

Linux Kernel Management

Michael Tsai
2016/05/04

Some “small” thing to talk about

- For productivity (and eye health)
 - Use your local terminal software to SSH into the host you are working in.
 - Use a font / size that looks comfortable to you
- Additional advantage: your “google window” on the side and copy-and-paste functionality
- Port forwarding function in VirtualBox might be helpful

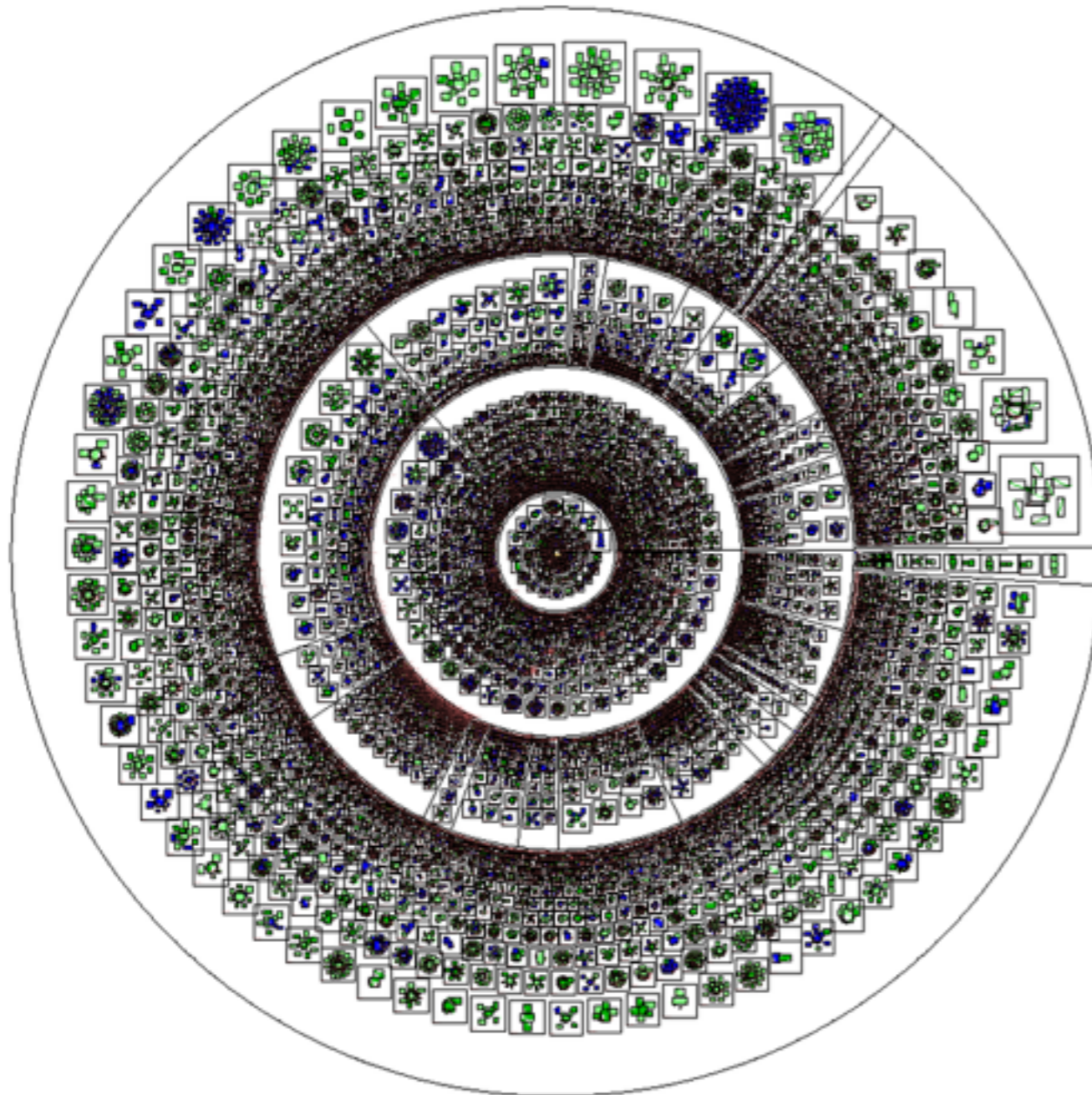
What does Kernel do?

