LPIC1 Exercises

Week 2 - Videos 17 - 27

Nika Soltani Tehrani
Spring 2025

1. **In fdisk, when using the p (print) command, there is a column labeled "sectors." What does "sector" refer to in this context?**
   A sector is the smallest physical unit of storage on a disk. The disk is divided into many such sectors, and partitions are defined by specifying the starting and ending sectors they cover.
   So, the "sectors" column shows the size of the partition in terms of how many of these basic units it contains.

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 multiple users require individual home directories and personal storage space

   Linux desktop systems:
   - **/boot** - 500 MB to 1 GB: Contains the EFI System Partition (ESP) and kernel/boot files. A separate partition is required for UEFI systems and protects boot files.
   - **/** - 20–50 GB: Holds the core OS files and applications. 20 GB is typically sufficient for most desktop uses, but increase up to 50 GB if you plan to install many applications or games.
   - **/home** - Remaining free space or dedicated partition: Stores user data and personal files. Separating /home from root ensures user data is safe if the OS needs reinstalling or upgrading.
   - **/var** - 2 GB or more (optional for desktops): Stores variable data such as logs and caches. For desktop systems, this can remain within root unless you expect heavy logging or server-like usage.
   - **/tmp** - Separate partition (optional): Used for temporary files. Separating /tmp can improve security and prevent temporary files from filling other partitions, but often not mandatory for typical desktops.
   - **swap** - Size depends on RAM: Typically equal to or 1–2 times the amount of RAM depending on whether hibernation is used. Modern systems often use swap files instead of dedicated partitions, but swap partitions are still common.
   - **/usr** (20 GB or more): Contains user-installed applications and libraries; separating it can protect applications during system upgrades.
   - **/opt** (500 MB to 5 GB): For optional or third-party software.

   Linux servers used for databases or web services with extensive logging:
   - Use LVM (Logical Volume Manager):
     i. Employ LVM to allow dynamic resizing and easier management of partitions as storage needs grow or change over time.
   - Separate Key Directories into Different Partitions:

This helps isolate system components, improves performance, and prevents one partition filling up from affecting others.

i. **/boot** - 250 MB to 1 GB: Contains kernel and boot files; a separate partition protects boot files and simplifies kernel upgrades.
ii. **/root** - 20–50 GB: Holds the core OS files and applications. 20 GB is usually sufficient, but increases if many applications or services are installed.
iii. **/var** - 2 GB or more: Stores variable data such as logs, databases, and spool files. Because extensive logging is expected, allocate more space here or consider a separate /var/log partition to isolate logs and prevent disk space issues from affecting the system.
iv. **/var/log** - Separate partition recommended: To handle extensive logging without impacting other /var contents or the root filesystem, isolating /var/log improves stability and performance.
v. **/tmp** - Size similar to swap (e.g., 4–8 GB): Temporary files can grow large; a separate /tmp partition prevents these files from filling critical partitions.
vi. **/home** - Remaining free space or dedicated partition: Stores user data and personal files, keeping them safe from OS issues.
vii. **/usr** - At least 20 GB: Contains user-installed applications and libraries; separating it can protect applications during system upgrades.
viii. **swap** - Size depends on RAM: Typically 1–2 times the amount of RAM, depending on workload and whether hibernation is used.
ix. **/opt** for optional software (500 MB to 5 GB)
x. **/etc, /bin, /sbin, /lib** can remain within root unless specific needs arise.

Linux servers used in university labs or staging environments where multiple users require individual home directories and personal storage space:

- **/boot** - 500 MB to 1 GB: Contains bootloader and kernel files; a separate partition ensures boot files are protected and simplifies kernel updates.
- **/** - 20–50 GB: Holds the core OS files and applications. Allocate enough space to accommodate installed software and system updates.
- **/home** - Large partition with remaining space: Stores individual user home directories and personal files. Keeping /home separate allows user data to be preserved independently of the OS and simplifies backups.
- **/var** - 5 GB or more (depending on usage): Stores variable data including logs, mail spools, and databases. Since university lab servers may have extensive logging and application data, allocate sufficient space here.
- **/var/log** - Separate partition recommended: To handle extensive logging without risking the root or /var partitions filling up, isolating /var/log improves system stability and log management.
- **/tmp** - 4–8 GB: Used for temporary files. A separate /tmp partition prevents temporary files from consuming space on critical partitions.

- **/swap** - Size depends on RAM: Typically equal to or 1–2 times the amount of RAM depending on workload and whether hibernation is used.

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

   When an operating system runs multiple applications but the total available RAM is less than the combined memory requirements, it uses several techniques to manage this efficiently. First, the OS schedules processes to run on the CPU by quickly switching between them, giving the appearance that they run simultaneously. This scheduling is managed by algorithms that decide which process gets CPU time based on priority and fairness.

