallel to a suspend/resume transition, such as from a device `->probe` callback or a boot-time PCI quirk.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 51); *backlink***
>
> Unknown interpreted text role "c:func".

Another example for an inconsistent state would be a device link that represents a driver presence dependency, yet is added from the consumer's `->probe` callback while the supplier hasn't started to probe yet: Had the driver core known about the device link earlier, it wouldn't have probed the consumer in the first place. The onus is thus on the consumer to check presence of the supplier after adding the link, and defer probing on non-presence. [Note that it is valid to create a link from the consumer's `->probe` callback while the supplier is still probing, but the consumer must know that the supplier is functional already at the link creation time (that is the case, for instance, if the consumer has just acquired some resources that would not have been available had the supplier not been functional then).]

If a device link with `DL_FLAG_STATELESS` set (i.e. a stateless device link) is added in the `->probe` callback of the supplier or consumer driver, it is typically deleted in its `->remove` callback for symmetry. That way, if the driver is compiled as a module, the device link is added on module load and orderly deleted on unload. The same restrictions that apply to device link addition (e.g. exclusion of a parallel suspend/resume transition) apply equally to deletion. Device links managed by the driver core are deleted automatically by it.

Several flags may be specified on device link addition, two of which have already been mentioned above: `DL_FLAG_STATELESS` to express that no driver presence dependency is needed (but only correct suspend/resume and shutdown ordering) and `DL_FLAG_PM_RUNTIME` to express that runtime PM integration is desired.

Two other flags are specifically targeted at use cases where the device link is added from the consumer's `->probe` callback: `DL_FLAG_RPM_ACTIVE` can be specified to runtime resume the supplier and prevent it from suspending before the consumer is runtime suspended. `DL_FLAG_AUTOREMOVE_CONSUMER` causes the device link to be automatically purged when the consumer fails to probe or later unbinds.

Similarly, when the device link is added from supplier's `->probe` callback, `DL_FLAG_AUTOREMOVE_SUPPLIER` causes the device link to be automatically purged when the supplier fails to probe or later unbinds.

If neither `DL_FLAG_AUTOREMOVE_CONSUMER` nor `DL_FLAG_AUTOREMOVE_SUPPLIER` is set, `DL_FLAG_AUTOPROBE_CONSUMER` can be used to request the driver core to probe for a driver for the consumer driver on the link automatically after a driver has been bound to the supplier device.

Note, however, that any combinations of `DL_FLAG_AUTOREMOVE_CONSUMER`, `DL_FLAG_AUTOREMOVE_SUPPLIER` or `DL_FLAG_AUTOPROBE_CONSUMER` with `DL_FLAG_STATELESS` are invalid and cannot be used.

Limitations

Driver authors should be aware that a driver presence dependency for managed device links (i.e. when `DL_FLAG_STATELESS` is not specified on link addition) may cause probing of the consumer to be deferred indefinitely. This can become a problem if the consumer is required to probe before a certain initcall level is reached. Worse, if the supplier driver is blacklisted or missing, the consumer will never be probed.

Moreover, managed device links cannot be deleted directly. They are deleted by the driver core when they are not necessary any more in accordance with the `DL_FLAG_AUTOREMOVE_CONSUMER` and `DL_FLAG_AUTOREMOVE_SUPPLIER` flags. However, stateless device links (i.e. device links with `DL_FLAG_STATELESS` set) are expected to be removed by whoever called :c:func:`device_link_add()` to add them with the help of either :c:func:`device_link_del()` or :c:func:`device_link_remove()`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 116); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 116); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 116); *backlink*

Unknown interpreted text role "c:func".

Passing `DL_FLAG_RPM_ACTIVE` along with `DL_FLAG_STATELESS` to :c:func:`device_link_add()` may cause the PM-runtime usage counter of the supplier device to remain nonzero after a subsequent invocation of either :c:func:`device_link_del()` or :c:func:`device_link_remove()` to remove the device link returned by it. This happens if :c:func:`device_link_add()` is called twice in a row for the same consumer-supplier pair without removing the link between these calls, in which case allowing the PM-runtime usage counter of the supplier to drop on an attempt to remove the link may cause it to be suspended while the consumer is still PM-runtime-active and that has to be avoided. [To work around this limitation it is sufficient to let the consumer runtime suspend at least once, or call :c:func:`pm_runtime_set_suspended()` for it with PM-runtime disabled, between the :c:func:`device_link_add()` and :c:func:`device_link_del()` or :c:func:`device_link_remove()` calls.]

