Driver Binding

Driver binding is the process of associating a device with a device driver that can control it. Bus drivers have typically handled this because there have been bus-specific structures to represent the devices and the drivers. With generic device and device driver structures, most of the binding can take place using common code.

Bus

The bus type structure contains a list of all devices that are on that bus type in the system. When device_register is called for a device, it is inserted into the end of this list. The bus object also contains a list of all drivers of that bus type. When driver_register is called for a driver, it is inserted at the end of this list. These are the two events which trigger driver binding.

device_register

When a new device is added, the bus's list of drivers is iterated over to find one that supports it. In order to determine that, the device ID of the device must match one of the device IDs that the driver supports. The format and semantics for comparing IDs is bus-specific. Instead of trying to derive a complex state machine and matching algorithm, it is up to the bus driver to provide a callback to compare a device against the IDs of a driver. The bus returns 1 if a match was found; 0 otherwise.

int match(struct device * dev, struct device_driver * drv);

If a match is found, the device's driver field is set to the driver and the driver's probe callback is called. This gives the driver a chance to verify that it really does support the hardware, and that it's in a working state.

Device Class

Upon the successful completion of probe, the device is registered with the class to which it belongs. Device drivers belong to one and only one class, and that is set in the driver's devclass field. devclass_add_device is called to enumerate the device within the class and actually register it with the class, which happens with the class's register_dev callback.

Driver

When a driver is attached to a device, the device is inserted into the driver's list of devices.

sysfs

A symlink is created in the bus's 'devices' directory that points to the device's directory in the physical hierarchy.

A symlink is created in the driver's 'devices' directory that points to the device's directory in the physical hierarchy.

A directory for the device is created in the class's directory. A symlink is created in that directory that points to the device's physical location in the sysfs tree.

A symlink can be created (though this isn't done yet) in the device's physical directory to either its class directory, or the class's top-level directory. One can also be created to point to its driver's directory also.

driver_register

The process is almost identical for when a new driver is added. The bus's list of devices is iterated over to find a match. Devices that already have a driver are skipped. All the devices are iterated over, to bind as many devices as possible to the driver.

Removal

When a device is removed, the reference count for it will eventually go to 0. When it does, the remove callback of the driver is called. It is removed from the driver's list of devices and the reference count of the driver is decremented. All symlinks between the two are removed.

When a driver is removed, the list of devices that it supports is iterated over, and the driver's remove callback is called for each one. The device is removed from that list and the symlinks removed.