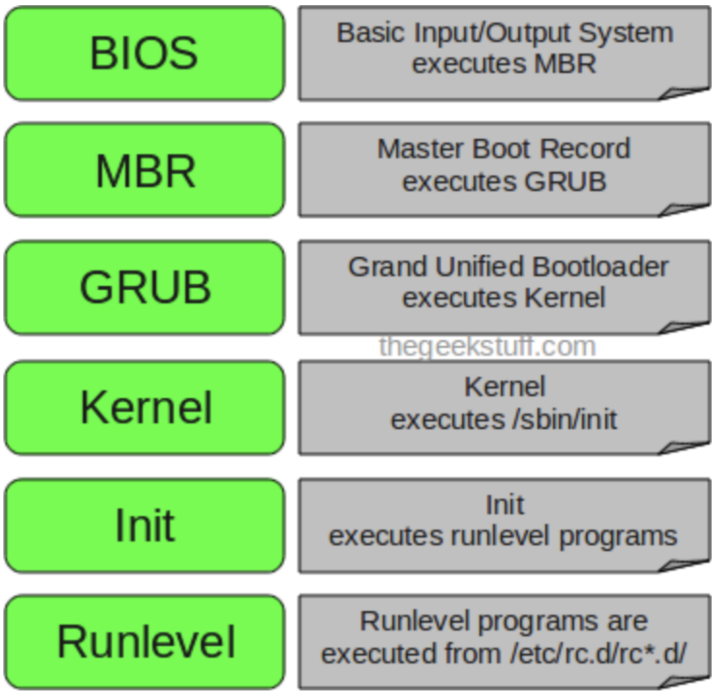
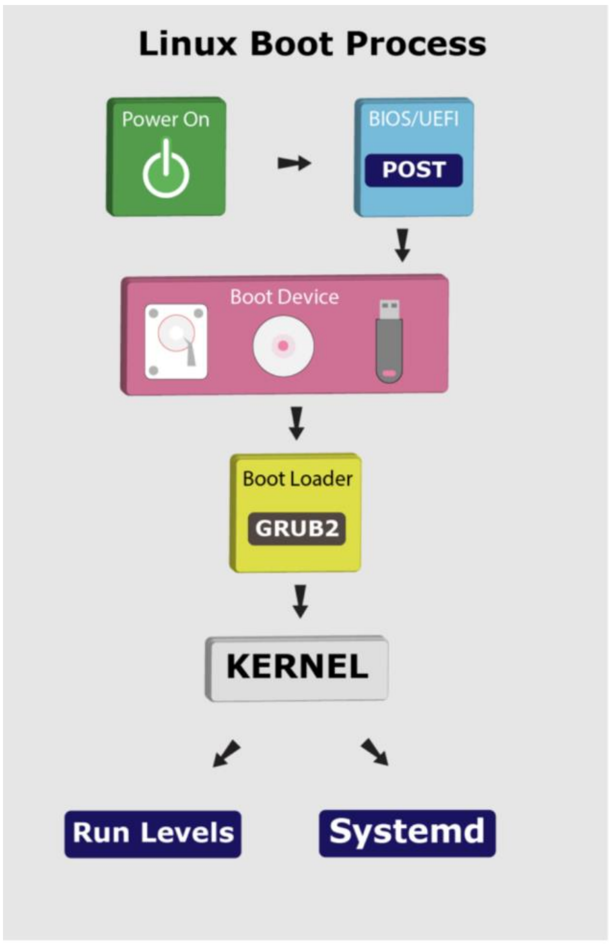
Topics To be Covered

Week	Broader Topic	Lecture	Topics	Tools to be covered
1	Boot Process	11-14	1. Boot Process: The boot process. Linux Boot Process POST Boot Loader GUI 2. Partitioning Partitioning Tools 2.1 Create a Partitioning 3. Virtual memory Address Space Paging Demand Paging Page Replacement Algorithm 3.1 Benefits of Virtual Memory 3.2 Virtual Memory on linux 4. “Mounting” and “Mount” 5. How Files Get stored in memory 6. Dual Boot 7. Introduction to Swap Space 7.1 Types of Swap Space	Linux

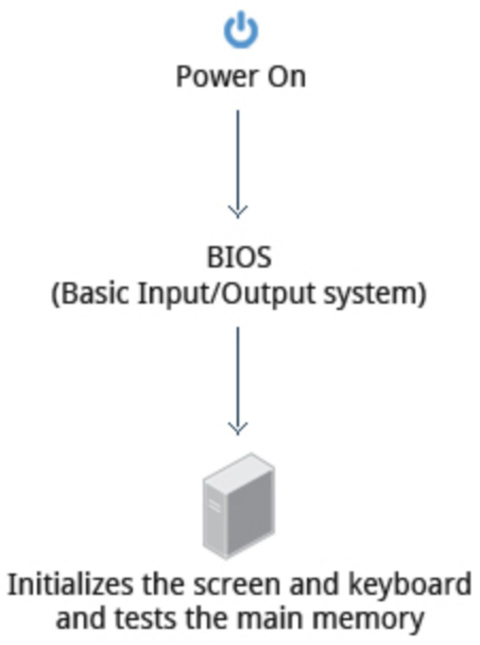
1. Linux Boot Process


BIOS
Basic Input/Output System executes MBR
MBR
Master Boot Record executes GRUB
GRUB
Grand Unified Bootloader executes Kernel
thegeekstuff.com
Kernel
Kernel executes /sbin/init
Init
Init executes runlevel programs
Runlevel
Runlevel programs are executed from /etc/rc.d/rc*.d/



The boot process, also known as booting, refers to the sequence of steps that a computer system follows to start up and load the operating system into memory. When you turn on or restart your computer, the boot process begins, and it typically involves several stages. Here is a generalized overview of the boot process:

1. **Power-on self-test (POST):** When you turn on your computer, the basic input/output system (BIOS) or the firmware in newer systems performs a POST. This is a series of diagnostic tests to check the hardware components (such as memory, processor, and storage devices) and ensure they are functioning correctly.


