WORK-CASE №7

***Виконали студенти Коваленко С. та Корольов Є.***

1. В ході роботи досить часто виникає завдання планування задач:

- Охарактеризуйте основні функції які може виконувати планувальник завдань в будь-якій ОС. Порівняйте можливості планування завдань в різних ОС на прикладі Windows та Linux.

The Task Scheduler is a system tool that allows you to automate the execution of tasks at a specified time or under certain conditions.

The main functions of the task scheduler include:

* Automation of task execution: Ensures that programs, scripts or commands are executed at a specified time or after a specific event.
* Flexibility in setting the execution time: Tasks can be run periodically (daily, hourly, daily, minutely) or at a single time.
* Interaction with system events: Respond to system events such as user logins, system startup, or triggers.
* Monitoring and logging: Maintain logs to record completed tasks, detect errors, and monitor the success of the execution.
* Task prioritisation: Allows you to set up a queue of tasks based on their importance.
* Multi-user support: Ability to run tasks on behalf of different users using their accounts.

**Task Scheduler in Windows**

Windows uses the Task Scheduler tool to automate the execution of tasks.

Features:

* Intuitive graphical interface for setting up tasks.
* Supports running programs, scripts, and commands at specified intervals.
* Ability to create tasks that are triggered by system events (for example, user login or system startup).
* Logging of completed tasks in the Event Viewer.
* Support for running tasks with administrator or other user rights.
* Example of use: Create a task to run a script daily at 12:00:
* Open Task Scheduler.
* Create a new task.
* Specify the trigger (daily) and action (script execution).

**Task scheduler in Linux**

In Linux, the main tools for scheduling tasks are cron (for regular tasks) and at (for one-time tasks).

Features:

*cron:*

* Suitable for periodic tasks.
* Tasks are configured in the crontab file.
* It is used to automate system maintenance (e.g., backups, clearing logs).

*at:*

* Allows you to run a task once in the future.
* Simple text-based command-line interface.
* Support for logs via the system log (/var/log).
* Flexibility due to the ability to execute any commands or scripts.

- Опишіть основні принципи роботи з планувальником Cron в ОС Linux. Як його налаштовувати? Чи є йому альтернативи (дайте їх характеристику).

Cron jobs are recorded and managed in a special file known as a crontab. Each user profile on the system can have their own crontab where they can schedule jobs, which is stored under /var/spool/cron/crontabs/.

To schedule a job, open up your crontab for editing and add a task written in the form of a *cron expression*. The syntax for cron expressions can be broken down into two elements: the schedule and the command to run.

The command can be virtually any command you would normally run on the command line. The schedule component of the syntax is broken down into 5 different fields, which are written in the following order:

|  |  |
| --- | --- |
| **Field** | **Allowed Values** |
| minute | *0-59* |
| hour | *0-23* |
| Day of the month | *1-31* |
| month | *1-12 or JAN-DEC* |
| Day of the week | *0-6 or SUN-SAT* |

Here’s a functional example of a cron expression. This expression runs the command curl http://www.google.com every Tuesday at 5:30 PM:

*30 17 \* \* 2 curl http://www.google.com*

**Two useful alternatives to Cron for Linux**

**1. Anacron**

Anacron is a scheduler that periodically executes commands just like cron. The only difference is that it doesn't require your computer to always be running. You can schedule a task to run at any time. When your computer is off, the task will be executed the next time you turn your computer on.

To get started, simply install the anacron application on your system. It is available in most package managers. Use the following command to install the application in Ubuntu:

sudo apt-get install anacron

Anacron works in a similar way to cron. You can add your job to the /etc/anacrontab file.

The syntax for adding a job is as follows:

period delay-after-computer-start job-identifier command

period delay-after-computer-start job-identifier command

For example, to run a backup script every day, you can add the following command:

1 15 cron.daily /bin/sh /home/damien/backup.sh

The disadvantage of anacron is that it can only be used to schedule daily, weekly and monthly jobs, you cannot use it for hourly or minute by minute jobs.

**2. fcron**

The fcron application is better than cron and anacron. It does not require your computer to run 24×7 and allows you to set hourly and minute by minute tasks.

During the installation you will be asked several questions and you can safely answer “Y” (yes) to each of them.

To start creating the fcron job, simply type fcron -e in the terminal.

