Instigate Design

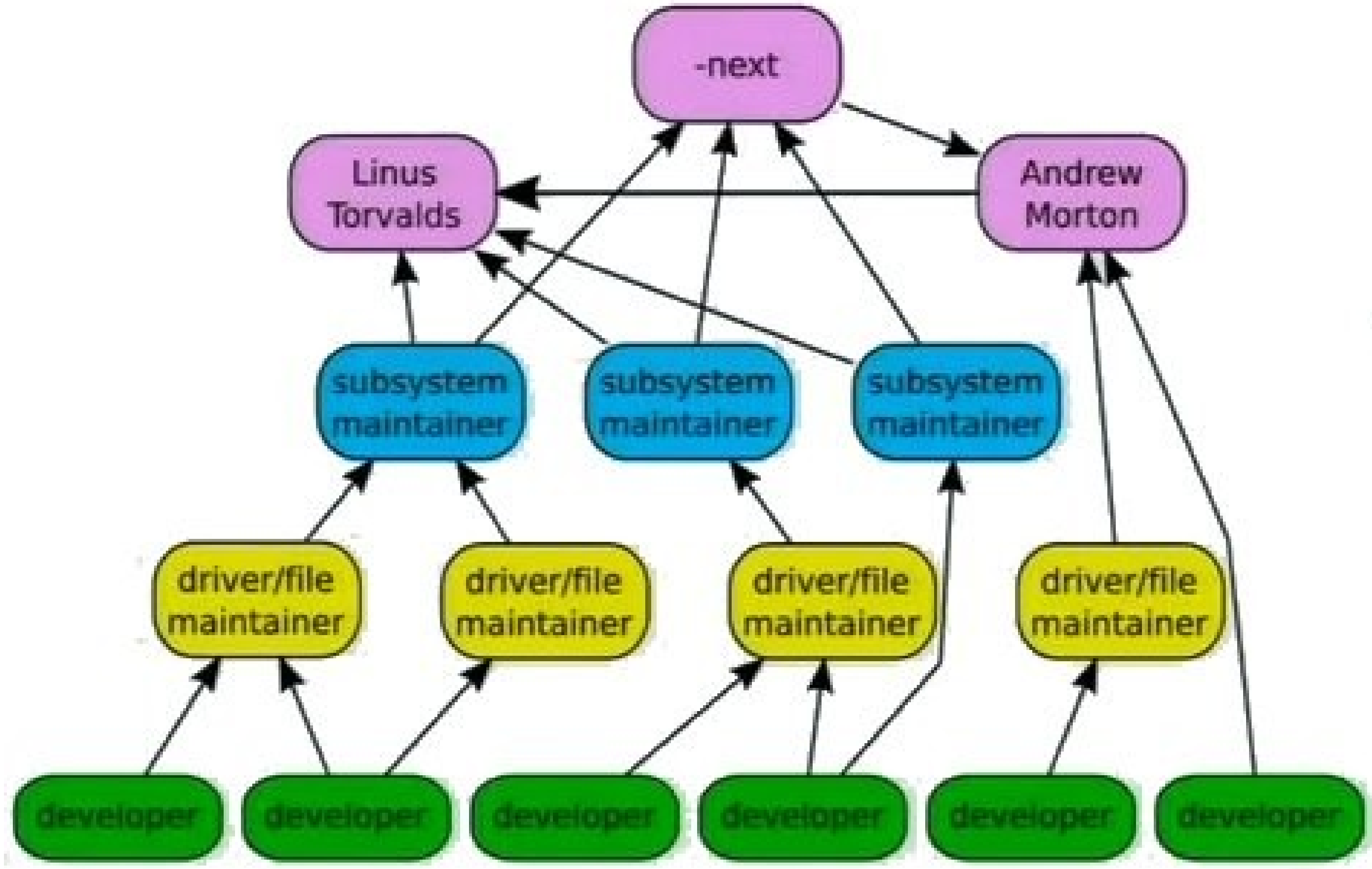
Parallel Systems Development

Linux kernel

instigate-training-center.am

Who is doing and how it is being done?