Power On
BIOS
(Basic Input/Output system)
Initializes the screen and keyboard and tests the main memory

2. **Initial startup program:** After the POST, the BIOS or firmware locates the initial startup program. In traditional BIOS systems, this program is stored in the firmware itself, while in modern UEFI systems, it is usually located in a separate partition on the storage device.
3. **Boot loader:** The initial startup program then loads the boot loader. The boot loader is responsible for locating the operating system's kernel and initializing the boot

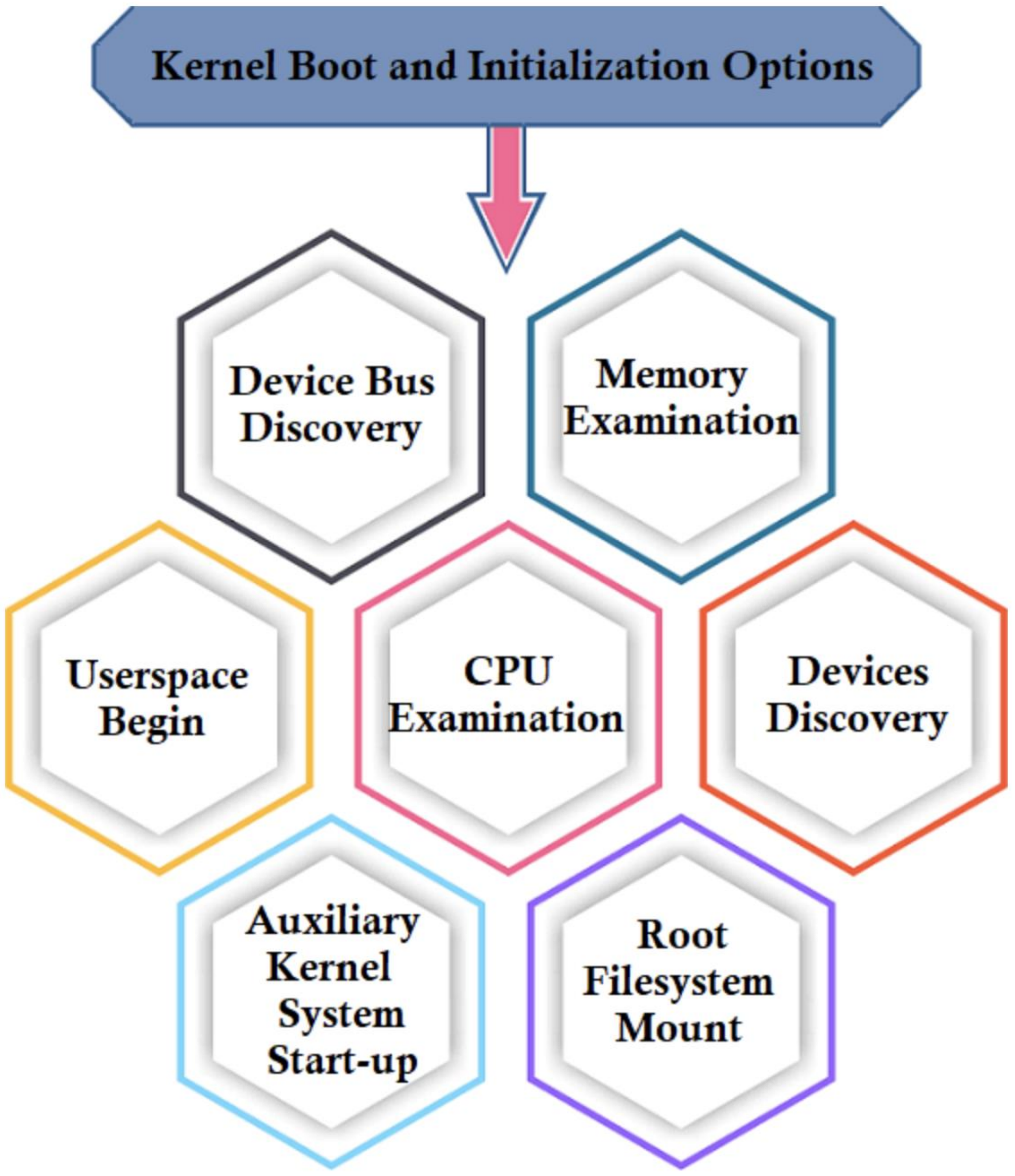
process. Examples of popular boot loaders include GRUB (used in many Linux distributions) and Windows Boot Manager. Grub configuration file is **/boot/grub/grub.conf** The following is a sample grub.conf of CentOS.

- #boot=/dev/sda
- default=0
- timeout=5
- splashimage=(hd0,0)/boot/grub/splash.xpm.gz
- hiddenmenu
- title CentOS (2.6.18-194.el5PAE)
-         root (hd0,0)
-         kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=
-         initrd /boot/initrd-2.6.18-194.el5PAE.img

4. **Operating system kernel:** Once the boot loader is loaded into memory, it transfers control to the operating system's kernel. The kernel is the core component of the operating system that manages system resources, and device drivers, and provides various services.
5. **Initialization and device detection:** The operating system kernel initializes the essential system components and starts detecting and configuring hardware devices such as the CPU, memory, storage devices, input/output devices, and network interfaces.

- Looks at the /etc/inittab file to decide the Linux run level.

6. **User space initialization:** After the kernel initializes the hardware, it starts launching user-space processes and services. These processes and services are responsible for providing the user interface and running applications. In Unix-like systems, the init process is typically the first user space process and serves as the ancestor of all other processes.



