Linux Internals

ECE 373

Overview

- Short History
- Brief description
- 20,000 foot level
- Navigating the source
- Top Down trace – user land to kernel
- Bottom Up trace – hardware to kernel

Linux, at a glance

- General and flexible OS kernel
- Supercomputers, Servers, Clients, Devices
- Configurable
- Open Source
- GPLv2 (controversy!!)
- Usually found in a full distribution with user environments and GUIs

Pre-History

- UNIX – 1969, Thompson & Ritchie, PDP-11
- BSD - 1977, Bill Joy and friends, PDP and VAX
- GNU - 1983, Richard Stallman, et al
- MINIX - 1987, Tannenbaum, 80286
- Linux – 1991, Torvalds, 80386





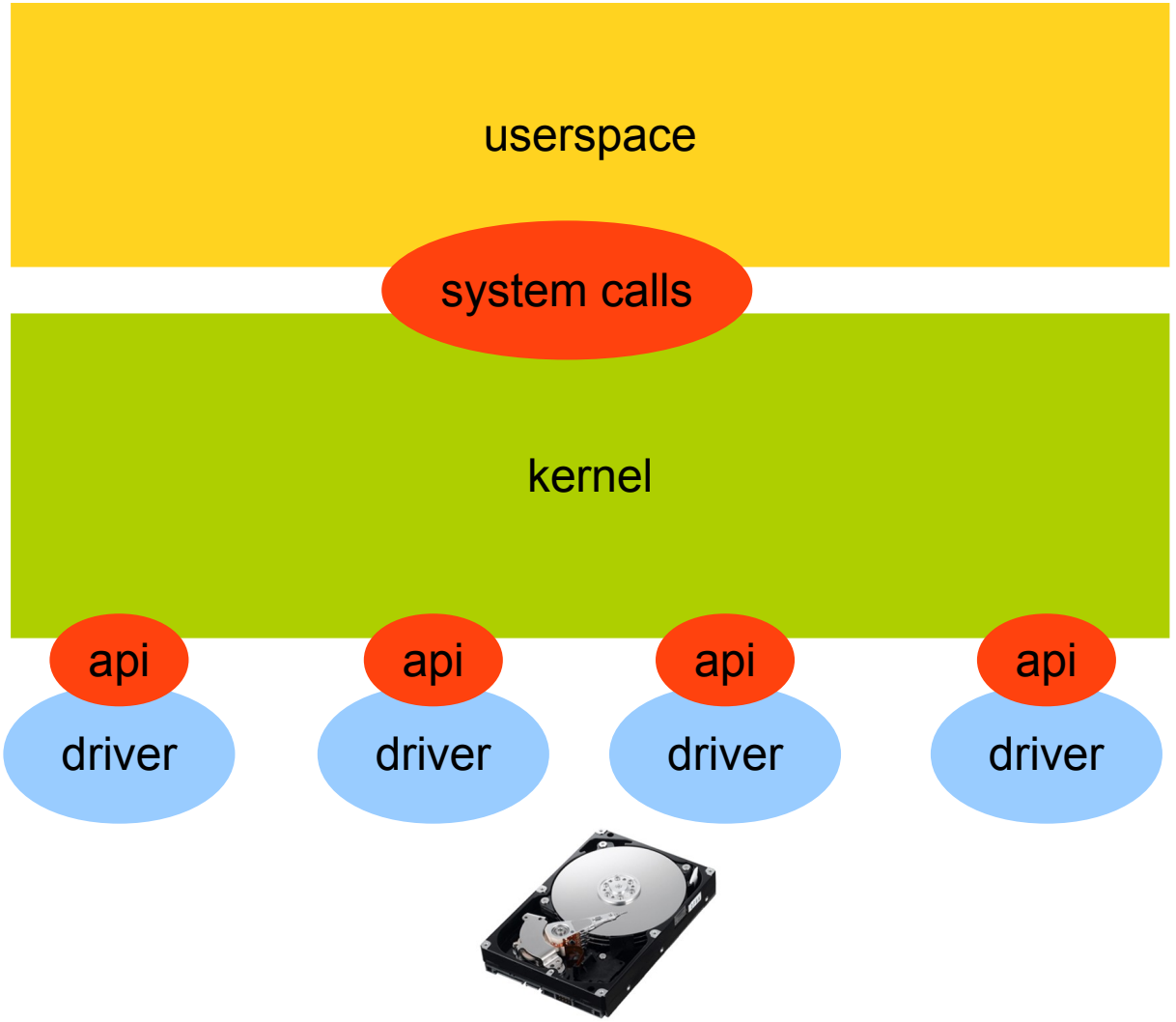
Linux Early History

- 1990 – Started as a terminal emulator project
- 1991 – Released to public
- 1993 – Slackware distro, Peter Volkerding
- 2000 – RedHat Commercial distro
- 1999 – Realtime extentions
- Embedded uses – Wind River, MontaVista, etc

