Contents

Using CRON

cron is a job scheduler in Linux used to execute scripts or commands at scheduled intervals.

Checking if CRON is Installed

```
sudo dnf install -y cronie
```

Ensure the cron service is running:

```
sudo systemctl enable --now crond
sudo systemctl status crond
```

Understanding the Crontab Format
A cron job follows this syntax:

```
MIN HOUR DOM MON DOW COMMAND
```

MIN → Minutes (0-59)
HOUR → Hours (0-23)
DOM → Day of the Month (1-31)
MON → Month (1-12)
DOW → Day of the Week (0-7, 0 and 7 = Sunday)
COMMAND → The command/script to execute

Sample Script for Testing
Create the Script

```
sudo vi /home/user/script.sh
```

Add the Following Content

```
#!/bin/bash
echo "Script executed on: $(date)" >> /home/user/script.log
echo "Hello from the at job!" >> /home/user/script.log
```

Make the Script Executable

```
chmod +x /home/user/script.sh
```

Scheduling Jobs with Crontab
View Existing Cron Jobs

```
crontab -l
```

Edit Crontab for a User

```
crontab -e
```

Example: Scheduling Jobs

Schedule	Crontab Entry
Run every 5 minutes	*/5 * * * * /home/user/script.sh
Run daily at 2 AM	0 2 * * * /home/user/backup.sh
Run every Sunday at midnight	0 0 * * 0 /home/user/log-cleanup.sh
Run on the 1st of every month at 3 AM	0 3 1 * * /home/user/report.sh

Using System-Wide Cron Jobs

There are predefined system cron directories:

```
/etc/cron.hourly/
/etc/cron.daily/
/etc/cron.weekly/
/etc/cron.monthly/
```

To schedule system-wide jobs, place your script inside one of these directories.

Special Syntax

Shortcut	Equivalent
@reboot	Runs once at startup
@hourly	0 * * * *
@daily	0 0 * * *
@weekly	0 0 * * 0
@monthly	0 0 1 * *

Example:

```
@reboot /home/user/startup.sh
```

*This will execute startup.sh every time the system reboots.

Logging and Troubleshooting CRON Jobs

Check CRON Logs

```
sudo journalctl -u crond --since "1 hour ago"
```

Redirect Output for Debugging

By default, cron does not output errors. To capture logs:

```
*/5 * * * * /home/user/script.sh >> /home/user/cron.log 2>&1
```

Where:

Field	Value	Meaning
Minute	*/5	Runs every 5 minutes
Hour	*	Runs every hour
Day of Month	*	Runs every day
Month	*	Runs every month
Day of Week	*	Runs every day of the week

*This means the script will execute every 5 minutes, all day, every day.

Symbol	Meaning
>>	Append output to a file (/home/user/cron.log)
2>&1	Redirect error messages (stderr, 2) to the same location as standard output (1)

Using AT

Install the at Package

By default, at is not always installed in RHEL 8 / 9. To install it:

```
sudo dnf install -y at
```

Enable and start the atd daemon:

```
sudo systemctl enable --now atd
sudo systemctl status atd
```

Scheduling a One-Time Job with at

The basic syntax is:

```
echo "<command>" | at <time>
```

or interactively:

```
at <time>
```

*Then enter the command and press Ctrl+D to save.

Example: Run a Command at a Specific Time

To schedule a script (/home/user/script.sh) to run at 2:30 PM today:

```
at 14:30
```

Type the command:

```
/home/user/script.sh
```

*Press Ctrl+D to save and exit.

Example: Schedule a Job in the Future

Run a command "in 10 minutes":

```
echo "echo 'Backup started' > /home/user/backup.log" | at now + 10 minutes
```

Run a command "tomorrow at 8 AM":

```
at 8:00 AM tomorrow
```

*Then enter the command and press Ctrl+D.

Run a command "next Monday at 7 PM":

```
at 19:00 Monday
```

Viewing Scheduled at Jobs
To list all pending at jobs:

```
atq
```

Example output:

```
1  Tue Feb 27 14:30:00 2025 a user
2  Wed Feb 28 08:00:00 2025 a user
```

*Each job has an ID.

Removing a Scheduled at Job
To remove a job, use its job ID from atq:

```
atrm <job-ID>
```

Example:

```
atrm 1
```

Running a Job as a Specific User
As root, you can schedule jobs for another user:

```
sudo -u username at 10:00
```

*Then enter the command and press Ctrl+D.