7. **Graphical user interface (optional):** If your operating system includes a graphical user interface (GUI), such as Windows or macOS, the boot process continues to initialize the graphical system components, including the window manager and desktop environment, providing a visual interface for the user.
8. **Login or desktop:** Finally, the boot process reaches a stage where the user can log in or directly access the desktop environment. The operating system has finished loading, and you can start using the computer and running applications.

It's important to note that the boot process can vary between different operating systems and configurations. The steps described here provide a general overview, but specific details mays differ based on the system's firmware, boot loader, and the operating system being use.

2. Partitioning

Partitioning is the process of dividing a storage device, such as a hard drive or solid-state drive (SSD), into multiple sections or partitions. Each partition acts as a separate logical unit with its own file system, allowing you to organize and manage data more efficiently. Here are some key points about partitioning:

1. **Primary and Extended Partitions:** On a disk, you can have primary and extended partitions. A primary partition is a standalone partition that can be used to install an operating system or store data. An extended partition, on the other hand, serves as a container for logical partitions.
2. **Logical partitions:** Logical partitions are created within the extended partition. They are used to further divide the available space and provide additional storage areas. Logical partitions cannot be used to install an operating system directly but can be used for data storage.
3. **File systems:** Each partition typically has a file system associated with it, such as ext4, NTFS, or FAT32. The file system determines how data is stored, accessed, and organized on the partition.
4. **System partition:** The system partition, often referred to as the root partition (denoted as '/'), is where the operating system is installed. It contains essential system files, directories, and configuration files.
5. **Data partitions:** Data partitions are used to store files, documents, media, and other user data. By separating the operating system and user data into different partitions, you can isolate the data from potential system issues and make it easier to back up or reinstall the operating system.
6. **Partitioning tools:** There are various partitioning tools available for Linux, such as fdisk, parted, and GParted. These tools allow you to create, delete, resize, and modify partitions on your storage devices.

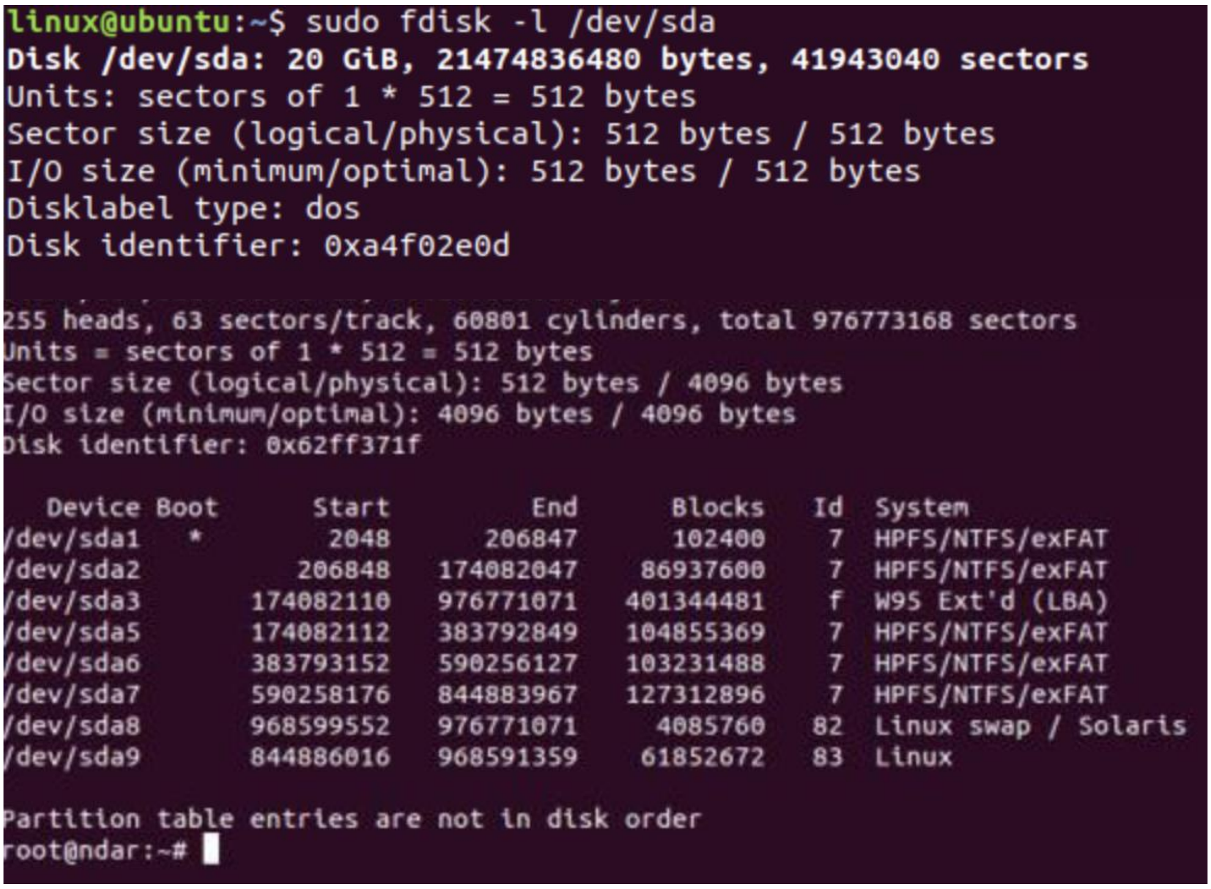
When partitioning a disk, it is important to plan the partition sizes and layout according to your specific needs. Consider factors such as the operating system requirements, the amount of data you expect to store, and any future expansion requirements. Properly partitioning your storage device can help improve your system's performance, organization, and data management.

