LPIC1 Exercises

Week 1 - Videos 01 - 16

Nika Soltani Tehrani

Spring 2025

1. **Please summarize the following server components:**
   - **Motherboard (System Board)**
     Motherboard means specifically a PCB with expansion capabilities. As the name suggests, this board is often referred to as the mother of all components attached to it, which often include peripherals, interface cards, and daughterboards: sound cards, video cards, network cards, host bus adapters, TV tuner cards, and a variety of other custom components.
     Unlike a backplane, a motherboard usually contains significant sub-systems, such as the central processor, the chipset's input/output and memory controllers, interface connectors, and other components integrated for general use.

   - **CPU (Central Processing Unit)**
     CPU is the primary processor in a given computer. Its electronic circuitry executes instructions of a computer program, such as arithmetic, logic, controlling, and input/output (I/O) operations.
   - **RAM (Memory)**
     RAM is a form of electronic computer memory that can be read and changed in any order, typically used to store working data and machine code. A random-access memory device allows data items to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory, in contrast with other direct-access data storage media (such as hard disks and magnetic tape), where the time required to read and write data items varies significantly depending on their physical locations on the recording medium, due to mechanical limitations such as media rotation speeds and arm movement.
   - **Storage Drives (HDD/SSD/NVMe)**
     HDD:
     HDD is an electro-mechanical data storage device that stores and retrieves digital data using magnetic storage with one or more rigid rapidly rotating platters coated with magnetic material. The platters are paired with magnetic heads, usually arranged on a moving actuator arm, which read and write data to the platter surfaces. Data is accessed in a random-access manner, meaning that individual blocks of data can be stored and retrieved in any order. HDDs are a type of non-volatile storage, retaining stored data when powered off.
     SSD:
     SSDs rely on non-volatile memory, typically NAND flash, to store data in memory cells. Unlike traditional hard disk drives (HDDs), SSDs have no moving parts, allowing them to deliver faster data access speeds, reduced latency, increased resistance to physical shock, lower power consumption, and silent operation.
     NVMe:
     NVM Express (NVMe) or Non-Volatile Memory Host Controller Interface Specification (NVMHCIS) is an open, logical-device interface specification for accessing a computer's non-volatile storage media usually attached via the PCI Express bus. The initial NVM stands for non-volatile memory, which is often

NAND flash memory that comes in several physical form factors, including solid-state drives (SSDs), PCIe add-in cards, and M.2 cards, the successor to mSATA cards. NVM Express, as a logical-device interface, has been designed to capitalize on the low latency and internal parallelism of solid-state storage devices.

- **RAID Controller (Smart Array)**
  A disk array controller is a device that manages the physical disk drives and presents them to the computer as logical units. It often implements hardware RAID, thus it is sometimes referred to as RAID controller. It also often provides additional disk cache.
- **Power Supply Unit (PSU)**
  A power supply unit (PSU) converts mains AC to low-voltage regulated DC power for the internal components of a desktop computer. Modern personal computers universally use switched-mode power supplies. Some power supplies have a manual switch for selecting input voltage, while others automatically adapt to the main voltage.
- **Network Interface Card (NIC)**
  NIC is a computer hardware component that connects a computer to a computer network. Early network interface controllers were commonly implemented on expansion cards that plugged into a computer bus. The low cost and ubiquity of the Ethernet standard means that most newer computers have a network interface built into the motherboard, or is contained into a USB-connected dongle.
  Modern network interface controllers offer advanced features such as interrupt and DMA interfaces to the host processors, support for multiple receive and transmit queues, partitioning into multiple logical interfaces, and on-controller network traffic processing such as the TCP offload engine.
- **Cooling System (Fans and Heat Sinks)**
  Computer cooling is required to remove the waste heat produced by computer components, to keep components within permissible operating temperature limits. Components that are susceptible to temporary malfunction or permanent failure if overheated include integrated circuits such as central processing units (CPUs), chipsets, graphics cards, hard disk drives, and solid state drives.
  Components are often designed to generate as little heat as possible, and computers and operating systems may be designed to reduce power consumption and consequent heating according to workload, but more heat may still be produced than can be removed without attention to cooling. Use of heatsinks cooled by airflow reduces the temperature rise produced by a given amount of heat. Attention to patterns of airflow can prevent the development of hotspots. Computer fans are widely used along with heatsink fans to reduce temperature by actively exhausting hot air. There are also other cooling techniques, such as liquid cooling. All modern day processors are designed to cut out or reduce their voltage or clock speed if the internal temperature of the processor exceeds a specified limit. This is generally known as Thermal

Throttling in the case of reduction of clock speeds, or Thermal Shutdown in the case of a complete shutdown of the device or system.

