

[PATCH 196/196] Driver core: coding style fixes

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-01/msg10231.html>

- *From:* Greg Kroah-Hartman <gregkh@xxxxxxx>
 - *Date:* Thu, 24 Jan 2008 23:33:45 -0800
-

Fix up a number of coding style issues in the drivers/base/ directory that have annoyed me over the years. checkpatch.pl is now very happy.

Signed-off-by: Greg Kroah-Hartman <gregkh@xxxxxxx>

```
---
drivers/base/base.h | 14 +-
drivers/base/bus.c | 290 ++++++-----
drivers/base/class.c | 140 ++++++-----
drivers/base/core.c | 203 ++++++-----
drivers/base/dd.c | 119 ++++++-----
drivers/base/driver.c | 120 ++++++-----
drivers/base/init.c | 9 +-
drivers/base/platform.c | 233 ++++++-----
8 files changed, 545 insertions(+), 583 deletions(-)
```

```
diff --git a/drivers/base/base.h b/drivers/base/base.h
index f7ad65a..c044414 100644
--- a/drivers/base/base.h
+++ b/drivers/base/base.h
@@ -50,15 +50,15 @@ extern int platform_bus_init(void);
extern int system_bus_init(void);
extern int cpu_dev_init(void);

-extern int bus_add_device(struct device * dev);
-extern void bus_attach_device(struct device * dev);
-extern void bus_remove_device(struct device * dev);
+extern int bus_add_device(struct device *dev);
+extern void bus_attach_device(struct device *dev);
+extern void bus_remove_device(struct device *dev);

-extern int bus_add_driver(struct device_driver *);
-extern void bus_remove_driver(struct device_driver *);
+extern int bus_add_driver(struct device_driver *drv);
+extern void bus_remove_driver(struct device_driver *drv);

-extern void driver_detach(struct device_driver * drv);
-extern int driver_probe_device(struct device_driver *, struct device *);
+extern void driver_detach(struct device_driver *drv);
+extern int driver_probe_device(struct device_driver *drv, struct device *dev);
```

[PATCH 196/196] Driver core: coding style fixes

```
extern void sysdev_shutdown(void);
extern int sysdev_suspend(pm_message_t state);
diff --git a/drivers/base/bus.c b/drivers/base/bus.c
index a377b65..f484495 100644
--- a/drivers/base/bus.c
+++ b/drivers/base/bus.c
@@ -46,10 +46,10 @@ static void bus_put(struct bus_type *bus)
kset_put(&bus->p->subsys);
}

-static ssize_t
-drv_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
+static ssize_t drv_attr_show(struct kobject *kobj, struct attribute *attr,
+ char *buf)
{
- struct driver_attribute * drv_attr = to_drv_attr(attr);
+ struct driver_attribute *drv_attr = to_drv_attr(attr);
struct driver_private *drv_priv = to_driver(kobj);
ssize_t ret = -EIO;

@@ -58,11 +58,10 @@ drv_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
return ret;
}

-static ssize_t
-drv_attr_store(struct kobject * kobj, struct attribute * attr,
- const char * buf, size_t count)
+static ssize_t drv_attr_store(struct kobject *kobj, struct attribute *attr,
+ const char *buf, size_t count)
{
- struct driver_attribute * drv_attr = to_drv_attr(attr);
+ struct driver_attribute *drv_attr = to_drv_attr(attr);
struct driver_private *drv_priv = to_driver(kobj);
ssize_t ret = -EIO;

@@ -89,16 +88,13 @@ static struct kobj_type driver_ktype = {
.release = driver_release,
};

-
-/*
- * sysfs bindings for buses
- */
-
-static ssize_t
-bus_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
+static ssize_t bus_attr_show(struct kobject *kobj, struct attribute *attr,
+ char *buf)
{
```

[PATCH 196/196] Driver core: coding style fixes

```
- struct bus_attribute * bus_attr = to_bus_attr(attr);
+ struct bus_attribute *bus_attr = to_bus_attr(attr);
struct bus_type_private *bus_priv = to_bus(kobj);
ssize_t ret = 0;

@@ -107,11 +103,10 @@ bus_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
return ret;
}

-static ssize_t
-bus_attr_store(struct kobject * kobj, struct attribute * attr,
- const char * buf, size_t count)
+static ssize_t bus_attr_store(struct kobject *kobj, struct attribute *attr,
+ const char *buf, size_t count)
{
- struct bus_attribute * bus_attr = to_bus_attr(attr);
+ struct bus_attribute *bus_attr = to_bus_attr(attr);
struct bus_type_private *bus_priv = to_bus(kobj);
ssize_t ret = 0;

@@ -125,7 +120,7 @@ static struct sysfs_ops bus_sysfs_ops = {
.store = bus_attr_store,
};

-int bus_create_file(struct bus_type * bus, struct bus_attribute * attr)
+int bus_create_file(struct bus_type *bus, struct bus_attribute *attr)
{
int error;
if (bus_get(bus)) {
@@ -135,14 +130,16 @@ int bus_create_file(struct bus_type * bus, struct bus_attribute * attr)
error = -EINVAL;
return error;
}
+EXPORT_SYMBOL_GPL(bus_create_file);

-void bus_remove_file(struct bus_type * bus, struct bus_attribute * attr)
+void bus_remove_file(struct bus_type *bus, struct bus_attribute *attr)
{
if (bus_get(bus)) {
sysfs_remove_file(&bus->p->subsys.kobj, &attr->attr);
bus_put(bus);
}
}
+EXPORT_SYMBOL_GPL(bus_remove_file);

static struct kobj_type bus_ktype = {
.sysfs_ops = &bus_sysfs_ops,
@@ -219,10 +216,13 @@ static ssize_t driver_bind(struct device_driver *drv,
if (dev->parent)
up(&dev->parent->sem);
```

[PATCH 196/196] Driver core: coding style fixes

```
- if (err > 0) /* success */
+ if (err > 0) {
+ /* success */
err = count;
- else if (err == 0) /* driver didn't accept device */
+ } else if (err == 0) {
+ /* driver didn't accept device */
err = -ENODEV;
+ }
}
put_device(dev);
bus_put(bus);
@@ -259,37 +259,36 @@ static ssize_t store_drivers_probe(struct bus_type *bus,
}
#endif
```

```
-static struct device * next_device(struct klist_iter * i)
+static struct device *next_device(struct klist_iter *i)
{
- struct klist_node * n = klist_next(i);
+ struct klist_node *n = klist_next(i);
return n ? container_of(n, struct device, knode_bus) : NULL;
}
```

/**

```
- * bus_for_each_dev - device iterator.
- * @bus: bus type.
- * @start: device to start iterating from.
- * @data: data for the callback.
- * @fn: function to be called for each device.
+ * bus_for_each_dev - device iterator.
+ * @bus: bus type.
+ * @start: device to start iterating from.
+ * @data: data for the callback.
+ * @fn: function to be called for each device.
*
- * Iterate over @bus's list of devices, and call @fn for each,
- * passing it @data. If @start is not NULL, we use that device to
- * begin iterating from.
+ * Iterate over @bus's list of devices, and call @fn for each,
+ * passing it @data. If @start is not NULL, we use that device to
+ * begin iterating from.
*
- * We check the return of @fn each time. If it returns anything
- * other than 0, we break out and return that value.
+ * We check the return of @fn each time. If it returns anything
+ * other than 0, we break out and return that value.
*
- * NOTE: The device that returns a non-zero value is not retained
- * in any way, nor is its refcount incremented. If the caller needs
- * to retain this data, it should do, and increment the reference
```