2.1 Create a Partition in Linux - A Step-by-Step Guide

Note: Creating a partition involves modifying the disk's partition table, which can result in data loss if not done carefully. Make sure to back up your important data before proceeding.

1. Open a terminal: Open a terminal window on your Linux system. You can usually find the terminal application in the applications menu or by using the keyboard shortcut (e.g., Ctrl+Alt+T).
2. Identify the disk: Run the following command to list the available disks on your system:

   ```
   sudo fdisk -l
   ```



```
linux@ubuntu:~$ sudo fdisk -l /dev/sda
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa4f02e0d

255 heads, 63 sectors/track, 60801 cylinders, total 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk identifier: 0x62ff371f

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      206847      102400    7  HPFS/NTFS/exFAT
/dev/sda2          206848   174082047    86937600    7  HPFS/NTFS/exFAT
/dev/sda3       174082110   976771071   401344481    f  W95 Ext'd (LBA)
/dev/sda5       174082112   383792849   104855369    7  HPFS/NTFS/exFAT
/dev/sda6       383793152   590256127   103231488    7  HPFS/NTFS/exFAT
/dev/sda7       590258176   844883967   127312896    7  HPFS/NTFS/exFAT
/dev/sda8       968599552   976771071     4085760   82  Linux swap / Solaris
/dev/sda9       844886016   968591359    61852672   83  Linux

Partition table entries are not in disk order
root@ndar:~#
```

Identify the disk on which you want to create a partition (e.g., /dev/sdb).

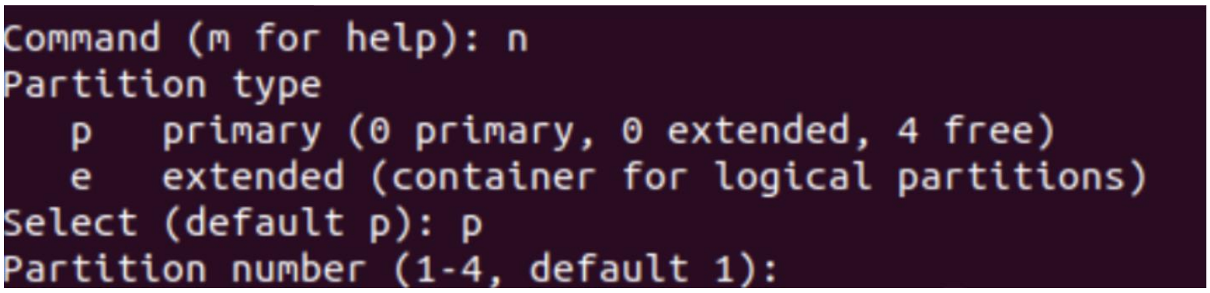
3. Run fdisk: Execute the following command, replacing /dev/sdX with the disk identifier from the previous step:

1. `sudo fdisk  -l /dev/sda`

4. Create a new partition: Within the fdisk prompt, use the following steps to create a new partition:

a. Type n and press Enter to create a new partition.

b. Choose the partition type:



```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
```

- For a primary partition, press p and Enter.
- For an extended partition, press e and Enter.
-

c. **Specify the partition number.** If it's the first partition, you can choose 1. For subsequent partitions, choose an appropriate number.

d. Set the starting and ending sectors for the partition. You can either specify the exact size in sectors or use default values for the entire disk space.

2. **Set the partition type:** After creating the partition, you need to set the partition type. Type t and press Enter.
   a. Select the partition number you just created.
   b. Choose the partition type code. For Linux filesystems, you can use 83 (Linux) or 82 (Linux swap). You can view the list of available partition types by typing L.
3. **Save and exit:** Once you have created the partition and set the partition type, type w and press Enter to save the changes and exit fdisk.
4. Format the partition: After creating the partition, you'll need to format it with a file system. For example, to format the partition as ext4, you can use the following command:
   sudo mkfs.ext4 /dev/sda1

Replace /dev/sda1 with the appropriate partition identifier.

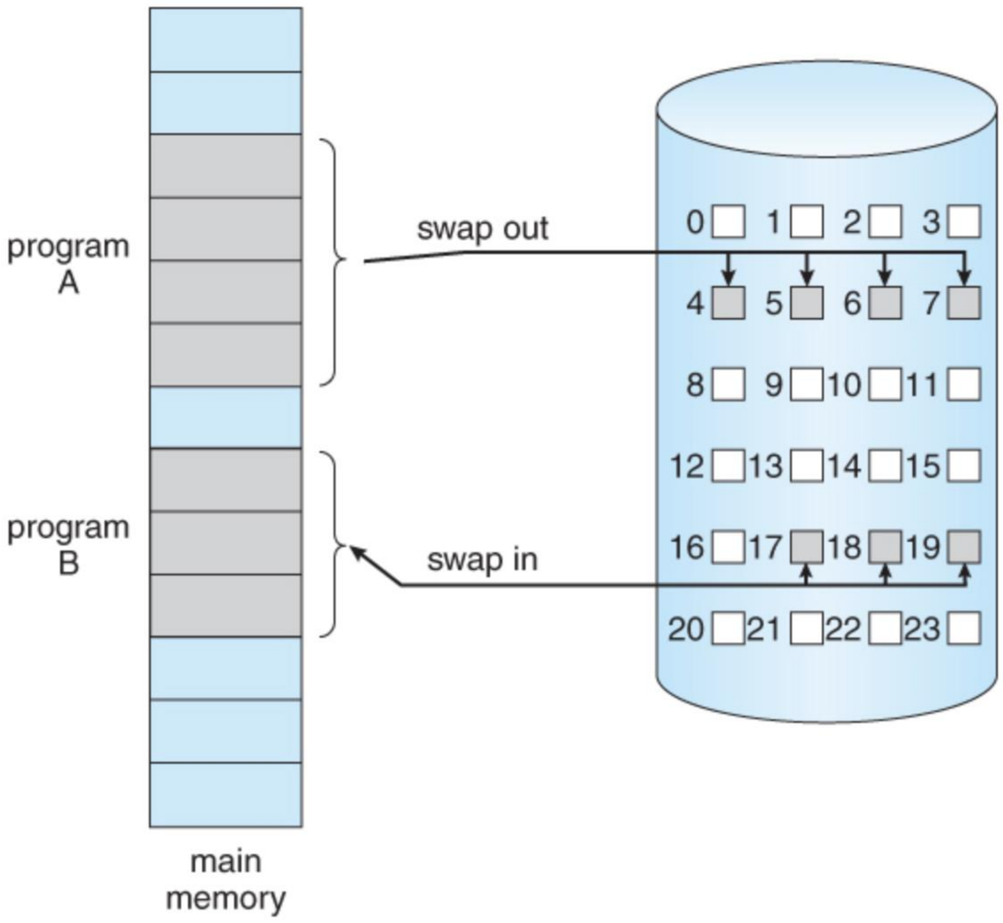
Congratulations! You have successfully created a partition in Linux using fdisk. You can now mount the partition and start using it to store files and data.

