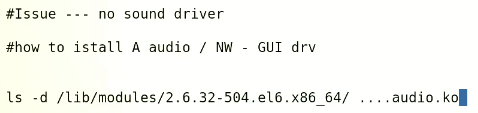
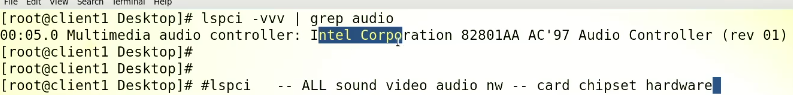
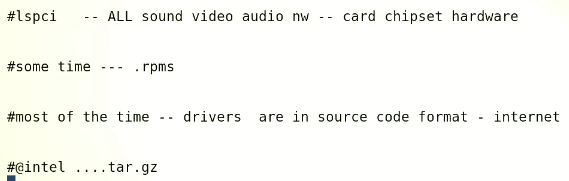
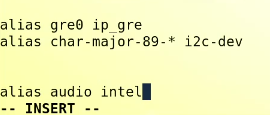
Lecture 27

Kernel-Management-Patching

* Drivers are in **/lib/modules/3.10.0-1160.6.1.el7.x86\_64/kernel** 🡪 CentOS 7
* how to install a driver in Linux (CentOS 7)?
* 
* The **dmesg** command in Linux is used to display the kernel ring buffer messages, which contain information about the system hardware, device drivers, and other low-level system information.
* When the Linux system boots up, it creates a ring buffer in memory to store kernel messages. The **dmesg** command reads these messages from the buffer and displays them on the terminal. The output of **dmesg** can help diagnose hardware and software problems, as well as provide insight into system performance and behavior.
* The **dmesg** command can also be useful in troubleshooting issues related to device drivers. For example, it can be used to determine if a device driver has loaded successfully or if there are any errors associated with a particular device.
* By default, the **dmesg** command shows the entire kernel ring buffer. However, it is possible to filter the output using various command-line options. For example, the **-T** option can be used to display timestamps for each message, and the **-l** option can be used to filter messages by log level (e.g., only show messages with a severity level of "warning" or higher).
* Overall, the **dmesg** command is a powerful tool for system administrators and developers to diagnose and troubleshoot issues with the Linux operating system.
* To install a driver first we need to know “chipset” of that specific hardware.
* .
* $ lspci 🡪 command used to list PCI devices information of the information related to the devices connected to a computer.
* The **lspci** command in Linux is used to display information about all PCI (Peripheral Component Interconnect) buses and devices connected to them on the system.
* PCI is a hardware interface standard used to connect devices like network adapters, sound cards, and graphics cards to a computer. The **lspci** command reads the configuration space of each PCI device and displays information such as the vendor and device IDs, the device class, the driver in use, and more.
* The output of the **lspci** command can be useful in identifying the hardware configuration of a system, as well as troubleshooting issues related to device drivers or hardware compatibility. For example, it can be used to determine the exact model of a network adapter, which can help in finding and installing the correct driver for that device.
* The **lspci** command can also be used in conjunction with other commands, such as **grep** or **awk**, to filter or manipulate the output. For example, you can use **lspci | grep -i network** to display only the information for network adapters.
* Overall, the **lspci** command is a useful tool for system administrators and developers to gather information about the PCI devices on a Linux system.
* 
* Drivers are available on internet –
* 
* At the end we need to have “.ko” file
* Graphical user interface, text, application

  Description automatically generated
* All the files are stored in /usr/local
* 
* After the “.ko” is available or driver file is to be loaded into the RAM
* $ insmod <driver\_.ko>
* And move to its proper location i.e 🡪 /lib/modules/3.6.4.6 …./ kernel/<specific\_directory>
* Another way is to use
* $ modprobe <driver\_.ko> 🡪 it is batter than insmod
* How to persist this driver?
* Makte entry in this file 🡪 $ vi /etc/modprobe.d/dist.conf
* 
* For CentOS 7

In CentOS 7, the **/etc/modprobe.conf** file is not present by default, but you can still add driver entries to the modprobe configuration.

1. Create a new file **/etc/modprobe.d/custom.conf** using a text editor such as nano or vi.
2. Add a line to the file that specifies the driver module you want to load. The format of the line should be:

**alias <device-name> <driver-module-name>**

Replace **<device-name>** with the name of the device you want to configure and **<driver-module-name>** with the name of the driver module.