Restricting at Access

- Users listed in /etc/at.allow can use at.
- Users in /etc/at.deny cannot use at.

To allow only specific users, create /etc/at.allow:

```
echo "user1" | sudo tee -a /etc/at.allow
```

Checking at Logs
To check executed jobs:

```
sudo journalctl -u atd --since "1 hour ago"
```

Tar, Gunzip and Bunzip2

Commands for compressing, archiving, and extracting files in Linux.

Using tar (Tape Archive)
Create a Tar Archive
To archive files without compression:

```
tar -cvf archive.tar file1 file2 directory/
```

c → Create an archive
v → Verbose (show progress)
f → Filename (archive.tar)

Example:

```
tar -cvf backup.tar /home/user/
```

*This creates backup.tar containing everything in /home/user/.

Extract a Tar Archive
To extract files:

```
tar -xvf archive.tar
```

x → Extract

Extract to a specific directory:

```
tar -xvf archive.tar -C /destination/path
```

List Contents of a Tar Archive

```
tar -tvf archive.tar
```

Using tar with Compression
Create a Tar Archive with Gzip Compression

```
tar -czvf archive.tar.gz /home/user/
```

z → Use gzip compression

Extract a Tar.gz Archive

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

Create a Tar Archive with Bzip2 Compression

```
tar -cjvf archive.tar.bz2 /home/user/
```

j → Use bzip2 compression (better compression than gzip)

Extract a Tar.bz2 Archive

```
tar -xjvf archive.tar.bz2
```

Using gunzip (for .gz files)
Compress a File Using gzip

```
gzip filename
```

*This creates filename.gz.

Decompress a .gz File

```
gunzip filename.gz
```

or

```
gzip -d filename.gz
```

Using bunzip2 (for .bz2 files)

Compress a File Using bzip2

```
bzip2 filename
```

*This creates filename.bz2.

Decompress a .bz2 File

```
bunzip2 filename.bz2
```

or

```
bzip2 -d filename.bz2
```

Review: File and Folder Permissions

In Linux, file and folder permissions determine who can read, write, or execute files and directories. They are managed using chmod, chown, and chgrp.

Understanding Linux File Permissions

When you run:

```
ls -l
```

You will see output like this:

```
-rwxr-xr--  1 user group  1234 Feb 27 12:00 script.sh
```

Breakdown of File Permissions:

Symbol	**Meaning**
-	Regular file (d for directory)
rwx	Owner (user) can read, write, execute
r-x	Group can read, execute (not write)
r--	Others can only read

Changing Permissions with chmod

Numeric Mode (Octal)

Each permission is assigned a number:

```
r (read) = 4
w (write) = 2
x (execute) = 1
```

Example:

```
chmod 755 script.sh
```

Owner	Group	Others
7 = rwx	5 = r-x	5 = r-x

Another example:
chmod 644 file.txt

Owner	Group	Others
6 = rw-	4 = r--	4 = r--

Symbolic Mode

```
chmod u+x script.sh      # Add execute permission to the user
chmod g-w file.txt       # Remove write permission from the group
chmod o+r file.txt       # Give others read permission
chmod a+x script.sh      # Give everyone execute permission
```

Changing Ownership with chown

Change File Owner

```
chown newuser file.txt
```

Change Group Ownership

```
chown :newgroup file.txt
```

Change Both Owner and Group

```
chown newuser:newgroup file.txt
```

Changing Group Ownership with chgrp

```
chgrp newgroup file.txt
```

Checking and Testing Permissions

Check File Permissions

```
ls -l file.txt
```

Test Write Access

```
touch testfile
```

*If you get Permission denied, you need write access.

Processes Management

Create the Script

```
sudo vi /home/user/script.sh
```

Add the Following Content

```
#!/bin/bash
# Infinite loop that logs the current time every 5 seconds
while true;
do
  echo "Script is running... $(date)" >> /home/user/script.log
  sleep 5
done
```

Make the Script Executable

```
chmod +x /home/user/script.sh
```

Test Process Management with the Script

Run in the Foreground

```
/home/user/script.sh
```

Press Ctrl + C to stop it.