3. Virtual Memory

Virtual memory is a memory management technique used by operating systems to provide the illusion of having more physical memory (RAM) than is actually available on the system. It allows programs to utilize more memory than the physical RAM by utilizing disk space as an extension of the available memory. Virtual memory serves as an intermediary layer between the physical memory and the program's memory requirements.

Here's how virtual memory works:

1. **Address space:** Each program running on a system has its own virtual address space. This address space is divided into fixed-size chunks called pages.
2. **Paging:** The virtual memory system breaks down the program's memory into fixed-size pages, typically 4 KB in size. Similarly, the physical memory (RAM) is also divided into pages of the same size.



3. **Page tables:** The operating system maintains a page table that maps the virtual pages of a program to the corresponding physical pages in RAM. This mapping allows the program to access its memory as if it were using physical memory directly.
4. **Page swapping:** When a program requests a memory page that is not currently present in physical memory, a page fault occurs. The operating system then selects a page from RAM to evict (if necessary) and writes it back to the disk. The required page is then fetched from disk and loaded into the newly freed page in RAM.
5. **Demand paging:** Virtual memory systems typically use a technique called demand paging. It means that pages are loaded into physical memory only when they are actually accessed by the program. This approach helps conserve physical memory resources by loading only the necessary pages on-demand.

6. **Page replacement algorithms:** When physical memory becomes full and a new page needs to be brought in, the operating system uses page replacement algorithms (such as LRU, FIFO, or clock algorithms) to select the page to be evicted from RAM. The goal is to minimize the number of page faults and maximize the efficiency of memory usage.

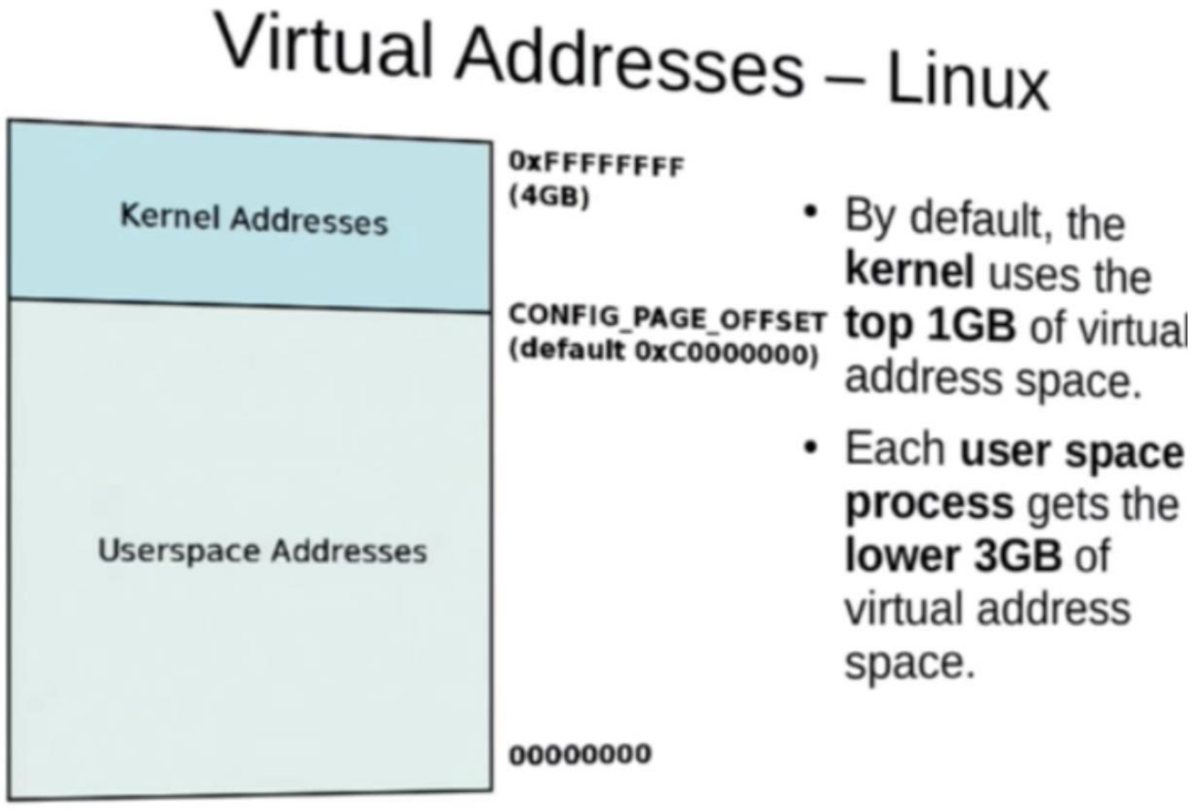
3.1 Virtual memory provides several benefits, including:

- It allows programs to run on systems with limited physical memory by utilizing disk space as additional memory.
- It provides memory protection by isolating the memory spaces of different programs, preventing one program from accessing or modifying the memory of another program.
- It enables efficient sharing of memory between multiple programs by mapping the same physical pages to different virtual pages.
- It simplifies memory management for programmers, as they can allocate and use memory without worrying about physical memory constraints.

