Linux Kernel & Related Topics

Michael Tsai
2025/4/21

Agenda

- Before class:
  please take a look at various readings on NTU COOL
  - Software licenses
  - The U. Minnesota incident
- Lab 9: LDAP
  (Light-weight Directory Access Protocol)
- HW8 (LDAP) will be announced today.

What does Kernel do?

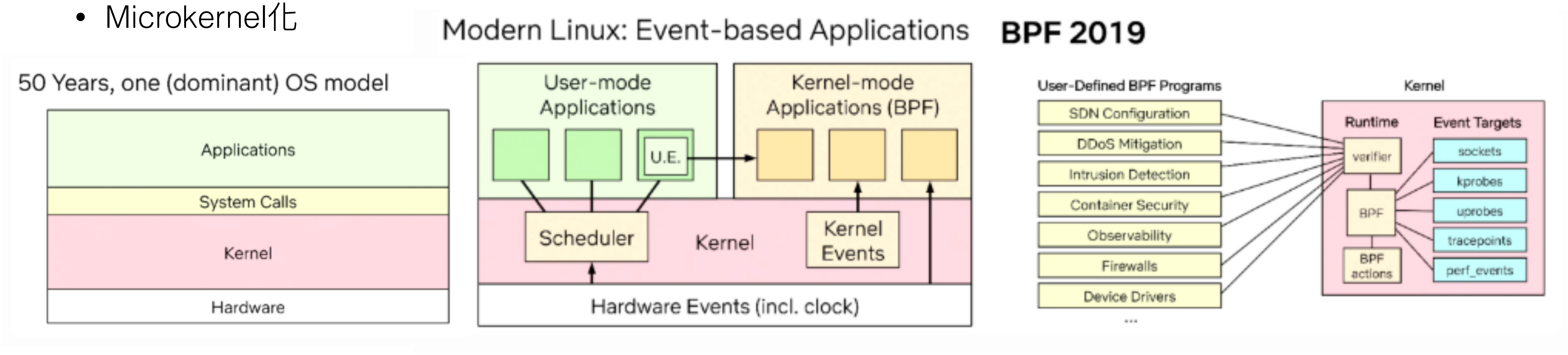
- Kernel creates these concepts from the low-level hardware features:
    - Processes (time-sharing, process address space)
    - Signals and semaphores
    - Virtual memory (swapping, paging, mapping)
    - The filesystem (files, directories, namespace)
    - General input/output (specialty hardware,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

Linux kernel in the past 20 years (FOSDEM 2020)

- SMP support from v2.0
  - Big kernel lock:
    lock_kernel() & unlock_kernel() - v2.4「暫時」的鎖定機制
  - BKL用於保護整個核心、spinlock保護特定的某共享資源
- Support for virtualization
  - KVM == Kernel-based Virtual Machine (AWS一開始用Xen, 後AWS & Google Cloud皆使用KVM)
  - x86架構非為虛擬化而生，因此軟體架構複雜
  - 直接把Linux轉為hypervisor，結合硬體的虛擬化支援
- Data Plane Development Kit
  - app-to-wire-to-app, TCP/UDP的延遲到達 6 us
  - 繞過作業系統的管理，使得應用程式可以直接操作、寫入網路介面緩衝區
    - 攔截libc呼叫、確認後把呼叫轉到使用者層級的TCP/IP stack
- Ref: jserv http://hackmd.io/@sysprog/linux-dev-review

Linux kernel in the past 20 years (FOSDEM 2020)

- Asynchronous I/O:
  - Async I/O: 從/往核心空間讀取或寫入資料的過程，交由核心內部處理。複製完資料後，再通知使用者的行程。這中間使用者行程不會被阻擋。
  - 更有效率、速度更快
- Container的支援：
  - Control groups (cgroups) —>
限制，控制與隔離 Process 所使用到的系統資源 [ref]
  - Namespaces —>
命名空間（Namespaces）是 Linux 核心的一項功能，它能夠將核心資源進行劃分，使得一組行程（processes）看到的是一組資源，而另一組行程則看到的是另一組不同的資源。
- Microkernel化