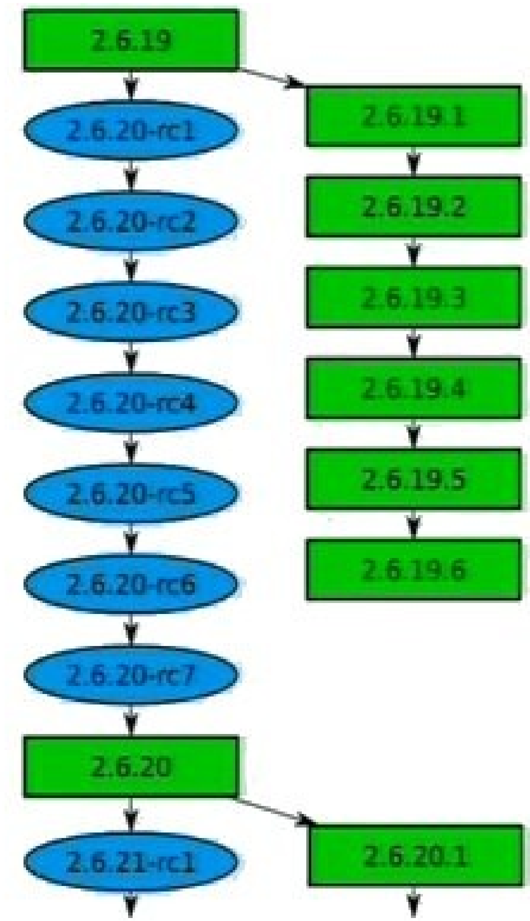
- Fastest moving project known to everybody (during 2007-2008, per day)
  - 4,300 lines added
  - 1,800 lines removed
  - 1,500 lines modified
- By 2008
  - 9.2 million lines of code
  - 2399 developers



Instigate Training Center

Releases

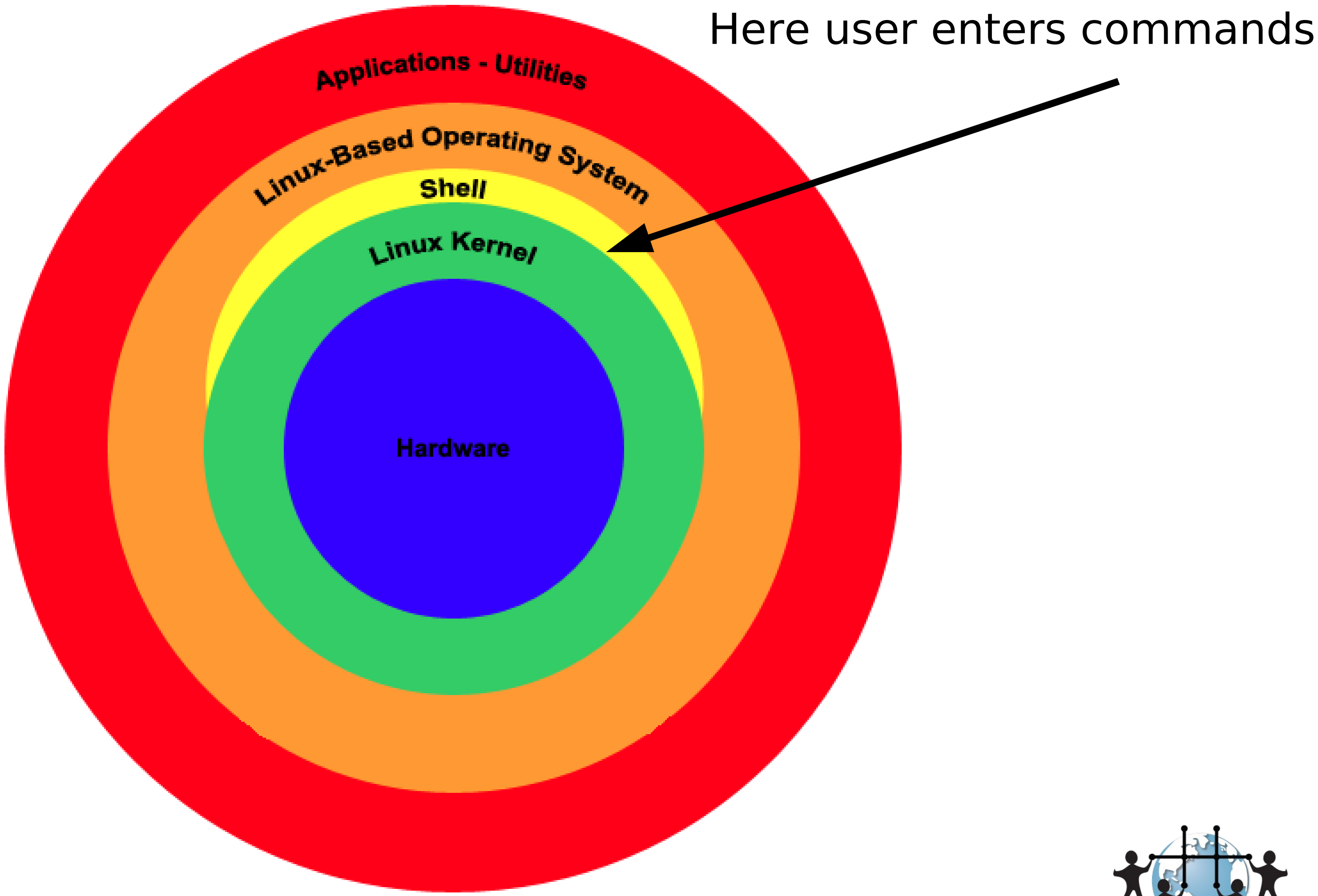
- New enhancements and bug fixes in rc-s
- Support of the previous version
- Important is to not break kernel for existing arcl
- The main testing is done by the users
- The patches done on old release should be done also on the upstream tree
- New release every 2 ¾ months