[PATCH 196/196] Driver core: coding style fixes

```
- * count in the supplied callback.
+ * NOTE: The device that returns a non-zero value is not retained
+ * in any way, nor is its refcount incremented. If the caller needs
+ * to retain this data, it should do, and increment the reference
+ * count in the supplied callback.
*/
-
-int bus_for_each_dev(struct bus_type * bus, struct device * start,
- void * data, int (*fn)(struct device *, void *))
+int bus_for_each_dev(struct bus_type *bus, struct device *start,
+ void *data, int (*fn)(struct device *, void *))
{
struct klist_iter i;
- struct device * dev;
+ struct device *dev;
int error = 0;

if (!bus)
@@ -302,6 +301,7 @@ int bus_for_each_dev(struct bus_type * bus, struct device * start,
klist_iter_exit(&i);
return error;
}
+EXPORT_SYMBOL_GPL(bus_for_each_dev);

/**
 * bus_find_device – device iterator for locating a particular device.
@@ -318,9 +318,9 @@ int bus_for_each_dev(struct bus_type * bus, struct device * start,
 * if it does. If the callback returns non-zero, this function will
 * return to the caller and not iterate over any more devices.
*/
-struct device * bus_find_device(struct bus_type *bus,
- struct device *start, void *data,
- int (*match)(struct device *, void *))
+struct device *bus_find_device(struct bus_type *bus,
+ struct device *start, void *data,
+ int (*match)(struct device *dev, void *data))
{
struct klist_iter i;
struct device *dev;
@@ -336,11 +336,11 @@ struct device * bus_find_device(struct bus_type *bus,
klist_iter_exit(&i);
return dev;
}
+EXPORT_SYMBOL_GPL(bus_find_device);

-
-static struct device_driver * next_driver(struct klist_iter * i)
+static struct device_driver *next_driver(struct klist_iter *i)
{
- struct klist_node * n = klist_next(i);
+ struct klist_node *n = klist_next(i);
```

[PATCH 196/196] Driver core: coding style fixes

```
struct driver_private *drv_priv;

if (n) {
@@ -351,30 +351,29 @@ static struct device_driver * next_driver(struct klist_iter * i)
}

/**
- * bus_for_each_drv - driver iterator
- * @bus: bus we're dealing with.
- * @start: driver to start iterating on.
- * @data: data to pass to the callback.
- * @fn: function to call for each driver.
+ * bus_for_each_drv - driver iterator
+ * @bus: bus we're dealing with.
+ * @start: driver to start iterating on.
+ * @data: data to pass to the callback.
+ * @fn: function to call for each driver.
*
- * This is nearly identical to the device iterator above.
- * We iterate over each driver that belongs to @bus, and call
- * @fn for each. If @fn returns anything but 0, we break out
- * and return it. If @start is not NULL, we use it as the head
- * of the list.
+ * This is nearly identical to the device iterator above.
+ * We iterate over each driver that belongs to @bus, and call
+ * @fn for each. If @fn returns anything but 0, we break out
+ * and return it. If @start is not NULL, we use it as the head
+ * of the list.
*
- * NOTE: we don't return the driver that returns a non-zero
- * value, nor do we leave the reference count incremented for that
- * driver. If the caller needs to know that info, it must set it
- * in the callback. It must also be sure to increment the refcount
- * so it doesn't disappear before returning to the caller.
+ * NOTE: we don't return the driver that returns a non-zero
+ * value, nor do we leave the reference count incremented for that
+ * driver. If the caller needs to know that info, it must set it
+ * in the callback. It must also be sure to increment the refcount
+ * so it doesn't disappear before returning to the caller.
*/
-
-int bus_for_each_drv(struct bus_type * bus, struct device_driver * start,
- void * data, int (*fn)(struct device_driver *, void *))
+int bus_for_each_drv(struct bus_type *bus, struct device_driver *start,
+ void *data, int (*fn)(struct device_driver *, void *))
{
struct klist_iter i;
- struct device_driver * drv;
+ struct device_driver *drv;
int error = 0;
```

[PATCH 196/196] Driver core: coding style fixes

```
if (!bus)
@@ -387,6 +386,7 @@ int bus_for_each_drv(struct bus_type * bus, struct device_driver * start,
klist_iter_exit(&i);
return error;
}
+EXPORT_SYMBOL_GPL(bus_for_each_drv);

static int device_add_attrs(struct bus_type *bus, struct device *dev)
{
@@ -397,7 +397,7 @@ static int device_add_attrs(struct bus_type *bus, struct device *dev)
return 0;

for (i = 0; attr_name(bus->dev_attrs[i]); i++) {
- error = device_create_file(dev, &bus->dev_attrs[i]);
+ error = device_create_file(dev, &bus->dev_attrs[i]);
if (error) {
while (--i >= 0)
device_remove_file(dev, &bus->dev_attrs[i]);
@@ -407,13 +407,13 @@ static int device_add_attrs(struct bus_type *bus, struct device *dev)
return error;
}

-static void device_remove_attrs(struct bus_type * bus, struct device * dev)
+static void device_remove_attrs(struct bus_type *bus, struct device *dev)
{
int i;

if (bus->dev_attrs) {
for (i = 0; attr_name(bus->dev_attrs[i]); i++)
- device_remove_file(dev, &bus->dev_attrs[i]);
+ device_remove_file(dev, &bus->dev_attrs[i]);
}
}

@@ -434,15 +434,15 @@ static inline void remove_deprecated_bus_links(struct device *dev) { }
#endif

/**
- * bus_add_device - add device to bus
- * @dev: device being added
+ * bus_add_device - add device to bus
+ * @dev: device being added
*
- * - Add the device to its bus's list of devices.
- * - Create link to device's bus.
+ * - Add the device to its bus's list of devices.
+ * - Create link to device's bus.
*/
-int bus_add_device(struct device * dev)
+int bus_add_device(struct device *dev)
{
```

[PATCH 196/196] Driver core: coding style fixes

```
- struct bus_type * bus = bus_get(dev->bus);
+ struct bus_type *bus = bus_get(dev->bus);
int error = 0;

if (bus) {
@@ -476,13 +476,13 @@ out_put:
}

/**
- * bus_attach_device - add device to bus
- * @dev: device tried to attach to a driver
+ * bus_attach_device - add device to bus
+ * @dev: device tried to attach to a driver
*
- * - Add device to bus's list of devices.
- * - Try to attach to driver.
+ * - Add device to bus's list of devices.
+ * - Try to attach to driver.
*/
-void bus_attach_device(struct device * dev)
+void bus_attach_device(struct device *dev)
{
struct bus_type *bus = dev->bus;
int ret = 0;
@@ -500,32 +500,34 @@ void bus_attach_device(struct device * dev)
}

/**
- * bus_remove_device - remove device from bus
- * @dev: device to be removed
+ * bus_remove_device - remove device from bus
+ * @dev: device to be removed
*
- * - Remove symlink from bus's directory.
- * - Delete device from bus's list.
- * - Detach from its driver.
- * - Drop reference taken in bus_add_device().
+ * - Remove symlink from bus's directory.
+ * - Delete device from bus's list.
+ * - Detach from its driver.
+ * - Drop reference taken in bus_add_device().
*/
-void bus_remove_device(struct device * dev)
+void bus_remove_device(struct device *dev)
{
if (dev->bus) {
sysfs_remove_link(&dev->kobj, "subsystem");
remove_deprecated_bus_links(dev);
- sysfs_remove_link(&dev->bus->p->devices_kset->kobj, dev->bus_id);
+ sysfs_remove_link(&dev->bus->p->devices_kset->kobj,
+ dev->bus_id);
```

[PATCH 196/196] Driver core: coding style fixes

```
device_remove_attrs(dev->bus, dev);
if (dev->is_registered) {
dev->is_registered = 0;
klist_del(&dev->knode_bus);
}
- pr_debug("bus: '%s': remove device %s\n", dev->bus->name, dev->bus_id);
+ pr_debug("bus: '%s': remove device %s\n",
+ dev->bus->name, dev->bus_id);
device_release_driver(dev);
bus_put(dev->bus);
}
}

-static int driver_add_attrs(struct bus_type * bus, struct device_driver * drv)
+static int driver_add_attrs(struct bus_type *bus, struct device_driver *drv)
{
int error = 0;
int i;
@@ -534,19 +536,19 @@ static int driver_add_attrs(struct bus_type * bus, struct device_driver * drv)
for (i = 0; attr_name(bus->drv_attrs[i]); i++) {
error = driver_create_file(drv, &bus->drv_attrs[i]);
if (error)
- goto Err;
+ goto err;
}
}
- Done:
+done:
return error;
- Err:
+err:
while (--i >= 0)
driver_remove_file(drv, &bus->drv_attrs[i]);
- goto Done;
+ goto done;
}

-
-static void driver_remove_attrs(struct bus_type * bus, struct device_driver * drv)
+static void driver_remove_attrs(struct bus_type *bus,
+ struct device_driver *drv)
{
int i;

@@ -623,9 +625,8 @@ static ssize_t driver_uevent_store(struct device_driver *drv,
static DRIVER_ATTR(uevent, S_IWUSR, NULL, driver_uevent_store);

/**
- * bus_add_driver - Add a driver to the bus.
- * @drv: driver.
- */
```