   Regarding memory, the OS loads process data into RAM as needed, using a system called virtual memory. Instead of loading the entire program into RAM at once, it loads parts of it on demand. When the RAM is insufficient to hold all active processes, the OS moves some data from RAM to disk storage, a process known as swapping or paging. This frees up RAM for other processes to run. Paging divides memory into small fixed-size blocks called pages, and only the necessary pages are loaded into RAM, rather than the whole program.

   The OS uses memory management strategies like demand paging, which means it loads pages into RAM only when they are actually needed during execution. This approach allows the system to run programs even if there isn't enough physical memory to hold all their data at once. Therefore, not all pages of a process must be loaded into RAM for it to execute; only the required parts are loaded, and the rest remain on disk until needed. This combination of scheduling, virtual memory, paging, and swapping enables the operating system to run multiple applications smoothly despite limited physical RAM.

   In swapping, When an operating system runs multiple applications and the total available RAM is less than the combined memory requirements, it relies on a feature called swap space to help manage memory. Swap space is a designated area on a hard drive or SSD that acts as an extension of physical RAM. When RAM becomes full, the operating system temporarily moves inactive or less frequently used data from RAM to the swap space. This process, known as swapping, frees up RAM for more important or active tasks, allowing the system to continue running smoothly instead of slowing down or crashing.

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

   The Translation Lookaside Buffer (TLB) is a specialized, high-speed cache used by a computer's memory management unit (MMU) to store recent translations of virtual

memory addresses to physical memory addresses. In modern operating systems that use virtual memory, every time a program accesses memory, the CPU must translate the program's virtual address into a physical address in RAM. This translation is typically done using page tables, which reside in main memory and can be slow to access repeatedly.

The TLB significantly speeds up this process by keeping a small, fast-access cache of the most recently used address translations. When a memory access occurs, the system first checks the TLB for the required translation-a situation called a TLB hit. If the translation is found, the physical address is returned quickly, greatly reducing memory access time and improving overall system performance. If the translation is not in the TLB, the system must look up the address in the page table, which is slower, and then the new translation is added to the TLB for future access.

By minimizing the number of slow page table lookups, the TLB plays a crucial role in efficient memory management, reducing latency and power consumption, and ensuring the system remains responsive even as it manages complex virtual memory mappings.

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?**

   A page is a fixed-size block of memory used in virtual memory systems to divide both physical memory (RAM) and virtual memory into manageable chunks. A virtual page refers specifically to one of these blocks in the virtual address space of a process, which the operating system maps to a physical page frame in RAM or to disk if not currently in memory.

   A context switch is the process where the CPU stops executing one process and starts executing another. During this switch, the operating system saves the current process's state-including CPU registers and memory management information like page tables-and loads the state of the next process to resume its execution. This allows multiple processes to share the CPU effectively, enabling multitasking.

   When RAM is insufficient, the operating system uses swap space on disk to store pages that are not actively needed in memory. Increasing swap space can help prevent out-of-memory errors and provide more virtual memory, but excessive swapping can degrade performance because disk access is much slower than RAM access. During context switches, if processes have many pages swapped out, the system may spend more time loading pages back into RAM, increasing latency.

   The page size influences these effects significantly. Larger pages reduce the number of pages and page table entries, which can lower overhead and reduce TLB misses, improving performance. However, large pages can also lead to wasted memory if processes do not fully use the allocated space and can increase the cost of swapping since more data moves between RAM and disk per page. Smaller pages allow finer-grained memory management but increase the overhead of page tables and TLB misses.

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

A huge page is a memory page that is significantly larger than the default page size used by most operating systems, which is typically 4 KiB. Common huge page sizes on modern x86_64 systems are 2 MiB and 1 GiB. The primary purpose of huge pages is to improve system performance by reducing the overhead associated with managing large amounts of memory.

Huge pages help reduce the pressure on the TLB. Because huge pages cover much larger memory regions, fewer TLB entries are needed to map the same amount of memory. This leads to fewer TLB misses, which in turn reduces the costly page table lookups and improves the speed of memory access for memory-intensive applications.

Huge pages are especially beneficial in scenarios where applications use large contiguous memory blocks, such as databases, virtualization, high-performance computing, and large-scale in-memory caches. Some systems use Transparent Huge Pages (THP), which automatically manage huge pages without explicit application support, but many high-performance applications prefer to use pre-allocated huge pages for better control and predictable performance.

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

   Memory fragmentation in RAM occurs when free memory is broken into many small, non-contiguous blocks over time, making it difficult for the system to allocate large contiguous chunks of memory even though the total free memory might be sufficient. This happens because programs request and release memory blocks of varying sizes at different times, leaving gaps that are too small to satisfy new allocation requests. As a result, the available memory is inefficiently used, and some allocation requests may fail despite enough total free memory existing. This is known as external fragmentation.