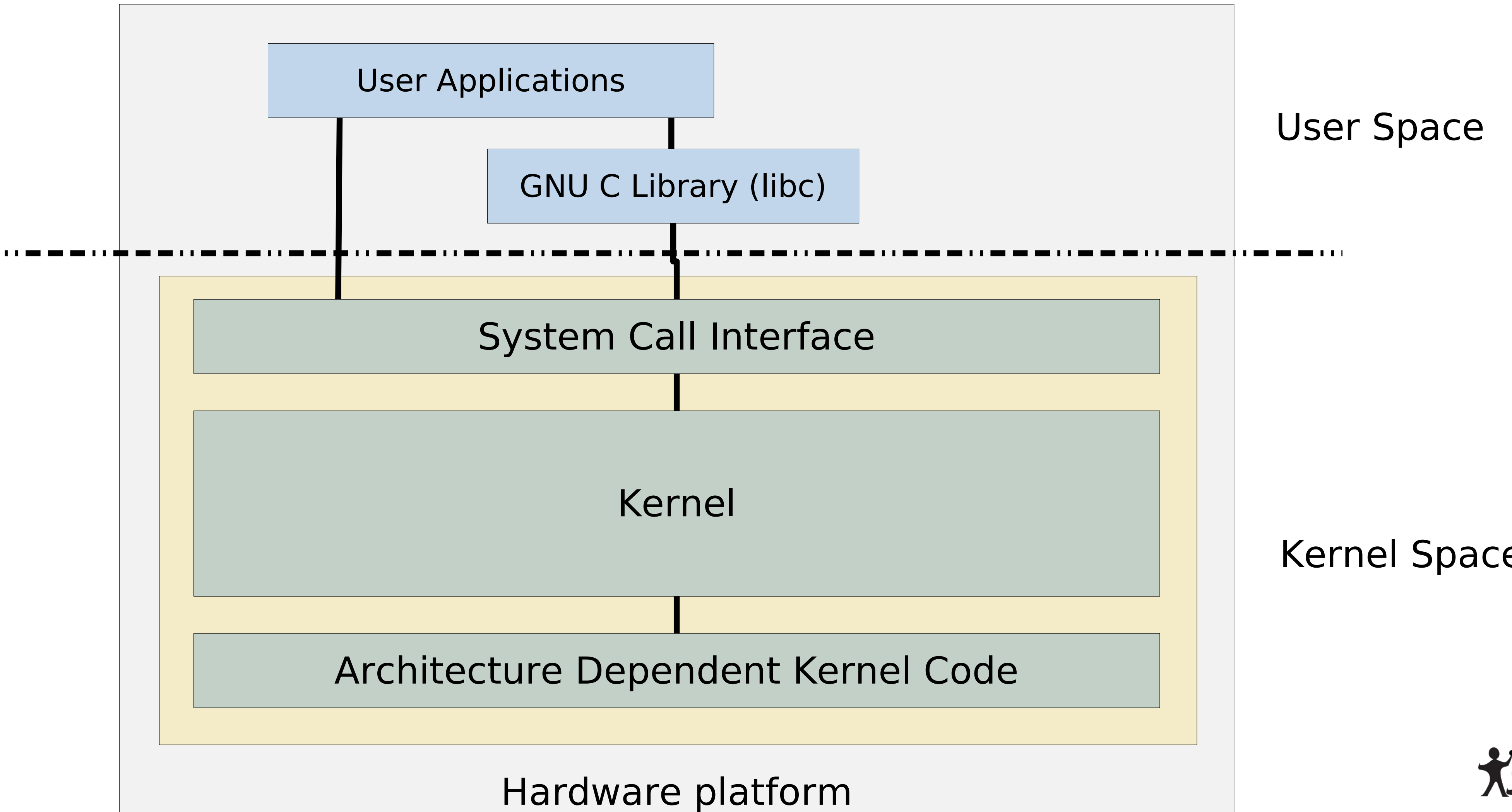
Architecture

Important parts

- Kernel
- Shell
- Graphical subsystem X
- Network subsystem

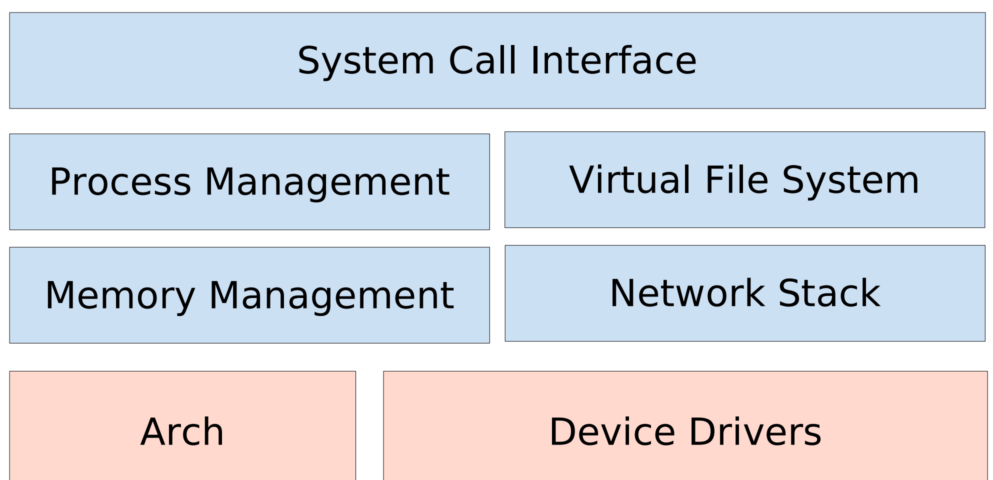


Kernel Architecture



User and Kernel Space Components

- User applications
- Glibc – provides the system call interface that connects to the kernel
- System call interface – provides the basic functions (read and write)
- Kernel
  - Architecture-independent kernel code common to all processor architectures
  - Architecture-Dependent Code - processor and platform specific code
- Kernel Subsystems