[PATCH 196/196] Driver core: coding style fixes

```
+ * bus_add_driver – Add a driver to the bus.
+ * @drv: driver.
*/
int bus_add_driver(struct device_driver *drv)
{
@@ -688,15 +689,14 @@ out_put_bus:
}

/**
- * bus_remove_driver – delete driver from bus's knowledge.
- * @drv: driver.
+ * bus_remove_driver – delete driver from bus's knowledge.
+ * @drv: driver.
*
- * Detach the driver from the devices it controls, and remove
- * it from its bus's list of drivers. Finally, we drop the reference
- * to the bus we took in bus_add_driver().
+ * Detach the driver from the devices it controls, and remove
+ * it from its bus's list of drivers. Finally, we drop the reference
+ * to the bus we took in bus_add_driver().
*/
-
-void bus_remove_driver(struct device_driver * drv)
+void bus_remove_driver(struct device_driver *drv)
{
if (!drv->bus)
return;
@@ -712,10 +712,9 @@ void bus_remove_driver(struct device_driver * drv)
bus_put(drv->bus);
}

-
/* Helper for bus_rescan_devices's iter */
static int __must_check bus_rescan_devices_helper(struct device *dev,
- void *data)
+ void *data)
{
int ret = 0;

@@ -737,10 +736,11 @@ static int __must_check bus_rescan_devices_helper(struct device *dev,
* attached and rescan it against existing drivers to see if it matches
* any by calling device_attach() for the unbound devices.
*/
-int bus_rescan_devices(struct bus_type * bus)
+int bus_rescan_devices(struct bus_type *bus)
{
return bus_for_each_dev(bus, NULL, NULL, bus_rescan_devices_helper);
}
+EXPORT_SYMBOL_GPL(bus_rescan_devices);

/**
```

[PATCH 196/196] Driver core: coding style fixes

* device_reprobe – remove driver for a device and probe for a new driver

@@ -765,55 +765,55 @@ int device_reprobe(struct device *dev)

EXPORT_SYMBOL_GPL(device_reprobe);

/**

– * find_bus – locate bus by name.

– * @name: name of bus.

+ * find_bus – locate bus by name.

+ * @name: name of bus.

*

– * Call kset_find_obj() to iterate over list of buses to

– * find a bus by name. Return bus if found.

+ * Call kset_find_obj() to iterate over list of buses to

+ * find a bus by name. Return bus if found.

*

– * Note that kset_find_obj increments bus' reference count.

+ * Note that kset_find_obj increments bus' reference count.

*/

#if 0

–struct bus_type * find_bus(char * name)

+struct bus_type *find_bus(char *name)

{

– struct kobject * k = kset_find_obj(bus_kset, name);

+ struct kobject *k = kset_find_obj(bus_kset, name);

return k ? to_bus(k) : NULL;

}

#endif /* 0 */

/**

– * bus_add_attrs – Add default attributes for this bus.

– * @bus: Bus that has just been registered.

+ * bus_add_attrs – Add default attributes for this bus.

+ * @bus: Bus that has just been registered.

*/

–static int bus_add_attrs(struct bus_type * bus)

+static int bus_add_attrs(struct bus_type *bus)

{

int error = 0;

int i;

if (bus->bus_attrs) {

for (i = 0; attr_name(bus->bus_attrs[i]); i++) {

– error = bus_create_file(bus, &bus->bus_attrs[i]);

+ error = bus_create_file(bus, &bus->bus_attrs[i]);

if (error)

– goto Err;

+ goto err;

}

}

[PATCH 196/196] Driver core: coding style fixes

```
- Done:
+done:
return error;
- Err:
+err:
while (--i >= 0)
- bus_remove_file(bus,&bus->bus_attrs[i]);
- goto Done;
+ bus_remove_file(bus, &bus->bus_attrs[i]);
+ goto done;
}

-static void bus_remove_attrs(struct bus_type * bus)
+static void bus_remove_attrs(struct bus_type *bus)
{
int i;

if (bus->bus_attrs) {
for (i = 0; attr_name(bus->bus_attrs[i]); i++)
- bus_remove_file(bus,&bus->bus_attrs[i]);
+ bus_remove_file(bus, &bus->bus_attrs[i]);
}
}

@@ -843,14 +843,14 @@ static ssize_t bus_uevent_store(struct bus_type *bus,
static BUS_ATTR(uevent, S_IWUSR, NULL, bus_uevent_store);

/**
- * bus_register - register a bus with the system.
- * @bus: bus.
+ * bus_register - register a bus with the system.
+ * @bus: bus.
*
- * Once we have that, we registered the bus with the kobject
- * infrastructure, then register the children subsystems it has:
- * the devices and drivers that belong to the bus.
+ * Once we have that, we registered the bus with the kobject
+ * infrastructure, then register the children subsystems it has:
+ * the devices and drivers that belong to the bus.
*/
-int bus_register(struct bus_type * bus)
+int bus_register(struct bus_type *bus)
{
int retval;
struct bus_type_private *priv;
@@ -922,15 +922,16 @@ bus_uevent_fail:
out:
return retval;
}
+EXPORT_SYMBOL_GPL(bus_register);
```

[PATCH 196/196] Driver core: coding style fixes

```
/**
- * bus_unregister – remove a bus from the system
- * @bus: bus.
+ * bus_unregister – remove a bus from the system
+ * @bus: bus.
*
- * Unregister the child subsystems and the bus itself.
- * Finally, we call bus_put() to release the refcount
+ * Unregister the child subsystems and the bus itself.
+ * Finally, we call bus_put() to release the refcount
*/
-void bus_unregister(struct bus_type * bus)
+void bus_unregister(struct bus_type *bus)
{
pr_debug("bus: '%s': unregistering\n", bus->name);
bus_remove_attrs(bus);
@@ -941,6 +942,7 @@ void bus_unregister(struct bus_type * bus)
kset_unregister(&bus->p->subsys);
kfree(bus->p);
}
+EXPORT_SYMBOL_GPL(bus_unregister);

int bus_register_notifier(struct bus_type *bus, struct notifier_block *nb)
{
@@ -973,15 +975,3 @@ int __init buses_init(void)
return -ENOMEM;
return 0;
}
-
-
-EXPORT_SYMBOL_GPL(bus_for_each_dev);
-EXPORT_SYMBOL_GPL(bus_find_device);
-EXPORT_SYMBOL_GPL(bus_for_each_drv);
-
-EXPORT_SYMBOL_GPL(bus_register);
-EXPORT_SYMBOL_GPL(bus_unregister);
-EXPORT_SYMBOL_GPL(bus_rescan_devices);
-
-EXPORT_SYMBOL_GPL(bus_create_file);
-EXPORT_SYMBOL_GPL(bus_remove_file);
diff --git a/drivers/base/class.c b/drivers/base/class.c
index 9f737ff..59cf358 100644
--- a/drivers/base/class.c
+++ b/drivers/base/class.c
@@ -23,11 +23,11 @@
#define to_class_attr(_attr) container_of(_attr, struct class_attribute, attr)
#define to_class(obj) container_of(obj, struct class, subsys.kobj)

-static ssize_t
-class_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
+static ssize_t class_attr_show(struct kobject *kobj, struct attribute *attr,
```

