Contents

Checking Linux System Stats

Check Kernel Version

Show kernel version	uname -r
Show full system info (includes kernel)	uname -a
Show detailed kernel and OS info	hostnamectl

Check Linux Distro and Version

Standard Oracle Linux / RHEL release info	cat /etc/os-release
Another detailed option	cat /etc/redhat-release
Another fallback (older systems)	cat /etc/issue

Check Enabled Repositories

List all enabled repos	dnf repolist
List all repos (enabled + disabled)	dnf repolist all
(If using older yum) List repos	yum repolist
Show full repository info	dnf repository-packages <repo-name> list

List All Services and Search Specific Services

List all services (systemd)	systemctl list-units --type=service
List all loaded and inactive services	systemctl list-units --type=service --all
Search for a specific service (ex: sshd)	` systemctl list-units --type=service
Check status of specific service	systemctl status <service-name>

List All Processes and Search Specific Process

List all running processes	ps aux
Search for a specific process (ex: sshd)	ps aux | grep sshd
Dynamic real-time process view	top
More advanced view	htop (you may need to install: dnf install htop)

Show Current User

Show current user	whoami
Show who is logged in	who
Show login session details	w

Show Firewall Status and Rules

Check firewall status	firewall-cmd --state
List all active firewall rules	firewall-cmd --list-all
List all zones	firewall-cmd --get-zones
List rules for a specific zone (ex: public)	firewall-cmd --zone=public --list-all

Alternative (if using iptables manually) show raw rules	iptables -L -n -v

List All Users

View all users (system + normal)	cat /etc/passwd

List All Groups

View all groups	cat /etc/group
View current user’s groups	groups
View another user's groups	groups <username>

Oracle Linux Administration Command Summary

System Information

Kernel version	uname -r
Full system info	uname -a
OS version	cat /etc/os-release
Hostname	hostnamectl
Uptime	uptime
Check IP address	ip a or hostname -I
Check CPU info	lscpu
Check memory info	free -h
Disk usage	df -h
Partition layout	lsblk

Package and Repository Management

Install a package	dnf install <package>
Remove a package	dnf remove <package>
Update system	dnf update
Search for a package	dnf search <package>
List enabled repositories	dnf repolist
Clean package cache	dnf clean all

Service and Process Management

List running services	systemctl list-units --type=service --state=running
List all services	systemctl list-units --type=service --all
Start a service	systemctl start <service>
Stop a service	systemctl stop <service>

Enable service on boot	systemctl enable <service>
Disable service from boot	systemctl disable <service>
Check service status	systemctl status <service>
List processes	ps aux
Search a process	ps aux | grep sshd
Real-time process monitor	top or htop

User and Group Management

Current user	whoami
List logged-in users	who
List all users	cat /etc/passwd
List all groups	cat /etc/group
Add new user	useradd <username>
Set user password	passwd <username>
Delete user	userdel <username>
Add user to group	usermod -aG <group> <user>

Firewall and Security

Firewall status	firewall-cmd --state
List firewall rules	firewall-cmd --list-all
Reload firewall	firewall-cmd --reload
Open a port (ex: 80/tcp)	firewall-cmd --permanent --add-port=80/tcp
Close a port	firewall-cmd --permanent --remove-port=80/tcp
List SELinux status	sestatus
Set SELinux to permissive (temporary)	setenforce 0
Set SELinux to enforcing (temporary)	setenforce 1

File System and Storage

Disk usage	df -h
Directory size	du -sh /path/to/dir
List block devices	lsblk
Mount a filesystem	mount /dev/sdX /mnt
Unmount a filesystem	umount /mnt
View fstab entries	cat /etc/fstab
Check disk I/O activity	iostat (requires sysstat package)

Logs and Troubleshooting

View system logs	journalctl
View boot logs	journalctl -b
View specific service logs	journalctl -u <service>

Check last system reboot	last reboot
System messages (old method)	cat /var/log/messages

System Backup and Restore

Create tar backup	tar -czvf backup.tar.gz /path/to/data
Extract tar backup	tar -xzvf backup.tar.gz
Copy files (preserving permissions)	cp -a source/ destination/
Rsync (synchronize files)	rsync -avh /source/ /destination/

Networking

View IP addresses	ip a
View routing table	ip route
Test network connectivity	ping <hostname>
Test DNS resolution	dig <hostname>
Check open ports	ss -tuln
Traceroute a network path	traceroute <hostname> (install if needed)

You can create a quick health report:

```
hostnamectl; uname -r; free -h; df -h; dnf repolist; systemctl list-units --type=service --state=running;
firewall-cmd –state
```

Designing the Disk Partitioning Scheme for a Linux System

Disk partitioning is about dividing a hard disk into sections, where each section can be treated as a separate "disk" by the OS.

Basic Linux partitions commonly created:

Mount Point	Purpose
/boot	Holds the kernel and early boot files
/ (root)	The main system partition (everything under the filesystem tree)
/home	Where user files (documents, downloads) are kept
/var	Stores logs, databases, websites, etc.
/tmp	Temporary files
swap	Virtual memory

Typical Simple Partitioning (Small Server Example)

Partition	Size Suggestion	Notes
/boot	1–2 GB	Separate /boot is recommended
/ (root)	20–50 GB	OS binaries, applications
/home	Rest of the space (or LVM)	For users

swap	See below for sizing	Virtual memory

More Complex Partitioning (Large Server Example)

Partition	**Size Suggestion**
/boot	1–2 GB
/ (root)	30–50 GB
/var	Separate 10–20 GB (or more if logging/databases)
/tmp	5–10 GB
/home	Separate large partition
swap	Based on RAM

Key Design Tips:

- Separate /var if running heavy logging, mail servers, databases (prevents / from filling up).
- Separate /home if multiple users exist (keeps user data isolated).
- Use Logical Volume Manager (LVM) to make disk resizing easier later
- Always create a small, separate /boot partition (important for some architectures like BIOS/UEFI systems).

Ensuring the /boot Partition Conforms to Hardware Architecture Requirements

What /boot needs to match:

Architecture	/boot Notes
BIOS/MBR systems	/boot must be on a standard partition, early in the disk (below 2 TB for old BIOS)
UEFI systems	You need a EFI System Partition (ESP) mounted at /boot/efi, formatted as FAT32 (not ext4!)