For example, to load the **mydriver** module for a device named **mydevice**, the line would look like:

**alias mydevice mydriver**

1. Save the changes to the file and exit the text editor.
2. Run the following command to ensure that the kernel module dependencies are updated:

**sudo depmod -a**

1. Restart the system to apply the changes.

Once the system is restarted, the kernel module loader should automatically load the specified driver module when the device is detected.

* The **modules.dep** file is a file that contains a list of kernel modules and their dependencies. It is used by the kernel module loader to ensure that modules are loaded in the correct order and that all necessary dependencies are satisfied.
* In CentOS, the **modules.dep** file is located in the **/lib/modules/<kernel-version>/** directory. For example, if you are running the 4.18.0-305.el8.x86\_64 kernel, the **modules.dep** file would be located in **/lib/modules/4.18.0-305.el8.x86\_64/modules.dep**.
* The **modules.dep** file is automatically generated by the kernel when new modules are installed or when the kernel is updated. You should not edit this file manually as it could cause issues with the module loading process.
* If you need to update the module dependencies, you can run the following command:
* **sudo depmod -a <kernel-version>**
* Replace **<kernel-version>** with the version of the kernel you are running. This will regenerate the **modules.dep** file for the specified kernel version.

How to disable / blacklist Kernel drivers or modules?

To blacklist a kernel module or driver in CentOS 7, follow these steps:

1. Create a new file in the **/etc/modprobe.d/** directory using a text editor such as nano or vi. You can name the file anything you like, but it should end in **.conf**. For example, you could create a file called **blacklist.conf**.
2. Add a line to the file that specifies the module or driver you want to blacklist. The format of the line should be:

**blacklist <module-name>**

Replace **<module-name>** with the name of the module or driver you want to blacklist.

For example, to blacklist the **nouveau** driver for NVIDIA graphics cards, you would add the following line:

**blacklist nouveau**

1. Save the changes to the file and exit the text editor.
2. Run the following command to ensure that the kernel module dependencies are updated:

**sudo depmod -a**

1. Reboot your system for the changes to take effect.

Once your system has restarted, the blacklisted module or driver will not be loaded by the kernel, even if it is present on the system.

* Additional information related to modeprobe.d

The **modprobe.d** directory is a directory in CentOS (and other Linux distributions) that contains configuration files for the modprobe utility. The modprobe utility is used to manage kernel modules, including loading and unloading them.

The files in the **modprobe.d** directory are used to configure the modprobe utility, including specifying module options and configuring module aliases.

To create a new configuration file in the **modprobe.d** directory in CentOS 7, follow these steps:

1. Open a terminal window and switch to the root user by typing the following command and entering the root password:

**su -**

1. Navigate to the **/etc/modprobe.d/** directory using the following command:

**cd /etc/modprobe.d/**

1. Create a new configuration file in the **modprobe.d** directory using a text editor such as nano or vi. You can name the file anything you like, but it should end in **.conf**. For example, you could create a file called **myconfig.conf**.

**nano myconfig.conf**

1. Add your configuration options to the file, using the appropriate syntax for the option you want to set. For example, to set an option for the **snd-hda-intel** module, you could add the following line to the file:

**options snd-hda-intel model=auto**

1. Save the changes to the file and exit the text editor.
2. Run the following command to ensure that the kernel module dependencies are updated:

**depmod -a**

1. Reboot your system for the changes to take effect.

Once your system has restarted, the modprobe utility will load the specified module options when the associated modules are loaded.

* .
* Visualize the Kernel

In CentOS 7, the /boot directory contains the files necessary to boot the system. It is a mount point for the partition that holds the kernel and initial ramdisk (initramfs) files that are loaded during the boot process.

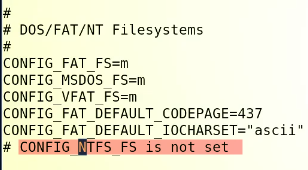
Here's a brief explanation of the files that are typically found in the /boot directory:

1. vmlinuz: This is the Linux kernel image file. It contains the code necessary to start up the Linux operating system.
2. initramfs: This is the initial ramdisk image file. It contains the necessary drivers and modules to load the Linux kernel and boot the system.
3. System.map: This file contains a symbol table that maps the names of kernel functions and variables to their memory addresses.
4. config: This file contains the configuration options used to build the Linux kernel. It is generated during the kernel compilation process.
5. grub: This directory contains the bootloader configuration files. The GRUB bootloader is responsible for loading the kernel and initramfs files during the boot process.

In summary, the /boot directory holds the essential files for booting the system, including the kernel, initial ramdisk, bootloader configuration files, and other system files.

Top of Form

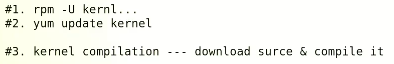
Bottom of Form

* Suppose there is a car named CentOS 7 and the Kernel is the engine in this car
  + There could be multiple engines or Kernels in this car 🡪 the upgraded engine.
  + But in there can only be one engine will be active and in working position
  + The Kernel file is loaded into the RAM and extracted.
  + /proc directory holds all the files extracted by the Kernel main file. (house of running Kernel)
  + The “initramfs” file mounts root partition at the time of boot.
  + Config file is just like a manual for Kernel.
    1. If we explore the config file,
    2. It shows that the Kernel doesn’t support NTFS.
    3. 
    4. How to enable NTFS Support in Linux?
    5. 
    6. We need to install ntfs.ko file 🡪 or driver or Kernel Module of NTFS
    7. 
    8. $ yum update kernel
  + 🡪 shows which file package installed Kernel package.
  + To update Kernel $ yum update kernel or $ rmp -U zsh
  + 🡪 this is Kernel patching. 🡪 remember if the Kernel is updated the previous one is not overwritten. The previous Kernel file remains in the directory.

When a kernel is upgraded in a Linux system, several changes occur in the **/boot** directory, including:

1. A new kernel image file is added: A new kernel image file is added to the **/boot** directory with a new version number. The new file is typically named **vmlinuz-version** and is stored alongside the previous kernel image files.
2. A new initramfs image file is added: An initramfs (initial RAM file system) image file is also generated and added to the **/boot** directory with a new version number. The initramfs image file contains a minimal file system that is used by the kernel during the boot process to load necessary drivers and modules.
3. A new System.map file is added: A new System.map file is generated and added to the **/boot** directory with a new version number. This file contains the symbols and function names used by the kernel and is used by system tools for debugging and troubleshooting.
4. A new configuration file is added: A new configuration file is added to the **/boot** directory that contains the configuration options used when compiling the new kernel.
5. Grub configuration is updated: The GRUB (Grand Unified Bootloader) configuration file, which is usually located in the **/boot/grub** directory, is updated to include the new kernel and initramfs image file. This allows the system to boot using the new kernel during the next boot process.

Overall, when a kernel is upgraded, the **/boot** directory is updated to include the necessary files for the new kernel version, and the GRUB configuration is updated to include the new kernel version in the boot menu.

* At the time of boot there would be 2 Kernels visible -🡪 which one is the running, Kernel?
* $ uname -r
* Kernels can’t switch while in OS mode.
* In order to switch reboot is must
* Ways to update Kernel,
* 

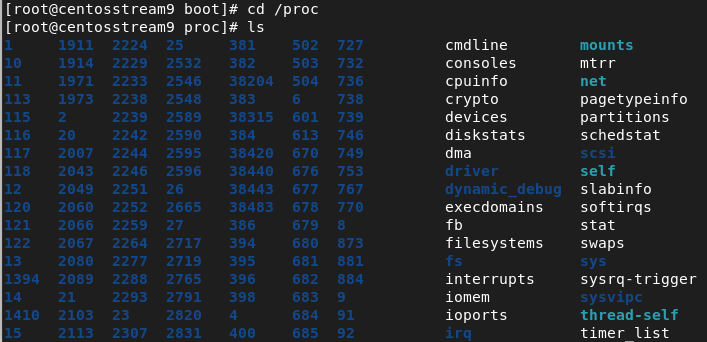
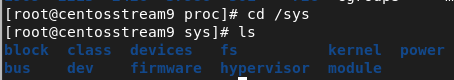
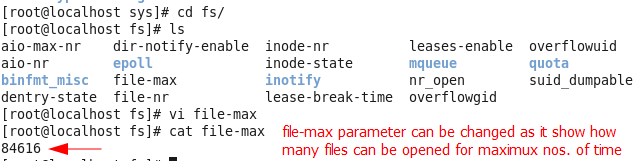
Kernel Tuning (interview topic)

* (It means) to change few parameters of Kernel. /to change values of running Kernel
* /proc 🡪 everything related to RAM
* Kernel is also loaded into RAM while running. 🡪 Kernel tuning is also done in /proc 🡪 it will be temporary.
* There r many types of Kernels
  + Monolethic Kernel / called modular user in Linux
    1. Drivers can be loaded in real time 🡪 $ modprobe
  + Micro Kernel – used in soleris /AIX/ HP -UniX
    1. Drivers can’t be loaded in real time.
  + .

In the context of Linux, a kernel refers to the core component of the operating system that manages system resources and provides a platform for running applications. Here are some types of Linux kernels:

1. Monolithic Kernel: A monolithic kernel is a type of kernel where all the core operating system functions are included in a single binary file. This type of kernel is relatively simple and efficient, but it can be difficult to modify and customize.
2. Microkernel: A microkernel is a type of kernel that only includes the most essential operating system functions, such as memory management and process scheduling. Additional functionality, such as device drivers and file systems, are implemented as separate modules that run in user space. This type of kernel is more modular and customizable than a monolithic kernel, but it can be less efficient due to the overhead of inter-process communication.
3. Hybrid Kernel: A hybrid kernel is a type of kernel that combines elements of both monolithic and microkernel designs. Like a monolithic kernel, a hybrid kernel includes many operating system functions in the kernel itself. However, it also allows some functionality, such as device drivers, to run in user space. This type of kernel aims to strike a balance between simplicity and flexibility.
4. Real-time Kernel: A real-time kernel is a type of kernel that is designed for systems that require predictable and consistent response times, such as industrial control systems and embedded devices. Real-time kernels typically prioritize certain types of tasks over others to ensure that critical operations are executed quickly and reliably.

These are some of the main types of kernels used in Linux. Each type has its own advantages and disadvantages, and the choice of kernel depends on the specific requirements of the system.

* For Kernel tuning 🡪 /proc
* 
* These values can’t be changed inside /proct 🡪 it can be done in “sys”
* $ /proc/sys 🡪 files values and parameters can be changed.
* 
* Go into “fs” directory.
* 
* Another command is used $ sysctl -a 🡪 show which parameters or values can be changed.
*  🡪 719 values can be tuned or changed
* The Kernel tuning in CentOS 9 is bit different, (in lecture Sir Kazim used CentOS 6)

Kernel tuning in CentOS 9 involves modifying kernel parameters to optimize the performance and behavior of the operating system. Here are some steps you can take to tune the kernel in CentOS 9:

1. Check the current kernel parameters: Before making any changes, it's a good idea to check the current kernel parameters using the **sysctl** command. This command displays the current kernel parameters and their values.
2. Modify kernel parameters: You can modify kernel parameters by editing the **/etc/sysctl.conf** file. This file contains a list of kernel parameters and their values. To modify a parameter, you can add or modify a line in the file that specifies the parameter and its new value.
3. Apply the changes: After modifying the kernel parameters, you need to apply the changes using the **sysctl** command. You can either reboot the system to apply the changes at boot time or run the **sysctl -p** command to apply the changes immediately.
4. Monitor system performance: After applying the kernel changes, it's important to monitor system performance to ensure that the changes are having the desired effect. You can use system monitoring tools such as **top**, **vmstat**, and **sar** to monitor system resource usage and identify any performance issues.

Some common kernel parameters that you might want to modify include:

* swappiness: This parameter controls the degree to which the kernel swaps out processes to disk. A lower value means that the kernel will swap out processes less aggressively, which can improve performance for memory-intensive applications.
* file-max: This parameter controls the maximum number of open file descriptors that the system can handle. Increasing this parameter can improve the performance of applications that require a large number of open file descriptors.
* tcp\_tw\_reuse: This parameter controls whether the kernel can reuse TCP connections that are in the TIME\_WAIT state. Enabling this parameter can improve the performance of applications that use a large number of short-lived connections.

These are just a few examples of the many kernel parameters that you can modify to optimize the performance of CentOS 9. It's important to carefully evaluate the impact of any changes and monitor system performance to ensure that the changes are having the desired effect.