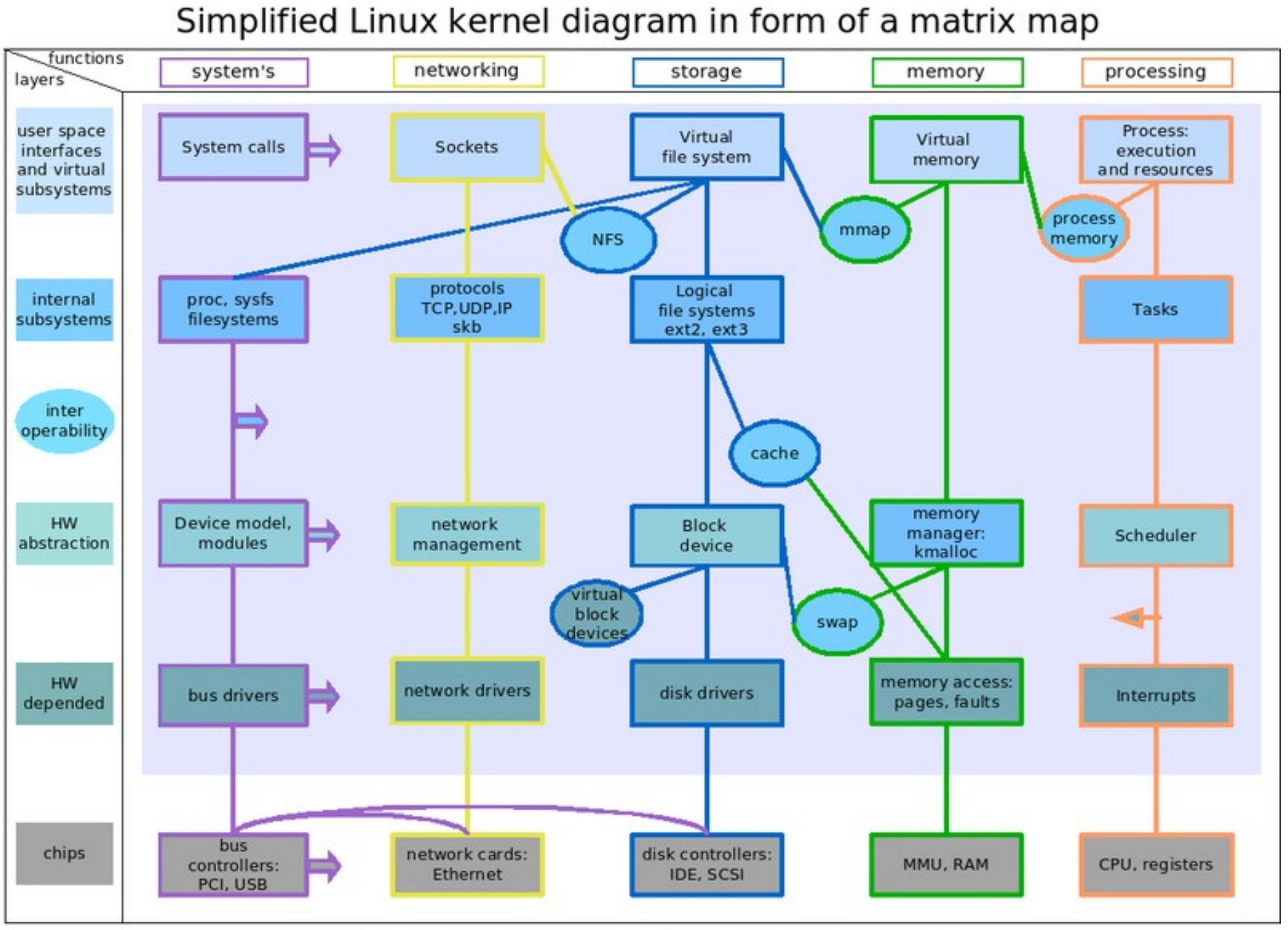
20,000 foot level

- Monolithic kernel
- Dynamically loadable modules
- Pre-emption

A little closer...



Too close?