Size Recommendations:

/boot	1 GB to 2 GB (ext4 filesystem)
/boot/efi (for UEFI systems)	300 MB - 600 MB (FAT32 filesystem)

Commands to check:

```
lsblk
mount | grep boot
```

Concept of Swap Space

Swap space is a reserved area on disk used when RAM is full. Think of it like virtual RAM.

Swap Uses:

- When the system RAM is fully used, swap temporarily holds inactive memory pages.
- Helps prevent crashing due to "out of memory" errors.
- Not a full substitute for RAM! (it's slower — disk access is way slower than RAM)

How much Swap to allocate?

RAM Size	Recommended Swap
< 2 GB	2x RAM
2–8 GB	Equal to RAM (1x)
8–64 GB	0.5x RAM
> 64 GB	4–32 GB fixed

If using hibernate (suspend to disk) — swap must be at least the size of RAM.
Check swap:

```
swapon --show
free -h
```

Create swap manually:

```
# Create swap file of 2G
dd if=/dev/zero of=/swapfile bs=1G count=2
chmod 600 /swapfile
mkswap /swapfile
swapon /swapfile
# To make it permanent
echo '/swapfile swap swap defaults 0 0' >> /etc/fstab
```

Understanding LVM (Logical Volume Manager)

LVM allows you to manage disk space more flexibly.

- Without LVM: Partitions are fixed. Hard to resize or move.
- With LVM: You can resize, move, combine disks easily without downtime.

How LVM Works:

Component	Description
Physical Volume (PV)	Real disk/partition initialized for LVM (e.g., /dev/sdb1)
Volume Group (VG)	A pool of storage made by combining one or more PVs
Logical Volume (LV)	"Virtual" partitions created from VG. These are like your /home, /var, etc.

Diagram:

```
Physical Disk -> Physical Volume (PV) -> Volume Group (VG) -> Logical Volumes (LVs)
```

LVM Commands:

Task	Command
Create a PV	pvcreate /dev/sdb1
Create a VG	vgcreate my_vg /dev/sdb1
Create an LV	lvcreate -L 10G -n my_lv my_vg
List LVM setup	pvs, vgs, lvs
Resize LV	lvextend -L +5G /dev/my_vg/my_lv + resize2fs /dev/my_vg/my_lv

Simple Example of LVM Setup

1. Prepare the physical volume

```
pvcreate /dev/sdb
```

2. Create a volume group

```
vgcreate my_vg /dev/sdb
```

3. Create a logical volume

```
lvcreate -L 10G -n my_home my_vg
```

4. Make filesystem

```
mkfs.ext4 /dev/my_vg/my_home
```

5. Mount it

```
mount /dev/my_vg/my_home /home
```

Booting the System

The BIOS Boot Sequence

The classic BIOS boot sequence goes like this:

Step	What Happens
1. Power ON	Power is supplied to the motherboard (cold boot).
2. POST (Power-On Self-Test)	BIOS checks hardware (CPU, RAM, disks, keyboard, etc.).
3. Find bootable device	BIOS reads boot order (HDD, CD, USB, etc.) and finds boot sector.
4. Load bootloader (like GRUB)	BIOS loads the first 512 bytes (MBR - Master Boot Record) into memory.
5. Handoff to bootloader	Bootloader (e.g., GRUB) takes over and loads the OS kernel.

- Modern systems use UEFI instead of BIOS.
- UEFI loads the EFI System Partition (/boot/efi) and a bootloader (like GRUB2) from there.

Introduction to Linux Boot Events

Once the bootloader starts loading Linux, the sequence is:

Step	Event
1. Load Linux kernel	GRUB loads the kernel into memory.
2. Load initial RAM filesystem	initramfs or initrd — temporary root filesystem needed early.
3. Mount real root filesystem	After drivers/modules are ready.
4. Start init system	(SysVinit, SystemD, or Upstart) to launch services and targets.
5. Reach a usable system	User can log in (multi-user or GUI mode).

Important utilities:

dmesg	View kernel messages from early boot and device detection
journalctl -k	Also shows kernel boot logs if using systemd
initramfs	Initial root filesystem loaded into memory (critical drivers, files)
/boot/initramfs-<kernel_version>.img	Actual initramfs image

Check boot messages:

```
dmesg | less
```

or

```
journalctl -b
# (-b = show logs from current boot)
```

Common Commands to the Bootloader (GRUB)

At the GRUB boot screen, you can:

Task	How
Edit kernel parameters temporarily	Press e key on the selected boot entry
Boot manually after editing	Press Ctrl + X or F10
Get into GRUB command line	Press c
Set boot timeout	Edit /etc/default/grub and update GRUB
Rebuild GRUB menu after changes	grub2-mkconfig -o /boot/grub2/grub.cfg

Example GRUB edit:

a. At boot ➔ highlight a kernel ➔ Press e
b. Find the line starting with linux and append kernel parameters like:

```
linux /vmlinuz-xxx root=/dev/sda2 ro quiet crashkernel=auto
```

c. Press Ctrl + X to boot with those changes.

Kernel Options at Boot Time

You can pass extra options to the kernel during boot via GRUB.

Example Option	Purpose
single	Boot into single-user mode (emergency maintenance)
init=/bin/bash	Bypass init system, drop directly to bash
selinux=0	Disable SELinux temporarily
nomodeset	Disable graphics drivers during boot (useful if graphical boot issues)
rescue	Boot into rescue mode
ro	Mount root filesystem read-only (safe)
rw	Mount root filesystem read-write

SysVinit, SystemD, UpStart

These are the init systems — the first real program started by the Linux kernel.

Init System	Notes
SysVinit	Old traditional init system. Uses /etc/inittab and scripts under /etc/rc.d/.
Upstart	Introduced by Ubuntu. Event-driven. Now mostly replaced by systemd.
SystemD	Modern init system. Used by Oracle Linux 7, 8, 9. Parallel service startup, journaling, targets, unit files.