- **Expansion Slots (PCIe)**
  PCI-E is a high-speed serial computer expansion bus standard, meant to replace the older PCI, PCI-X and AGP bus standards. It is the common motherboard interface for personal computers' graphics cards, capture cards, sound cards, hard disk drive host adapters, SSDs, Wi-Fi, and Ethernet hardware connections. The PCI Express electrical interface is measured by the number of simultaneous lanes.
- **Chassis (Rack/Tower/Blade)**

  Rack Server
  A rack server is a versatile server housed within a rack. Typically, rack servers are designed to accommodate various applications and serve diverse computing infrastructure needs. Rack-mounted servers are designed according to uniform standards, usually installed in a rack, and can be stacked and placed in a metal casing for easy management.
  Blade Server
  A blade server is an enclosure hosting multiple modular circuit boards known as server blades. These blades typically encompass essential components like the server CPU, memory, and network controls. Although primarily focused on these modular elements, some blade servers also incorporate storage drives, facilitating network storage functionalities like SAN or NAS devices.
  Tower Server
  Tower servers are essentially integrated desktop computers with a substantial footprint. As a result, they boast higher CPU power, fast read memory, and are designed to manage multi-user requests efficiently. Additionally, they offer various services, including DHCP or DNS.
- **BIOS/UEFI Firmware**
  Firmware is a basic software that enables the connection between a hardware and motherboard's software. Both BIOS and UEFI are linux firmwares, but BIOS stands for Basic Input Output System is the older version, and It is stored on an EPROM (Erasable Programmable Read-Only Memory), allowing the

manufacturer to push out updates easily. BIOS provides many helper functions that allow reading boot sectors of attached storage and printing things on screen.
UEFI stands for Unified Extensible Firmware Interface is the newer version, and it stores all data about initialization and startup in an .efi file, instead of storing it on the firmware. This .efi file is stored on a special partition called EFI System Partition (ESP) on the hard disk. This ESP partition also contains the bootloader.

- **Backplane**
  A backplane or backplane system is a group of electrical connectors in parallel with each other, so that each pin of each connector is linked to the same relative pin of all the other connectors, forming a computer bus. It is used to connect several printed circuit boards together to make up a complete computer system. Backplanes commonly use a printed circuit board, but wire-wrapped backplanes have also been used in minicomputers and high-reliability applications.
  A backplane is generally differentiated from a motherboard by the lack of on-board processing and storage elements. A backplane uses plug-in cards for storage and processing.

2. What are IPMI and iLO, and what are their functions?
IPMI, iDRAC, and iLO all provide essentially the same functionality, which one is active in your dedicated server depends on the manufacturer of your dedicated server hardware and motherboard. iDRAC is how Dell brands this feature for its dedicated servers, HP brands it as iLO, and SuperMicro brands it as IPMI. They all are a way for a server administrator to connect to a server and launch a remote KVM session or remotely control the server's power and do a reboot or stop/start the server's power. There are no real security differences between the three, the remote console for each operates via a Java pop-up so it is necessary for the server administrator to have the latest version of Java installed on their workstation when they want to use the remote KVM function. It is also important to note that with iDRAC (from Dell) and iLO (from HP) there might be licensing fees that need to be paid, whereas with IPMI (from SuperMicro) there currently is no additional licensing fees that need to be paid.

3. How do IPMI or iLO relate to the BIOS or UEFI firmware?

IPMI and iLO are technologies that enable out-of-band management of servers, typically through a dedicated controller known as the BMC (Baseboard Management Controller). BIOS and UEFI are types of firmware responsible for initializing hardware and bootstrapping the operating system when a server powers on.