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 124); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 124); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 124); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 124); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 124); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 124); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 124); *backlink*

Unknown interpreted text role "c:func".

Sometimes drivers depend on optional resources. They are able to operate in a degraded mode (reduced feature set or performance) when those resources are not present. An example is an SPI controller that can use a DMA engine or work in PIO mode. The controller can determine presence of the optional resources at probe time but on non-presence there is no way to know whether they will become available in the near future (due to a supplier driver probing) or never. Consequently it cannot be determined whether to defer probing or not. It would be possible to notify drivers when optional resources become available after probing, but it would come at a high cost for drivers as switching between modes of operation at runtime based on the availability of such resources would be much more complex than a mechanism based on probe deferral. In any case optional resources are beyond the scope of device links.

Examples

- An MMU device exists alongside a busmaster device, both are in the same power domain. The MMU implements DMA address translation for the busmaster device and shall be runtime resumed and kept active whenever and as long as the busmaster device is active. The busmaster device's driver shall not bind before the MMU is bound. To achieve this, a device link with runtime PM integration is added from the busmaster device (consumer) to the MMU device (supplier). The effect with regards to runtime PM is the same as if the MMU was the parent of the master device.

  The fact that both devices share the same power domain would normally suggest usage of a struct dev_pm_domain or struct generic_pm_domain, however these are not independent devices that happen to share a power switch, but rather the MMU device serves the busmaster device and is useless without it. A device link creates a synthetic hierarchical relationship between the devices and is thus more apt.

- A Thunderbolt host controller comprises a number of PCIe hotplug ports and an NHI device to manage the PCIe switch. On resume from system sleep, the NHI device needs to re-establish PCI tunnels to attached devices before the hotplug ports can resume. If the hotplug ports were children of the NHI, this resume order would automatically be enforced by the PM core, but unfortunately they're aunts. The solution is to add device links from the hotplug ports (consumers) to the NHI device (supplier). A driver presence dependency is not necessary for this use case.

- Discrete GPUs in hybrid graphics laptops often feature an HDA controller for HDMI/DP audio. In the device hierarchy the HDA controller is a sibling of the VGA device, yet both share the same power domain and the HDA controller is only ever needed when an HDMI/DP display is attached to the VGA device. A device link from the HDA controller (consumer) to the VGA device (supplier) aptly represents this relationship.
- ACPI allows definition of a device start order by way of _DEP objects. A classical example is when ACPI power management methods on one device are implemented in terms of I^2C accesses and require a specific I^2C controller to be present and functional for the power management of the device in question to work.
- In some SoCs a functional dependency exists from display, video codec and video processing IP cores on transparent memory access IP cores that handle burst access and compression/decompression.

Alternatives

- A struct dev_pm_domain can be used to override the bus, class or device type callbacks. It is intended for devices sharing a single on/off switch, however it does not guarantee a specific suspend/resume ordering, this needs to be implemented separately. It also does not by itself track the runtime PM status of the involved devices and turn off the power switch only when all of them are runtime suspended. Furthermore it cannot be used to enforce a specific shutdown ordering or a driver presence dependency.
- A struct generic_pm_domain is a lot more heavyweight than a device link and does not allow for shutdown ordering or driver presence dependencies. It also cannot be used on ACPI systems.

Implementation

The device hierarchy, which -- as the name implies -- is a tree, becomes a directed acyclic graph once device links are added.

Ordering of these devices during suspend/resume is determined by the dpm_list. During shutdown it is determined by the devices_kset. With no device links present, the two lists are a flattened, one-dimensional representations of the device tree such that a device is placed behind all its ancestors. That is achieved by traversing the ACPI namespace or OpenFirmware device tree top-down and appending devices to the lists as they are discovered.

Once device links are added, the lists need to satisfy the additional constraint that a device is placed behind all its suppliers, recursively. To ensure this, upon addition of the device link the consumer and the entire sub-graph below it (all children and consumers of the consumer) are moved to the end of the list. (Call to :c:func:`device_reorder_to_tail()` from :c:func:`device_link_add()`.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 228); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 228); *backlink*

Unknown interpreted text role "c:func".