However, the use of virtual memory can introduce performance overhead due to the need for disk I/O when swapping pages. Therefore, optimizing memory usage and minimizing page faults is essential for maintaining good system performance.

3.2 Virtual Memory on Linux and How to manage it

Virtual memory on Linux is managed by the kernel, and it employs various mechanisms to efficiently handle memory resources. Here's an overview of virtual memory management on Linux and how to manage it effectively:



Virtual Addresses – Linux

- By default, the **kernel** uses the **top 1GB** of virtual address space.
- Each **user space process** gets the **lower 3GB** of virtual address space.

1. **Swapping and Swappiness:**

    - Linux uses a technique called swapping to move pages between physical memory (RAM) and disk (swap space). When the system runs out of physical memory, less frequently used pages are swapped out to free up space.
    - The swappiness parameter determines the willingness of the system to swap out pages. It ranges from 0 to 100, with lower values preferring to keep more pages in physical memory and higher values being more aggressive in swapping out pages.
    - To check the current swappiness value: `cat /proc/sys/vm/swappiness`

- To modify the swappiness value temporarily (e.g., set it to 10): `sudo sysctl vm.swappiness=10`
- To make the swappiness value persistent across reboots, edit the `/etc/sysctl.conf` file and add or modify the line: `vm.swappiness = 10`

2. **Page Cache Management:**
    - Linux uses a page cache to speed up file I/O operations. The page cache holds recently accessed files and their corresponding disk pages in memory.
    - The `drop_caches` interface allows you to manually free up the page cache. However, it's generally recommended to avoid doing this unless necessary, as the kernel is designed to manage the page cache efficiently.
    - To drop page cache (requires root privileges): `sudo sh -c 'echo 1 > /proc/sys/vm/drop_caches'`
3. **Monitoring Memory Usage:**
    - Linux provides various tools to monitor memory usage and virtual memory statistics.
    - The free command displays information about total, used, and available memory, as well as swap usage.
    - The top command provides real-time information about memory usage by individual processes.
    - The vmstat command reports various virtual memory statistics, including page-ins, page-outs, swap activity, and more.
4. **Tuning and Optimizing:**
    - Linux offers several tunable parameters related to virtual memory management, allowing you to optimize memory usage for specific scenarios.
    - Parameters can be adjusted through the `/proc/sys/vm` interface or via sysctl settings.

- Some commonly tuned parameters include `dirty_ratio and `dirty_background_ratio` (control when the kernel starts writing dirty pages to disk), vm.vfs_cache_pressure (adjusts the kernel's tendency to reclaim inode and dentry cache), and more.
- To modify a parameter temporarily: sudo sysctl parameter_name=value
- To make a parameter change persistent, edit the /etc/sysctl.conf file and add or modify the corresponding line.

It's essential to carefully manage virtual memory on Linux to maintain system performance and prevent excessive swapping. However, it's generally recommended to rely on the default settings and let the kernel handle memory management, as it is designed to optimize performance automatically. Only make specific adjustments if you encounter specific memory-related issues or have specific workload requirements.

4. What do we mean by “Mounting” and “Mount”

In the context of computer systems, "mounting" refers to the process of associating a file system or storage device with a directory in the file hierarchy. When a file system or storage device is mounted, it becomes accessible to the operating system and applications, allowing them to read from and write to the files stored within it.

The concept of mounting is often used in operating systems like Linux and Unix. When a storage device such as a hard drive, solid-state drive, or network share is mounted, the operating system makes it available for use by assigning it a mount point, which is a directory in the file system. The mount point acts as a reference or entry point for accessing the files and directories stored on the mounted device.

For example, if you have a separate hard drive with files that you want to access, you can mount the hard drive to a specific directory in your file system, such as "/mnt/mydrive". Once mounted, the contents of the hard drive will be accessible through the "/mnt/mydrive" directory, and you can read from or write to the files as if they were part of the local file system.

In addition to physical storage devices, mounting can also be used for virtual file systems, network shares, or other types of storage resources. The process of mounting typically requires administrative privileges or root access, depending on the operating system and the specific configuration.

The term "mount" can refer to different things depending on the context. It can be used as a verb, describing the action of associating a file system or storage device with a directory (as explained above). It can also be used as a noun, representing the mounted file system or device itself. For example, you might say, "I mounted the USB drive to the /media/usb directory," where "mounted" is a verb, and "USB drive" is the mount, representing the mounted device.

5. How files get stored in memory

In Linux, the process of storing files in memory involves several layers, including the file system, buffer cache, and virtual memory subsystem. Here's an overview of how files are typically stored in memory in Linux:

- **File System:** Linux uses various file systems like ext4, XFS, and others to manage storage devices and organize files. When a file is created or accessed, the file system keeps track of its metadata (such as file name, size, and permissions) and the location of its data blocks on the storage device.
- **Buffer Cache:** The buffer cache is a portion of the system's memory reserved for caching frequently accessed data from storage devices. When a file is read or written, the file system driver fetches the corresponding data blocks from the storage device and stores them in the buffer cache. Subsequent accesses to the same file can be served directly from the cache, improving performance.
- **Page Cache:** The page cache is a component of the virtual memory subsystem in Linux. It caches file data from the buffer cache and holds it in memory pages. Each page in the page cache represents a fixed-size chunk of data from a file. The page cache provides a unified view of data from various files and allows efficient access and management of file data in memory.