Run in the Background

```
/home/user/script.sh &
```

Check background jobs:

```
jobs
```

Bring it back to the foreground:

```
fg %1
```

Pause a running process:

```
Ctrl + Z
```

Send it back to the background:

```
Ctrl + Z
bg
```

Find the Process

```
ps aux | grep script.sh
```

or

```
pgrep -l script.sh
```

Stop the Process

Kill by PID:

```
kill <PID>
```

Force kill:

```
kill -9 <PID>
```

or

```
pkill -9 script.sh
```

Check Logs
After stopping the process, view the logs:

```
cat /home/user/script.log
```

Viewing Running Processes
To list all processes:

```
ps aux
```

a → Show all users' processes
u → Show detailed user info
x → Show processes not attached to a terminal

To filter by a specific user:

```
ps -u username
```

To find a process by name:

```
ps aux | grep apache
```

Using top (Real-Time Process Monitoring)

```
top
```

- Shows CPU, memory usage, and active processes
- Press q to quit
- Press k to kill a process

To sort by memory usage:

```
top -o %MEM
```

Using htop (Interactive Process Viewer)
If htop is not installed:

```
sudo dnf install -y htop
```

Run:

```
htop
```

- Arrow keys → Navigate
- F9 → Kill a process
- F4 → Filter processes

Processes Prioritization with RENICE

- renice adjusts the priority of a running process.
- Linux schedules CPU time using niceness values (ranges from -20 (highest priority) to 19 (lowest priority)).
- The lower the value, the higher the priority.

Check Process Priority with nice and ps

View the Niceness of Running Processes

```
ps -eo pid,ni,comm | grep script.sh
```

PID → Process ID
NI → Niceness value (0 by default)
COMM → Process name

Example output:

```
12345  0  script.sh
```

*This means script.sh is running with a niceness of 0 (default priority).

Run a New Process with a Specific Niceness

Start a process with low priority (nice 10):

```
nice -n 10 ./script.sh
```

Start a process with high priority (nice -5):

```
sudo nice -n -5 ./script.sh
```

*Lower values mean higher priority, but negative values require sudo.

Adjust Priority of a Running Process with renice

Increase Priority (Requires Root)

```
sudo renice -5 -p <PID>
```

*Makes the process faster (higher priority).

Decrease Priority

```
renice 10 -p <PID>
```

*Makes the process slower (lower priority).

Check Updated Priority

```
ps -eo pid,ni,comm | grep script.sh
```

Example output:

```
12345  -5  script.sh
```

*Now script.sh has higher priority (-5).

Practical Example with script.sh

Run the Script in the Background

```
./script.sh &
```

Find its PID

```
ps aux | grep script.sh
```

Example output:

```
user   12345  0.3  0.2  123456  5678 ?    S   12:00   0:01 script.sh
```

Lower Priority (Less CPU Usage)

```
renice 10 -p 12345
```

Increase Priority (Faster Execution)

```
sudo renice -5 -p 12345
```

Using TUNED

Tuned is a dynamic performance tuning daemon in RHEL 8 / 9 that automatically adjusts system settings based on workload and hardware.

Install and Enable tuned

Check if tuned is installed:

```
rpm -q tuned
```

If not installed, install it:

```
sudo dnf install -y tuned
```

Enable and start tuned:

```
sudo systemctl enable --now tuned
sudo systemctl status tuned
```

List Available Tuning Profiles

To view all available tuning profiles:

```
tuned-adm list
```

Example output:

```
Available profiles:
- balanced
- throughput-performance
- latency-performance
- powersave
- virtual-guest
- virtual-host
```

```
- network-latency
- network-throughput
- desktop
- server-powersave
- oracle
```

Apply a Performance Profile
To apply a profile, use:

```
sudo tuned-adm profile <profile-name>
```

Example Profiles:
Optimized for general usage (default):

```
sudo tuned-adm profile balanced
```

Optimized for performance (databases, high workloads):

```
sudo tuned-adm profile throughput-performance
```

Optimized for low-latency applications (real-time systems):

```
sudo tuned-adm profile latency-performance
```

Optimized for network latency-sensitive applications:

```
sudo tuned-adm profile network-latency
```

Optimized for network throughput (servers with high network loads):

```
sudo tuned-adm profile network-throughput
```

Optimized for power saving (laptops, low-power servers):

```
sudo tuned-adm profile powersave
```

Optimized for virtual machines (VMs):

```
sudo tuned-adm profile virtual-guest
```

Verify the Active Profile
To check the current active profile:

```
tuned-adm active
```

Example output:

```
Current active profile: throughput-performance
```

Disk Partitions and File Systems

Managing disk partitions and file systems is essential for storage administration in RHEL 8 / 9. This guide covers creating, formatting, mounting, and managing partitions.