How IPMI/iLO Interact with BIOS/UEFI

- Communication Channel:
    - The BIOS or UEFI firmware communicates with the BMC (which implements IPMI or iLO functionality) via specific interfaces, such as the KCS (Keyboard Controller Style) interface. This allows the BIOS/UEFI to send commands and receive information from the BMC during the boot process.
- Data Exchange:
    - During server initialization, the BIOS/UEFI may send hardware status, configuration, or operational state information to the BMC using IPMI commands. For example, the BIOS can report its version and operational status to the BMC, and receive configuration data or commands that may affect system behavior.
- Remote Management:
    - IPMI and iLO enable administrators to monitor, control, and manage servers remotely, even when the main operating system is not running. This includes functions like power cycling, hardware monitoring, and remote BIOS/UEFI configuration. For instance, iLO provides a RESTful API that exposes UEFI BIOS settings for remote management and configuration.
- Firmware Upgrades:
    - The BMC firmware (IPMI/iLO) can be upgraded independently of the BIOS/UEFI. However, the BIOS/UEFI can initiate or assist in BMC firmware upgrades through IPMI commands, and the BMC can report its firmware status back to the BIOS/UEFI.

Example Interactions

- During Boot:
    - When a server powers on, the BIOS/UEFI checks the status of the BMC via the IPMI protocol. If the link is normal, the BIOS/UEFI and BMC exchange necessary data (such as hardware inventory or configuration settings) before proceeding to boot the operating system.
- Error Handling:
    - If the BIOS/UEFI cannot communicate with the BMC (e.g., due to a faulty IPMI link), it may retry communication, log errors, or take corrective actions such as rebooting or reporting the failure via other channels.
- Configuration Management:
    - With iLO, administrators can remotely read or modify UEFI BIOS settings through a RESTful API, enabling automated or remote configuration of BIOS parameters without direct physical access to the server.

4. What are CPU sockets on a server, and what is their purpose?
A CPU socket is a physical component on a server's motherboard designed to securely hold and connect a central processing unit (CPU) to the rest of the system. It provides both the mechanical support and the electrical interface necessary for the CPU to communicate with the motherboard and, by extension, with other hardware components in the server.
The CPU socket contains a dense array of pins or contact points that facilitate the transfer of power, data, and control signals between the CPU and the motherboard. This connection allows the CPU to execute instructions, process data, and interact with memory, storage, and peripheral devices.
By using a socket instead of soldering the CPU directly onto the motherboard, servers can have their CPUs replaced or upgraded without replacing the entire motherboard. This is especially valuable in data centers and enterprise environments where hardware flexibility and maintenance are crucial.

5. Why was the pseudo file system introduced in Linux?
The pseudo file system was introduced in Linux to provide a standardized, hierarchical interface for accessing and interacting with dynamic kernel and system information as if it were regular files, even though this information does not reside on a physical disk.
Pseudo file systems like /proc and /sys allow users and applications to access real-time information about the running kernel, processes, hardware, and system configuration using familiar file operations (such as cat, ls, and echo). This makes system introspection and configuration straightforward and scriptable.

6. What are the differences between a pseudo file system and a normal file system?

- Nature of Files:
    - Pseudo file systems present virtual files that represent system or kernel information, while normal file systems manage real files stored on disk.
- Persistence:
    - Pseudo file system entries are temporary and exist only during system runtime; normal file systems retain data across reboots.
- Function:
    - Pseudo file systems facilitate access to operating system internals and hardware interfaces, whereas normal file systems are designed for general data storage and retrieval.
- Interaction:
    - Both types can be accessed using standard file operations (read, write, open, etc.), but the underlying data source and behavior differ significantly.

