# How to Create and Set Up a Cron Job in Linux

January 31, 2024

CRON CRONTAB LINUX

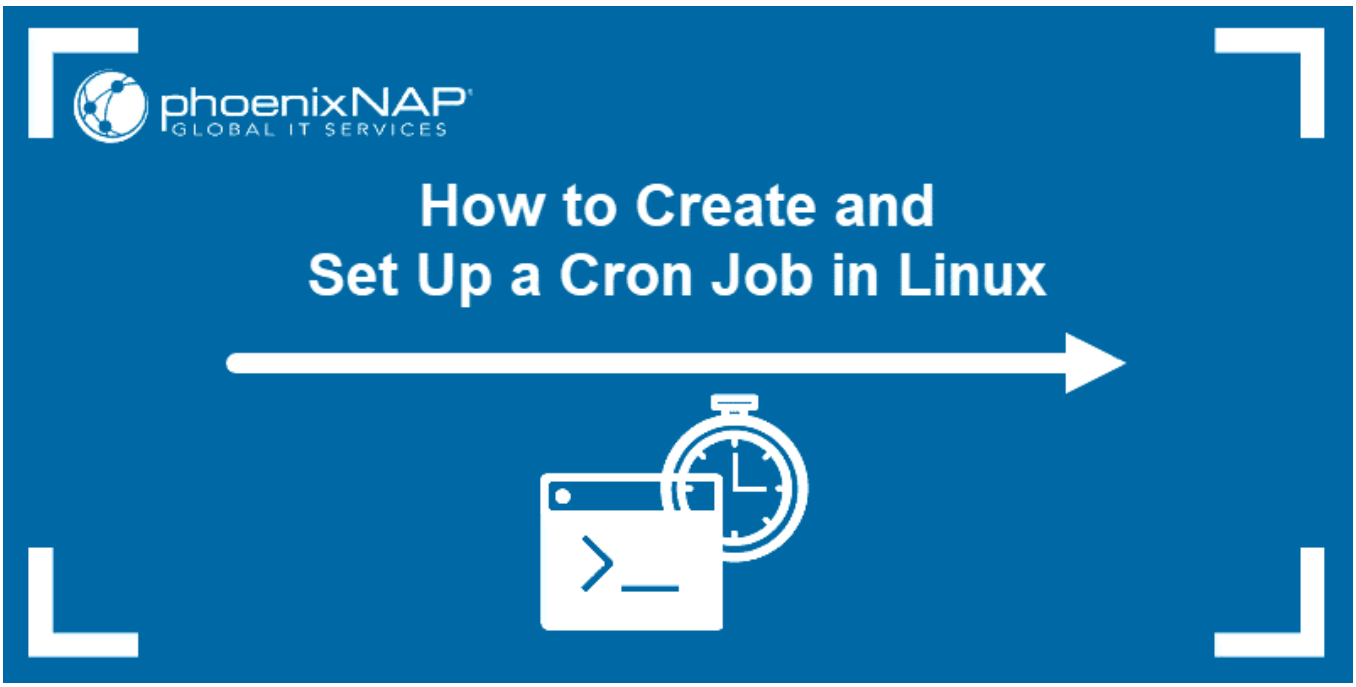


## Introduction

The Cron daemon is a built-in Linux utility that reads the **crontab** (cron table) file and executes commands and scripts at predefined times and intervals.

Users set up **cron jobs** in the **crontab** to streamline routine maintenance activities, such as updating software, creating backups, or clearing caches.

**Learn how to set up a cron job in Linux and lighten your workload by automating repetitive tasks.**




## Prerequisites

- A system running Linux.
- Access to a command line or terminal window.

# Basic Crontab Syntax

The syntax of a cron job line in a crontab file must use the following format:

```
MIN HOUR DOM MON DOW CMD
```

The first five fields, each separated by a single space, represent time intervals: **MIN** for *minutes*, **HOUR** for *hours*, **DOM** for *day of the month*, **MON** for *month*, and **DOW** for *day of the week*. They tell Cron when to initiate the cron job.

These fields are followed by the command (**CMD**), which is usually a path to a script or a system command. For example, the following line prompts Cron to execute a script on January 1st at 9 AM:

```
0 9 1 1 * /path/to/your_script.sh
```

Each field has its own set of permissible values, which can be accompanied or swapped for a special character. The asterisk (*) operator in this example instructs Cron to execute the job regardless of which day of the week falls on January 1st.

