Task Scheduling

Task scheduling is a feature in Linux systems that allows users to schedule and automate tasks. It allows administrators and users to run tasks at a specific time or within specific frequencies without having to start them manually. It can be used in Linux systems such as Ubuntu, Redhat Linux, and Solaris to manage a variety of tasks. Examples include automatically updating software, running scripts, cleaning databases, and automating backups. This also allows users to schedule regular and repetitive tasks to ensure they are run regularly. In addition, alerts can be set up to display when certain events occur or to contact administrators or users. There are many different use cases for automation of this type, but these cover most cases.

Systemd

Systemd is a service used in Linux systems such as Ubuntu, Redhat Linux, and Solaris to start processes and scripts at a specific time. With it, we can set up processes and scripts to run at a specific time or time interval and can also specify specific events and triggers that will trigger a specific task. To do this, we need to take some steps and precautions before our scripts or processes are automatically executed by the system.

1. Create a timer
2. Create a service
3. Activate the timer

Create a Timer

To create a timer for systemd, we need to create a directory where the timer script will be stored.

Create a Timer

```
amit8986@htb[/htb]$ sudo mkdir /etc/systemd/system/mytimer.timer.d
amit8986@htb[/htb]$ sudo vim /etc/systemd/system/mytimer.timer
```

Next, we need to create a script that configures the timer. The script must contain the following options: "Unit", "Timer" and "Install". The "Unit" option specifies a description for the timer. The "Timer" option specifies when to start the timer and when to activate it. Finally, the "Install" option specifies where to install the timer.

Mytimer.timer

Code: txt

```
[Unit]
Description=My Timer

[Timer]
OnBootSec=3min
OnUnitActiveSec=1hour

[Install]
WantedBy=timers.target
```

Here it depends on how we want to use our script. For example, if we want to run our script only once after the system boot, we should use `OnBootSec` setting in `Timer`. However, if we want our script to run regularly, then we should use the `OnUnitActiveSec` to have the system run the script at regular intervals. Next, we need to create our `service`.

Create a Service

Create a Service

```
amit8986@htb[/htb]$ sudo vim /etc/systemd/system/mytimer.service
```

Here we set a description and specify the full path to the script we want to run. The "multi-user.target" is the unit system that is activated when starting a normal multi-user mode. It defines the services that should be started on a normal system startup.

Code: txt

```
[Unit]
Description=My Service

[Service]
ExecStart=/full/path/to/my/script.sh

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

After that, we have to let `systemd` read the folders again to include the changes.

Reload Systemd

Reload Systemd

```
amit8986@htb[/htb]$ sudo systemctl daemon-reload
```

After that, we can use `systemctl` to `start` the service manually and `enable` the autostart.

Start the Timer & Service

Start the Timer & Service

```
amit8986@htb[/htb]$ sudo systemctl start mytimer.service
amit8986@htb[/htb]$ sudo systemctl enable mytimer.service
```

Cron

Cron is another tool that can be used in Linux systems to schedule and automate processes. It allows users and administrators to execute tasks at a specific time or within specific intervals. For the above examples, we can also use Cron to automate the same tasks. We just need to create a script and then tell the cron daemon to call it at a specific time.

With Cron, we can automate the same tasks, but the process for setting up the Cron daemon is a little different than Systemd. To set up the cron daemon, we need to store the tasks in a file called `crontab` and then tell the daemon when to run the tasks. Then we can schedule and automate the tasks by configuring the cron daemon accordingly. The structure of Cron consists of the following components:

Time Frame	Description
Minutes (0-59)	This specifies in which minute the task should be executed.
Hours (0-23)	This specifies in which hour the task should be executed.
Days of month (1-31)	This specifies on which day of the month the task should be executed.
Months (1-12)	This specifies in which month the task should be executed.
Days of the week (0-7)	This specifies on which day of the week the task should be executed.

For example, such a crontab could look like this:

Code: txt

```
# System Update
* */6 * * /path/to/update_software.sh

# Execute scripts
0 0 1 * * /path/to/scripts/run_scripts.sh

# Cleanup DB
0 0 * * 0 /path/to/scripts/clean_database.sh

# Backups
0 0 * * 7 /path/to/scripts/backup.sh
```

The "System Update" should be executed every sixth hour. This is indicated by the entry `*/6` in the hour column. The task is executed by the script `update_software.sh`, whose path is given in the last column.

The task `execute scripts` is to be executed every first day of the month at midnight. This is indicated by the entries `0` and `0` in the minute and hour columns and `1` in the days-of-the-month column. The task is executed by the `run_scripts.sh` script, whose path is given in the last column.

The third task, `Cleanup DB`, is to be executed every Sunday at midnight. This is specified by the entries `0` and `0` in the minute and hour columns and `0` in the days-of-the-week column. The task is executed by the `clean_database.sh` script, whose path is given in the last column.

The fourth task, `backups`, is to be executed every Sunday at midnight. This is indicated by the entries `0` and `0` in the minute and hour columns and `7` in the days-of-the-week column. The task is executed by the `backup.sh` script, whose path is given in the last column.

It is also possible to receive notifications when a task is executed successfully or unsuccessfully. In addition, we can create logs to monitor the execution of the tasks.

Systemd vs. Cron

Systemd and Cron are both tools that can be used in Linux systems to schedule and automate processes. The key difference between these two tools is how they are configured. With Systemd, you need to create a timer and services script that tells the operating system when to run the tasks. On the other hand, with Cron, you need to create a `crontab` file that tells the cron daemon when to run the tasks.

Start Instance

1 / 1 spawns left

Waiting to start...

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

+ 0 What is the type of the service of the "syslog.service"?

Submit your answer here...

Submit

Previous Next

Cheat Sheet

Go to Questions

Table of Contents

Introduction

Linux Structure

Linux Distributions

Introduction to Shell

The Shell

Prompt Description

Getting Help

System Information