Dynamic Major Number

- The above example uses the older *static* method to assign a device number
- Today dynamic allocation is preferred
- Here is how:

```
#include <linux/kdev_t.h>
dev_t dev;
```

- This declares **`dev`** to be a device number (both major and minor). Now assign it a value
- **`dev = MKDEV(234, 0);`**

Requesting a number

- Now request a number

```
#include <linux/fs.h>
int register_chrdev_region(dev, 4, "hello");
```

  - This requests a device number starting with 234 (previous page)
  - It asks for 4 minor numbers
  - Uses the name "hello"
- When done with the device use:

```
void unregister_chrdev_region(dev, 4);
```

Using **`mknod`**

- If you major number is assigned dynamically, how do you use **`mknod`**? Try the following

```
module="hello"
device="hello"
mode="664"
# remove stale nodes
/sbin/insmod ./$module.ko $* || exit 1
rm -f /dev/${device}0
major=`awk "\\$2==\"$module\" {print \\$1} /proc/devices`
mknod /dev/${device}0 c $major 0
```

/proc/devices

```
Character
devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  7 vcs
 10 misc
 13 input
 14 sound
 21 sg
 29 fb
 81 video4linux
89 i2c
90 mtd
116 alsa
128 ptm
136 pts
153 spi
161 ircomm
180 usb
189 usb_device
216 rfcomm
247 bccat
248 pvrsrvkm
249 rtc
250 ttySDIO
251 omap-resizer
252 omap-
previewer
253 usbmon
254 bsg

Block devices:
  1 ramdisk
259 blkext
  7 loop
  8 sd
 11 sr
 31 mtdblock
 65 sd
 66 sd
 67 sd
 68 sd
 69 sd
70 sd
 71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
179 mmc
```

Assignment

- See http://elinux.org/EBC_Exercise_16_Device_Drivers

Module dependencies

- Some kernel modules can depend on other modules, which need to be loaded first
- Example: the usb-storage module depends on the scsi_mod, libusual and usbcore modules
- Dependencies are described in /lib/modules/<kernel-version>/modules.dep

/lib/modules/2.6.32/models.dep

```
kernel/drivers/char/examples/hello1.ko:
kernel/crypto/twofish_common.ko:
kernel/crypto/ctr.ko:
kernel/crypto/blowfish.ko:
kernel/crypto/ghash-generic.ko:
kernel/crypto/gf128mul.ko
kernel/crypto/xts.ko:
kernel/crypto/gf128mul.ko
kernel/crypto/gcm.ko:
kernel/crypto/cryptd.ko:
kernel/crypto/md4.ko:
kernel/crypto/lrw.ko:
kernel/crypto/gf128mul.ko
```

Kernel log

When a new module is loaded,
related information is available in the kernel log

- The kernel keeps its messages in a circular buffer (so that it doesn't consume more memory with many messages)
- Kernel log messages are available through the `dmesg` command
 (“**d**iagnostic **mes**sa**g**e”)
- Kernel log messages are also displayed in the system console (messages can be filtered by level using `/proc/sys/kernel/printk`)

printk

- `/proc/sys/kernel/printk`
- The four values in this file are
  - *console_loglevel*,
  - *default_message_loglevel*,
  - *minimum_console_level* and
  - *default_con- sole_loglevel*.
- These values influence `printk()` behavior when printing or logging error messages
- Messages with a higher priority than *console_loglevel* will be printed to the console
- Messages without an explicit priority will be printed with priority *default_message_level*

http://www.tin.org/bin/man.cgi?section=5&topic=proc

Kernel log levels

0 (KERN_EMERG)	The system is unusable
1 (KERN_ALERT)	Actions that must be taken care of immediately
2 (KERN_CRIT)	Critical conditions
3 (KERN_ERR)	Noncritical error conditions
4 (KERN_WARNING)	Warning conditions that should be taken care of
5 (KERN_NOTICE)	Normal, but significant events
6 (KERN_INFO)	Informational messages that require no action
7 (KERN_DEBUG)	Kernel debugging messages, output by the

Module utilities (1)

- modinfo <module_name>
  modinfo <module_path>.ko
  Gets information about a module: parameters, license, description and dependencies.
  Very useful before deciding to load a module or not.
- sudo insmod <module_path>.ko
  Tries to load the given module. The full path to the module object file must be given.

Understanding module loading

- When loading a module fails,
  insmod often doesn't give you enough details!
- Details are often available in the kernel log
- Example:

```
beagle$ sudo insmod ./intr_monitor.ko
insmod: error inserting './intr_monitor.ko': -1
Device or resource busy
beagle$ dmesg
[17549774.552000] Failed to register handler for
irq channel 2
```

Module utilities (2)

- sudo modprobe <module_name>
  Most common usage of modprobe: tries to load all the modules the given module depends on, and then this module. Lots of other options are available. modprobe automatically looks in `/lib/modules/<version>/` for the object file corresponding to the given module name.
- lsmod
  Displays the list of loaded modules
  Compare its output with the contents of /proc/modules!

lsmod

```
beagle$ lsmod
Module                  Size  Used by
bufferclass_ti          4768  0
omaplfb                 8733  0
pvrsrvkm              154248  2 bufferclass_ti,omaplfb
rfcomm                 33484  0
ircomm_tty             30305  0
ircomm                 16429  1 ircomm_tty
irda                  162973  2 ircomm_tty,ircomm
ipv6                  249063  14
hidp                   11193  0
l2cap                  30104  4 rfcomm,hidp
bluetooth              49221  3 rfcomm,hidp,l2cap
…
```

Module utilities (3)

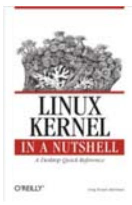
- sudo rmmod <module_name>
  Tries to remove the given module.
  Will only be allowed if the module is no longer in use (for example, no more processes opening a device file)
- sudo modprobe -r <module_name>
  Tries to remove the given module and all dependent modules (which are no longer needed after the module removal)

Passing parameters to modules

- Find available parameters:
  modinfo snd-intel8x0m
- Through insmod:
  sudo insmod ./snd-intel8x0m.ko index=-2
- Through modprobe:
  Set parameters in /etc/modprobe.conf or in any file in /etc/modprobe.d/:
  options snd-intel8x0m index=-2
- Through the kernel command line, when the module is built statically into the kernel:
  snd-intel8x0m.index=-2

module name
module parameter name
module parameter value

Useful reading



Linux Kernel in a Nutshell, Dec 2006

- By Greg Kroah-Hartman, O'Reilly
  http://www.kroah.com/lkn/
- A good reference book and guide on configuring, compiling and managing the Linux kernel sources.
- Freely available on-line!
  Great companion to the printed book for easy electronic searches!
  Available as single PDF file on
  http://free-electrons.com/community/kernel/lkn/

Useful reading too

Linux Device Drivers, Third Edition, February 2005

- By Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman, O'Reilly
  http://lwn.net/Kernel/LDD3/
- Freely available on-line!
  Great companion to the printed book for easy electronic searches!
  Available as single PDF file
- LDD3 is current as of the 2.6.10 kernel (Old?)