[PATCH 196/196] Driver core: coding style fixes

```
+ char *buf)
{
- struct class_attribute * class_attr = to_class_attr(attr);
- struct class * dc = to_class(kobj);
+ struct class_attribute *class_attr = to_class_attr(attr);
+ struct class *dc = to_class(kobj);
ssize_t ret = -EIO;

if (class_attr->show)
@@ -35,12 +35,11 @@ class_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
return ret;
}

-static ssize_t
-class_attr_store(struct kobject * kobj, struct attribute * attr,
- const char * buf, size_t count)
+static ssize_t class_attr_store(struct kobject *kobj, struct attribute *attr,
+ const char *buf, size_t count)
{
- struct class_attribute * class_attr = to_class_attr(attr);
- struct class * dc = to_class(kobj);
+ struct class_attribute *class_attr = to_class_attr(attr);
+ struct class *dc = to_class(kobj);
ssize_t ret = -EIO;

if (class_attr->store)
@@ -48,7 +47,7 @@ class_attr_store(struct kobject * kobj, struct attribute * attr,
return ret;
}

-static void class_release(struct kobject * kobj)
+static void class_release(struct kobject *kobj)
{
struct class *class = to_class(kobj);

@@ -75,17 +74,17 @@ static struct kobj_type class_ktype = {
static struct kset *class_kset;

-int class_create_file(struct class * cls, const struct class_attribute * attr)
+int class_create_file(struct class *cls, const struct class_attribute *attr)
{
int error;
- if (cls) {
+ if (cls)
error = sysfs_create_file(&cls->subsys.kobj, &attr->attr);
- } else
+ else
error = -EINVAL;
return error;
}
```

[PATCH 196/196] Driver core: coding style fixes

```
-void class_remove_file(struct class * cls, const struct class_attribute * attr)
+void class_remove_file(struct class *cls, const struct class_attribute *attr)
{
if (cls)
sysfs_remove_file(&cls->subsys.kobj, &attr->attr);
@@ -94,48 +93,48 @@ void class_remove_file(struct class * cls, const struct class_attribute * attr)
static struct class *class_get(struct class *cls)
{
if (cls)
- return container_of(kset_get(&cls->subsys), struct class, subsys);
+ return container_of(kset_get(&cls->subsys),
+ struct class, subsys);
return NULL;
}

-static void class_put(struct class * cls)
+static void class_put(struct class *cls)
{
if (cls)
kset_put(&cls->subsys);
}

-
-static int add_class_attrs(struct class * cls)
+static int add_class_attrs(struct class *cls)
{
int i;
int error = 0;

if (cls->class_attrs) {
for (i = 0; attr_name(cls->class_attrs[i]); i++) {
- error = class_create_file(cls,&cls->class_attrs[i]);
+ error = class_create_file(cls, &cls->class_attrs[i]);
if (error)
- goto Err;
+ goto error;
}
}
- Done:
+done:
return error;
- Err:
+error:
while (--i >= 0)
- class_remove_file(cls,&cls->class_attrs[i]);
- goto Done;
+ class_remove_file(cls, &cls->class_attrs[i]);
+ goto done;
}
}
```

[PATCH 196/196] Driver core: coding style fixes

```
-static void remove_class_attrs(struct class * cls)
+static void remove_class_attrs(struct class *cls)
{
int i;

if (cls->class_attrs) {
for (i = 0; attr_name(cls->class_attrs[i]); i++)
- class_remove_file(cls,&cls->class_attrs[i]);
+ class_remove_file(cls, &cls->class_attrs[i]);
}
}

-int class_register(struct class * cls)
+int class_register(struct class *cls)
{
int error;

@@ -167,7 +166,7 @@ int class_register(struct class * cls)
return error;
}

-void class_unregister(struct class * cls)
+void class_unregister(struct class *cls)
{
pr_debug("device class '%s': unregistering\n", cls->name);
remove_class_attrs(cls);
@@ -249,8 +248,8 @@ void class_destroy(struct class *cls)

/* Class Device Stuff */

-int class_device_create_file(struct class_device * class_dev,
- const struct class_device_attribute * attr)
+int class_device_create_file(struct class_device *class_dev,
+ const struct class_device_attribute *attr)
{
int error = -EINVAL;
if (class_dev)
@@ -258,8 +257,8 @@ int class_device_create_file(struct class_device * class_dev,
return error;
}

-void class_device_remove_file(struct class_device * class_dev,
- const struct class_device_attribute * attr)
+void class_device_remove_file(struct class_device *class_dev,
+ const struct class_device_attribute *attr)
{
if (class_dev)
sysfs_remove_file(&class_dev->kobj, &attr->attr);
@@ -281,12 +280,11 @@ void class_device_remove_bin_file(struct class_device *class_dev,
sysfs_remove_bin_file(&class_dev->kobj, attr);
}
}
```

[PATCH 196/196] Driver core: coding style fixes

```
-static ssize_t
-class_device_attr_show(struct kobject * kobj, struct attribute * attr,
- char * buf)
+static ssize_t class_device_attr_show(struct kobject *kobj,
+ struct attribute *attr, char *buf)
{
- struct class_device_attribute * class_dev_attr = to_class_dev_attr(attr);
- struct class_device * cd = to_class_dev(kobj);
+ struct class_device_attribute *class_dev_attr = to_class_dev_attr(attr);
+ struct class_device *cd = to_class_dev(kobj);
ssize_t ret = 0;

if (class_dev_attr->show)
@@ -294,12 +292,12 @@ class_device_attr_show(struct kobject * kobj, struct attribute * attr,
return ret;
}

-static ssize_t
-class_device_attr_store(struct kobject * kobj, struct attribute * attr,
- const char * buf, size_t count)
+static ssize_t class_device_attr_store(struct kobject *kobj,
+ struct attribute *attr,
+ const char *buf, size_t count)
{
- struct class_device_attribute * class_dev_attr = to_class_dev_attr(attr);
- struct class_device * cd = to_class_dev(kobj);
+ struct class_device_attribute *class_dev_attr = to_class_dev_attr(attr);
+ struct class_device *cd = to_class_dev(kobj);
ssize_t ret = 0;

if (class_dev_attr->store)
@@ -312,10 +310,10 @@ static struct sysfs_ops class_dev_sysfs_ops = {
.store = class_device_attr_store,
};

-static void class_dev_release(struct kobject * kobj)
+static void class_dev_release(struct kobject *kobj)
{
struct class_device *cd = to_class_dev(kobj);
- struct class * cls = cd->class;
+ struct class *cls = cd->class;

pr_debug("device class '%s': release.\n", cd->class_id);

@@ -324,8 +322,8 @@ static void class_dev_release(struct kobject * kobj)
else if (cls->release)
cls->release(cd);
else {
- printk(KERN_ERR "Class Device '%s' does not have a release() function, "
- "it is broken and must be fixed.\n",
```

[PATCH 196/196] Driver core: coding style fixes

```
+ printk(KERN_ERR "Class Device '%s' does not have a release() "  
+ "function, it is broken and must be fixed.\n",  
cd->class_id);  
WARN_ON(1);  
}  
@@ -436,7 +434,8 @@ static int class_uevent(struct kset *kset, struct kobject *kobj,  
add_uevent_var(env, "PHYSDEVBUS=%s", dev->bus->name);  
  
if (dev->driver)  
- add_uevent_var(env, "PHYSDEVDRIVER=%s", dev->driver->name);  
+ add_uevent_var(env, "PHYSDEVDRIVER=%s",  
+ dev->driver->name);  
}  
  
if (class_dev->uevent) {  
@@ -469,40 +468,40 @@ static struct kset class_obj_subsys = {  
.uevent_ops = &class_uevent_ops,  
};  
  
-static int class_device_add_attrs(struct class_device * cd)  
+static int class_device_add_attrs(struct class_device *cd)  
{  
int i;  
int error = 0;  
- struct class * cls = cd->class;  
+ struct class *cls = cd->class;  
  
if (cls->class_dev_attrs) {  
for (i = 0; attr_name(cls->class_dev_attrs[i]); i++) {  
error = class_device_create_file(cd,  
- &cls->class_dev_attrs[i]);  
+ &cls->class_dev_attrs[i]);  
if (error)  
- goto Err;  
+ goto err;  
}  
}  
- Done:  
+done:  
return error;  
- Err:  
+err:  
while (--i >= 0)  
- class_device_remove_file(cd,&cls->class_dev_attrs[i]);  
- goto Done;  
+ class_device_remove_file(cd, &cls->class_dev_attrs[i]);  
+ goto done;  
}  
  
-static void class_device_remove_attrs(struct class_device * cd)  
+static void class_device_remove_attrs(struct class_device *cd)
```

[PATCH 196/196] Driver core: coding style fixes

```
{
int i;
- struct class * cls = cd->class;
+ struct class *cls = cd->class;

if (cls->class_dev_attrs) {
for (i = 0; attr_name(cls->class_dev_attrs[i]); i++)
- class_device_remove_file(cd,&cls->class_dev_attrs[i]);
+ class_device_remove_file(cd, &cls->class_dev_attrs[i]);
}
}

-static int class_device_add_groups(struct class_device * cd)
+static int class_device_add_groups(struct class_device *cd)
{
int i;
int error = 0;
@@ -512,7 +511,8 @@ static int class_device_add_groups(struct class_device * cd)
error = sysfs_create_group(&cd->kobj, cd->groups[i]);
if (error) {
while (--i >= 0)
- sysfs_remove_group(&cd->kobj, cd->groups[i]);
+ sysfs_remove_group(&cd->kobj,
+ cd->groups[i]);
goto out;
}
}
@@ -521,14 +521,12 @@ out:
return error;
}

-static void class_device_remove_groups(struct class_device * cd)
+static void class_device_remove_groups(struct class_device *cd)
{
int i;
- if (cd->groups) {
- for (i = 0; cd->groups[i]; i++) {
+ if (cd->groups)
+ for (i = 0; cd->groups[i]; i++)
sysfs_remove_group(&cd->kobj, cd->groups[i]);
- }
- }
}

static ssize_t show_dev(struct class_device *class_dev, char *buf)
@@ -653,7 +651,7 @@ int class_device_add(struct class_device *class_dev)
out3:
kobject_del(&class_dev->kobj);
out2:
- if(parent_class_dev)
+ if (parent_class_dev)
```

[PATCH 196/196] Driver core: coding style fixes

```
class_device_put(parent_class_dev);
class_put(parent_class);
out1:
@@ -670,9 +668,11 @@ int class_device_register(struct class_device *class_dev)
/**
 * class_device_create – creates a class device and registers it with sysfs
 * @cls: pointer to the struct class that this device should be registered to.
 * @parent: pointer to the parent struct class_device of this new device, if any.
 * @parent: pointer to the parent struct class_device of this new device, if
 * any.
 * @devt: the dev_t for the char device to be added.
 * @device: a pointer to a struct device that is associated with this class device.
 * @device: a pointer to a struct device that is associated with this class
 * device.
 * @fmt: string for the class device's name
 *
 * This function can be used by char device classes. A struct
@@ -796,7 +796,7 @@ void class_device_destroy(struct class *cls, dev_t devt)
class_device_unregister(class_dev);
}

-struct class_device * class_device_get(struct class_device *class_dev)
+struct class_device *class_device_get(struct class_device *class_dev)
{
if (class_dev)
return to_class_dev(kobject_get(&class_dev->kobj));
@@ -973,7 +973,7 @@ int class_interface_register(struct class_interface *class_intf)

void class_interface_unregister(struct class_interface *class_intf)
{
- struct class * parent = class_intf->class;
+ struct class *parent = class_intf->class;
struct class_device *class_dev;
struct device *dev;

diff --git a/drivers/base/core.c b/drivers/base/core.c
index f09dde3..edf3bbe 100644
--- a/drivers/base/core.c
+++ b/drivers/base/core.c
@@ -24,8 +24,8 @@
#include "base.h"
#include "power/power.h"

-int (*platform_notify)(struct device * dev) = NULL;
-int (*platform_notify_remove)(struct device * dev) = NULL;
+int (*platform_notify)(struct device *dev) = NULL;
+int (*platform_notify_remove)(struct device *dev) = NULL;

/*
 * sysfs bindings for devices.
@@ -51,11 +51,11 @@ EXPORT_SYMBOL(dev_driver_string);
```

[PATCH 196/196] Driver core: coding style fixes

```
#define to_dev(obj) container_of(obj, struct device, kobj)
#define to_dev_attr(_attr) container_of(_attr, struct device_attribute, attr)

-static ssize_t
-dev_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
+static ssize_t dev_attr_show(struct kobject *kobj, struct attribute *attr,
+ char *buf)
{
- struct device_attribute * dev_attr = to_dev_attr(attr);
- struct device * dev = to_dev(kobj);
+ struct device_attribute *dev_attr = to_dev_attr(attr);
+ struct device *dev = to_dev(kobj);
ssize_t ret = -EIO;

if (dev_attr->show)
@@ -63,12 +63,11 @@ dev_attr_show(struct kobject * kobj, struct attribute * attr, char * buf)
return ret;
}

-static ssize_t
-dev_attr_store(struct kobject * kobj, struct attribute * attr,
- const char * buf, size_t count)
+static ssize_t dev_attr_store(struct kobject *kobj, struct attribute *attr,
+ const char *buf, size_t count)
{
- struct device_attribute * dev_attr = to_dev_attr(attr);
- struct device * dev = to_dev(kobj);
+ struct device_attribute *dev_attr = to_dev_attr(attr);
+ struct device *dev = to_dev(kobj);
ssize_t ret = -EIO;

if (dev_attr->store)
@@ -90,9 +89,9 @@ static struct sysfs_ops dev_sysfs_ops = {
* reaches 0. We forward the call to the device's release
* method, which should handle actually freeing the structure.
*/
-static void device_release(struct kobject * kobj)
+static void device_release(struct kobject *kobj)
{
- struct device * dev = to_dev(kobj);
+ struct device *dev = to_dev(kobj);

if (dev->release)
dev->release(dev);
@@ -101,8 +100,8 @@ static void device_release(struct kobject * kobj)
else if (dev->class && dev->class->dev_release)
dev->class->dev_release(dev);
else {
- printk(KERN_ERR "Device '%s' does not have a release() function, "
- "it is broken and must be fixed.\n",
+ printk(KERN_ERR "Device '%s' does not have a release() "
```

[PATCH 196/196] Driver core: coding style fixes

```
+ "function, it is broken and must be fixed.\n",
dev->bus_id);
WARN_ON(1);
}
@@ -185,7 +184,8 @@ static int dev_uevent(struct kset *kset, struct kobject *kobj,
add_uevent_var(env, "PHYSDEVBUS=%s", dev->bus->name);

if (dev->driver)
- add_uevent_var(env, "PHYSDEVDRIVER=%s", dev->driver->name);
+ add_uevent_var(env, "PHYSDEVDRIVER=%s",
+ dev->driver->name);
}
#endif

@@ -327,7 +327,8 @@ static int device_add_groups(struct device *dev,
error = sysfs_create_group(&dev->kobj, groups[i]);
if (error) {
while (--i >= 0)
- sysfs_remove_group(&dev->kobj, groups[i]);
+ sysfs_remove_group(&dev->kobj,
+ groups[i]);
break;
}
}
@@ -406,14 +407,12 @@ static struct device_attribute devt_attr =
/* kset to create /sys/devices/ */
struct kset *devices_kset;

-
/**
- * device_create_file - create sysfs attribute file for device.
- * @dev: device.
- * @attr: device attribute descriptor.
+ * device_create_file - create sysfs attribute file for device.
+ * @dev: device.
+ * @attr: device attribute descriptor.
*/
-
-int device_create_file(struct device * dev, struct device_attribute * attr)
+int device_create_file(struct device *dev, struct device_attribute *attr)
{
int error = 0;
if (get_device(dev)) {
@@ -424,12 +423,11 @@ int device_create_file(struct device * dev, struct device_attribute * attr)
}

/**
- * device_remove_file - remove sysfs attribute file.
- * @dev: device.
- * @attr: device attribute descriptor.
+ * device_remove_file - remove sysfs attribute file.
```