## Cron Job Time Format

This table lists possible values for the time fields in a cron expression:

| Field | Possible Values | Syntax | Description |
|---|---|---|---|
| **Minute Within the Hour (MIN)** | 0 – 59 | **7** * * * * | The cron job is initiated every time the system clock shows 7 in the minute's position. |
| **Hour of the Day (HOUR)** | 0 – 23 | **0 7** * * * | The cron job runs any time the system clock shows 7 AM (7 PM would be coded as 19). |
| **Day of the Month (DOM)** | 1 – 31 | **0 0 7** * * | The day of the month is 7, which means that the job runs every 7th day of the month. |
| **Month of the Year (MON)** | 1 - 12 or JAN - DEC | **0 0 0 7** * | The numerical month is 7, which determines that |

| Field | Possible Values | Syntax | Description |
|---|---|---|---|
| | | | the job runs only in July. The *Month* field value can be a number or the month abbreviation. |
| **Day of the Week (DOW)** | 0 - 7 or SUN - SAT | **0 0 * * 7** | 7 in the current position means that the job would only run on Sundays. The *Day of the Week* field can be a number or the day abbreviation. |

## Command to Execute

After defining the schedule, enter the absolute path to the script or executable command you want Cron to complete. For example, the following command tells Cron to execute the **`backup.sh`** script, located in the **`root`** directory, every day at midnight:

```
0 0 * * * /root/backup.sh
```

Users must ensure their scripts have appropriate execute permissions and that the Cron service can access and run scripts in the specified directories.

## Using Operators

Operators are special characters used instead of or in conjunction with allowed field values. They simplify and shorten cron expressions and are integral to almost all cron jobs. The most frequently used operators include:

- **An asterisk (*)** substitutes all possible values for a time field. For example, when used in the day field, it indicates that the task should be executed every day.
- **A comma (,)** is used to separate individual values within a field. For example, **`20,40`** in the minute field runs the task at 20 and 40 minutes past the hour.
- **A dash (-)** defines a range of values in a single field. Entering **`15-18`** in the minute field instructs Cron to run a task every minute between and including the 15th and 18th minute.
- **A forward slash (/)** divides a field value into increments. For example, **`*/30`** in the minute field means that the task is run every 30 minutes.

**Note:** If you also manage Windows systems, learn how to set up a Cron Job on Windows.

# Setting Up a Cron Job

To configure a cron job, open the crontab file using a preferred text editor and input the syntax for the command or script you want to run.

Follow the steps below to configure a cron job.

## 1. Write a Script (Optional)

This section explains how to create an example script. If you already have a script ready, skip to the next section.

To write a simple UNIX shell script:

1. Use a text editor, like nano, and create a new *.sh* file. The file in this example is named *script.sh*:

```
nano script.sh
```

2. Utilize a preferred Linux shell to write a script for the cron job to run. For example, to create a Bash script, start with the shebang expression. Enter the path to the Bash binary and list the commands that you want to execute:

```
#!/bin/bash
echo "Current Date and Time: $(date)"
```

The `echo "Current Date and Time: $(date)"` command displays the current date and time when the cron job is executed.

3. **Save and exit** the file.

4. Enter the chmod command to ensure that *script.sh* has appropriate execute permissions:

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

## 2. Create or Edit Crontab File

Open the crontab configuration file for the current user by entering the following command:

```
crontab -e
```

If this is your first time accessing the crontab, the system creates a new file. In Ubuntu 22.04, users are prompted to select a preferred text editor. Enter the corresponding number, for example, **1** for nano, to open the crontab file.

To schedule a job for a different user, add the **-u** option and the username:

```
crontab -u [username] -e
```

**Note:** Use the sudo command when accessing crontab for system-level tasks or tasks requiring administrative privileges. For regular user-level tasks, **sudo** is not necessary.

## 3. Create the Cron Job

Add a line containing a cron expression and the path to a script. This example uses the path to *script.sh* created earlier:

```
15 22 * * * /home/phoenixnap/script.sh
```

The cron job entry will run *script.sh* every day at 10:15 PM.

Remember to specify the complete, absolute path to where the script is located. You can add an infinite number of scheduled tasks. Each task must be in a separate line.

## 4. Output (Optional)

Cron sends an email to the owner of the crontab file with the cron job output. The format of cron emails can vary based on the system email configuration and the script's output.