- **Memory Mapping:** Linux supports memory mapping, which allows files to be mapped directly into the virtual memory space of a process. With memory mapping, a portion or the entire file is mapped to memory pages, providing direct access to the file's contents without explicit read or write operations. This can be useful for efficient file processing, especially for large files.
- **Virtual Memory Management:** Linux uses virtual memory management techniques to manage memory allocation and provide a logical address space for processes. When a file is accessed, the necessary memory pages are allocated in the virtual memory space of the process. The page cache manages the association between the file data and the allocated memory pages.
- **Caching and Swapping:** The page cache in Linux not only caches file data but also acts as a general-purpose cache for various data, including executable code, shared libraries, and other system data. The operating system dynamically manages the contents of the page cache based on the available memory and the demands of running processes. In situations of memory pressure, less frequently accessed pages can be evicted from memory and swapped out to disk.

It's worth noting that the specific mechanisms and optimizations related to file caching and memory management can vary based on the Linux distribution, kernel version, file system type, and system configuration. Linux provides a flexible and extensible architecture that allows administrators and developers to tune and customize these aspects to meet specific performance or resource requirements.

6. Dual Boot

Dual boot refers to the process of setting up a computer to run two different operating systems on the same machine. With dual boot, you can choose which operating system to boot into when you start your computer. This allows you to have the flexibility of using multiple operating systems on a single device. Here's a step-by-step guide on setting up dual boot with Linux and Windows:

Note: Before proceeding, it's crucial to back up your data to prevent any potential data loss during the installation process.

1. **Prepare your system:**

   a. Make sure your computer meets the system requirements for both Linux and Windows.

   b. Create a backup of all your important files and documents.

2. **Install Windows:**

   a. If you don't have Windows installed, install it first. Follow the instructions provided by Microsoft for the installation process.

   1. **Partition your hard drive:**

      a. In Windows, open the Disk Management tool by searching for "Disk Management" in the Start menu

      b. Shrink the Windows partition to create free space for the Linux installation. Right-click on the Windows partition, select "Shrink Volume," and choose the size for the new partition.

      c. Leave the free space unallocated for now. It will be used for Linux installation.

2. **Install Linux:**

   a. Download the Linux distribution of your choice and create a bootable USB drive using tools like Rufus or BalenaEtcher.

   b. Restart your computer and boot from the Linux USB drive. This usually involves pressing a specific key (e.g., F12, Esc, or Del) during startup to access the boot menu and selecting the USB drive as the boot option.

   c. Follow the installation wizard for your chosen Linux distribution. When prompted for the installation type, select the "Something Else" or "Manual" option to manually partition the disk.

   d. Create a root partition (typically mounted as "/") and, if desired, additional partitions for swap, home, or other purposes. Assign the unallocated space you created earlier to the Linux partitions.

   e. Complete the installation process by following the on-screen instructions.

3. **Configure the bootloader:**

a. After the Linux installation is complete, restart your computer. You may find that it automatically boots into Linux.

b. To boot into Windows, you need to configure the bootloader. In Linux, open a terminal and run the following command to update the bootloader:

```
sudo update-grub
```

c. The bootloader should detect and add the Windows installation to the boot menu.

4. Choose the operating system:

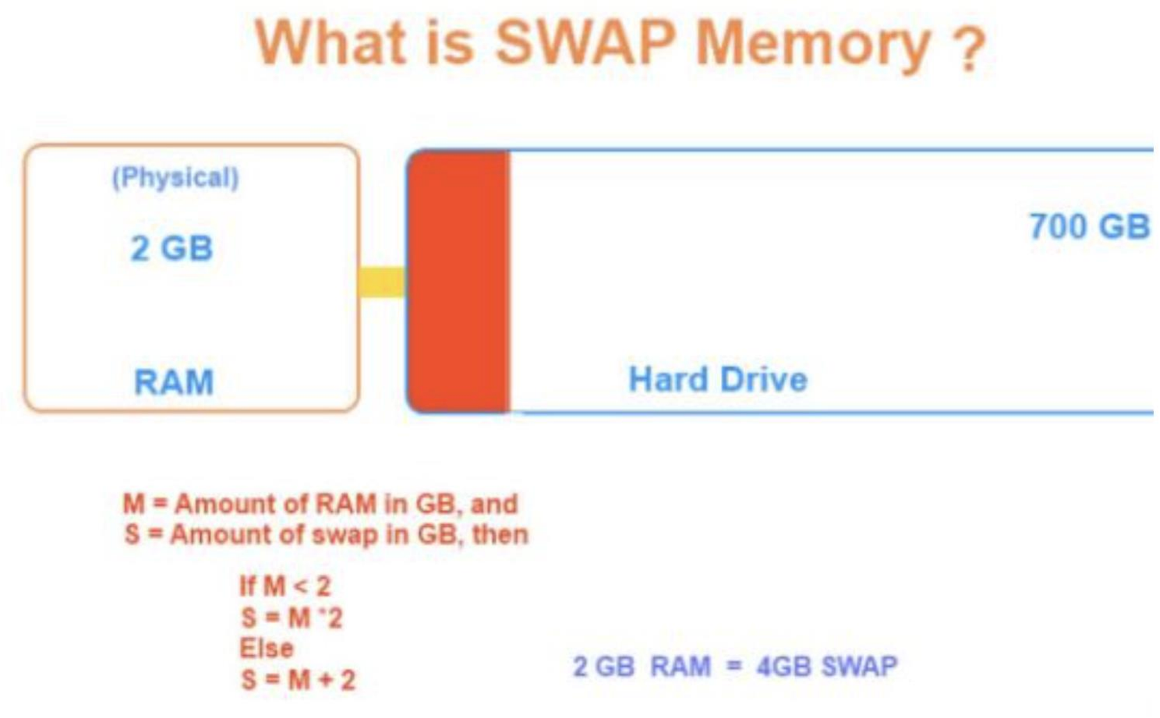
a. Restart your computer again. You should now see a boot menu during startup that allows you to choose between Linux and Windows.

b. Use the arrow keys to select the desired operating system, and press Enter to boot into it.