To check what init system you're using:

```
ps -p 1 -o comm=
```

(Shows the process at PID 1 — likely systemd or init)

If it's systemd, you will mainly work with:

- systemctl commands
- journalctl logs
- /etc/systemd/ configs

Changing Runlevels, Boot Targets, Single User Mode

In SysVinit: Linux runlevels are numbers (0-6):

Runlevel	Purpose
0	Halt (shutdown)
1	Single-user (rescue)
3	Multi-user (no GUI)
5	Multi-user + GUI (graphical)
6	Reboot

In SystemD: We use targets instead:

Target Purpose

rescue.target	Single-user mode
multi-user.target	Multi-user mode (text mode)
graphical.target	Multi-user + GUI

Check current target:

```
systemctl get-default
```

Change target (temporary):

```
systemctl isolate rescue.target
```

Set target permanently:

```
systemctl set-default graphical.target
```

Boot into single user mode temporarily:

a. At GRUB screen, press e
b. Append *single* or systemd.unit=rescue.target at the end of linux line
c. Press Ctrl+X to boot

Reboot and Shutdown Commands (With User Alerts)

Graceful shutdown/reboot (notifying users):

Task	Command
Reboot immediately	reboot or shutdown -r now
Shutdown immediately	shutdown -h now
Schedule reboot in 5 minutes	shutdown -r +5
Broadcast a custom message	shutdown -r +10 "Server rebooting for maintenance!"
Cancel a scheduled shutdown/reboot	shutdown -c

Example:

```
shutdown -r +15 "System will reboot in 15 minutes. Please save your work."
```

(All logged-in users will see the broadcast warning.)

The Linux Terminal / Command Line

Common Linux Terminal Commands

bash

Starts a Bash shell session (Bourne Again SHell).

```
bash
```

Exit the new bash:

```
exit
```

echo

Prints text or variables to the terminal.

```
echo [options] [string]
```

Examples:

Print a simple message:

```
echo "Hello, world!"
```

Print the value of a variable:

```
echo $HOME
```

Print text to a file:

```
echo "This is a test" > file.txt
```

Append text to a file:

```
echo "Add another line" >> file.txt
```

env

Lists all environment variables currently active.

```
env
```

*You'll see variables like PATH, HOME, USER, etc.

Get value of a specific environment variable:

```
echo $PATH
```

export

Sets environment variables — makes them available to child processes.

```
export VARIABLE_NAME=value
```

Examples:

Create and export a variable:

```
export MY_VAR="HelloWorld"
```

Now MY_VAR is available to any scripts or programs you run.

Check:

```
echo $MY_VAR
```

pwd

Prints the current working directory.

```
pwd
```

set

Lists all shell variables and functions — very detailed view of your environment.

```
set
```

Common usage:

Setting shell options:

```
set -x
```

(Debug mode — shows commands as they are executed.)

Disable debug:

```
set +x
```

unset

Removes a shell variable or function.

```
unset VARIABLE_NAME
```

Example:

Set and then unset a variable:

```
export TESTVAR="something"
echo $TESTVAR
unset TESTVAR
echo $TESTVAR
```

*After unset, TESTVAR is no longer available.

man

Displays the manual page for a command.

```
man command_name
```

Example:

```
man ls
```

*View the manual for ls.

uname

Prints system information.

```
uname [options]
```

Examples:

Print kernel name:

```
uname
```

Print full system info:

```
uname -a
```

Sample Output

```
Linux oraclelinux9 5.15.0-1032.el9.x86_64 #1 SMP Tue Jan 23 21:20:00 UTC 2024 x86_64 x86_64
x86_64 GNU/Linux
```

Other useful flags:

```
-r      Kernel release
-m      Machine hardware (architecture)
```

history
Shows the history of previously run commands.

```
history
```

You'll see an output like:

```
  1  ls -l
  2  cd /etc
  3  cat /etc/os-release
```

Repeat a specific command:

```
!3
```

(runs command #3)

.bash_history
The file that stores the command history for Bash.

```
~/.bash_history
```

(~ = your home directory)

.bash_profile

- .bash_profile is a startup script that is executed when you log in to a Linux system via a login shell.
- It sets environment variables (like PATH, EDITOR, etc.)
- It runs commands you want when the user first logs in (like starting services, setting colors, aliases, etc.)

It is specific to each user and is located in the user's home directory:

```
~/.bash_profile
```

Only login shells read .bash_profile.
Examples of login shells:

- SSH sessions
- Console login (Ctrl + Alt + F2, etc.)
- Logging into the system via graphical login that starts a shell

Typical contents of .bash_profile:

```
# Set PATH
export PATH=$PATH:$HOME/.local/bin:$HOME/bin

# Set a default editor
export EDITOR=vim

```

```
# Set environment variables
export HISTSIZE=1000
export HISTFILESIZE=2000

# Load .bashrc (non-login shell settings)
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

Determining attached hardware and Computer peripherals

How Linux Detects Hardware (sysfs, udev, dbus)

When Linux boots:

a. The kernel detects hardware (CPU, disks, network cards, USBs, etc.).
b. It uses drivers (modules) to control devices.
c. It shows hardware in the /sys (sysfs) and /proc virtual filesystems.

Important Components:

sysfs (/sys)	Virtual filesystem exposing hardware information.
udev	Dynamic device manager. Creates device files (like /dev/sda) automatically.
dbus	Message bus system that lets processes (like desktop apps) talk to devices and react to hardware events.

Example:
When you plug a USB drive:

a. Kernel detects it.
b. udev creates /dev/sdb1.
c. dbus notifies graphical apps (like "Removable device detected").

Use of Linux Tools to List Hardware Information

Tool	Purpose	Command
lscpu	CPU details	lscpu
lsblk	List block devices (disks)	lsblk
lspci	List PCI devices (network cards, graphics)	lspci
lsusb	List USB devices	lsusb
dmidecode	BIOS, motherboard, memory info	sudo dmidecode
hwinfo	Very detailed hardware report	sudo hwinfo (install if missing)
inxi	Easy full summary	inxi -Fx (install if missing)

Example:
shows disks and partitions.

```
lsblk
```

shows Ethernet network cards.

```
lspci | grep -i ethernet
```

Tools to Manipulate USB Devices (like modprobe)

modprobe — used to load or remove kernel modules (drivers).

Load a module	sudo modprobe module_name
Remove a module	sudo modprobe -r module_name
List loaded modules	lsmod

Example:
Load the USB storage driver manually:

```
sudo modprobe usb-storage
```

Remove it:

```
sudo modprobe -r usb-storage
```

Check loaded modules:

```
lsmod
```

Manually Mount and Unmount Filesystems

To mount a device (attach it to the filesystem):

Create mount point	sudo mkdir /mnt/myusb
Mount device	sudo mount /dev/sdb1 /mnt/myusb
View mounted filesystems	mount or findmnt

Example:

```
sudo mkdir /mnt/usbdrive
sudo mount /dev/sdb1 /mnt/usbdrive
```

*Now you can access files under /mnt/usbdrive.

To unmount a device:

Unmount	sudo umount /mnt/myusb

Or if you only know device:

```
sudo umount /dev/sdb1
```

Configuring Filesystems to Automatically Mount at Boot

Use the /etc/fstab file.

- fstab = file systems table.

Example entry in /etc/fstab:

```
/dev/sdb1  /mnt/usbdrive  ext4  defaults  0 0
```

Field	Meaning
Device	/dev/sdb1
Mount point	/mnt/usbdrive
Filesystem type	ext4, xfs, vfat, etc.
Options	defaults, noatime, rw, etc.
Dump	Backup option (usually 0)
Pass	Filesystem check order (0 = no check)

After editing /etc/fstab, test it without rebooting:

```
sudo mount -a
```

Sample /etc/fstab for USB:

```
UUID=xxxx-yyyy /mnt/usbdrive vfat defaults,noauto 0 0
```

(UUID can be found using blkid)

Tip: Use UUIDs instead of device names (/dev/sdb1) — safer because device names can change!

Package Management Overview

In Linux, Package Managers automate installing, updating, and removing software.

Package Type	System Example	Manager
.deb	Debian, Ubuntu	apt, dpkg
.rpm	Red Hat, Oracle Linux, Fedora	yum, dnf, rpm

Linux Filesystem Structure

Directory	**Purpose**
/	Root of everything
/bin	Essential system commands (ls, cp, mv)
/sbin	Essential system administration commands
/usr	User installed software
/var	Variable data (logs, spools)
/tmp	Temporary files
/etc	Configuration files
/home	User directories
/lib	Shared libraries for programs
/boot	Kernel and bootloader files
/dev	Device files (disks, terminals)
/proc	Virtual filesystem for system info
/sys	Kernel and hardware info (sysfs)

Managing Shared Libraries

Linux programs rely on shared libraries (.so files) to save space and memory.

Tool/Concept	Purpose
ldd	Show shared libraries required by a binary
ldconfig	Update the cache of shared libraries
/etc/ld.so.conf	Lists directories for dynamic linker to search for libraries
LD_LIBRARY_PATH	Environment variable to add custom library paths temporarily

Example Commands:

Check required libraries:

```
ldd /bin/ls
```

Manually update library cache:

```
sudo ldconfig
```

Temporarily set library path:

```
export LD_LIBRARY_PATH=/opt/myapp/lib:$LD_LIBRARY_PATH
```

Debian Package Management

Tools for .deb packages (Debian/Ubuntu systems):

sources.list

File where software sources (repositories) are defined:

```
cat /etc/apt/sources.list
```

Example line output:

```
deb http://deb.debian.org/debian stable main
```

apt-get

Command-line tool to install, remove, and manage packages.

Update package list	sudo apt-get update
Upgrade all packages	sudo apt-get upgrade
Install a package	sudo apt-get install package-name
Remove a package	sudo apt-get remove package-name

dpkg

Low-level tool to install .deb files manually.

Install a .deb package	sudo dpkg -i package.deb
Remove a package	sudo dpkg -r package-name
List installed packages	dpkg -l

apt-cache

Query the APT cache for package information.

Search a package	apt-cache search keyword
Show package details	apt-cache show package-name

aptitude

Text-based front-end for APT (optional but powerful).

Launch aptitude UI	sudo aptitude
Install a package	sudo aptitude install package-name

RPM and YUM/DNF Package Management (RHEL / Oracle Linux)

Tools for .rpm packages:

/etc/yum.conf and /etc/yum.repos.d/

- /etc/yum.conf → main yum configuration.
- /etc/yum.repos.d/ → stores individual repository .repo files.

Example repo file:

```
cat /etc/yum.repos.d/oraclelinux.repo
```

Contains:

```
[ol9_baseos_latest]
name=Oracle Linux 9 BaseOS Latest ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL9/baseos/latest/$basearch/
enabled=1
gpgcheck=1
```

yumdownloader

Tool to download RPM packages without installing them.

Install yum-utils (if missing)	sudo dnf install yum-utils
Download a package	yumdownloader package-name

Example:
Saves the RPM file locally.

```
yumdownloader bash
```

rpm and rpm2cpio

rpm	Install, query, remove RPM packages manually
rpm2cpio	Convert RPM to a CPIO archive (to extract files without installing)

Common rpm examples:

Install RPM manually:

```
sudo rpm -i package.rpm
```

Remove a package:

```
sudo rpm -e package-name
```

Query if package installed:

```
rpm -q package-name
```

List files inside an installed package:

```
rpm -ql package-name
```

rpm2cpio + cpio example (extract contents):

```
rpm2cpio package.rpm | cpio -idmv
```

(Extracts RPM contents without installing.)

Creating and Changing Symbolic and Hard Links (ln vs ls)

ln	create links.
ls	list files/directories (different tool).

Hard Link (ln)

Points directly to the same inode (actual data).

If original is deleted, hard link still works.

```
ln file1.txt file1_hardlink.txt
```

*Now file1_hardlink.txt is another name for file1.txt.

Symbolic (Soft) Link (ln -s)

Points to the filename.

If original is deleted, symlink breaks ("dangling link").

```
ln -s file1.txt file1_symlink.txt
```

*file1_symlink.txt points to file1.txt.

List links:

```
ls -l
```

(you'll see links as ->)

Basic Linux File Management

Common Archiving Tools

tar	Combine many files into one archive
gzip	Compress a file
bzip2	Compress with better compression
zip/unzip	Create/extract zip files

Example:

Create a tar.gz archive:

```
tar -czvf archive.tar.gz /path/to/dir
```

Extract:

```
tar -xzvf archive.tar.gz
```

Making Directories

```
mkdir mydir
mkdir -p parentdir/childdir
```

(-p creates parent directories as needed)

Copy, Move, Delete Files and Directories

Copy file	cp file1.txt file2.txt
Copy directory	cp -r dir1/ dir2/
Move/rename file	mv oldname.txt newname.txt
Move directory	mv dir1/ dir2/
Delete file	rm file.txt
Delete directory	rm -r dir1/

Finding and Locating Files

find	Powerful search by name, type, size, etc.
locate	Very fast database-based search
updatedb	Updates the locate database
whereis	Find binary, source, and man pages
which	Show the full path of a command
type	Show how a command will be interpreted

Examples:

Find file by name:

```
find /home -name "*.txt"
```

Locate file (fast):

```
locate file1.txt
```

Update locate database:

```
sudo updatedb
```

Where a command is stored:

```
which bash
```

Check what a command is:

```
type ls
```

Output Redirection and Text Stream Modification

Useful GNU textutils commands:

cat	View or concatenate files
cut	Extract sections from each line
sort	Sort lines
expand	Convert tabs to spaces
join	Join lines from two files
split	Split a file into pieces
wc	Word/line/char count
sed	Stream editor for search/replace
tail	Show last lines of a file

Examples:

View a file:

```
cat file.txt
```

Count lines:

```
wc -l file.txt
```

Sort a file:

```
sort names.txt
```

Tail last 10 lines:

```
tail -n 10 log.txt
```

Replace text:

```
sed 's/oldword/newword/g' file.txt
```

Use of Streams, Pipes, and Redirects

Streams:

- STDIN (input)
- STDOUT (output)
- STDERR (error)

Redirection Examples:

Redirect output to file	ls > filelist.txt
Append output to file	ls >> filelist.txt
Redirect errors	command 2> error.log
Redirect both output and error	command > out.log 2>&1

Pipes (|)

Connect output of one command to input of another.

Example:
list files, then filter txt files

```
ls | grep txt
```

tee

Send output to file AND screen:

```
ls | tee filelist.txt
```

xargs

Build and execute commands from input.
cats all .txt files

```
ls *.txt | xargs cat
```

Searching Text Files Using Regular Expressions

grep	Basic text search
egrep	Extended grep (supports extended regex)
fgrep	Fixed grep (no regex)
sed	Stream editing, search/replace
regex	Regular expressions — pattern matching rules

Examples:

Find all lines containing "error":

```
grep error logfile.txt
```

Find case-insensitive:

```
grep -i error logfile.txt
```

Use regex to find lines starting with "Error":

```
grep '^Error' logfile.txt
```

Using sed to replace words:

```
sed 's/foo/bar/g' file.txt
```

File Permissions and Ownership

Default File Permissions & File Creation Mask (umask)

Default permissions:

- New files usually get 666 permissions (rw-rw-rw-).
- New directories get 777 (rwxrwxrwx).

umask

Subtracts permissions from defaults.

View current umask	umask
Set umask	umask 022

Example:

- Default file = 666
- umask 022 → actual permissions = 644 (rw-r--r--)

Reading the output of ls -l

Example:

```
-rw-r--r-- 1 user group 1234 Apr 27 20:00 file.txt
```

Part	Meaning
-	File type (- file, d directory, l symlink)
rw-	Owner permissions (read, write)
r--	Group permissions (read only)
r--	Others permissions (read only)
user	Owner
group	Group

Managing Access Permissions

Numeric Permissions

- Read = 4
- Write = 2
- Execute = 1

Add the values:

Who	Value
Owner	7 (rwx = 4+2+1)
Group	5 (r-x = 4+0+1)
Others	5 (r-x = 4+0+1)

Set with chmod:

```
chmod 755 file.txt
```

Symbolic (Text) Permissions

Use letters:

Symbol	Meaning
u	User (owner)
g	Group
o	Others
a	All

Example:

Give execute permission to user:

```
chmod u+x script.sh
```

Remove write permission for group:

```
chmod g-w file.txt
```

Group Permissions

Change group ownership:

```
chgrp developers file.txt
```

Change file owner:

```
chown user:group file.txt
```

Example:

```
chown john:developers report.doc
```

(owned by john, group is developers)

Basic Network Configuration and Troubleshooting

IP Addressing Essentials

An IP address = address to identify a device on a network.

- IPv4 example: 192.168.1.10/24
- IPv6 example: 2001:db8::1/64

Routing Essentials

- Routers move traffic between different networks.

Common routing protocols:

- Static (manual routes)
- Dynamic (OSPF, BGP, RIP)

Route Metrics:

- Lower metric = more preferred route.
- Metrics measure cost like hop count, bandwidth, reliability.

Basic TCP/IP Host Configuration

Task	**Command**
Set IP manually (older)	ifconfig eth0 192.168.1.10 netmask 255.255.255.0
Set IP manually (modern)	ip addr add 192.168.1.10/24 dev eth0
Bring interface up	ifconfig eth0 up or ip link set eth0 up

Manual vs Automatic Interface Configuration

Manual	Set static IP yourself
Automatic (DHCP)	IP assigned dynamically

Config file locations:

- RHEL/Oracle: /etc/sysconfig/network-scripts/ifcfg-eth0
- Debian: /etc/network/interfaces

DNS Resolution

Set DNS servers manually:

```
sudo nano /etc/resolv.conf #List of nameservers
```

Example /etc/resolv.conf:

```
nameserver 8.8.8.8
nameserver 1.1.1.1
```

Query remote DNS:

```
dig www.example.com
host www.example.com
```

Local Name Resolution

Order of resolution defined in:

```
/etc/nsswitch.conf    # Controls lookup order for hosts (DNS, files)
```

Setting Up Routes

View routes:

```
ip route show
```

Add static route:

```
ip route add 192.168.2.0/24 via 192.168.1.1
```

Delete route:

```
ip route del 192.168.2.0/24
```

Basic Troubleshooting

Tool	Purpose
ping	Test network reachability (IPv4)
ping6	Test network reachability (IPv6)
traceroute	Track path of packets (IPv4)
traceroute6	Track path (IPv6)
ifconfig	View network interfaces
ip a	View IPs with ip tool
ifup, ifdown	Bring network interfaces up/down
host	Query DNS
hostname	View/set hostname

Examples:
Ping test:

```
ping 8.8.8.8
```

Traceroute test:

```
traceroute google.com
```

Bring interface down:

```
sudo ifdown eth0
```

Advanced Troubleshooting

Tool	Purpose
ping6	IPv6 ping
traceroute6	IPv6 trace route
tracepath	Path tracing (works without needing root)
tracepath6	IPv6 version
netcat (nc)	Port scanner, banner grabber, socket testing
dig	Advanced DNS querying
route	Show/edit routing table (legacy command)

Example with netcat:

```
nc -v google.com 80
```

(check if TCP port 80 is open)

Example with dig:

```
dig @8.8.8.8 google.com
```

(use specific DNS server)

Custom Partitions and Filesystem Administration

MBR, GUID, GPT (Introduction)

Partition table = tells the system how the disk is divided.

MBR (Master Boot Record)	Old standard, supports disks up to 2 TB, max 4 primary partitions.
GPT (GUID Partition Table)	Modern standard, supports huge disks (>2 TB), 128+ partitions.
GUID	Globally Unique Identifier for partitions in GPT.

*GPT is the standard today for new systems.

Formatting Disk Drives

Format a partition:

```
mkfs.ext4 /dev/sdb1
```

Other filesystems:

- ext4 (most common)
- xfs (very scalable, default in RHEL 7+)
- vfat (for USB, FAT32)

Example for XFS:

```
mkfs.xfs /dev/sdb1
```

Introduction to gdisk, parted, and gparted

gdisk	Partitioning tool for GPT disks (like fdisk but for GPT).
parted	Versatile tool for MBR/GPT partitioning.
gparted	GUI version of parted.

Example with gdisk:

```
sudo gdisk /dev/sdb
```

n = new partition
w = write changes

Example with parted:

```
sudo parted /dev/sdb
```

```
(parted) mklabel gpt
(parted) mkpart primary ext4 0% 100%
```

Monitoring Disk Space and Inodes

df -h	View disk usage (human-readable)
df -i	View inode usage

Example:

```
df -h
df -i
```

Checking Disk Integrity and Repairing Filesystems

fsck	Filesystem checker for Linux filesystems.
e2fsck	Specifically for ext2/3/4 filesystems.
mke2fs	Make ext2/3/4 filesystem.
dumpe2fs	Show filesystem superblock info.
tunefs	Tune filesystem parameters (ext filesystems).
xfs_repair	Repair XFS filesystems.

Check and repair example:

```
sudo fsck /dev/sdb1
```

XFS repair:

```
xfs_repair /dev/sdb1
```

Setting Up and Managing Disk Quotas

Quotas limit how much disk space or how many files a user/group can use.

Enable quotas:

- Mount filesystem with usrquota and/or grpquota.
- Edit /etc/fstab.

Commands:

Initialize quota files	quotacheck -cug /mountpoint
Turn on quotas	quotaon /mountpoint
Set user quota	edquota username
Check quotas	repquota /mountpoint

Linux Security Administration

Password Management for User Accounts

Change password:

```
passwd username
```

Force password change next login:

```
passwd -e username
```

Set password aging policy:

```
chage username
```

Options:
-M → Max days
-m → Min days
-W → Warning before expiry

Example:

```
chage -M 90 -W 7 username
```

(Password expires every 90 days, warning 7 days before.)

Limiting Number of Login Attempts

Use /etc/pam.d/system-auth with pam_tally2 or pam_faillock.

Example (in modern systems, faillock):

```
auth required pam_faillock.so deny=3 unlock_time=300
```

(3 failed attempts lock for 5 minutes.)

Host Security and TCP Wrappers

TCP Wrappers control access to services.

File	Purpose
/etc/hosts.allow	Allow specific hosts.
/etc/hosts.deny	Deny specific hosts.

Example:
Allow SSH only from 192.168.1.0/24:

```
sshd: 192.168.1.
```

Block all others:

```
sshd: ALL
```

Determine Currently Logged In Users and Access Logs

who	Who is logged in.
w	Logged in users + what they are doing.
last	Historical login sessions.

Important log files:

File	Purpose
/var/log/secure	Authentications and login attempts.
/var/log/wtmp	Binary file for login history (used by last).

Discovering Open Ports (nmap, netstat)

Find open ports:

```
sudo netstat -tulnp
```

(tcp/udp listening ports)

Using nmap:

```
sudo nmap localhost
```

Memory Usage and Process Limits

Configure in /etc/security/limits.conf.

Example (limit user memory):

```
username soft memlock 50000
username hard memlock 100000
```

View current limits:

```
ulimit -a
```

Setting Process Priorities

Tool	Purpose
nice	Start process with a custom priority.
renice	Change priority of running process.
top	View and change priorities interactively.
ps	View running processes.

Examples:
Start process with lower priority:

```
nice -n 10 command
```

Change priority of running process:

```
renice +5 -p 1234
```

(change process 1234)

Add, Modify, Remove Users and Groups

Add user:

```
useradd username
passwd username
```

Modify user:

```
usermod -aG groupname username
```

Remove user:

```
userdel username
```

Add group:
```
groupadd groupname
```

Remove group:
```
groupdel groupname
```

Introduction to sudo

sudo allows regular users to run commands as root.

Configure sudo:
```
visudo
```

Add a line like:
```
john ALL=(ALL) ALL
```
(john can run all commands as root.)

Use:
```
sudo command
```

Example:
```
sudo yum update
```

Configuring SSH Access and Encryption

Introduction to SSH (Secure Shell)

SSH = Secure encrypted remote login between two machines.

SSH Versions:

Version	Notes
SSHv1	Obsolete, insecure — do not use.
SSHv2	Current standard (secure encryption, authentication).

*SSH default port = 22.

Enabling or Disabling SSH Server

Start/enable SSH server:

```
sudo systemctl start sshd
sudo systemctl enable sshd
```

Stop/disable SSH server:

```
sudo systemctl stop sshd
sudo systemctl disable sshd
```

Config file:

```
/etc/ssh/sshd_config
```

*Change settings like Port, PermitRootLogin, etc.

After editing, reload SSH:

```
sudo systemctl reload sshd
```

Restricting SSH Access Using Allowed Hosts and Authorized Keys

Allow SSH only from specific IP addresses:

- Use AllowUsers in /etc/ssh/sshd_config.

Example:

```
AllowUsers john@192.168.1.*
```

Use authorized keys:

User generates SSH key:

```
ssh-keygen
```

(Saves public/private key in ~/.ssh/)

Copy public key to server:

```
ssh-copy-id user@server
```

or manually paste into:

```
~/.ssh/authorized_keys
```

Passkey-less SSH Access (Key-Based Authentication)

Steps:
Generate key pair:

```
ssh-keygen
```

Copy public key to remote server:

```
ssh-copy-id username@remote-server
```

Now you can SSH without typing password:

```
ssh username@remote-server
```

*Important: Private key (id_rsa) must be protected (chmod 600).

Configuring X11 Tunnels and SSH Port Tunnels

X11 forwarding (run graphical apps remotely):
Enable in /etc/ssh/sshd_config:

```
X11Forwarding yes
```

Then SSH with -X:

```
ssh -X user@server
```

SSH Port Forwarding (Tunneling):
Local Forward:

```
ssh -L 8080:internalserver:80 user@gateway
```

(Access internalserver's port 80 via localhost:8080)

Remote Forward:

```
ssh -R 9090:localhost:22 user@server
```

Very useful for secure access to internal systems!

Basic GnuPG Configuration, Usage, and Revocation

GnuPG (GPG) = Open-source tool for encrypting/signing files and emails.

Generate a GPG key:

```
gpg --full-generate-key
```

List keys:

```
gpg --list-keys
```

Encrypt a file:

```
gpg -e -r recipient_user file.txt
```

Decrypt a file:

```
gpg -d file.txt.gpg
```

Revoke a key:

```
gpg --gen-revoke your_key_id
```

*GPG keys are critical for secure communication (encrypt + sign).

Customizing the Shell Environment

Setting Environment Variables

Environment variable example:

```
export VAR_NAME=value
```

Example:

```
export EDITOR=vim
```

View environment variables:

```
printenv
```

or

```
env
```

Make it permanent by adding to:

- ~/.bashrc (for interactive shells)
- ~/.bash_profile (for login shells)

Setting Command Search Path (PATH)

PATH = where shell looks for commands.
View current path:

```
echo $PATH
```

Add a new directory:

```
export PATH=$PATH:/opt/myprogram/bin
```

*Again, place this in ~/.bashrc to make it permanent.

Setting Default Environment for New User Accounts

Modify skeleton directory (/etc/skel):

- Files inside /etc/skel are copied to every new user's home.

Example:

```
Edit /etc/skel/.bashrc to set default environment variables, aliases, etc.
```

When you create a new user:

```
useradd newuser
```

*Their home directory copies the /etc/skel defaults.

Introduction to Bash Functions

Bash functions = reusable mini-scripts inside your shell.

Basic syntax:

```
function_name () {
   commands
}
```

Example:

```
greet() {
   echo "Hello, $1!"
}

greet John
```

(output: Hello, John!)

- Bash functions can make your shell scripts much cleaner and reusable.
- Save functions inside your ~/.bashrc to always have them ready.

Shell Scripting

Introduction to Shell Scripting (Basics)

- Shell scripts are text files with a list of shell commands.
- Shebang line (#!) at top tells system which interpreter to use.

Example start of script:

```
#!/bin/bash
echo "Hello, World!"
```

Save as myscript.sh, make executable:

```
chmod +x myscript.sh
./myscript.sh
```

Scripting best practices:

- Use clear comments (#)
- Always set #!/bin/bash at the top.
- Group related code using functions.

Loops and Command Substitution

For loop example:

```
for file in *.txt
do
  echo "$file"
done
```

While loop:

```
count=1
while [ $count -le 5 ]
do
  echo $count
  ((count++))
done
```

Command substitution (capture output):

```
today=$(date)
echo "Today is: $today"
```

Alternative syntax:

```
echo "Today is: $(date)"
```

Managing Ownership and SUID Rights of Scripts

Ownership:

- Normal chown and chmod rules apply.

SUID Bit:

- Normally ignored on shell scripts for security reasons!

Example to set SUID (works mainly on binaries):

```
chmod u+s myscript.sh
```

But:

```
Most modern systems ignore SUID on scripts.
Best practice: use sudo instead for privilege escalation inside scripts.
```

Smarter Organizational Decisions in Scripts

Tips:

- Use functions to organize repeated code.
- Use case statements instead of messy if-else chains:

```
case $1 in
start)
  echo "Starting..."
  ;;
stop)
  echo "Stopping..."
  ;;
*)
  echo "Usage: $0 {start|stop}"
  ;;
esac
```

- Always validate user input

Automating System Administration Jobs

Configuring and Managing Cron and at Jobs

Cron = schedule recurring tasks.

View crontab:

```
crontab -l
```

Edit crontab:

```
crontab -e
```

Crontab syntax:

```
*    *    *    *    *    command
|    |    |    |    |
|    |    |    |    └── Day of the week (0-7)
|    |    |    └────── Month (1-12)
|    |    └────────── Day of the month (1-31)
|    └────────────── Hour (0-23)
└────────────────── Minute (0-59)
```

at = schedule one-time jobs.

Schedule a job:

```
echo "reboot" | at 2am
```

List at jobs:

```
atq
```

Configuring Anacron

Anacron = ensures missed cron jobs still run on next boot (good for laptops/servers that shut down).
Main config:

```
/etc/anacrontab
```

Syntax:

```
period  delay  job-identifier  command
```

Example:

```
1   5   cron.daily  run-parts /etc/cron.daily
```

Managing Other Essential System Services

Local Settings (Language, Timezone, Keyboard)

Set Language (Locale):

```
localectl set-locale LANG=en_US.UTF-8
```

Set Timezone:

```
timedatectl set-timezone Asia/Manila
```

Keyboard:

```
localectl set-keymap us
```

Setting Time and NTP

View time settings:

```
timedatectl
```

Enable NTP (automatic time sync):

```
timedatectl set-ntp true
```

Install and configure chronyd if using a dedicated NTP client.

Configuring and Auditing System Logs

System logs handled by rsyslog and journald.

View logs:

```
journalctl
```

View boot logs:

```
journalctl -b
```

Manual log files:

```
/var/log/messages
/var/log/secure
/var/log/cron
```

Configure rsyslog in:

```
/etc/rsyslog.conf
```

Audit logs:

```
ausearch
```

and

```
auditctl
```

(if auditd is installed)

Mail Transfer Agent Basics

MTAs (like Postfix, Exim) deliver system mail.

Send test mail:

```
mail -s "Test Subject" user@domain.com
```

Check mail queue:

```
mailq
```

Basic MTA configuration file:

```
/etc/postfix/main.cf (Postfix)
```

Managing Printers (CUPS)

CUPS = Common Unix Printing System.
Install and start CUPS:

```
sudo systemctl start cups
sudo systemctl enable cups
```

Web interface:

```
http://localhost:631
```

*Manage printers there.

List printers:

```
lpstat -p -d
```

Print a file:

```
lpr filename.txt
```

Setting Up a Display Manager

Display managers = graphical login screens (like GDM, LightDM, SDDM).

Install and set default:

```
sudo systemctl set-default graphical.target
sudo systemctl enable gdm
```

Installing and Configuring X11 (Graphical)

Install X11 server:

```
sudo dnf groupinstall "Server with GUI"
```

Set to boot into GUI:

```
sudo systemctl set-default graphical.target
```

Basic X11 config file:

```
/etc/X11/xorg.conf
```

*Modern systems auto-configure X11, so manual xorg.conf editing is rare now.

Setup Samba File Server (SMB/CIFS Sharing)

Samba = lets Linux share files/folders with Windows and Linux clients.

Install Samba:

```
sudo dnf install samba samba-client samba-common
```

Configure Samba:
Edit config file:

```
sudo nano /etc/samba/smb.conf
```

Add at the bottom:

```
[shared]
  path = /srv/samba/shared
  writable = yes
  browsable = yes
  guest ok = yes
```

Create the shared folder:

```
sudo mkdir -p /srv/samba/shared
sudo chmod -R 0777 /srv/samba/shared
```

Start and enable Samba services:

```
sudo systemctl start smb nmb
sudo systemctl enable smb nmb
```

Optional: Create Samba user (authenticated shares)

```
sudo smbpasswd -a username
```

Now access it from Windows/Linux:

```
\\server-ip\shared
```

Setup Oracle Database Server (Basic Install)

Setup Oracle Database Server 21c Enterprise Edition

1. Prepare the Linux System
Install OS prerequisites using Oracle's preinstall package (this sets limits, packages, kernel settings):

```
sudo dnf install -y oracle-database-preinstall-21c
```

Add necessary users if not already done:

```
sudo useradd oracle
sudo groupadd oinstall
sudo groupadd dba
sudo usermod -aG oinstall,dba oracle
passwd oracle
```

Create necessary directories for database installation:

```
sudo mkdir -p /u01/app/oracle
sudo mkdir -p /u01/app/oraInventory
sudo chown -R oracle:oinstall /u01
sudo chmod -R 775 /u01
```

2. Download Oracle Database 21c Software (Enterprise Edition)
    a. Go to Oracle Downloads page.
    b. Download the LINUX.X64_213000_db_home.zip (Oracle 21c Database Base ZIP file).
    c. Transfer it to your server (using scp or wget if direct download).

As oracle user:

```
su - oracle
mkdir -p /u01/app/oracle/product/21c/dbhome_1
cd /u01/app/oracle/product/21c/dbhome_1
unzip /path/to/LINUX.X64_213000_db_home.zip
```

Set environment variables (temporary):

```
export ORACLE_BASE=/u01/app/oracle
export ORACLE_HOME=/u01/app/oracle/product/21c/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH
```

* (Later, add these permanently in .bash_profile.)

3. Install the Oracle Database Software (Binaries Only)
Run the Oracle installer:

```
cd $ORACLE_HOME
./runInstaller
```

Choose:

- Set up Software Only (install binaries first — recommended for DBAs).
- Oracle Home location should be /u01/app/oracle/product/21c/dbhome_1.
- Software Owner = oracle, groups = oinstall, dba.

After installation, execute as root (installer will prompt):

```
sudo /u01/app/oraInventory/orainstRoot.sh
sudo /u01/app/oracle/product/21c/dbhome_1/root.sh
```

4. Create a New Database (After Software Installation)
Run DBCA (Database Configuration Assistant):

```
dbca
```

Or silent mode (no GUI):

```
dbca -silent -createDatabase \
 -templateName General_Purpose.dbc \
 -gdbname ORCL21 \
 -sid ORCL21 \
 -responseFile NO_VALUE \
 -characterSet AL32UTF8 \
 -memoryMgmtType auto_sga \
 -totalMemory 2048 \
 -emConfiguration NONE \
 -dbsnmpPassword Welcome1 \
 -sysPassword Welcome1 \
 -systemPassword Welcome1 \
 -createAsContainerDatabase true \
 -numberOfPDBs 1 \
 -pdbName PDB1 \
 -pdbAdminPassword Welcome1
```

*Customize the memory (-totalMemory) as needed based on your server.

5. Configure the Listener

Start lsnrctl:

```
lsnrctl start
```

Listener config file:

```
$ORACLE_HOME/network/admin/listener.ora
```

*Typical connection port = 1521.

6. Enable Autostart
To automatically start Oracle Database at boot:

Create a systemd service unit /etc/systemd/system/oracle-db.service like:

```
[Unit]
Description=Oracle Database 21c Service
After=network.target

[Service]
Type=forking
User=oracle
ExecStart=/u01/app/oracle/product/21c/dbhome_1/bin/dbstart $ORACLE_HOME
ExecStop=/u01/app/oracle/product/21c/dbhome_1/bin/dbshut $ORACLE_HOME
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Enable and start:

```
sudo systemctl daemon-reload
sudo systemctl enable oracle-db
sudo systemctl start oracle-db
```

Setup Nginx Web Server

Nginx = high-performance lightweight web server.

Install Nginx:

```
sudo dnf install nginx
```

Start and enable Nginx:

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

Check if it's running:

```
sudo systemctl status nginx
```

Access in browser:

```
http://server-ip/
```

(Default welcome page)

Basic Configuration File:

```
sudo nano /etc/nginx/nginx.conf
```

Website config (for vhosts):

```
sudo nano /etc/nginx/conf.d/your-site.conf
```

Example minimal server block:

```
server {
    listen 80;
    server_name your-domain.com;
    root /var/www/html;
    index index.html index.htm;
}
```

After editing configs:

```
sudo nginx -t
sudo systemctl reload nginx
```

Setup NTP Server (Network Time Protocol)

- NTP server keeps system time synchronized across devices.
- On modern Oracle Linux/RHEL, use chronyd.

Install Chrony:

```
sudo dnf install chrony
```

Configure Chrony:

```
sudo nano /etc/chrony.conf
```

Add/modify:

```
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
allow 192.168.1.0/24   # Allow your internal LAN clients
```

"allow" lets clients sync with your server.

Start and enable Chrony:

```
sudo systemctl start chronyd
sudo systemctl enable chronyd
```

Check Status:

```
chronyc tracking
chronyc sources
```

If using old ntpd:

```
sudo dnf install ntp
sudo systemctl start ntpd
sudo systemctl enable ntpd
```

Config file:

```
/etc/ntp.conf
```