7. What kind of information is available in the /sys/ directory?
The /sys/ directory in Linux is a virtual file system (sysfs) that provides a structured, hierarchical interface to kernel data structures, offering detailed information about the system's hardware, devices, drivers, and kernel parameters. The information available in /sys/ includes:

- Hardware Devices:

    - Details about all hardware devices detected by the kernel, such as CPUs, memory, storage devices, network interfaces, USB devices, and more.
- Device Drivers:
    - Information about drivers currently loaded and associated with hardware devices, found in directories like /sys/bus/ and /sys/class/.
- Kernel Parameters and Status:
    - Real-time kernel parameters and status information, including power management (/sys/power/), kernel modules (/sys/module/), and kernel configuration (/sys/kernel/).
- Device Attributes and Configuration:
    - Many files in /sys/ allow you to read and sometimes write configuration settings for hardware devices and kernel features, such as CPU frequency scaling, device power states, and device-specific parameters.
- System Firmware:
    - Information about system firmware, such as BIOS or UEFI details, found in /sys/firmware/.
- Tracing and Debugging:
    - Facilities for kernel tracing and debugging, accessible via /sys/tracing/.

All entries in /sys/ are dynamically generated by the kernel and reflect the current state of the system. They do not represent actual files on disk but are instead views into the kernel's internal data structures, providing a powerful mechanism for system inspection and management.

8. What is DMA (Direct Memory Access), and what is its use case in Linux?

DMA is a feature in computer systems that allows certain hardware devices, such as disk drives, network cards, graphics cards, and sound cards, to transfer data directly to or from the main system memory (RAM) without requiring continuous intervention by the CPU.

Instead of the CPU managing every byte of data during a transfer (as in programmed I/O), DMA enables the CPU to initiate a transfer and then continue with other tasks while the DMA controller handles the actual movement of data. Once the transfer is complete, the DMA controller typically notifies the CPU via an interrupt.

Use Case in Linux

In Linux, DMA is widely used to improve system performance and efficiency, especially for high-speed or high-volume data transfers. Common use cases include:

- Disk I/O: Hard drives and SSDs use DMA to transfer large blocks of data to and from memory, reducing CPU load and speeding up file operations.
- Network Operations: Network cards leverage DMA to move packets directly between memory and the network interface, allowing for fast data transmission and reception with minimal CPU overhead.
- Graphics and Multimedia: Graphics cards and sound cards use DMA to stream data efficiently, which is critical for video playback, gaming, and audio processing.
- Memory-to-Memory Transfers: DMA can also be used for copying or moving large blocks of data within memory, offloading these tasks from the CPU.

DMA is essential in Linux for any scenario where large or frequent data transfers would otherwise monopolize the CPU, enabling multitasking and higher overall system throughput.

9. What does the lsblk command do internally when executed in Linux? Do lsusb, lspci, and lshw function similarly?
When you run the lsblk command in Linux, it lists information about all available block devices (such as hard drives, SSDs, and their partitions) in a tree-like format. Internally, lsblk gathers its information primarily by reading from the sysfs virtual file system, specifically from /sys/class/block, and also consults the udev database for additional device details. If the udev database is unavailable or lsblk is compiled without udev support, it may attempt to read labels, UUIDs, and filesystem types directly from the block devices, which may require root permissions. The output includes device names, sizes, types, mount points, and their hierarchical relationships.

Do lsusb, lspci, and lshw Function Similarly?
While lsblk, lsusb, lspci, and lshw all provide hardware information, their internal mechanisms and data sources differ:
lsusb: Lists USB devices by querying the USB subsystem, typically reading from /proc/bus/usb or using the libusb library to interact with the kernel's USB stack.
lspci: Lists PCI devices by querying the PCI subsystem, often reading from /proc/bus/pci or using the libpci library to directly access PCI configuration space.
lshw: Provides a comprehensive hardware overview by aggregating information from multiple sources, including /proc, /sys, and direct queries to hardware via various kernel interfaces.

10. How can we simulate a shutdown operation via the /sys file system?
We can go to the specific target level, in this case, power off mode to not only halt the computer but also to detach its powe, using the following command:

```
systemctl isolate poweroff
```

11. What are the different types of kernels?
Monolithic vs. Microkernel vs. Hybrid
What are the advantages and disadvantages of each?