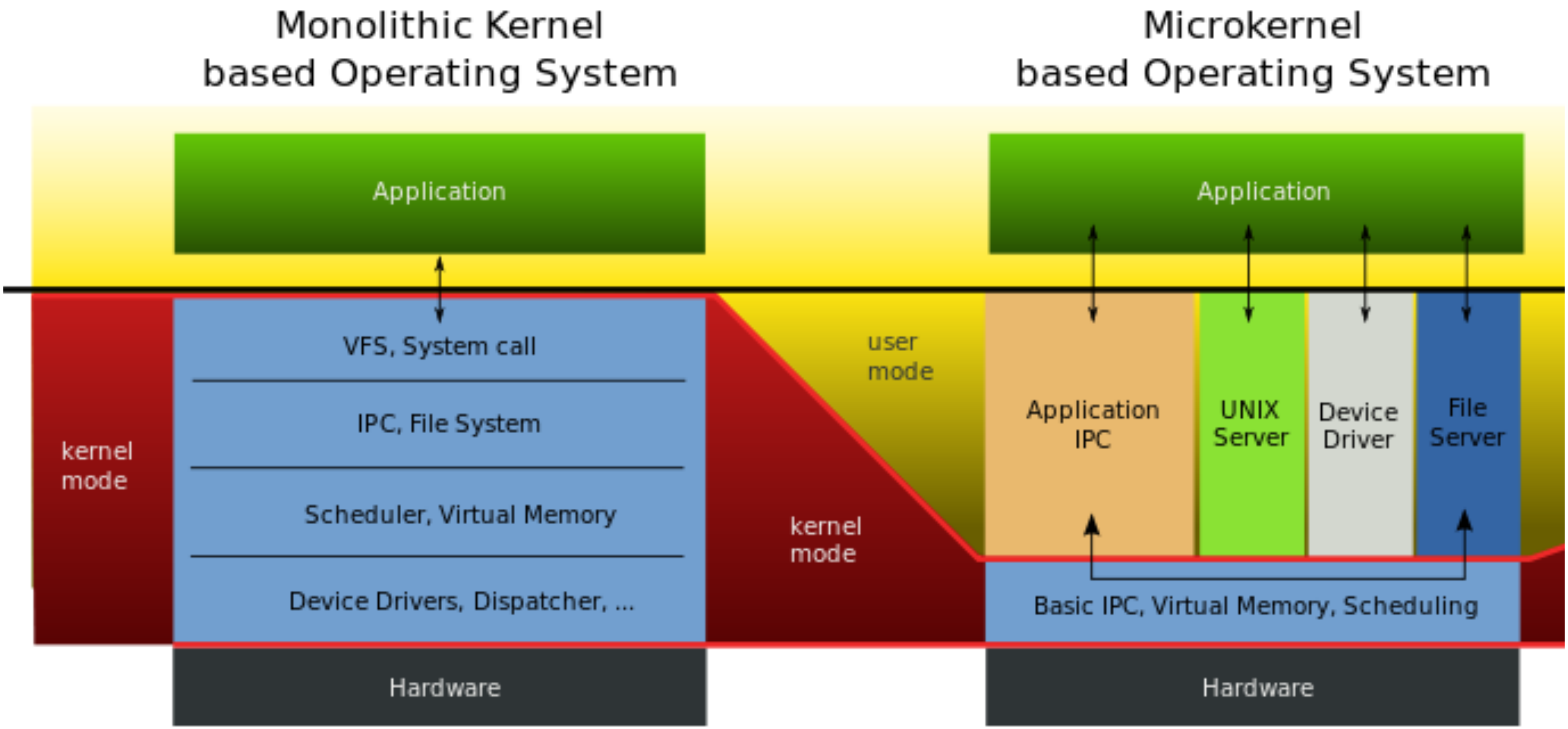


Kernel Adaptation

- Linux is a monolithic kernel at heart (at the beginning)
    - 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
- Modern OS'es (e.g., Windows, Mac OS X) use hybrid kernel
- Linux is moving toward having microkernel

https://en.wikiversity.org/wiki/Operating_Systems/Kernel_Models#Monolithic_Kernel

https://en.wikipedia.org/wiki/Tanenbaum–Torvalds_debate



http://en.wikipedia.org/wiki/Image:OS-structure.svg

GNU
General Public License (GPL)

- Find online about GPL, L-GPL & BSD licenses. What are their major differences?

- Since version 0.12, Linux kernel is released under GPL. How has the use of GPL helped the development of Linux?

- Choose a license
  https://choosealicense.com/licenses/

- Read:
  GPL開源(Open Source)許可證的風險與感染性

- (20 minutes)

University of Minnesota Banned from Linux Kernel Contributions (2021)

- **Research Experiment:** UMN researchers submitted patches with intentional vulnerabilities to the Linux kernel to study the feasibility of introducing stealthy bugs into open-source software. An IEEE S&P 2021 paper was accepted based on the research.
- **Community Reaction:** The Linux kernel maintainers were unaware of the experiment and felt their trust was violated, leading to significant backlash.
- **Consequences:** Greg Kroah-Hartman, a senior Linux kernel maintainer, banned UMN from contributing to the Linux kernel. All previous contributions from UMN were also removed and re-evaluated.
- **Apology Issued:** UMN researchers apologized, acknowledging the lack of prior consultation with the Linux community and the unintended harm caused. The authors retract the paper from IEEE S&P 2021.
- **Ethical Discussions:** The incident sparked broader discussions about ethics in open-source research and the responsibilities of contributors.

