**Lecture 26**

**Kernel-Driver-Management**

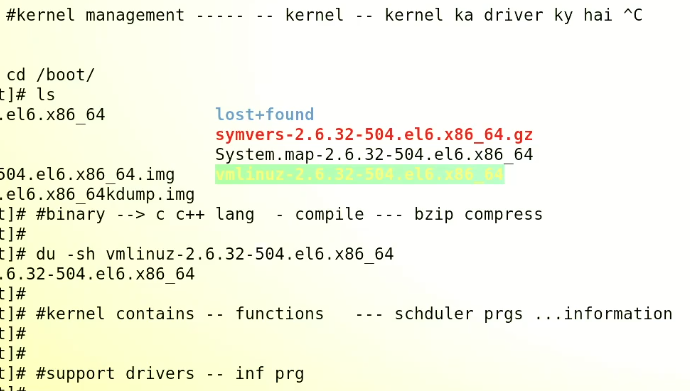
**What is a driver?**

In the context of Linux, a driver is a piece of software that facilitates communication between the operating system and a hardware device. It serves as an interface between the operating system and the hardware device, allowing the operating system to interact with the hardware and control its functions.

Linux drivers are typically written in the C programming language and are loaded into the operating system kernel as kernel modules. They provide a standardized way for the operating system to interact with different types of hardware, such as network adapters, graphics cards, and input/output devices.

- /boot

- Vmlinuz... file



In computing, a kernel is the central component of an operating system. It is a low-level software layer that provides essential services to enable other parts of the operating system and applications to communicate with the hardware of a computer.

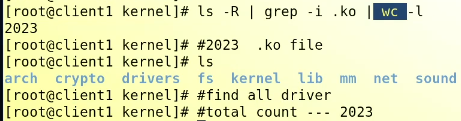
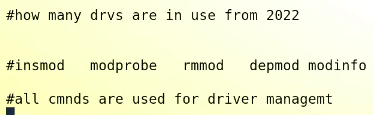
The kernel is responsible for managing system resources such as memory, CPU time, and input/output devices, as well as providing mechanisms for interprocess communication, system calls, and other low-level functions. It acts as a bridge between the software and hardware, providing a layer of abstraction that shields applications from the complexities of the underlying hardware.

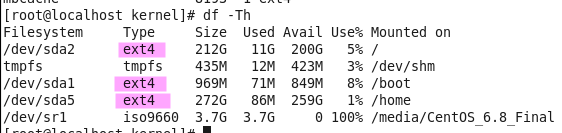
The kernel is typically loaded into memory when a computer is started, and remains there until the computer is shut down. It runs in privileged mode, which gives it direct access to the hardware and allows it to execute critical system functions.

Different operating systems use different kernels. For example, Linux uses the Linux kernel, while macOS uses the XNU kernel, and Windows uses the Windows NT kernel.

* The Linux kernel is primarily written in the C programming language, with some portions written in assembly language. The C language provides a good balance between performance, portability, and maintainability, making it a popular choice for system-level programming.
* Additionally, the Linux kernel is an open source project, which means that it is developed by a large community of volunteer developers from around the world. The source code is freely available for anyone to download, modify, and distribute, subject to the terms of the GNU General Public License (GPL). This open development model has led to a highly collaborative and transparent development process, where contributors can review, comment on, and improve the code contributed by others.

- Kernel is like an engine of a car, -->

* Doesn't store drivers (Kernel drivers, .ko --> Kernel Object) --\_ Loadable Kernel Module (LKM)-> drivers of motherboard, chipset etc , their size would be in Kbs --> stored in,
* 
* And these are all Kernel Drivers
* Inserting image...
* These r segregated directories containing drivers
* Lets open fs directory
* 
* By looking at these drivers we can know what type of drivers can be used in the Linux or any distro of Linux,
* 
* **ext4.ko,xz** --> it is the driver
* If it is removed, the ext4 partitions will not load and system will crash
* Because it mounts the ext4 partitions,
* How to check how many Kernel drivers are in the system ?
* $ ls – R | grep –i .ko | wc –l --> **go into every directory (-R) and grep every file with “.ko” extension and count it.**
* Another way is to use tree command --> yum install tree if package is not installed
* $ tree . --> go into every directory for .ko files 🡪 **$ yum install tree if the package is not installed**
* Explanation of “tree . “ command,
* The tree . command generates a directory tree of the current directory (represented by the .). The output of the command will show the directory structure, subdirectories, and files present in the current directory and its subdirectories.
* The output of the tree . command will vary depending on the contents of the directory and the operating system being used. Here's an example of what the output of the command might look like on a Unix-based system:
* .
* ├── file1.txt
* ├── file2.txt
* ├── folder1
* │ ├── file3.txt
* │ ├── file4.txt
* │ └── subfolder1
* │ └── file5.txt
* ├── folder2
* │ ├── file6.txt
* │ └── subfolder2
* └── folder3
* ├── file7.txt
* └── file8.txt
* 4 directories, 8 files
* In this example, the . directory contains 3 subdirectories **(folder1, folder2, and folder3)** and 2 files **(file1.txt and file2.txt).** The folder1 directory contains 1 subdirectory **(subfolder1)** and 2 files **(file3.txt and file4.txt)**. The subfolder1 directory contains 1 file **(file5.txt).** The folder2 directory contains 1 subdirectory **(subfolder2)** and 1 file **(file6.txt).** The subfolder2 directory is empty. Finally, the folder3 directory contains 2 files **(file7.txt and file8.txt).**
* The “find” command is also used for this purpose.
* $ find . -iname “\*.ko” | wc –l
* 
* How many drivers are in use?
* 
* Another one is -> **$ lsmod**
* $ **lsmod** show how many drivers are in RAM or being used
* 
* This shows ext4 is used by 3 programs
* Text

  Description automatically generated
* To check use $ df -Th
* 
* **If any partition is unmounted, the no of drivers being used will decrease, similary if a new volume of partition is attached which use ext4 fs 🡪 no of inuse ext4 will increase**
* **How to remove a driver from RAM**
* **$ rmmod <driver\_name> 🡪** 
* **To add a drived again bacy to RAM**
* **$ modprob <drivername>**

Here are some driver management commands in Linux:

1. **lsmod -** This command lists all currently loaded kernel modules (i.e., drivers) on the system.
2. **modprobe -** This command is used to add, remove, or manipulate kernel modules on the system. It can be used to load a module into the kernel, remove a module from the kernel, or set parameters for a module. (e.g a wifi driver or a sound card river)
3. **insmod -** This command is used to insert a kernel module into the running kernel.
4. **rmmod -** This command is used to remove a kernel module from the running kernel.
5. **depmod -** This command is used to create a dependency map of kernel modules and their dependencies.
6. **modinfo -** This command displays information about a kernel module, such as its name, version, author, and description.
7. **lsusb -** This command lists all USB devices connected to the system.
8. **lspci -** This command lists all PCI devices connected to the system.

These commands can be useful for managing and troubleshooting drivers on a Linux system, particularly if you need to load or unload specific drivers or determine which drivers are currently in use on the system. Note that some of these commands may require root or administrator privileges to run.

* **How to remove a driver permanently?**
* 
* **Locate the specific driver and remove it with “rm” from storage.**
* **$ moinfo <driver\_name> 🡪 information of a driver**
* 