The syntax of the fcron application is similar to the syntax of the cron application:

minute hour day-of-month month day-of-week command-to-be-executed

minute hour hour day-of-month month-day-of-week command-to-be-executed

But it has many more customization options than cron, for example, you can create jobs that will run in a specific time interval.

***Виконав студент Горохов Данило.***

2. Для вашої віртуальної машини зі встановленою ОС Linux здійсніть планування обраних вами задач (запуск додатків, вмикання/вимикання машини, очистка каталогів, видалення файлів, резервне копіювання, архівування тощо на ваш вибір) через планувальник Cron:  
- Виконання спланованої задачі в чітко визначений Вами час (наприклад о 8 ранку, 18.30 і т.д.).  
- Виконання однієї й тієї ж задачі двічі в день (час також визначаєте самостійно).  
- Виконання однієї й тієї ж задачі тільки в будні (або тільки у вихідні дні) у чітко визначений проміжок часу (наприклад з 8 до 18 години).  
- Виконання задач тільки раз у рік, раз у місяць, раз у день, щогодини, при вмиканні (після перезавантаження).

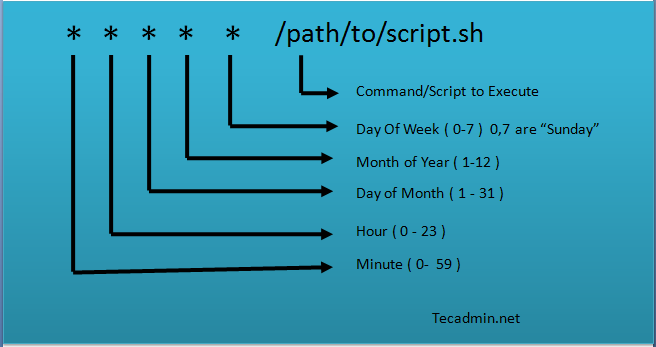
I don't have Cron installed on my virtual machine, so to install it, I need to write the command:



To get started with Cron, you need to write the command:



And choose the editor where we will write the planned tasks (I chose the one that is offered by default by Linux - the GNU Nano editor.

Cron Recording Format:

Executing the planned task at a clearly defined time (as a task – I chose to clean up the temporary /tmp folder at 08:00 in the morning:



Performing the same task twice a day (e.g. run the backup script /home/user/backup.sh at 09:00 and 21:00):



Performing the same task only on weekdays in a clearly defined period of time (for example, run the resource monitoring script /home/user/monitor.sh every hour from 08:00 to 18:00 on weekdays):



Performing a task once a year (e.g. system update on January 1, 05:00):



Performing the task once a month (delete old displayed messages (logs) every 1st day of the month at 02:00):



Perform the task once a day (check the disk status at 03:00):



Executing the task every hour (updating the database every hour):

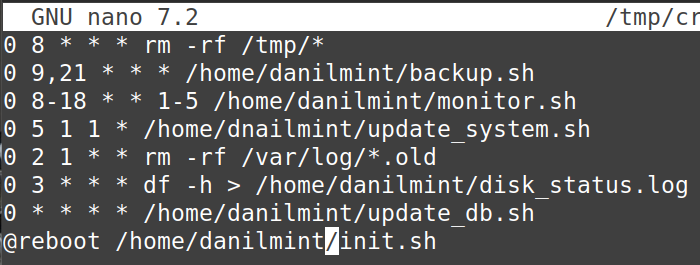


On reboot (run the initialization script):

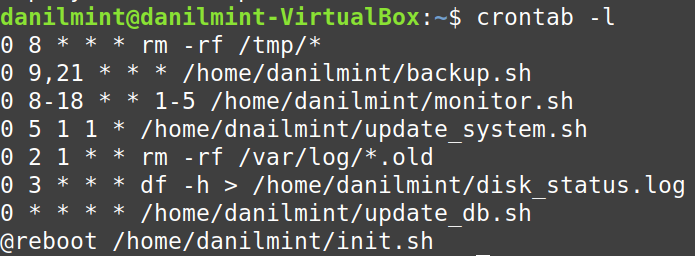


To exit the terminal, ctrl+x;

Instead of user, you need to specify the current username of the user, in my case danilmint.