[PATCH 196/196] Driver core: coding style fixes

```
+ * @dev: device.
+ * @attr: device attribute descriptor.
*/
-
-void device_remove_file(struct device * dev, struct device_attribute * attr)
+void device_remove_file(struct device *dev, struct device_attribute *attr)
{
if (get_device(dev)) {
sysfs_remove_file(&dev->kobj, &attr->attr);
@@ -510,18 +508,16 @@ static void klist_children_put(struct klist_node *n)
put_device(dev);
}

-
/**
- * device_initialize - init device structure.
- * @dev: device.
+ * device_initialize - init device structure.
+ * @dev: device.
*
- * This prepares the device for use by other layers,
- * including adding it to the device hierarchy.
- * It is the first half of device_register(), if called by
- * that, though it can also be called separately, so one
- * may use @dev's fields (e.g. the refcount).
+ * This prepares the device for use by other layers,
+ * including adding it to the device hierarchy.
+ * It is the first half of device_register(), if called by
+ * that, though it can also be called separately, so one
+ * may use @dev's fields (e.g. the refcount).
*/
-
void device_initialize(struct device *dev)
{
dev->kobj.kset = devices_kset;
@@ -754,15 +750,15 @@ static void device_remove_class_symlinks(struct device *dev)
}

/**
- * device_add - add device to device hierarchy.
- * @dev: device.
+ * device_add - add device to device hierarchy.
+ * @dev: device.
*
- * This is part 2 of device_register(), though may be called
- * separately _iff_ device_initialize() has been called separately.
+ * This is part 2 of device_register(), though may be called
+ * separately _iff_ device_initialize() has been called separately.
*
- * This adds it to the kobject hierarchy via kobject_add(), adds it
- * to the global and sibling lists for the device, then
```

[PATCH 196/196] Driver core: coding style fixes

```

- * adds it to the other relevant subsystems of the driver model.
+ * This adds it to the kobject hierarchy via kobject_add(), adds it
+ * to the global and sibling lists for the device, then
+ * adds it to the other relevant subsystems of the driver model.
*/
int device_add(struct device *dev)
{
@@ -870,70 +866,63 @@ int device_add(struct device *dev)
goto Done;
}

-
/**
- * device_register - register a device with the system.
- * @dev: pointer to the device structure
+ * device_register - register a device with the system.
+ * @dev: pointer to the device structure
*
- * This happens in two clean steps - initialize the device
- * and add it to the system. The two steps can be called
- * separately, but this is the easiest and most common.
- * I.e. you should only call the two helpers separately if
- * have a clearly defined need to use and refcount the device
- * before it is added to the hierarchy.
+ * This happens in two clean steps - initialize the device
+ * and add it to the system. The two steps can be called
+ * separately, but this is the easiest and most common.
+ * I.e. you should only call the two helpers separately if
+ * have a clearly defined need to use and refcount the device
+ * before it is added to the hierarchy.
*/
-
int device_register(struct device *dev)
{
device_initialize(dev);
return device_add(dev);
}

-
/**
- * get_device - increment reference count for device.
- * @dev: device.
+ * get_device - increment reference count for device.
+ * @dev: device.
*
- * This simply forwards the call to kobject_get(), though
- * we do take care to provide for the case that we get a NULL
- * pointer passed in.
+ * This simply forwards the call to kobject_get(), though
+ * we do take care to provide for the case that we get a NULL
+ * pointer passed in.

```

[PATCH 196/196] Driver core: coding style fixes

```
*/
-
-struct device * get_device(struct device * dev)
+struct device *get_device(struct device *dev)
{
return dev ? to_dev(kobject_get(&dev->kobj)) : NULL;
}

-
/**
- * put_device - decrement reference count.
- * @dev: device in question.
+ * put_device - decrement reference count.
+ * @dev: device in question.
*/
-void put_device(struct device * dev)
+void put_device(struct device *dev)
{
/* might_sleep(); */
if (dev)
kobject_put(&dev->kobj);
}

-
/**
- * device_del - delete device from system.
- * @dev: device.
+ * device_del - delete device from system.
+ * @dev: device.
*
- * This is the first part of the device unregistration
- * sequence. This removes the device from the lists we control
- * from here, has it removed from the other driver model
- * subsystems it was added to in device_add(), and removes it
- * from the kobject hierarchy.
+ * This is the first part of the device unregistration
+ * sequence. This removes the device from the lists we control
+ * from here, has it removed from the other driver model
+ * subsystems it was added to in device_add(), and removes it
+ * from the kobject hierarchy.
*
- * NOTE: this should be called manually _iff_ device_add() was
- * also called manually.
+ * NOTE: this should be called manually _iff_ device_add() was
+ * also called manually.
*/
-
-void device_del(struct device * dev)
+void device_del(struct device *dev)
{
- struct device * parent = dev->parent;
```

[PATCH 196/196] Driver core: coding style fixes

```
+ struct device *parent = dev->parent;
struct class_interface *class_intf;

device_pm_remove(dev);
@@ -979,47 +968,46 @@ void device_del(struct device * dev)
}

/**
- * device_unregister – unregister device from system.
- * @dev: device going away.
+ * device_unregister – unregister device from system.
+ * @dev: device going away.
*
- * We do this in two parts, like we do device_register(). First,
- * we remove it from all the subsystems with device_del(), then
- * we decrement the reference count via put_device(). If that
- * is the final reference count, the device will be cleaned up
- * via device_release() above. Otherwise, the structure will
- * stick around until the final reference to the device is dropped.
+ * We do this in two parts, like we do device_register(). First,
+ * we remove it from all the subsystems with device_del(), then
+ * we decrement the reference count via put_device(). If that
+ * is the final reference count, the device will be cleaned up
+ * via device_release() above. Otherwise, the structure will
+ * stick around until the final reference to the device is dropped.
*/
-void device_unregister(struct device * dev)
+void device_unregister(struct device *dev)
{
pr_debug("device: '%s': %s\n", dev->bus_id, __FUNCTION__);
device_del(dev);
put_device(dev);
}

-
-
-static struct device * next_device(struct klist_iter * i)
+static struct device *next_device(struct klist_iter *i)
{
- struct klist_node * n = klist_next(i);
+ struct klist_node *n = klist_next(i);
return n ? container_of(n, struct device, knode_parent) : NULL;
}

/**
- * device_for_each_child – device child iterator.
- * @parent: parent struct device.
- * @data: data for the callback.
- * @fn: function to be called for each device.
+ * device_for_each_child – device child iterator.
+ * @parent: parent struct device.
+ * @data: data for the callback.
```

[PATCH 196/196] Driver core: coding style fixes

```
+ * @fn: function to be called for each device.
*
- * Iterate over @parent's child devices, and call @fn for each,
- * passing it @data.
+ * Iterate over @parent's child devices, and call @fn for each,
+ * passing it @data.
*
- * We check the return of @fn each time. If it returns anything
- * other than 0, we break out and return that value.
+ * We check the return of @fn each time. If it returns anything
+ * other than 0, we break out and return that value.
*/
-int device_for_each_child(struct device * parent, void * data,
- int (*fn)(struct device *, void *))
+int device_for_each_child(struct device *parent, void *data,
+ int (*fn)(struct device *dev, void *data))
{
struct klist_iter i;
- struct device * child;
+ struct device *child;
int error = 0;

klist_iter_init(&parent->klist_children, &i);
@@ -1044,8 +1032,8 @@ int device_for_each_child(struct device * parent, void * data,
* current device can be obtained, this function will return to the caller
* and not iterate over any more devices.
*/
-struct device * device_find_child(struct device *parent, void *data,
- int (*match)(struct device *, void *))
+struct device *device_find_child(struct device *parent, void *data,
+ int (*match)(struct device *dev, void *data))
{
struct klist_iter i;
struct device *child;
@@ -1312,8 +1300,7 @@ static int device_move_class_links(struct device *dev,
class_name);
if (error)
sysfs_remove_link(&dev->kobj, "device");
- }
- else
+ } else
error = 0;
out:
kfree(class_name);
@@ -1344,7 +1331,7 @@ int device_move(struct device *dev, struct device *new_parent)
return -EINVAL;

new_parent = get_device(new_parent);
- new_parent_kobj = get_device_parent (dev, new_parent);
+ new_parent_kobj = get_device_parent(dev, new_parent);
```

