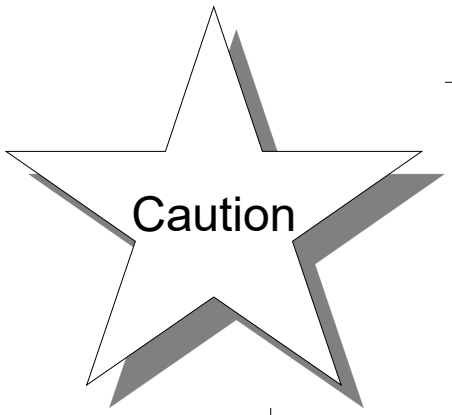
Intro to Linux Kernel



Kernel coding is different!

Can be hard to understand different syntax, functions, advanced C code in kernel!

Topics

1) How can we see an application’s sys-calls?
2) How does Linux kernel work with hardware?
3) How do we build and load a kernel image?

strace:
Viewport to the Kernel

Accelerometer Motivation Demo

- See Accelerometer Data Sheet: p22 for who-am-i register
  - I2C address 0x1C
  - Who-am-i Register 0x0D
- Setup

```
(bbg)$ config-pin p9_18 i2c
(bbg)$ config-pin p9_17 i2c
(bbg)$ i2cdetect -l
(bbg)$ i2cdetect -y -r 1
```

- Run i2cget

```
(bbg)$ i2cget -y 1 0x1C 0x0D
= 0x2a
```

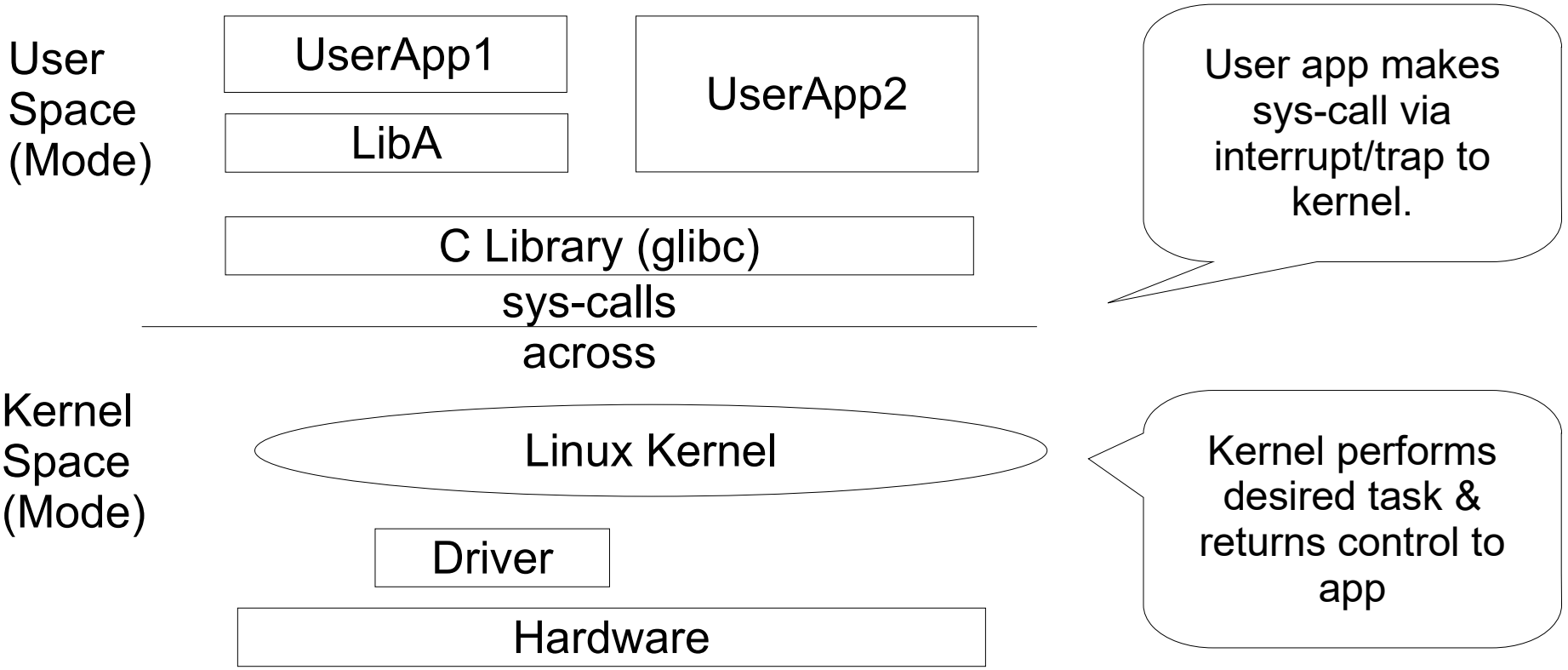
- Run my tool

```
(bbg)$ ./myi2cget
= 0x??
```

?!? Why ?!?