Monolithic Kernel

- Definition:
  - A monolithic kernel integrates all essential operating system services—including device drivers, file system management, process scheduling, and networking—directly into a single large kernel running in a single address space
- Advantages:
  - High performance: Direct function calls within the kernel space minimize overhead, making monolithic kernels generally faster and more efficient
  - Simplicity in development: Easier and faster to develop since all components are tightly integrated
  - Efficient communication: No need for inter-process communication (IPC) between kernel components
- Disadvantages:
  - Stability and security risks: A bug in any kernel component can crash the entire system or expose vulnerabilities, as everything runs with high privileges in the same address space
  - Less modular: Difficult to extend or modify without affecting the whole kernel
  - Larger codebase: More code in kernel space increases the risk of errors and makes maintenance harder
- Examples: Linux, traditional UNIX, BSD.

Microkernel

- Definition:
  - A microkernel keeps only the most essential functions (such as low-level memory management, process scheduling, and IPC) in kernel space. All other services (device drivers, file systems, networking) run as user-space processes.
- Advantages:
  - Modularity and flexibility: Easier to add, remove, or update services without affecting the core kernel.
  - Stability and security: Crashes or bugs in user-space services do not bring down the entire system; the attack surface is reduced
  - Portability: Easier to port to new hardware since most services are user-space
- Disadvantages:
  - Performance overhead: Communication between user-space services and the kernel relies on IPC, which can introduce significant overhead and slow down system performance
  - Complex development: More code is needed to manage communication and service isolation
- Examples: QNX, MINIX, Mach (basis for macOS), L4.

Hybrid Kernel

- Definition:

- ○ A hybrid kernel blends aspects of both monolithic and microkernel architectures. Some services (often performance-critical ones) run in kernel space, while others run in user space, aiming to balance performance, modularity, and security.
- Advantages:
  - Balanced performance and modularity: Selectively places services in kernel or user space to optimize for both speed and maintainability.
  - Flexibility: Developers can decide which components benefit from running in kernel mode versus user mode.
  - Improved stability (compared to monolithic): Some isolation of services reduces the risk of total system failure.
- Disadvantages:
  - Complexity: Can be more complex to design and maintain due to the mix of architectures.
  - Trade-offs: May not achieve the full performance of a pure monolithic kernel or the full modularity/security of a pure microkernel.
- Examples: Windows NT, macOS (XNU), modern versions of Linux with loadable kernel modules.

12. Why is the first sector of a disk used for the MBR?

The first sector of a disk is used for the Master Boot Record (MBR) because it provides a standardized, fixed location for essential boot and partition information that the system firmware (BIOS) can reliably access during the boot process. When a computer starts, the BIOS is programmed to read the very first sector of the boot disk—Cylinder 0, Head 0, Sector 1—which is always 512 bytes in size. This sector contains:

- The bootloader code, which initiates the process of loading the operating system.
- The disk partition table, which describes how the disk is divided into partitions.
- A signature (55AAH) that identifies the sector as a valid MBR

13. If the MBR is located in the first 512 bytes, how does it know the location of GRUB or another bootloader to load the kernel?

The MBR (Master Boot Record) is limited to 512 bytes and contains only a small piece of boot code, a partition table, and a signature. It does not directly know the location of GRUB or a full-featured bootloader. Instead, it follows a multi-stage process to locate and load the next stage of the bootloader, such as GRUB, which then loads the kernel.
How does the MBR locate GRUB or Another Bootloader

- Partition Table Lookup:
    - The MBR contains a partition table listing up to four primary partitions, each with information about its type, status (active/bootable), and starting location on the disk.
- Active Partition Selection:
    - The MBR boot code scans the partition table to find the "active" (bootable) partition.
- Volume Boot Record (VBR) Loading:
    - The MBR loads the first sector (the Volume Boot Record, or VBR) of the active partition into memory and transfers control to it.
    - The VBR is also called the partition boot sector or boot sector.
- Bootloader Stage 2:
    - The VBR contains code to load the next stage of the bootloader—such as GRUB Stage 2, NTLDR, or another OS loader—which is typically stored elsewhere on the disk (often within the partition itself).
- Kernel Loading:
    - The full bootloader (like GRUB) then presents a menu, loads the operating system kernel, and starts the OS.