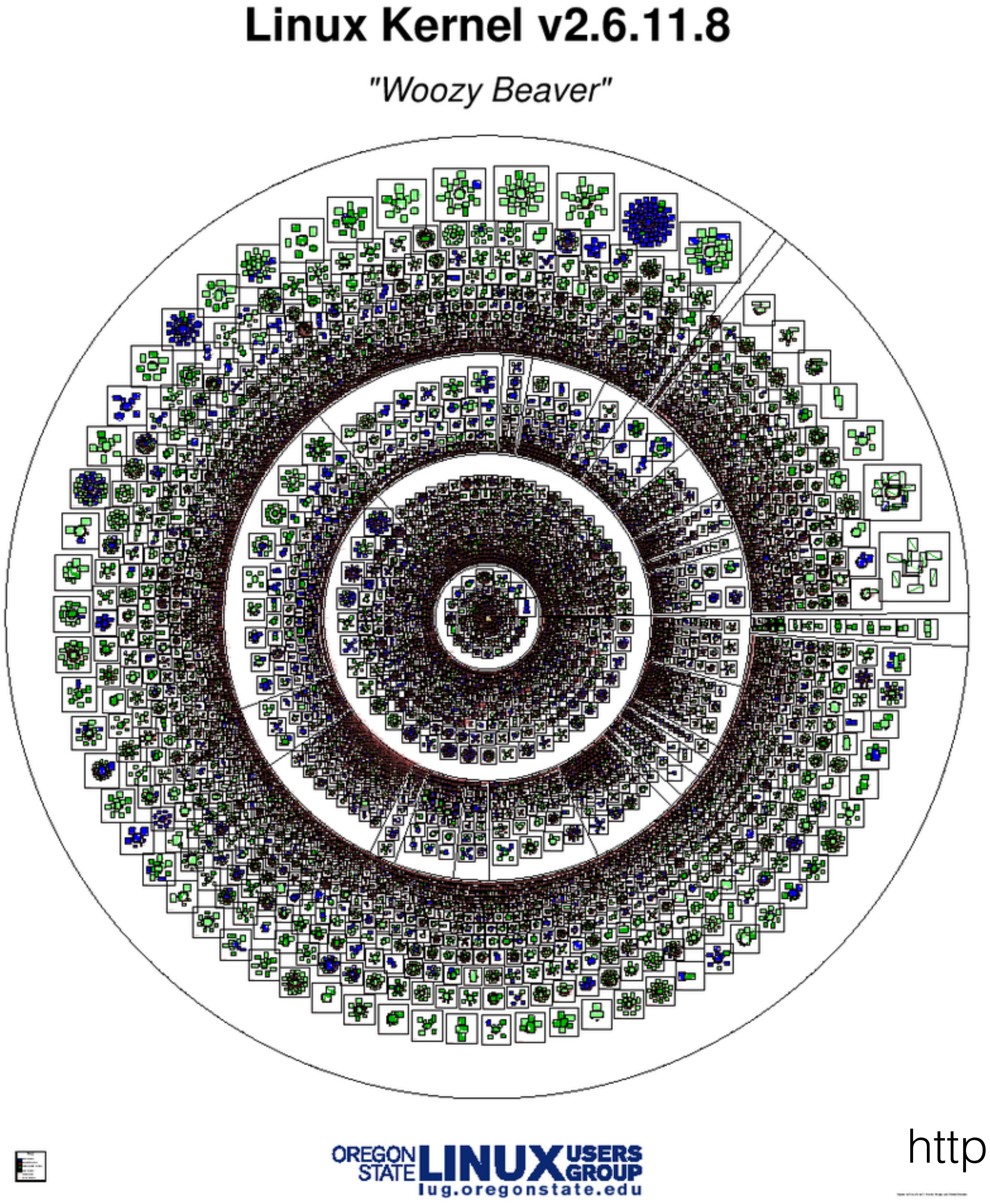
Recommend reading

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

- (Linux “chapter”, “After the Software Wars,” Keith Curtis, 2010)