What is i2cget doing that it works? Let's find out!

User vs Kernel Space (“Mode”)



- Kernel is..
  - ..
  - Errors in user application don’t crash system.

strace

- strace: ..
  - “**S**ys-call **trace**”
  - Command:
    (bbg)$ strace ./myApp some args 2> outputFile.txt
    - 2> redirects stderr to a file
- strace Output format

```
sysCallFunction(args,...)   = ReturnValue
```

ioctl()

```
debian@beaglebone:/dev$ ls -l ttyS*
crw------- 1 debian tty     4, 64 Mar  1  2021 ttyS0
crw-rw---- 1 root   dialout 4, 65 Feb 27 06:57 ttyS1
crw-rw---- 1 root   dialout 4, 66 Feb 27 06:57 ttyS2
```

- Device Nodes in /dev

  `(bbg)$ ls -l /dev`

  - c = character, b = block
  - Node #'s (before date):
    major = which driver; minor = sub-part of driver
- ioctl()..

- Arguments
  1. File descriptor
  2. Device-dependent request code
  3. void* or an unsigned long
     (dependent on request code)

I2C strace Demo

- Run strace
  - (bbg)$ sudo apt-get install strace
    (bbg)$ cd /mnt/remote/myApps
    (bbg)$ **strace ./myi2cget 2> myi2cget.txt**
    (bbg)$ **strace i2cget -y 1 0x1C 0x0D 2> i2cget.txt**
- Look at myi2cget.txt

```
open("/dev/i2c-1", O_RDWR)              = 3
ioctl(3, 0x703, 0x1c)                   = 0
write(3, "\r", 1)                       = 1
read(3, "\0", 1)                        = 1
close(3)                                = 0
```

Analysis next slide

I2C strace Demo Analysis

myi2cget.txt

```
open("/dev/i2c-1", O_RDWR)  = 3



ioctl(3, 0x703, 0x1c)       = 0
   Set Slave Mode

write(3, "\r", 1)           = 1
   Set  reg addr: '\r' = 0x0d

read(3, "\0", 1)            = 1
   Read 1 byte

close(3)                    = 0
```

i2cget.txt

```
open("/dev/i2c-1", O_RDWR)  = 3
ioctl(3, 0x705, 0xbe8f5b50) = 0
   Get capabilities; 2nd arg is
*long

ioctl(3, 0x703, 0x1c)       = 0
   Set Slave Mode

ioctl(3, 0x720, 0xbe8f5b50) = 0
   SMBUS operations (pass pointer)

close(3)                    = 0
```

0x720?

..

Following I2C_SLAVE into i2c-dev.h

0x720

= I2C_SMBUS (system management bus)

= protocol built on top of I2C

So, we're using I2C, i2cget uses SMBus

Linux Kernel Basics

Kernel Basics

- Monolithic kernel
  - ..
  - fully linked (no run-time dependencies)
  - no fool-proof internal memory protection
- Kernel source directory structure

```
Documentation/  - Kernel docs   (Ex: coding style guide)
include/        - Kernel header files
drivers/        - Source code to drivers
  .../char/     - Byte-based drivers
arch/arm/       - ARM specific code
init/               - General startup code
```

Drivers

- ..
- Types of Drivers
  - Packet: Networks
  - Block: Disk and memory
  - Character: ..
    Ex: tty, input, console, frame buffer, sound, ...
- Can compile module into the kernel image
  - good for network, file-system, etc.
- Can compile driver into a..
  - Compiled for kernel's internal interface (functions)
    -- specific to a kernel version
  - Creates a .ko file: Kernel Object; in /lib/modules/...