Congratulations! You have successfully set up dual boot on your computer with Linux and Windows. Now you can choose the operating system you want to use each time you start your computer.

7. Introduction to Swap Space

Swap space is an integral part of Linux systems and plays a crucial role in managing memory resources. It provides a way to extend the available memory beyond physical RAM by utilizing disk space as virtual memory. Here's an introduction to swap space on Linux systems:



M = Amount of RAM in GB, and
S = Amount of swap in GB, then

If M < 2
S = M *2
Else
S = M + 2

1. **Purpose of Swap Space:**
    - Swap space serves as a supplement to physical memory (RAM) when the system encounters memory limitations.
    - It allows the operating system to transfer less frequently used pages of memory from RAM to disk, freeing up RAM for more critical processes or data.
    - Swap space enables efficient memory management by providing a safety net for handling memory demands and preventing system crashes due to memory exhaustion.
2. **Swap Partition vs. Swap File:**

In Linux, swap space can be implemented using different types. The two main types of Linux swap are:

- Swap space can be implemented using either a dedicated swap partition or a swap file.
- A swap partition is a separate, specially designated partition on a hard disk solely dedicated to swap space.
- A swap file is a regular file stored on an existing filesystem, which is used as swap space.
- Both swap partitions and swap files serve the same purpose, but the flexibility of using swap files makes them more commonly used.

3. **Creating Swap Space:**
   - To create a swap partition, you need to repartition your disk and allocate a portion of it specifically for swap.
   - To create a swap file, you can use the dd command to create a file of the desired size and then format it as swap using the mkswap command.
   - Once created, the swap partition or file needs to be activated to make it usable by the system.
4. **Activating and Managing Swap Space:**
   - To activate a swap partition or file, you can use the swapon command followed by the path to the swap device or file.
   - The swapon command makes the swap space immediately available for use by the system.
   - The swapoff command is used to deactivate a swap partition or file and remove it from the system's available swap space.
   - The free command displays information about system memory, including the total swap space, used swap space, and available swap space.
5. **Swap Space Configuration:**
   - The configuration file for managing swap space is /etc/fstab. It contains entries that specify the swap devices or files to be activated at system startup.

- Editing the /etc/fstab file allows you to define the swap space persistence and configure the priority order of multiple swap devices or files.

6. **Monitoring Swap Space:**
   - Linux provides tools to monitor swap space usage and performance.
   - The free command displays information about swap usage, including the total swap space, used swap space, and available swap space.
   - Monitoring tools like top, htop, or vmstat can provide real-time insights into system memory and swap usage.

Overall, swap space on Linux systems provides a vital mechanism for managing memory resources and ensuring system stability even in memory-constrained situations.

7.1 Types of Linux swap

swap space disk partition

Swap space in Linux is an area on a disk that is designated for use as virtual memory. It acts as an extension to physical memory (RAM) and allows the operating system to swap out less frequently used pages from RAM to the disk when the memory is under pressure. Creating a dedicated swap partition is one way to allocate swap space in Linux. Here's a step-by-step guide to creating a swap partition:

Note: The following steps assume you have administrative privileges on your Linux system.

1. **Check available disk space:** Use the **df -h** command to check the available disk space on your system and identify the disk or partition where you want to create the swap partition.

2. **Determine swap space size:** Decide on the size of the swap partition you want to create. The general recommendation is to allocate swap space equivalent to 1.5 to 2 times the size of your RAM. For example, if you have 4 GB of RAM, you might allocate 6 to 8 GB of swap space.

3. Create the swap partition:

a. Open a terminal and run the following command to start the disk partitioning tool:

```
sudo fdisk /dev/sdX
```

Replace /dev/sdX with the identifier of the disk where you want to create the swap partition (e.g., /dev/sda).

b. Inside the fdisk prompt, type n and press Enter to create a new partition.

c. Choose the partition type:

- For a primary partition, press p and Enter.
- For an extended partition, press e and Enter.

d. Specify the partition number. If it's the first partition, you can choose 1. For subsequent partitions, choose an appropriate number.

e. Set the starting and ending sectors for the partition. You can either specify the exact size in sectors or use default values to allocate the remaining disk space.

f. Set the partition type to "Linux swap" by typing t and pressing Enter, then selecting the partition number and choosing the swap partition type code (82).

g. Type w and press Enter to save the changes and exit fdisk.

4. Create the swap file system:

Run the following command to create a swap file system on the newly created swap partition

```
sudo mkswap /dev/sdX1
```

Replace /dev/sdX1 with the appropriate identifier for your swap partition.

5. **Enable the swap partition:**

Activate the swap partition using the following command:

```
sudo swapon /dev/sdX1
```

Again, replace /dev/sdX1 with your swap partition identifier.

6. **Make the swap partition persistent:**

Open the /etc/fstab file in a text editor (e.g., sudo nano /etc/fstab) and add the following entry at the end of the file to make the swap partition mount automatically on system startup:

```
/dev/sdX1 none swap sw 0 0
```

Save the file and exit the text editor.

7. **Verify the swap space:**

Run the command **free -h** to verify that the swap partition is active and available.

Congratulations! You have successfully created a swap partition in Linux. The system will now use the swap space on the designated partition when necessary to support virtual memory.