- Kernel creates these concepts from the low-level hardware features:
 - Processes (time-sharing, protected address space)
 - Signals and semaphores
 - Virtual memory (swapping, paging, mapping)
 - The filesystem (files, directories, namespace)
 - General input/output (specialty hardware, keyboard, mouse, USB, etc.)
 - Interprocess communication (pipes and network connections)

Linux kernel in a diagram

Linux Kernel v2.6.11.8

"Woozy Beaver"



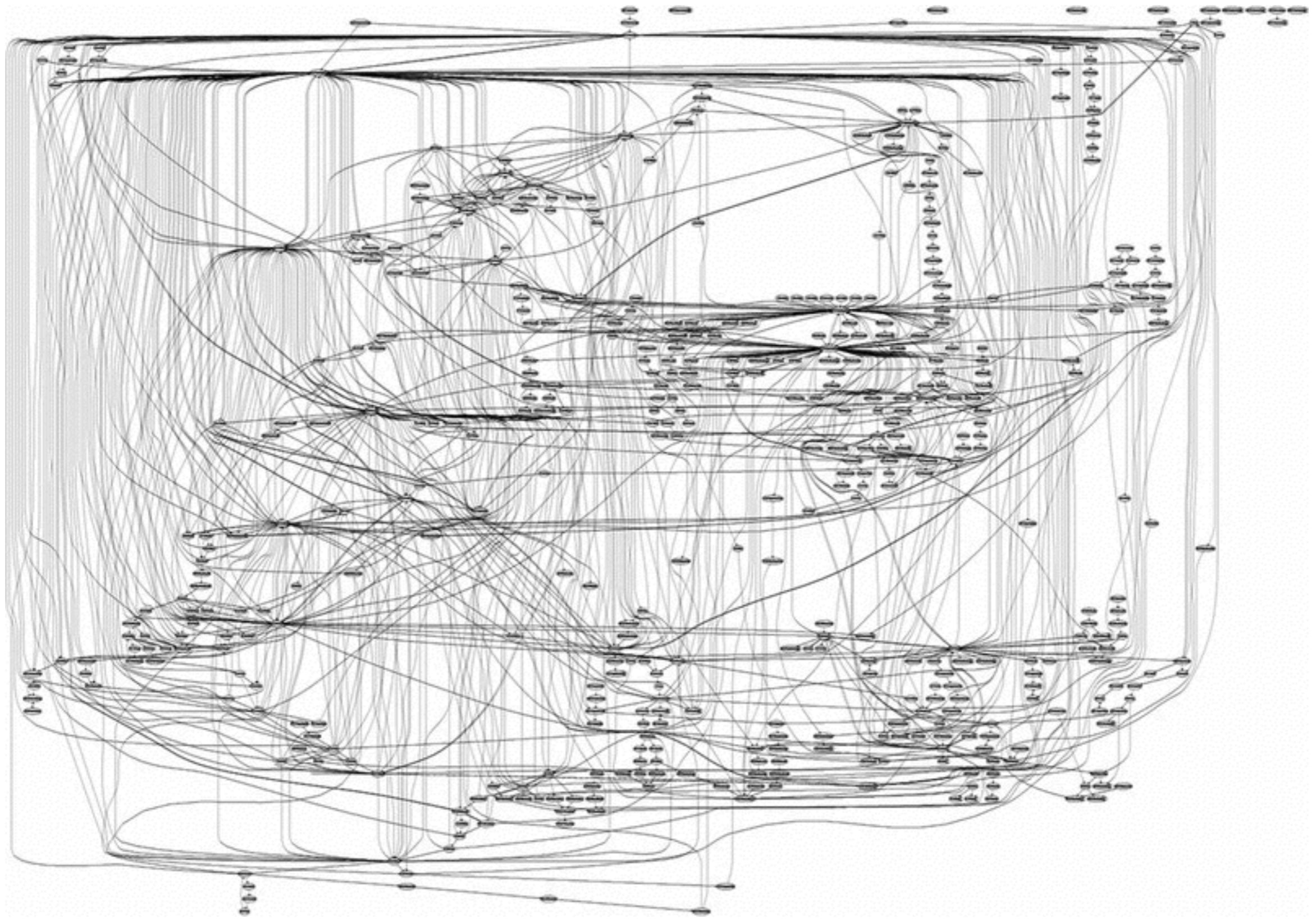
- 50% device driver
 - “Read the first 64 bytes of of /etc/passwd” —> “fetch block 3,348 from device 3”
- 25% CPU specific code
- Two inner layers are generic
- Mostly written in C + some assembly language to interface with hardware or chip specific functions