To prevent introduction of dependency loops into the graph, it is verified upon device link addition that the supplier is not dependent on the consumer or any children or consumers of the consumer. (Call to :c:func:`device_is_dependent()` from :c:func:`device_link_add()`.) If that constraint is violated, :c:func:`device_link_add()` will return `NULL` and a `WARNING` will be logged.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 235); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 235); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 235); *backlink*

Unknown interpreted text role "c:func".

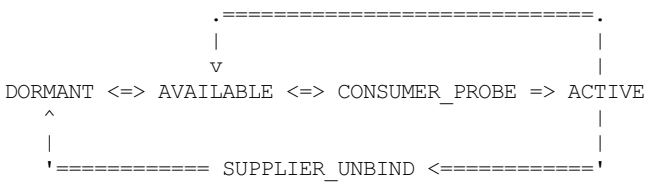
Notably this also prevents the addition of a device link from a parent device to a child. However the converse is allowed, i.e. a device link from a child to a parent. Since the driver core already guarantees correct suspend/resume and shutdown ordering between parent and child, such a device link only makes sense if a driver presence dependency is needed on top of that. In this case driver authors should weigh carefully if a device link is at all the right tool for the purpose. A more suitable approach might be to simply use deferred probing or add a device flag causing the parent driver to be probed before the child one.

State machine

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 256)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/linux/device.h
   :functions: device_link_state
```



- The initial state of a device link is automatically determined by :c:func:`device_link_add()` based on the driver presence on the supplier and consumer. If the link is created before any devices are probed, it is set to `DL_STATE_DORMANT`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 269); *backlink*

  Unknown interpreted text role "c:func".

- When a supplier device is bound to a driver, links to its consumers progress to `DL_STATE_AVAILABLE`. (Call to :c:func:`device_links_driver_bound()` from :c:func:`driver_bound()`.)

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 274); *backlink*

  Unknown interpreted text role "c:func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 274); *backlink*

  Unknown interpreted text role "c:func".

- Before a consumer device is probed, presence of supplier drivers is verified by checking the consumer device is not in the wait_for_suppliers list and by checking that links to suppliers are in `DL_STATE_AVAILABLE` state. The state of the links is updated to `DL_STATE_CONSUMER_PROBE`. (Call to :c:func:`device_links_check_suppliers()` from :c:func:`really_probe()`.) This prevents the supplier from unbinding. (Call to :c:func:`wait_for_device_probe()` from :c:func:`device_links_unbind_consumers()`.)

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 279); *backlink*

  Unknown interpreted text role "c:func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 279); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 279); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 279); *backlink*

Unknown interpreted text role "c:func".

- If the probe fails, links to suppliers revert back to `DL_STATE_AVAILABLE`. (Call to :c:func:`device_links_no_driver()` from :c:func:`really_probe()`.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 289); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 289); *backlink*

Unknown interpreted text role "c:func".

- If the probe succeeds, links to suppliers progress to `DL_STATE_ACTIVE`. (Call to :c:func:`device_links_driver_bound()` from :c:func:`driver_bound()`.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 292); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 292); *backlink*

Unknown interpreted text role "c:func".

- When the consumer's driver is later on removed, links to suppliers revert back to `DL_STATE_AVAILABLE`. (Call to :c:func:`__device_links_no_driver()` from :c:func:`device_links_driver_cleanup()`, which in turn is called from :c:func:`__device_release_driver()`.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 295); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 295); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst, line 295); *backlink*

Unknown interpreted text role "c:func".

- Before a supplier's driver is removed, links to consumers that are not bound to a driver are updated to `DL_STATE_SUPPLIER_UNBIND`. (Call to :c:func:`device_links_busy()` from :c:func:`__device_release_driver()`.) This prevents the consumers from binding. (Call to :c:func:`device_links_check_suppliers()` from :c:func:`really_probe()`.) Consumers that are bound are freed from their driver; consumers that are probing are waited for until they are done. (Call to :c:func:`device_links_unbind_consumers()` from :c:func:`__device_release_driver()`.) Once all links to consumers are in `DL_STATE_SUPPLIER_UNBIND` state, the supplier driver is released and the links revert to `DL_STATE_DORMANT`. (Call to :c:func:`device_links_driver_cleanup()` from :c:func:`__device_release_driver()`.)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]device_link.rst`, line 301); *backlink*

Unknown interpreted text role "c:func".

API

See device_link_add(), device_link_del() and device_link_remove().