See the list of tasks:

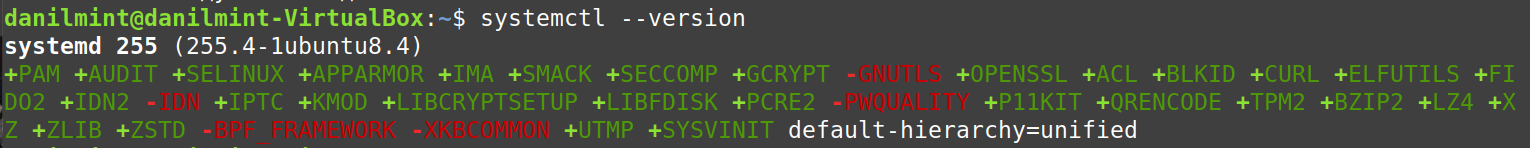


The scripts I mentioned above are not valid, so they need to be created and edited to suit our needs. To do this, use the commands: touch file.sh and

chmod +x file.sh. Then select the editor and edit the nano file.sh in it

3. Встановіть альтернативний Cron’у планувальник задач (на Ваш вибір). Виконані у завданні 2 дії продемонструйте через нього.

As an alternative to cron, I decided to install systemd timers. It is already integrated into most modern distributions. You can run a check to make sure:

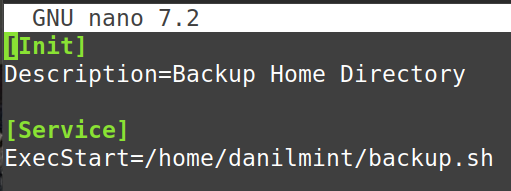
To complete tasks, you need to create a timer and then perform actions with it.

To do this:

Go to the directory to configure services and create a service file for the task (the name can be any):



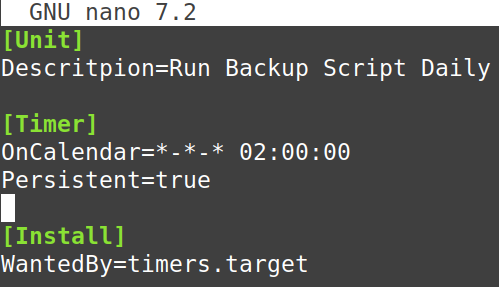
Adding content:



Creating a timer file:



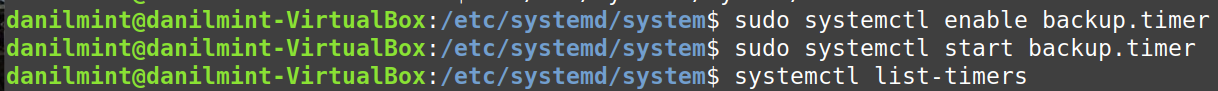
Adding content:



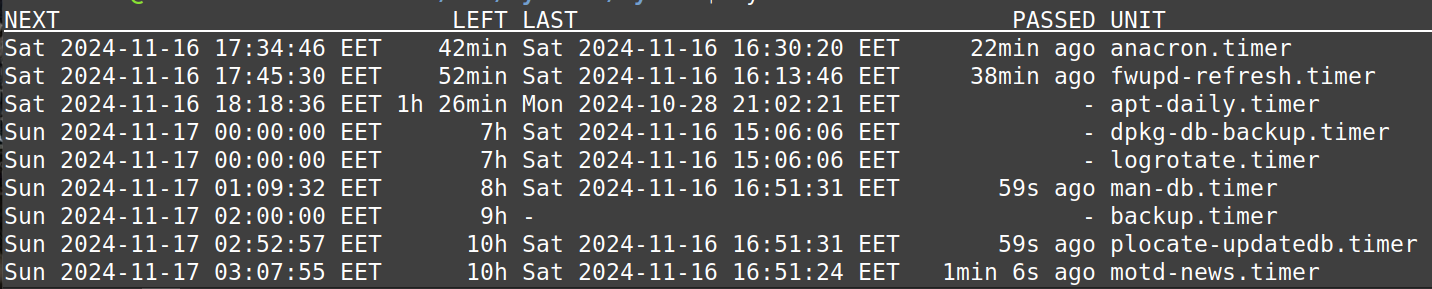
Restart systemctl to apply the changes:



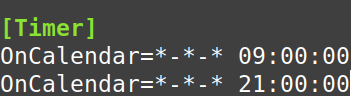
Turn off the timer, then start, check the status:



We get a timer:



Run the script twice a day:



Launch only on weekdays:



Launch once a year:



Starting when overloaded:

