1. In fdisk, when using the p (print) command, there is a column labeled "sectors." What does "sector" refer to in this context?

In `fdisk`, the "sectors" column refers to the number of 512-byte disk sectors allocated to a partition, indicating its size and position on the disk.

2. There are three types of servers mentioned. Please summarize your partitioning recommendations for each:

- Linux desktop systems
- Linux servers used for databases or web services with extensive logging
- Linux servers used in university labs or staging environments where

- **Linux Desktop Systems**: On a desktop computer, it is good to have one swap, one `/boot`, and allocate all other space to `/` (root).
- **Linux Database/Web Servers with Extensive Logging**: Allocate a small `/` (20-30 GB), a large `/var` (100-500 GB) for logs and databases, a `/home` (50-100 GB) for user files, and swap (1-2x RAM). Use LVM for flexibility.
- **Linux University Lab/Staging Servers**: Create a `/` (20-30 GB), a large `/home` (200-500 GB) for individual user directories, a `/var` (50-100 GB) for shared data, and swap (1-2x RAM). Use quotas to manage user storage.

3. How does an operating system run multiple applications when the total available RAM is less than the combined memory requirements? Please summarize:

- How processes are scheduled to run on the CPU
- How process data is loaded into RAM
- What happens when RAM is insufficient for a new process
- How and when the operating system uses disk space (paging and swapping)
- The memory management strategies involved (e.g., demand paging)
- Whether all pages of a process must be loaded into RAM for execution

- CPU Scheduling

- The OS uses a **scheduler** to alternate CPU time between processes (round-robin, priority-based, etc.).

- Loading into RAM

- Only **needed parts** of a program are loaded into RAM using **demand paging**.

- When RAM Is Insufficient

- The system uses **paging** or **swapping**:
- Moves inactive memory pages to **disk (swap space)**.
- This frees up RAM for active processes.

- Disk Usage (Paging vs. Swapping)

- **Paging**: Moves individual memory pages (4KB typically).
- **Swapping**: May move the **entire process** out of RAM.
- Stored in a special swap partition or file.

- Memory Management Strategies

- **Demand Paging**: Only load pages as they are accessed.
- **Copy-on-Write (COW)**: Share memory between processes until modification.
- **Page Replacement Algorithms**: Like LRU (Least Recently Used).

- Must All Pages Be in RAM?

- **No** — only active pages are needed.
- Inactive pages remain on disk until required (lazy loading).

4. What is the Translation Lookaside Buffer (TLB), and what role does it play in memory management?

The **Translation Lookaside Buffer (TLB)** is a **CPU cache** used to speed up **virtual-to-physical address translations**.

- Every memory access by a program uses **virtual addresses**.
- The **MMU** (Memory Management Unit) consults the **page table**, which is slow.
- The TLB caches recent translations:
- **Hit**: Fast access
- **Miss**: Consult full page table (slower)

5. What are a page, a virtual page, and a context switch? Additionally, how does increasing swap space affect context-switching performance? How does page size influence these effects?

- **Page**: Fixed-length block of memory (commonly 4KB).
- **Virtual Page**: A page in the virtual address space.
- **Context Switch**: When the CPU switches between processes:
  - Saves current state, loads the next
  - Requires **memory context change**, possibly TLB flush

6. What is a huge page? Please explain its purpose and when it is used.

A **huge page** is a large memory page (e.g., 2 MB or 1 GB) used to reduce TLB overhead and improve performance for memory-intensive applications like databases or virtual machines, enabled via `hugetlbfs` in Linux.

7. What is memory fragmentation in RAM, and what problems can it cause?

Memory fragmentation occurs when free RAM is split into small, non-contiguous chunks, preventing allocation of large blocks. It causes allocation failures, inefficient memory use, and performance degradation.