- http://keithcu.com/SoftwareWars.pdf

Linux kernel in a diagram



- 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

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

Advantages of Open Source Kernel

- Refactoring (smoothing, refining, simplifying, polishing) is done continuously in Linux

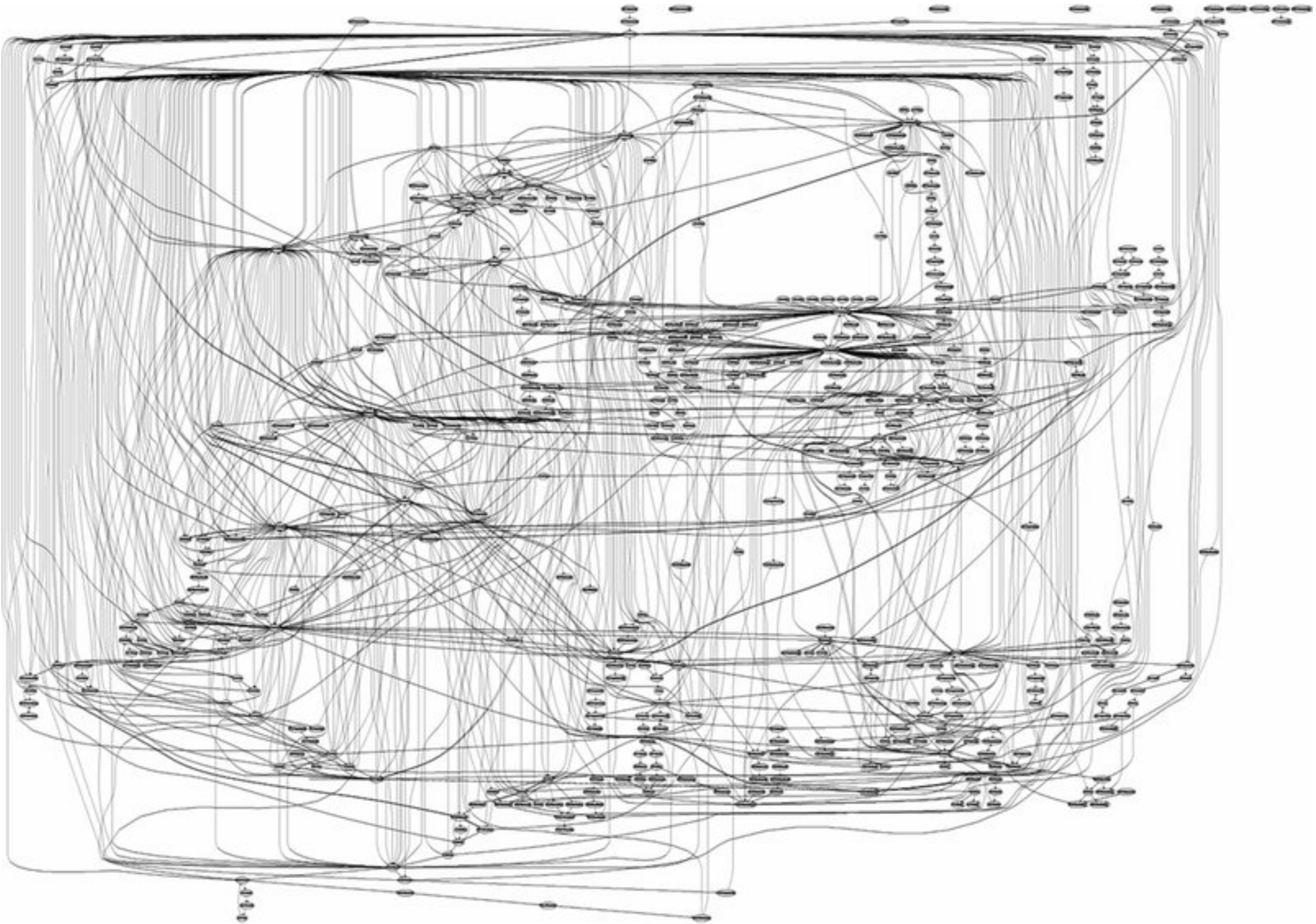
- Code is not freely available & in one place —> hard to evolve

  - Comparison: Microsoft's proprietary kernel — The need to keep the old version alive for old code/hardware

  - Example: If many drivers have similar tasks, duplicate logic can be pulled out and put into a new subsystem that can then be used by all drivers.