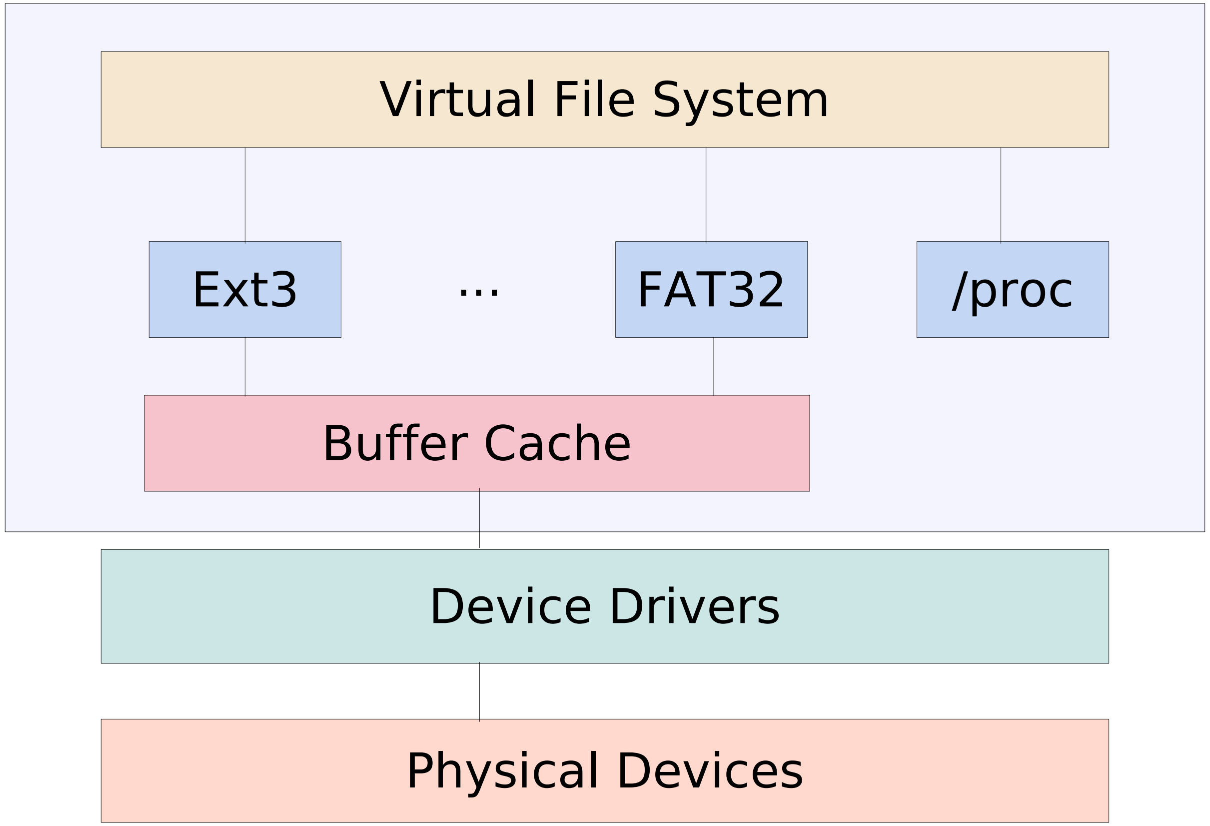


Kernel Subsystems

- System Call Interface
  - Provides the means to perform function calls from user space into the kernel
- Process Management
  - Focused on the execution of the processes
  - Each has an individual virtualization of the processor (data, stack, registers)
  - Kernel provides API to start start, stop and communicate with processes
  - Kernel implements constant time scheduling algorithm regardless the # of threads
- Memory Management
  - Memory is managed in pages (typically 4KB per page)
  - Keeps tracks of which pages are full, partially used, empty,
  - Swaps to disk if physical memory runs out

Virtual File System

- Virtual File system
  - Presents a common API (open, close, read, write)
  - Support over 50 different file systems
- Buffer Cache - Optimizes access to the physical devices by caching
- Network Stack
  - Managing connections
  - Moving data between endpoints
- Device Drivers – most of the kernel source code

