****

**“AZƏRBAYCAN HAVA YOLLARI” CJSC**

**NATIONAL AVIATION ACADEMY**

**Topic**: Cronjobs in Linux

**Subject**: Operating Systems

**Teacher**: Mammad Sahmaliyev

**Group**: 2450i

**Student**: Alakbarova Rahimakhanim

**Date**: 24.10.22

**Signature**: Alakbarova Rahimakhanim

**Baku 2022**

**What is a cron job in Linux?**

Cron is a command line utility to run small and quick commands on a scheduled basis. This is a handy, classic sysadmin tool for automating various tasks by combining it with othe tools. For example, some people combine rsync and cron to automatically create a daily or weekly backup at a certain time. Some people use it to analyze server logs and combine it with mail function to send an email if there is certain kind of error detected in the logs.

Cron is like the Swiss army knife. It can be used for a variety of use cases. It’s really up to your imagination on what to use it for.

Getting started with cron is super easy, and only takes a matter of seconds to get started. But before I show you that, I’ll discuss something else that often confuses Linux users.

**Basic Crontab Syntax**

Cron reads the configuration files for a list of commands to execute. The daemon uses a specific syntax to interpret the lines in the crontab configuration tables.

To be able to set up a cron job, we need to understand the basic elements that make up this syntax. The standard form for a crontab line is as follows:

a b c d e /directory/command output

So, the parts of a cron command are:

1. The first five fields a b c d e specify the time/date and recurrence of the job.

2. In the second section, the /directory/command specifies the location and script you want to run.

3. The final segment output is optional. It defines how the system notifies the user of the job completion.

1. Cron Job Time Format

The first five fields in the command represent numbers that define when and how often the command runs. A space separates each position, which represents a specific value.

The table below summarizes possible values for the fields and the example syntax:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Possible Values** | **Syntax** | **Description** |
| [a] – Minute | 0 – 59 | **7 \* \* \* \*** | The cron job is initiated every time the system clock shows 7 in the minute’s position. |
| [b] – Hour | 0 – 23 | **0 7 \* \* \*** | The cron job runs any time the system clock shows 7am (7pm would be coded as 19). |
| [c] – Day | 0 – 31 | **0 0 7 \* \*** | The day of the month is 7 which means that the job runs every 7th day of the month. |
| [d] – Month | 0 = none and 12 = December | **0 0 0 7 \*** | The numerical month is 7 which determines that the job runs only in July. |
| [e] – Day of the Week | 0 = Sunday and 7 = Sunday | **0 0 \* \* 7** | 7 in the current position means that the job would only run on Sundays. |

2. Command to Execute

The next section specifies the command to execute. It represents the exact directory and filename of the script or commands you want cron to complete. For example:

/root/backup.sh

In our example, the command looks at the root directory of the system and runs the backup.sh script. You may specify any script or command you wish.

3. Output (Optional)

By default, cron sends an email to the owner of the crontab file when it runs. This is a convenient way to keep track of tasks. Keep in mind that regular or minor tasks can fill up your inbox quickly.

As this is an optional feature, you can prevent that scenario by disabling the output email. To turn off email output, add the following string, >/dev/null 2>&1, after the timing and command fields.

\* \* \* \* \* directory/command >/dev/null 2>&1

4. Using Operators (Optional)

For efficiency, cron syntax also uses operators. Operators are special characters that perform operations on the provided values in the cron field.

An asterisk (\*) stands for all values. Use this operator to keep tasks running during all months, or all days of the week.

A comma (,) specifies separate individual values.

A dash (–) indicates a range of values.

A forward-slash (/) is used to divide a value into steps. (\*/2 would be every other value, \*/3 would be every third, \*/10 would be every tenth, etc.)

**Cron Job Examples**

When specifying jobs, use the asterisk to specify all values. Putting a value in one of the fields only runs the command on that value. For example:

\* 2 0 \* 4 /root/backup.sh

Even though it’s set to run at 2 am, it only runs when the first of the month (0) falls on a Wednesday (4). If you change to the following:

\* 2 0 \* \* /root/backup.sh

The command runs the first of every month at 2 am. The following table provides a few basic commands using the /root/backup.sh file from our previous examples.

|  |  |
| --- | --- |
| **Cron Job** | **Command** |
| **Run Cron Job Every Minute** | ***\* \* \* \* \* /root/backup.sh*** |
| **Run Cron Job Every 30 Minutes** | ***30 \* \* \* \* /root/backup.sh*** |
| **Run Cron Job Every Hour** | ***0 \* \* \* \*/root/backup.sh*** |
| **Run Cron Job Every Day at Midnight** | ***0 0 \* \* \* /root/backup.sh*** |
| **Run Cron Job at 2 am Every Day** | ***0 2 \* \* \* /root/backup.sh*** |
| **Run Cron Job Every 1st of the Month** | ***0 0 1 \* \* /root/backup.sh*** |
| **Run Cron Job Every 15th of the Month** | ***0 0 15 \* \* /root/backup.sh*** |
| **Run Cron Job on December 1st – Midnight** | ***0 0 0 12 \* /root/backup.sh*** |
| **Run Cron Job on Saturday at Midnight** | ***0 0 \* \* 6 /root/backup.sh*** |