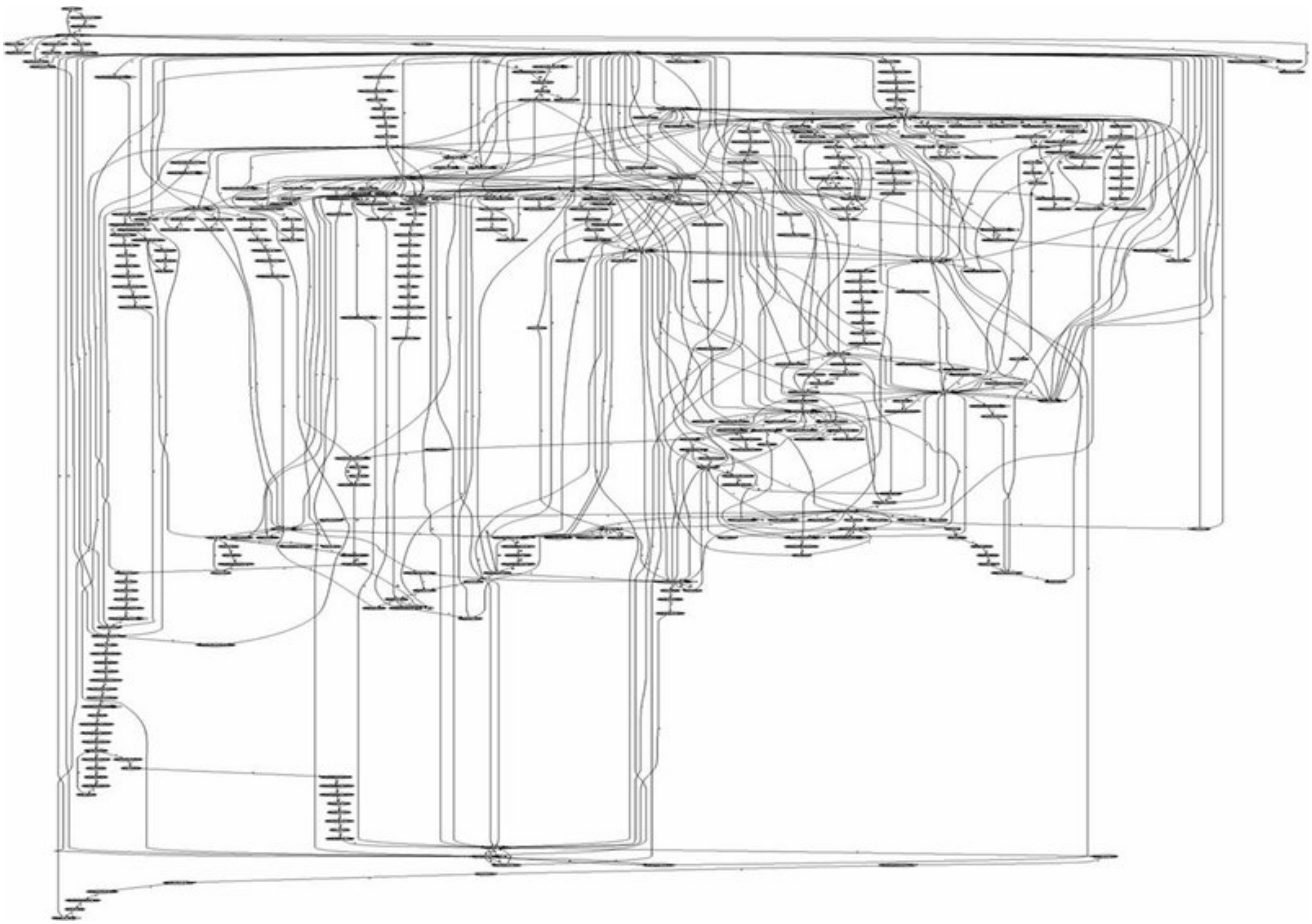
- Stanford research: Linux kernel has **0.17 bugs per 1,000 lines of code**, **150 times less** than average commercial code containing 20-30 bugs per 1,000 lines.

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



https://ma.ttias.be/system-calls-in-apache-linux-vs-iis-windows/

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



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

In-class: learn “screen” or “tmux”

- Allow you to have multiple “tabs” in terminal window
- Ctrl-a c to create a new “tab” (screen)
- Ctrl-a <number> to switch to that window (screen)
- Ctrl-a d to detach (screen)
- Re-attach using “screen -r”; useful when waiting for results
- tmux has different details but similar usage
- Use the workstation in the department or your VM to practice!

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

When to upgrade the kernel?

- Resist the temptation to keep up with the latest version
- Weigh between needs & risks
- Good rule of thumb:
  Upgrade or apply patches only when expected productivity gains is larger than the effort to install.