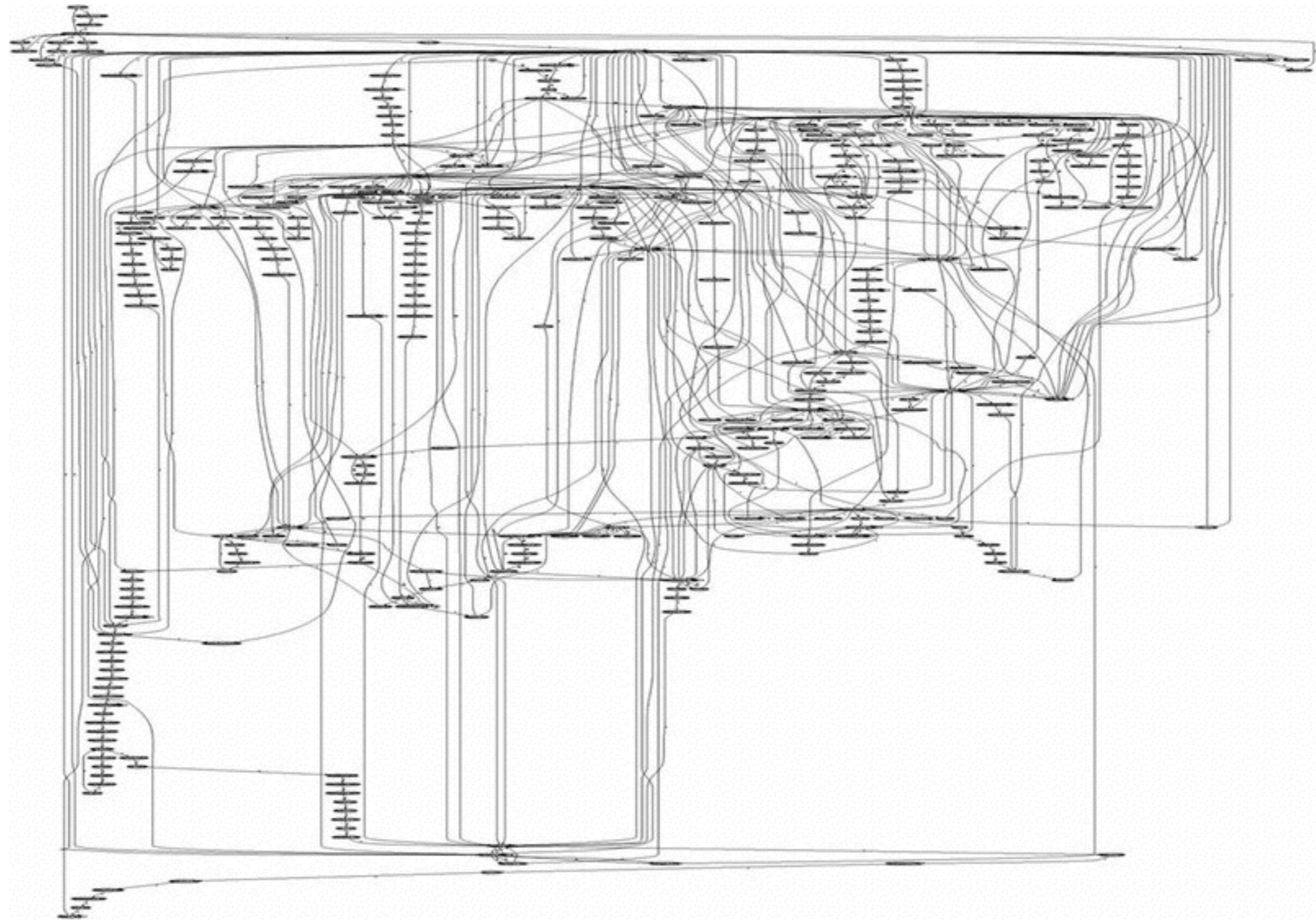
Advantages of Open Source Kernel

- Refactoring (smoothing, refining, simplifying, polishing) is done continuously in Linux
- Code is not freely available & in one place —> hard to evolve
 - Comparison: Microsoft's proprietary kernel
The need to keep the old version alive for old code/hardware
