**Word Count: 1084**

Video, Automating and Scheduling Tasks

In this video, we will cover Basic Shell Scripting to Automate Tasks and Scheduling of Tasks using crontab.

It is essential to know about Shell Scripting to automate tasks in Raspberry Pi 4. A shell is a special user program that provides an interface to the user to use operating system services. Shell accepts human-readable commands from the user and convert them into something which the kernel can understand. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.

A shell is kind of like a spell that you come up with on the spot to solve some problem, then scripts are a lot like the pages of a spellbook where you keep the incantations that you've found really useful.

To speak with the shell, a special programming language called Bash was invented. Shell scripts allow you to automate pretty much anything you can do on the Linux command line. A shell script is simply a text file containing a series of shell commands that are executed one by one, as if they were input directly on the command line. They can save time if you need to execute lots of commands at once, or if you want to execute commands automatically, like when the Pi boots up.

The shell language is a high level programming language. It’s more removed from system and memory than lower level languages like assembly or C. This basically means that it has commands and functions to perform tasks that would otherwise take a lot of code to execute.

So what can a shell script do?

Shell scripts are great if you need to enter long sequences of commands into the command line to do something. Most operations can be accomplished with a single command if you know how to write a shell script for it. For example, they can be used to:

* Control what happens when the computer boots up
* Start applications when an event occurs
* Use the output of one command as the input for another command
* Batch rename and move files
* Batch convert file formats

One handy feature of a shell script is the ability to create pipes. A pipe allows the output of one command to be forwarded to the input of the next command. Pipes can be used with as many commands as you want.

Shell scripts are simply an executable text file with the extension “.sh”. In this example, we’ll write a simple “hello world” script to demonstrate how to create and run them.

To begin, log in to your Raspberry Pi, and navigate to the directory where you want to save the script. Then open the Nano text editor and create a new file named “hello-world.sh” by entering this at the command prompt:

**sudo nano hello-world.sh**

Now, enter this code into the text editor:

The first line of this program, **#!/bin/sh**, is called a shebang. This tells the BASH shell to execute the commands in the script. Every shell script you create will need this on the first line of the script. Exit and save the file in Nano by pressing Ctrl-X to save and exit.

Next, we’ll need to make the hello-world.sh file executable. To do that, enter this at the command prompt:

**sudo chmod +x hello-world.sh**

Now that the shell script has been made executable, we can run it. Navigate to the directory where the file is saved, and enter either of this. The words “Hello World!” will be printed to the line below the command prompt. This “hello world” script isn’t particularly useful, but it will show you the basics of how to create and run a shell script.

Next, shall we look at how to schedule a task. Well you’ll have to use a tool called crontab that allows you to launch tasks with the user you want, when starting the Raspberry or whenever you want.

Cron is a service, automatically started at each boot of the Raspberry Pi, which allows the user to execute scheduled commands. What we’re going to see today is how to tell cron to execute our command or script when needed.

A crontab is a tool that will allow us to list what we want to start, in a format understandable by the cron service.

A crontab will contain two things:

* The list of commands to run
* When to run them

Crontab is also a command. Here’s the syntax to open the crontab

**crontab -e**

First thing, you have to choose an editor. I advise you to stay on nano, so keep the default choice and hit enter :

That’s it. You are now in the editor of crontab, which is empty and can be a little intimidating if it’s the first time you access it. I’ll explain what to do.

First of all, all the lines starting with a # are comments, they do nothing, and we can ignore them.

Our changes will be made at the end of the file.

You are probably wondering what the five stars mean

The syntax of an entry in the crontab is as follows:

Now that you understand the theory, let’s look at a simple example to be sure it’s clear.

Imagine that you want to run the hello world script every Wednesday at midnight

You must add a line like this.

**0 0 \* \* 3 /home/pi/helloworld.sh**

There are a lot of flexibility with scheduling in crontab like

* Launch a script at fixed hours
* Start a script during a fixed interval
* Schedule a script only during the weekdays
* You can also start something on boot

Example scripts for all the above cases are given in the resources.

Now, I will give you an activity. You have to create a shell script to do **apt-update** and **apt-upgrade** and use crontab to make it start every time on boot. This will make sure that your raspberry pi will stay updated without manual intervention.

Summary,

In this video, we have covered the following topics

* Basic Shell Scripting to Automate Tasks
* Scheduling Tasks using crontab

Section Summary,

In this section, we have covered the following topics

* Basics of Linux
* Basic Linux commands
* Advanced Linux commands
* Automating & Scheduling Tasks

In the next section, we will learn basic Python programming.