Checking Available Disks and Partitions

List All Disks

```
lsblk
```

Example output:

```
NAME        MAJ:MIN  RM  SIZE  RO  TYPE  MOUNTPOINT
sda         8:0      0   100G  0   disk
 ├─sda1     8:1      0   50G   0   part  /
 ├─sda2     8:2      0   40G   0   part  /data
 └─sda3     8:3      0   10G   0   part  [SWAP]
```

sda → Primary disk (100GB)
sda1, sda2, sda3 → Partitions

View Disk Partition Table

```
sudo fdisk -l
```

*This displays disk details including partition type, size, and usage.

Creating a New Partition

Using fdisk
Select a disk (e.g., /dev/sdb):

```
sudo fdisk /dev/sdb
```

Inside fdisk, follow these steps:

a) Press n → Create a new partition
b) Press p → Primary partition
c) Select Partition Number (default: 1)
d) Choose First sector (default: press Enter)
e) Choose Last sector (example: +20G for a 20GB partition)
f) Press w → Write changes and exit

Refresh partition table:

```
sudo partprobe
```

Formatting the Partition

Format as ext4 File System

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

Format as xfs File System (RHEL Default)

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

Mounting and Using the New Partition

Create a Mount Point

```
sudo mkdir -p /mnt/newdisk
```

Mount the Partition

```
sudo mount /dev/sdb1 /mnt/newdisk
```

Verify the Mount

```
df -h | grep /mnt/newdisk
```

Example output:

```
/dev/sdb1   20G   1G   19G   5% /mnt/newdisk
```

Persistent Mounting (FSTAB Configuration)

To make the mount permanent after reboot:

Get the UUID of the partition:

```
sudo blkid /dev/sdb1
```

Example output:

```
/dev/sdb1: UUID="a1b2c3d4" TYPE="ext4"
```

Edit /etc/fstab:

```
sudo vi /etc/fstab
```

Add the following line:

```
UUID=a1b2c3d4 /mnt/newdisk ext4 defaults 0 0
```

Reload fstab:

```
sudo mount -a
```

Unmounting and Removing a Partition

Unmount a Partition

```
sudo umount /mnt/newdisk
```

Delete a Partition
Use fdisk:

```
sudo fdisk /dev/sdb
```

- Press d → Delete partition
- Press w → Write changes and exit

Checking and Repairing File Systems
Check ext4 File System

```
sudo fsck.ext4 /dev/sdb1
```

Check xfs File System

```
sudo xfs_repair /dev/sdb1
```

SELinux Basics

SELinux (Security-Enhanced Linux) is a security mechanism that enforces mandatory access control (MAC) to protect the system from unauthorized access.

Checking SELinux Status
To check if SELinux is enabled and its mode:

```
sestatus
```

Example output:

```
SELinux status: enabled
Current mode: enforcing
Policy version: 31
```

Possible SELinux Modes

Mode	Description
Enforcing	Fully enforces policies (default in RHEL 8/9)
Permissive	Logs violations but does not block actions
Disabled	SELinux is completely turned off

Changing SELinux Modes
Temporarily Change SELinux Mode
Switch to permissive mode (without reboot):

```
sudo setenforce 0
```

Switch back to enforcing mode:

```
sudo setenforce 1
```

Verify the mode:

```
getenforce
```

Permanently Change SELinux Mode

Edit the SELinux configuration file:

```
sudo vi /etc/selinux/config
```

Change the line:

```
SELINUX=enforcing
```

to:

```
SELINUX=permissive
```

Save and reboot:

```
sudo reboot
```

Understanding SELinux Contexts

Every file in SELinux has a security context. To view a file's SELinux label:

```
ls -Z /var/www/html/index.html
```

Example output:

```
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

- system_u:object_r:httpd_sys_content_t:s0 → SELinux context
- httpd_sys_content_t → Allowed for Apache web server

Fixing SELinux Denials

If a process cannot access a file due to SELinux restrictions, check logs:

```
sudo journalctl -xe | grep AVC
```

Changing File Contexts (chcon)

If Apache cannot read a file:

```
sudo chcon -t httpd_sys_content_t /var/www/html/index.html
```

Restoring Default Contexts (restorecon)

To restore default SELinux contexts:

```
sudo restorecon -Rv /var/www/html/
```

Managing SELinux Booleans

List SELinux Booleans

```
semanage boolean -l
```

Modify a Boolean Value

Enable HTTPD to access home directories:

```
sudo setsebool -P httpd_enable_homedirs on
```

*-P makes the change permanent.

Disabling SELinux (Not Recommended)

To disable SELinux completely:

```
sudo vi /etc/selinux/config
```

Change:

```
SELINUX=disabled
```

Reboot:

```
sudo reboot
```

Shell Scripting Basic

A shell script is a program written for the Linux shell that automates tasks using command sequences.

Creating a Simple Shell Script

Create a Script File

```
sudo vi /home/user/script.sh
```

Add Script Content

```
#!/bin/bash
echo "Hello, this is my first shell script!"
date
```

- #!/bin/bash → Shebang (defines the shell to execute the script)
- echo → Prints a message
- date → Shows the current date/time

Making the Script Executable

```
chmod +x /home/user/script.sh
```

Running the Script

Execute it using:

```
./script.sh
```

or with full path:

```
/home/user/script.sh
```

Using Variables in Shell Scripts

```
#!/bin/bash
NAME="John"
echo "Hello, $NAME!"
```

Run:

```
./script.sh
```

Output:

```
Hello, John!
```

User Input in Scripts

```
#!/bin/bash
echo "Enter your name:"
read NAME
echo "Hello, $NAME!"
```

Conditional Statements

if Statement

```
#!/bin/bash
echo "Enter a number:"
read num
if [ $num -gt 10 ]; then
  echo "The number is greater than 10"
else
  echo "The number is 10 or less"
fi
```

Looping in Shell Scripts

for Loop

```
#!/bin/bash
for i in {1..5}; do
  echo "Iteration: $i"
done
```

while Loop

```
#!/bin/bash
count=1
while [ $count -le 5 ]; do
  echo "Count: $count"
  ((count++))
done
```

Functions in Shell Scripts

```
#!/bin/bash
greet() {
  echo "Hello, $1!"
}
```

```
greet "Alice"
```

Run:

```
./script.sh
```

Output:

```
Hello, Alice!
```

Scheduling Shell Scripts with cron

To run a script every hour:

```
crontab -e
```

Add:

```
0 * * * * /home/user/script.sh
```

Podman (AWS EC2 Instance)

Podman is a daemonless container engine that is compatible with Docker. It allows running, managing, and deploying containers without requiring root privileges.

Installing Podman on Amazon Linux 2023

Amazon Linux 2023 comes with Podman available in the default repositories.

Update the System

```
sudo dnf update -y
```

Install Podman

```
sudo dnf install -y podman
```

Verify the Installation

```
podman --version
```

Running a Basic Container

Run an Nginx Container

```
podman run -d -p 8080:80 --name my-nginx docker.io/library/nginx
```

- -d → Run in detached mode
- -p 8080:80 → Map port 8080 on the host to port 80 inside the container
- --name my-nginx → Assign a name to the container

Verify Running Containers

```
podman ps
```

Expected output:

CONTAINER ID	IMAGE	STATUS	PORTS	NAMES
abcdef123456	docker.io/library/nginx:latest	Up	0.0.0.0:8080->80/tcp	my-nginx

Access the Web Server

```
curl http://localhost:8080
```

Managing Containers

Stop a Running Container

```
podman stop my-nginx
```

Restart a Stopped Container

```
podman start my-nginx
```

Remove a Container

```
podman rm my-nginx
```

Remove All Stopped Containers

```
podman rm $(podman ps -aq)
```

Managing Container Images

List Available Images

```
podman images
```

Pull a Specific Image

```
podman pull alpine
```

Remove an Image

```
podman rmi alpine
```

Running Containers as a Non-Root User

Podman allows running containers without sudo.

Enable Rootless Mode

```
podman system migrate
```

Run a Rootless Container

```
podman run --rm -it alpine sh
```

Working with Podman Volumes

Create a Volume

```
podman volume create mydata
```

Use a Volume with a Container

```
podman run -d -v mydata:/data --name my-container alpine
```

Check Mounted Volumes

```
podman volume ls
```

Remove a Volume

```
podman volume rm mydata
```

Running Systemd Containers (Podman Compose)

To manage multiple containers like Docker Compose, install Podman Compose:
sudo dnf install -y podman-compose

Create a podman-compose.yml:

```
version: '3'
services:
 web:
  image: nginx
  ports:
   - "8080:80"
```

Start the container:
podman-compose up -d

Podman on AWS EC2 Security Considerations

Allow Firewall Rules for External Access

```
sudo firewall-cmd --permanent --add-port=8080/tcp
sudo firewall-cmd --reload
```

Configure SELinux for Containers

```
sudo setsebool -P container_manage_cgroup on
```