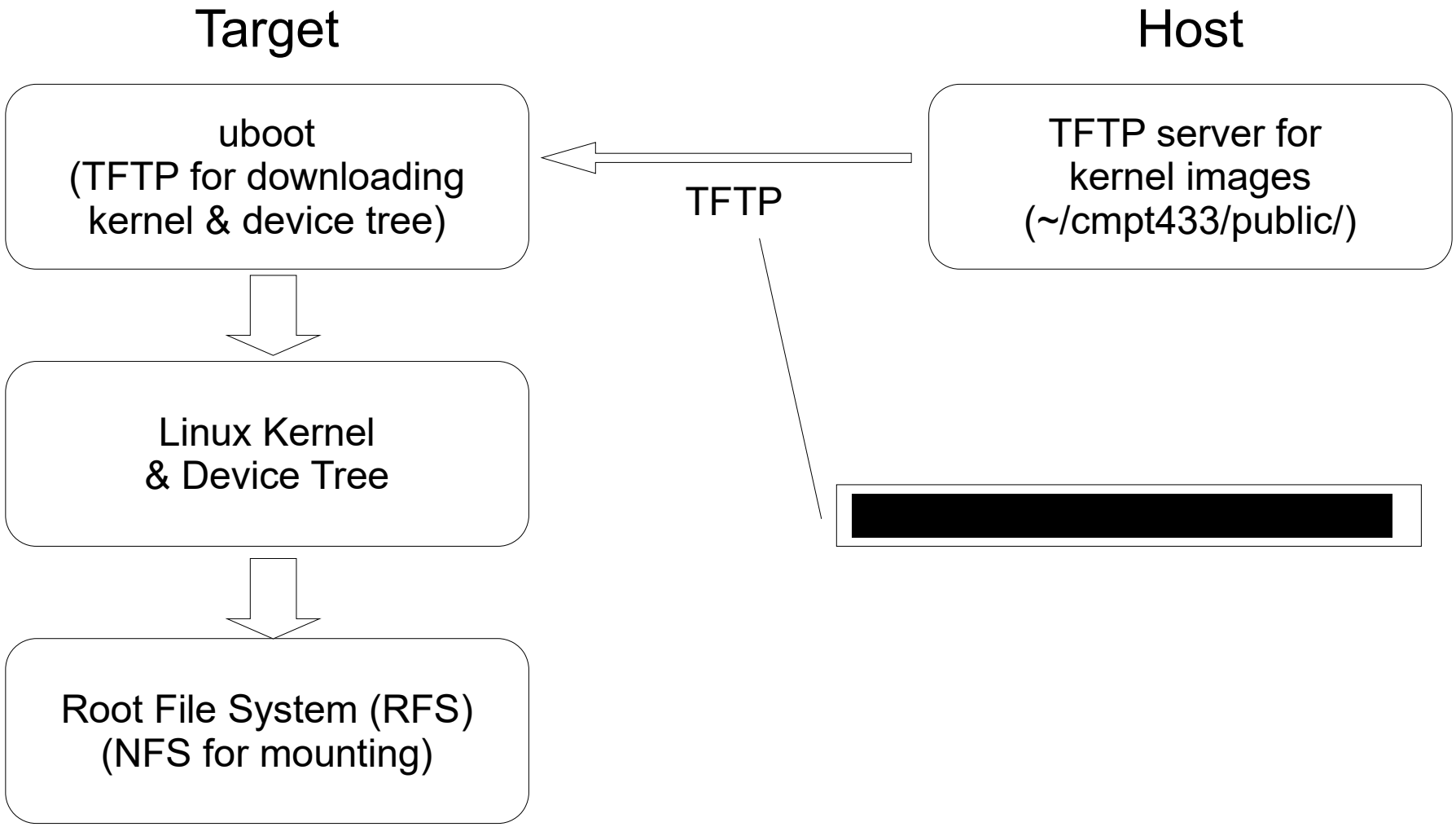
How Kernel knows Hardware

- Kernel must be told about the hardware in product
  - Many embedded board configurations!
  - "Old" way: board specific headers with hardware info:
    - serial ports, memory size, peripheral addresses, ...
- Problem?
  - Every new board/change requires push of code into Linux kernel.
  - Maintainers getting inundated with pushes (Linus rant)
- Solution: (Kernel 3.8+)
  - Create a special file to store hardware description =..

Device Tree

- Device Tree..
  - Kernel needs this to provide services.
    Ex:
    - What serial ports are connected?
    - What LEDs are connected? Where?
- Device Tree’s File Types
  - .dts:..
    in arch/arm/boot/dts
  - .dtb:..
    Passed to kernel via U-Boot
  - .dtbo:..
    Change the device tree at runtime

Boot Sequence if Downloading new Kernel



Summary & Demo

- strace: view app's sys-calls
- Kernel drivers ("modules")
  - run-time loadable or compiled into kernel image
- Device Tree: config file describing the hardware
- Boot Sequence
  - uboot: download kernel and device tree
  - run Linux & device tree
  - Loads root file system
- **DEMO:**
  - Kernel build, download & boot demo.
  - See Driver Creation Guide for details.