14. What are .efi files, and what is their role in the boot process?

“.efi” files are executables used by systems that follow the UEFI (Unified Extensible Firmware Interface) standard, which is the modern replacement for the older BIOS system. These files are typically stored on a special partition called the EFI System Partition (ESP), and they play a crucial role in the system’s boot process.

When a computer using UEFI firmware powers on, it doesn't immediately hand over control to an operating system. Instead, it looks into the EFI System Partition for .efi files, which are programs the firmware can execute to initiate the boot process. One of these .efi files is usually a bootloader, such as bootx64.efi for 64-bit systems. This bootloader is responsible for loading the operating system kernel into memory and starting the OS.

In the case of Linux, for example, we might have a file like grubx64.efi, which is the GRUB bootloader compiled as an EFI binary. For Windows, the file might be bootmgfw.efi. These files serve as the bridge between the firmware and the OS. They can also support additional features like secure boot, where the firmware verifies that the .efi file has not been tampered with and is signed by a trusted authority before executing it.

15. What is the ESP (EFI System Partition) in UEFI, and how is it used?

The EFI System Partition, or ESP, is a special partition on a UEFI-based system's storage drive that plays a key role in the boot process. It is formatted with a FAT32 file system so that the UEFI firmware can reliably read from it, regardless of the operating system installed. When a computer starts up, the UEFI firmware looks to the ESP for boot information, particularly for executable files with the `.efi` extension. These files include bootloaders for installed operating systems, as well as drivers or other UEFI applications.

Each installed operating system places its own bootloader on the ESP, typically in a subdirectory named after the OS or its vendor. For example, Windows stores its boot manager in a folder named `EFI/Microsoft/Boot`, while a Linux distribution using GRUB might place its files in `EFI/Ubuntu`. The firmware uses a boot entry list, stored in NVRAM (non-volatile RAM), to locate and launch these bootloaders during startup. The ESP can also contain tools like EFI shells, diagnostics, and recovery utilities.

In essence, the ESP acts as a standardized, OS-independent space where the firmware can find and launch the appropriate software to start the system, making it a central component of the UEFI boot process.

16. Please explain the following section from /etc/grub/grub.conf:

```
menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os $menuentry_id_option 'gnulinux-simple-3e3d2181-a1f5-4456-867c-a69f52c910e6'
```

This line defines a GRUB boot menu entry titled "Ubuntu", with associated class tags like ubuntu, gnu-linux, and os, which help GRUB themes or scripts categorize the entry.

```
recordfail
```

This tells GRUB to record if a previous boot attempt failed. It's used for fallback or recovery logic.

load_video, gfxmode $linux_gfx_mode

These lines initialize video settings for graphical boot menus or splash screens. GRUB is preparing to show a graphical interface rather than a simple text-based one.

insmod lines

These lines load specific GRUB modules:

- gzio: allows GRUB to read compressed files.
- xzio and lzopio: additional compression modules, loaded conditionally if the platform is xen (used in virtualized environments).
- part_gpt: allows GRUB to read GPT-partitioned disks.
- ext2: allows GRUB to read ext2/ext3/ext4 file systems.

set root='hd0,gpt2'

This tells GRUB which partition holds the OS, in this case, the second GPT partition of the first disk (hd0 = first disk, gpt2 = second partition). This is where the kernel and initrd are located.

if x$feature_platform_search_hint = xy; then

This conditional block checks if GRUB supports platform search hints, which help GRUB find the correct boot partition more efficiently and reliably.

search ... --set=root ...

This command searches for the partition containing the given UUID (49ee8c4e-7d13-455b-b287-488a33286e30) and sets it as the root for the boot process. It uses hints like --hint-bios, --hint-efi, or --hint-baremetal to speed up the search depending on the platform.

linux /vmlinuz-5.4.0-65-generic root=/dev/mapper/vg0-root ro maybe-ubiquity

This line loads the Linux kernel image, passes the root file system location (/dev/mapper/vg0-root), sets it to read-only at first (ro), and includes maybe-ubiquity (a boot parameter related to whether the Ubuntu installer GUI should run).

initrd /initrd.img-5.4.0-65-generic

This loads the initial RAM disk image (initrd), which contains necessary drivers and scripts needed to boot before the main system is accessible.