[PATCH 196/196] Driver core: coding style fixes

```
pr_debug("device: '%s': %s: moving to '%s'\n", dev->bus_id,
__FUNCTION__, new_parent ? new_parent->bus_id : "<NULL>");
@@ -1390,7 +1377,7 @@ EXPORT_SYMBOL_GPL(device_move);
*/
void device_shutdown(void)
{
- struct device * dev, *devn;
+ struct device *dev, *devn;

list_for_each_entry_safe_reverse(dev, devn, &devices_kset->list,
kobj.entry) {
diff --git a/drivers/base/dd.c b/drivers/base/dd.c
index b0726eb..a5cde94 100644
--- a/drivers/base/dd.c
+++ b/drivers/base/dd.c
@@ -1,20 +1,20 @@
/*
- * drivers/base/dd.c - The core device/driver interactions.
+ * drivers/base/dd.c - The core device/driver interactions.
*
- * This file contains the (sometimes tricky) code that controls the
- * interactions between devices and drivers, which primarily includes
- * driver binding and unbinding.
+ * This file contains the (sometimes tricky) code that controls the
+ * interactions between devices and drivers, which primarily includes
+ * driver binding and unbinding.
*
- * All of this code used to exist in drivers/base/bus.c, but was
- * relocated to here in the name of compartmentalization (since it wasn't
- * strictly code just for the 'struct bus_type'.
+ * All of this code used to exist in drivers/base/bus.c, but was
+ * relocated to here in the name of compartmentalization (since it wasn't
+ * strictly code just for the 'struct bus_type'.
*
- * Copyright (c) 2002-5 Patrick Mochel
- * Copyright (c) 2002-3 Open Source Development Labs
- * Copyright (c) 2007 Greg Kroah-Hartman <gregkh@xxxxxxx>
- * Copyright (c) 2007 Novell Inc.
+ * Copyright (c) 2002-5 Patrick Mochel
+ * Copyright (c) 2002-3 Open Source Development Labs
+ * Copyright (c) 2007 Greg Kroah-Hartman <gregkh@xxxxxxx>
+ * Copyright (c) 2007 Novell Inc.
*
- * This file is released under the GPLv2
+ * This file is released under the GPLv2
*/

#include <linux/device.h>
@@ -71,18 +71,18 @@ static void driver_sysfs_remove(struct device *dev)
}
```

[PATCH 196/196] Driver core: coding style fixes

```
/**
- * device_bind_driver - bind a driver to one device.
- * @dev: device.
+ * device_bind_driver - bind a driver to one device.
+ * @dev: device.
*
- * Allow manual attachment of a driver to a device.
- * Caller must have already set @dev->driver.
+ * Allow manual attachment of a driver to a device.
+ * Caller must have already set @dev->driver.
*
- * Note that this does not modify the bus reference count
- * nor take the bus's rwsem. Please verify those are accounted
- * for before calling this. (It is ok to call with no other effort
- * from a driver's probe() method.)
+ * Note that this does not modify the bus reference count
+ * nor take the bus's rwsem. Please verify those are accounted
+ * for before calling this. (It is ok to call with no other effort
+ * from a driver's probe() method.)
*
- * This function must be called with @dev->sem held.
+ * This function must be called with @dev->sem held.
*/
int device_bind_driver(struct device *dev)
{
@@ -93,6 +93,7 @@ int device_bind_driver(struct device *dev)
driver_bound(dev);
return ret;
}
+EXPORT_SYMBOL_GPL(device_bind_driver);

static atomic_t probe_count = ATOMIC_INIT(0);
static DECLARE_WAIT_QUEUE_HEAD(probe_waitqueue);
@@ -183,7 +184,7 @@ int driver_probe_done(void)
* This function must be called with @dev->sem held. When called for a
* USB interface, @dev->parent->sem must be held as well.
*/
-int driver_probe_device(struct device_driver * drv, struct device * dev)
+int driver_probe_device(struct device_driver *drv, struct device *dev)
{
int ret = 0;

@@ -201,27 +202,27 @@ done:
return ret;
}

-static int __device_attach(struct device_driver * drv, void * data)
+static int __device_attach(struct device_driver *drv, void *data)
{
- struct device * dev = data;
+ struct device *dev = data;
```

[PATCH 196/196] Driver core: coding style fixes

```
return driver_probe_device(drv, dev);
}

/**
- * device_attach – try to attach device to a driver.
- * @dev: device.
+ * device_attach – try to attach device to a driver.
+ * @dev: device.
*
- * Walk the list of drivers that the bus has and call
- * driver_probe_device() for each pair. If a compatible
- * pair is found, break out and return.
+ * Walk the list of drivers that the bus has and call
+ * driver_probe_device() for each pair. If a compatible
+ * pair is found, break out and return.
*
- * Returns 1 if the device was bound to a driver;
- * 0 if no matching device was found;
- * -ENODEV if the device is not registered.
+ * Returns 1 if the device was bound to a driver;
+ * 0 if no matching device was found;
+ * -ENODEV if the device is not registered.
*
- * When called for a USB interface, @dev->parent->sem must be held.
+ * When called for a USB interface, @dev->parent->sem must be held.
*/
-int device_attach(struct device * dev)
+int device_attach(struct device *dev)
{
int ret = 0;

@@ -240,10 +241,11 @@ int device_attach(struct device * dev)
up(&dev->sem);
return ret;
}
+EXPORT_SYMBOL_GPL(device_attach);

-static int __driver_attach(struct device * dev, void * data)
+static int __driver_attach(struct device *dev, void *data)
{
- struct device_driver * drv = data;
+ struct device_driver *drv = data;

/*
* Lock device and try to bind to it. We drop the error
@@ -268,26 +270,27 @@ static int __driver_attach(struct device * dev, void * data)
}

/**
- * driver_attach – try to bind driver to devices.
- * @drv: driver.
```

[PATCH 196/196] Driver core: coding style fixes

```
+ * driver_attach – try to bind driver to devices.
+ * @drv: driver.
*
– * Walk the list of devices that the bus has on it and try to
– * match the driver with each one. If driver_probe_device()
– * returns 0 and the @dev->driver is set, we've found a
– * compatible pair.
+ * Walk the list of devices that the bus has on it and try to
+ * match the driver with each one. If driver_probe_device()
+ * returns 0 and the @dev->driver is set, we've found a
+ * compatible pair.
*/
–int driver_attach(struct device_driver * drv)
+int driver_attach(struct device_driver *drv)
{
return bus_for_each_dev(drv->bus, NULL, drv, __driver_attach);
}
+EXPORT_SYMBOL_GPL(driver_attach);

/*
– * __device_release_driver() must be called with @dev->sem held.
– * When called for a USB interface, @dev->parent->sem must be held as well.
+ * __device_release_driver() must be called with @dev->sem held.
+ * When called for a USB interface, @dev->parent->sem must be held as well.
*/
–static void __device_release_driver(struct device * dev)
+static void __device_release_driver(struct device *dev)
{
– struct device_driver * drv;
+ struct device_driver *drv;

drv = dev->driver;
if (drv) {
@@ -310,13 +313,13 @@ static void __device_release_driver(struct device * dev)
}

/**
– * device_release_driver – manually detach device from driver.
– * @dev: device.
+ * device_release_driver – manually detach device from driver.
+ * @dev: device.
*
– * Manually detach device from driver.
– * When called for a USB interface, @dev->parent->sem must be held.
+ * Manually detach device from driver.
+ * When called for a USB interface, @dev->parent->sem must be held.
*/
–void device_release_driver(struct device * dev)
+void device_release_driver(struct device *dev)
{
/*
```

[PATCH 196/196] Driver core: coding style fixes

```
* If anyone calls device_release_driver() recursively from
@@ -327,15 +330,15 @@ void device_release_driver(struct device * dev)
__device_release_driver(dev);
up(&dev->sem);
}
-
+EXPORT_SYMBOL_GPL(device_release_driver);

/**
 * driver_detach – detach driver from all devices it controls.
 * @drv: driver.
 */
-void driver_detach(struct device_driver * drv)
+void driver_detach(struct device_driver *drv)
{
- struct device * dev;
+ struct device *dev;

for (;;) {
spin_lock(&drv->p->klist_devices.k_lock);
@@ -359,9 +362,3 @@ void driver_detach(struct device_driver * drv)
put_device(dev);
}
}
-
-EXPORT_SYMBOL_GPL(device_bind_driver);
-EXPORT_SYMBOL_GPL(device_release_driver);
-EXPORT_SYMBOL_GPL(device_attach);
-EXPORT_SYMBOL_GPL(driver_attach);
-
diff --git a/drivers/base/driver.c b/drivers/base/driver.c
index 94b697a..a35f041 100644
--- a/drivers/base/driver.c
+++ b/drivers/base/driver.c
@@ -19,27 +19,26 @@
#define to_dev(node) container_of(node, struct device, driver_list)

-static struct device * next_device(struct klist_iter * i)
+static struct device *next_device(struct klist_iter *i)
{
- struct klist_node * n = klist_next(i);
+ struct klist_node *n = klist_next(i);
return n ? container_of(n, struct device, knode_driver) : NULL;
}

/**
- * driver_for_each_device – Iterator for devices bound to a driver.
- * @drv: Driver we're iterating.
- * @start: Device to begin with
- * @data: Data to pass to the callback.
```