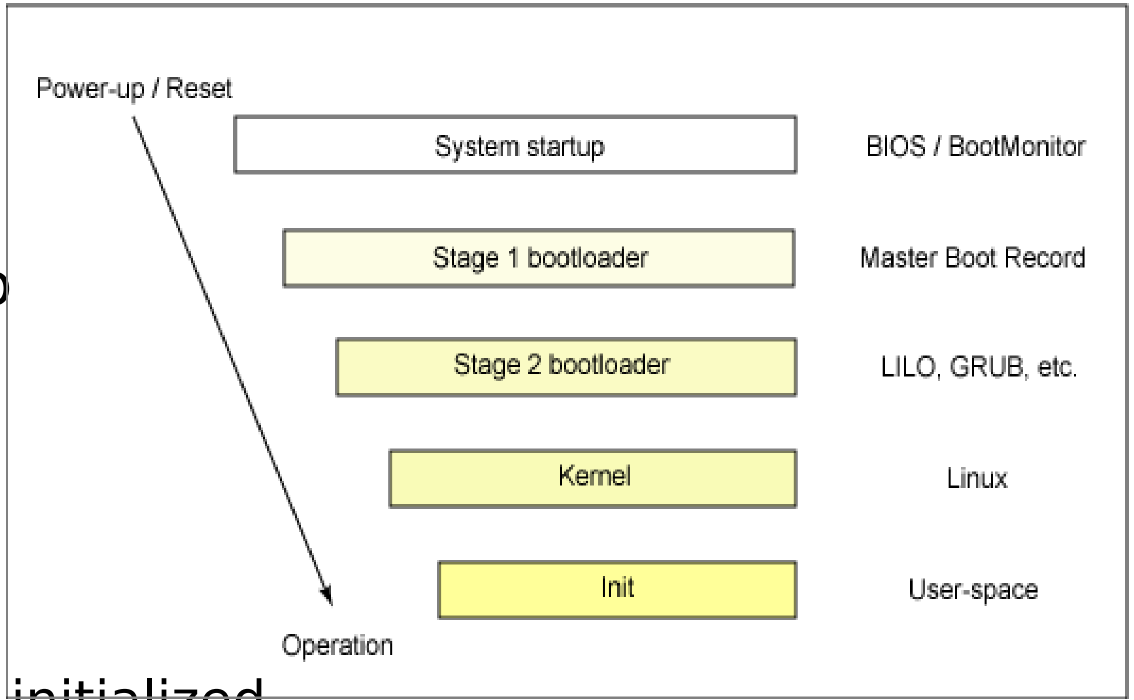


Modularity

- Many components of the Linux kernel may be compiled as modules which the kernel can dynamically load and remove as required.
- The best components to modularize are ones not required at boot time, for example peripheral devices and supplementary file systems.
  - lsmod - list currently loaded modules
  - rmmod - remove a single module
  - insmod - insert a single module
  - depmod - create the database of module dependencies modules.dep
  - modprobe - insert a module and dependencies listed in modules.dep
  - modinfo - list information about the author, license type and module parameters

Booting

- Processor executes code at a well-known location – BIOS, located in the flash memory of MB
  - BIOS determines devices to boot
- First-stage boot loader is loaded into RAM and executed (< 512 bytes)
  - Loads the second-stage boot loader
- Second-stage boot loader is loaded into RAM and executed
  - Passes control to the kernel image and the kernel is decompressed and initialized
- Kernel - checks the system hardware, enumerates the attached hardware devices, mounts the root device, loads the necessary kernel modules and starts init.



Instigate
Training Center

Building Linux Kernel

- Why upgrade or build your own kernel?
  - You want to optimize a kernel (for speed, size, etc.)
  - Optimize for security with extra enhancements (e.g. remove modules support, remove networking support, limit drivers)
  - You want to patch your kernel to use non-standard features.
  - You are adding a new hardware to machine not supported under current kernel

References

- Architecture of the Linux Kernel
- Greg Kroah Hartman on the Linux Kernel
- Booting process