- Stanford research: Linux kernel has **0.17 bugs per 1,000 lines of code, 150 times less** than average commercial code containing 20-30 bugs per 1,000 lines.

Ref: http://keithcu.com/wordpress/?page_id=599



System call graph in Microsoft's proprietary web server, **IIS**.



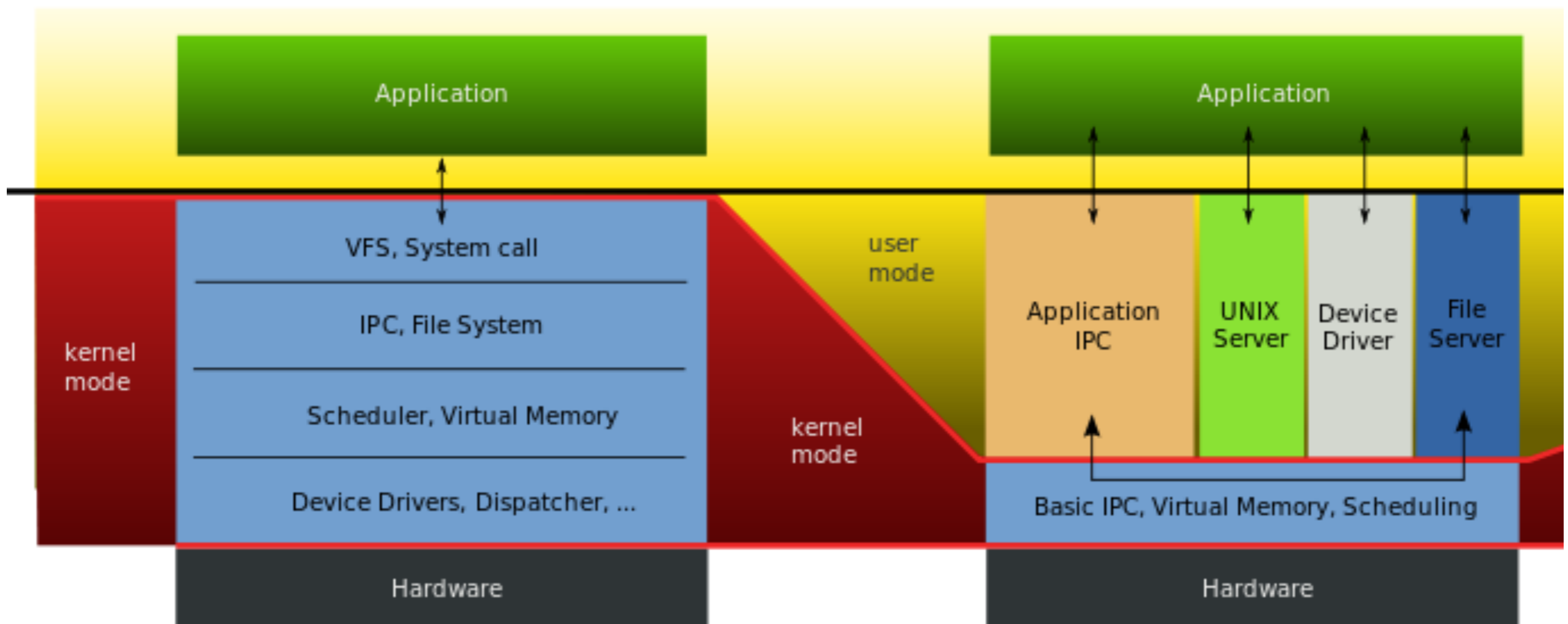
System call graph to return a picture in the free web server **Apache**.