[PATCH 196/196] Driver core: coding style fixes

```
- * @fn: Function to call for each device.
+ * driver_for_each_device – Iterator for devices bound to a driver.
+ * @drv: Driver we're iterating.
+ * @start: Device to begin with
+ * @data: Data to pass to the callback.
+ * @fn: Function to call for each device.
*
- * Iterate over the @drv's list of devices calling @fn for each one.
+ * Iterate over the @drv's list of devices calling @fn for each one.
*/
-
-int driver_for_each_device(struct device_driver * drv, struct device * start,
- void * data, int (*fn)(struct device *, void *))
+int driver_for_each_device(struct device_driver *drv, struct device *start,
+ void *data, int (*fn)(struct device *, void *))
{
struct klist_iter i;
- struct device * dev;
+ struct device *dev;
int error = 0;

if (!drv)
@@ -52,10 +51,8 @@ int driver_for_each_device(struct device_driver * drv, struct device * start,
klist_iter_exit(&i);
return error;
}
-
EXPORT_SYMBOL_GPL(driver_for_each_device);

-
/**
 * driver_find_device – device iterator for locating a particular device.
 * @drv: The device's driver
@@ -71,9 +68,9 @@ EXPORT_SYMBOL_GPL(driver_for_each_device);
 * if it does. If the callback returns non-zero, this function will
 * return to the caller and not iterate over any more devices.
 */
-struct device * driver_find_device(struct device_driver *drv,
- struct device * start, void * data,
- int (*match)(struct device *, void *))
+struct device *driver_find_device(struct device_driver *drv,
+ struct device *start, void *data,
+ int (*match)(struct device *dev, void *data))
{
struct klist_iter i;
struct device *dev;
@@ -92,12 +89,12 @@ struct device * driver_find_device(struct device_driver *drv,
EXPORT_SYMBOL_GPL(driver_find_device);

/**
- * driver_create_file – create sysfs file for driver.
```

[PATCH 196/196] Driver core: coding style fixes

```
- * @drv: driver.
- * @attr: driver attribute descriptor.
+ * driver_create_file – create sysfs file for driver.
+ * @drv: driver.
+ * @attr: driver attribute descriptor.
*/
-
-int driver_create_file(struct device_driver * drv, struct driver_attribute * attr)
+int driver_create_file(struct device_driver *drv,
+ struct driver_attribute *attr)
{
int error;
if (get_driver(drv)) {
@@ -107,22 +104,22 @@ int driver_create_file(struct device_driver * drv, struct driver_attribute * attr
error = -EINVAL;
return error;
}
-
+EXPORT_SYMBOL_GPL(driver_create_file);

/**
- * driver_remove_file – remove sysfs file for driver.
- * @drv: driver.
- * @attr: driver attribute descriptor.
+ * driver_remove_file – remove sysfs file for driver.
+ * @drv: driver.
+ * @attr: driver attribute descriptor.
*/
-
-void driver_remove_file(struct device_driver * drv, struct driver_attribute * attr)
+void driver_remove_file(struct device_driver *drv,
+ struct driver_attribute *attr)
{
if (get_driver(drv)) {
sysfs_remove_file(&drv->p->kobj, &attr->attr);
put_driver(drv);
}
}
-
+EXPORT_SYMBOL_GPL(driver_remove_file);

/**
* driver_add_kobj – add a kobject below the specified driver
@@ -149,10 +146,10 @@ int driver_add_kobj(struct device_driver *drv, struct kobject *kobj,
EXPORT_SYMBOL_GPL(driver_add_kobj);

/**
- * get_driver – increment driver reference count.
- * @drv: driver.
+ * get_driver – increment driver reference count.
+ * @drv: driver.
```

[PATCH 196/196] Driver core: coding style fixes

```
*/
-struct device_driver * get_driver(struct device_driver * drv)
+struct device_driver *get_driver(struct device_driver *drv)
{
if (drv) {
struct driver_private *priv;
@@ -164,16 +161,17 @@ struct device_driver * get_driver(struct device_driver * drv)
}
return NULL;
}
-
+EXPORT_SYMBOL_GPL(get_driver);

/**
- * put_driver - decrement driver's refcount.
- * @drv: driver.
+ * put_driver - decrement driver's refcount.
+ * @drv: driver.
*/
-void put_driver(struct device_driver * drv)
+void put_driver(struct device_driver *drv)
{
kobject_put(&drv->p->kobj);
}
+EXPORT_SYMBOL_GPL(put_driver);

static int driver_add_groups(struct device_driver *drv,
struct attribute_group **groups)
@@ -205,24 +203,23 @@ static void driver_remove_groups(struct device_driver *drv,
sysfs_remove_group(&drv->p->kobj, groups[i]);
}

-
/**
- * driver_register - register driver with bus
- * @drv: driver to register
+ * driver_register - register driver with bus
+ * @drv: driver to register
*
- * We pass off most of the work to the bus_add_driver() call,
- * since most of the things we have to do deal with the bus
- * structures.
+ * We pass off most of the work to the bus_add_driver() call,
+ * since most of the things we have to do deal with the bus
+ * structures.
*/
-int driver_register(struct device_driver * drv)
+int driver_register(struct device_driver *drv)
{
int ret;
```

[PATCH 196/196] Driver core: coding style fixes

```
if ((drv->bus->probe && drv->probe) ||
(drv->bus->remove && drv->remove) ||
- (drv->bus->shutdown && drv->shutdown)) {
- printk(KERN_WARNING "Driver '%s' needs updating - please use bus_type methods\n", drv->name);
- }
+ (drv->bus->shutdown && drv->shutdown))
+ printk(KERN_WARNING "Driver '%s' needs updating - please use "
+ "bus_type methods\n", drv->name);
ret = bus_add_driver(drv);
if (ret)
return ret;
@@ -231,29 +228,30 @@ int driver_register(struct device_driver * drv)
bus_remove_driver(drv);
return ret;
}
+EXPORT_SYMBOL_GPL(driver_register);

/**
- * driver_unregister - remove driver from system.
- * @drv: driver.
+ * driver_unregister - remove driver from system.
+ * @drv: driver.
*
- * Again, we pass off most of the work to the bus-level call.
+ * Again, we pass off most of the work to the bus-level call.
*/
-
-void driver_unregister(struct device_driver * drv)
+void driver_unregister(struct device_driver *drv)
{
driver_remove_groups(drv, drv->groups);
bus_remove_driver(drv);
}
+EXPORT_SYMBOL_GPL(driver_unregister);

/**
- * driver_find - locate driver on a bus by its name.
- * @name: name of the driver.
- * @bus: bus to scan for the driver.
+ * driver_find - locate driver on a bus by its name.
+ * @name: name of the driver.
+ * @bus: bus to scan for the driver.
*
- * Call kset_find_obj() to iterate over list of drivers on
- * a bus to find driver by name. Return driver if found.
+ * Call kset_find_obj() to iterate over list of drivers on
+ * a bus to find driver by name. Return driver if found.
*
- * Note that kset_find_obj increments driver's reference count.
+ * Note that kset_find_obj increments driver's reference count.
*/
```

[PATCH 196/196] Driver core: coding style fixes

```
struct device_driver *driver_find(const char *name, struct bus_type *bus)
{
@@ -266,12 +264,4 @@ struct device_driver *driver_find(const char *name, struct bus_type *bus)
}
return NULL;
```