This feature is convenient for tracking tasks, but emails for frequent or minor tasks can be spammy. Users can disable email output for specific cron jobs.

To turn off email output, add **`>/dev/null 2>&1`** at the end of the cron job line:

```
0 0 * * * /path/to/script.sh > /dev/null 2>&1
```

The output is redirected to the */dev/null* file that discards all data written to it. Use this method to redirect output to any file in the filesystem.

Alternatively, to direct Cron emails to a specific address, add the **`MAILTO`** variable followed by a valid email address above a cron job.

If `MAILTO` is left empty (`MAILTO=''`), Cron will not send emails for jobs listed below the variable.

## 5. Save

Once you have finished adding tasks, **save and exit** the crontab file. There is no need to restart Cron to apply the changes.

The daemon will automatically read and execute the provided instructions.

## 6. Check Active Cron Jobs

Enter the following command to list all cron jobs on your system without opening the crontab configuration file:

```
crontab -l
```

# Cron Job on Linux: Examples

The following table provides basic cron job command examples. Replace */path/to/script* with the actual, absolute path of your script on your system.

| Run Cron Job | Command |
|---|---|
| Every Minute | `* * * * * /path/to/script` |
| Every 15 Minutes | `*/15 * * * * /path/to/script` |

| Run Cron Job | Command |
|---|---|
| On the 30th Minute of Every Hour | `30 * * * * /path/to/script` |
| At the Beginning of Every Hour | `0 * * * * /path/to/script` |
| Every Day at Midnight | `0 0 * * * /path/to/script` |
| At 2 AM Every Day | `0 2 * * * /path/to/script` |
| Every 1st of the Month | `0 0 1 * * /path/to/script` |
| Every 15th of the Month | `0 0 15 * * /path/to/script` |
| On December 1st - Midnight | `0 0 1 12 * /path/to/script` |
| Saturdays at Midnight | `0 0 * * 6 /path/to/script` |
| Every Weekday at 4 AM | `0 4 * * 1-5 /path/to/script` |
| At 4 AM on Tuesdays and Thursdays | `0 4 * * 2,4 /path/to/script` |
| Every Other Day at 37 Minutes Past the Hour | `37 1-23/2 * * * /path/to/script` |
| Every 20 Minutes - Multiple Scripts | `*/20 * * * * /path/to/script1; /path/to/script2` |
| On Saturdays and Sundays at 12 PM | `0 12 * * 6,0 /path/to/script` |
| Monday to Friday - Every Hour 9 AM to 5 PM | `0 9-17 * * 1-5 /path/to/script` |
| Every Hour from 5 PM on Wednesday to 5 AM on Thursday (Job Spans Two Days) | `0 17-23 * * 3 /path/to/script`<br>`0 0-5 * * 4 /path/to/script` |
| Midnight Every Day - Send Output to a Different File | `0 0 * * * /path/to/script > /path/to/output.log 2>&1` |

## Conclusion

Thanks to the examples presented in this tutorial, you can now create and schedule cron jobs in Linux and automate routine sysadmin tasks.

Next, learn how to schedule a cron job to run at system reboot.

## Vladimir Kaplarevic

Vladimir is a resident Tech Writer at phoenixNAP. He has more than 7 years of experience in implementing e-commerce and online payment solutions with various global IT services providers. His articles aim to instill a passion for innovative technologies in others by providing practical advice and using an engaging writing style.

---

## Next you should read

SysAdmin, Web Servers

### Linux Commands Cheat Sheet

**November 2, 2023**

A list of all the important Linux commands in one place. Find the command you need, whenever you need it or download our Linux Commands Cheat Sheet and save it for future reference.

READ MORE

SysAdmin

### How to Run Linux Commands in Background

**July 5, 2023**

Executing Linux commands in the background allows users to avoid waiting for one process to complete before

entering another command. This guide explains different ways to run Linux commands in the background.

READ MORE

SysAdmin

### How to Create Symbolic Link (Symlink) in Linux

**December 4, 2023**

In Linux, symbolic links are used to easily manage and access files. This tutorial covers how to create, manage, and delete links in the Linux terminal using the ln command.

READ MORE

Security, SysAdmin

### User Management in Linux

**November 23, 2023**

This article explains how user management in Linux works and outlines the most commonly used commands for managing users.

READ MORE