Kernel Adaptation

- Linux is a monolithic kernel at heart
 - Monolithic kernel: the entire OS runs in **kernel space** - a section of memory reserved for privileged operating system functions (device drivers, IPC, virtual memory, scheduling all run in the same address space)
 - Microkernel: many of the “services” run in user mode as regular processes
- Modern monolithic kernels support on-demand loading of modules
 - No need to re-build the entire kernel & **reboot**
 - Example: filesystem and device drivers

Monolithic Kernel based Operating System

Microkernel based Operating System



Drivers and Device Files

- Device driver: manages the interaction between the system and the hardware
- User space access: from /dev. Kernel maps operations “on these files” to calls to the driver code.
- Major and minor device numbers - (use 'ls -l /dev' to see them) map device file references to drivers
- Block device - read or write one block (multiples of 512) at a time
- Character device - read or write one byte at a time
- Some of the device driver's functions
attach close dump ioctl open probe
prize read receive reset select stop
strategy timeout transmit write

“Phantom devices”

- `/dev/zero` & `/dev/null`:
data written here is discarded.
read from `/dev/zero` always returns 0.
read from `/dev/null` always returns EOF.
- `/dev/random` and `/dev/urandom`:
read from `/dev/random` will return random bytes.
(interface to kernel's random number generator)

Custom kernels v.s. loadable modules

- When installed, a system comes with a generic kernel
- Linux's udev system can manage real-time device changes
- Do we need custom-built kernels?
 - Pros: Opportunity for performance gain
 - Cons: Patch & system upgrade could be difficult
- For stability reasons:
using the stock kernel is recommended.

Kernel module related commands / files

- Files:
 - /vmlinuz: the actual kernel binary file
 - /lib/modules: kernel modules (separated by version)
- lsmod: list all kernel modules that have been loaded
- depmod: generate kernel module dependency file (modules.dep)
- modinfo: list information about a particular kernel module
- modprobe: load a kernel module (and its dependency)
- insmod, rmmod: manually load and unload kernel module

Initial ramdisk

- You might have seen this: `initrd.tbz`
- This is initial RAM disk (a disk in the memory)
- This is loaded before the root file system is mounted.
- Reason: some essential kernel modules might be needed for initial boot operation (storage, filesystem), but they are in `/lib/modules`
- Solution: load `initrd`, where these modules are stored.

Kernel Configuration

1. Modify tunable kernel configuration parameters

- Parameters can be adjusted via hooks in /proc (procfs)
- /proc/sys contains various special files for users to view and set kernel options
- Try it:
cat /proc/sys/fs/file-max
(maximum number of files the system can open at once)
sudo sh -c "echo <an integer> > /proc/sys/fs/file-max"
- Note that the changes are not carried across reboots

In-class HW today

See NASA In-class HW 2016/05/04 text file for detailed instructions.

1. PXEBoot Attempt #2:
boot Ubuntu 14.04.4 LTS without a disk
2. Compile your own kernel