Simplified Linux kernel diagram in form of a matrix map
functions
layers
system's
networking
storage
memory
processing
user space interfaces and virtual subsystems
internal subsystems
inter operability
HW abstraction
HW depended
chips
System calls
Sockets
Virtual file system
Virtual memory
Process: execution and resources
NFS
mmap
process memory
proc, sysfs filesystems
protocols TCP,UDP,IP skb
Logical file systems ext2, ext3
Tasks
cache
Device model, modules
network management
Block device
memory manager: kmalloc
Scheduler
virtual block devices
swap
bus drivers
network drivers
disk drivers
memory access: pages, faults
Interrupts
bus controllers: PCI, USB
network cards: Ethernet
disk controllers: IDE, SCSI
MMU, RAM
CPU, registers
Designed with OpenOffice.org by (cc) (by-nc-sa) Constantine Shulyupin, www.linuxdriver.co.il

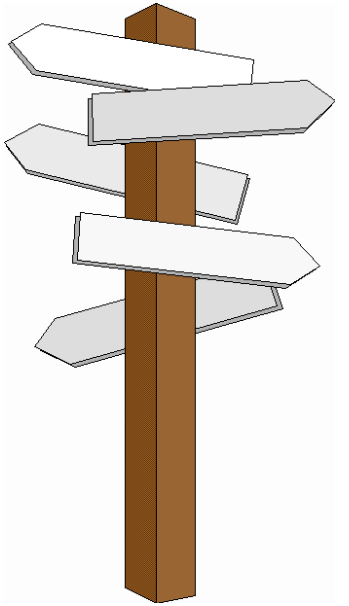
Navigating the dark and creepy places

- Where to start when diving in?
- Many resources to read kernel
- kernel.org + text editor
  - https://www.kernel.org/
- ctags / cscope
- LXR resources
  - http://lxr.free-electrons.com/

Other Kernel Resources

http://elinux.org/Linux_Kernel_Resources

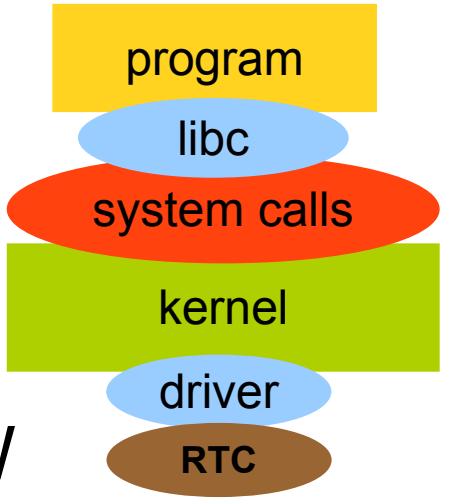
- References
  - kernel source
  - mailing lists
  - books
  - news
  - alternate architectures



Top down

What happens when a program requests the current time?

- program calls gettimeofday()
- libc
- Systrap with user buffer
- Kernel finds RTC driver
- RTC driver gets time value from clock HW
- Kernel service writes time to user space
- Systrap return



Following gettimeofday()

- First look at the manpage!
- Call starts in libc (not within today's scope)
- gettimeofday() wrapper called in kernel
- System call initiated, trap invoked through fast software interrupt
- Clock source read
- Buffers returned
- Trap complete

Walking the path...

- http://elixir.free-electrons.com/linux/latest/source/include/uapi/asm-generic/unistd.h#L483
- http://lxr.free-electrons.com/source/include/linux/syscalls.h#L212
- http://lxr.free-electrons.com/source/arch/arc/kernel/sys.c#L10
- http://lxr.free-electrons.com/source/kernel/time/time.c#L102
- http://lxr.free-electrons.com/source/kernel/time/timekeeping.c#L695
- http://lxr.free-electrons.com/source/kernel/time/timekeeping.c#L526
- http://lxr.free-electrons.com/source/kernel/time/timekeeping.c#L493
- http://lxr.free-electrons.com/source/kernel/time/timekeeping.c#L194
- … to a registered clock driver

- http://en.wikipedia.org/wiki/System_call

Bottom up

- What happens when the user hits "<enter>"?
  - Serial line read by UART
  - UART buffers characters, pulls interrupt line #x
  - CPU stores current process A, searches table for handler for interrupt #x
  - Handler code extracts character from UART through I/O register access
  - Find process B waiting on device semaphore, give it the character
  - Reschedule process B as Ready
  - Return from interrupt and resume process A

This was only an overview...

- Linux is open, readable, but big
- Tons of resources available to traverse the kernel and its code
- Will drill much deeper into the kernel as class goes on, don't worry!
- Ask questions!!