Work-case №7

*Готувала матеріал студентка Малієнко А.*

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.

Task scheduling is a crucial concept in multitasking and multiprocessor systems, both in general-purpose operating systems and real-time operating systems. Scheduling involves assigning priorities to processes in a queue based on their priority levels. The utility that performs this task is called a scheduler.

The primary goal of task scheduling is to maximize the utilization of available resources. To ensure overall system productivity, the scheduler relies on the following principles:

1. Processor Utilization: Assign tasks to the processor(s) whenever possible.

2. Throughput: Maximize the number of processes executed in a unit of time.

3. Turnaround Time: Minimize the time taken for the completion of a specific process.

4. Waiting Time: Minimize the time a process spends waiting in the ready queue.

5. Response Time: Minimize the time from submitting a request to receiving the first response.

6. Fairness: Ensure equal distribution of processor time among threads or processes.

In the middle of calculating the real hour, for example, on devices intended for automatic control in industry (for example, robotics), the planner is responsible for ensuring the control of processes in the passage of specified hourly intervals (hours); This is critical for maintaining a correct robotic real-time system.

Task scheduling (also known as task planning or job scheduling) is an important feature of operating systems (OS) that allows you to automate the execution of certain tasks according to a specific schedule or at a specific time.

*Windows:*

1. task scheduler:

- Interface:

- It has a graphical interface that makes it easy to create and manage tasks.

- You can customize tasks by schedule, events, time events, etc.

- Supports creating tasks with administrator rights.

2. powershell:

- It is possible to create and manage tasks using PowerShell.

- Provides extensive capabilities for complex automated scheduling.

*Linux:*

1. cron:

- Interface:

- Usually configured through a text file or command line.

- Easily used to run tasks on a regular basis.

- Ability to use different levels of access rights.

2. at:

- Allows you to run a task once at a specified time.

- Uses a text file to define tasks.

3. systemd:

- Used to manage system processes, but can also be used to schedule tasks.

- Provides more advanced integration capabilities with other system services.

Both OSes have powerful task scheduling tools, and the choice between them may depend on the user's specific requirements and preferences.

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

In Linux, periodic execution is managed by the cron daemon. It starts at system startup and runs in the background until the system is shut down. Cron reads configuration files that contain lists of commands and their schedules. Commands are usually processed by the sh shell (the Linux system console), so almost all commands that can be executed manually in the console can be delegated to the cron daemon.

Daemon configuration files are managed by the crontab command, so they are called crontab files. There are many crontab files in the system, but we will edit /etc/crontab. Note that in CentOS and Fedora distributions, the cron daemon is renamed crond.

Each line of a crontab file that is not a comment contains six or seven fields and constitutes a single command. A standard line looks like this:

*хвилина година день місяць день\_тижня [ім’я\_користувача] команда*

The user\_name field defines the user on whose behalf the command will be executed. The minute\_hour, day\_month, and day\_week fields contain information about the time when the command is run.

Consider an example of a record:

30 2 \* \* 1 root tar -czf /root/backups/sites.tar.gz /var/www

This line will archive the /var/www directory, where all site data is located in most configurations, and create an archive in /root/backups. The command will be executed on Mondays at 2:30. However, this method of creating backups of site data has a significant drawback: backups will be created until the VPS runs out of hard disk space. To solve this problem, you need to write a script that will allow you to store no more than 4 copies of the archives.

**Backup of site data**

So, to start a site data backup, we log in to the VPS using SFTP as the root user. In the working directory (/root), create the folder ./backup/www-backup, where the backups will be placed. In the /root/backup folder, create a script that will run the cron scheduler to create backups. Create a new file, name it www-backup.sh, open it with a text editor and write the script code into it:

#!/bin/bash

cd /root/backup/www-backup

rm ./www-dump-3.tar.gz

mv www-dump-2.tar.gz www-dump-3.tar.gz

mv www-dump-1.tar.gz www-dump-2.tar.gz

mv www-dump-0.tar.gz www-dump-1.tar.gz

tar -czf www-dump-0.tar.gz /var/www

*Готував матеріал студент Мішин Артем*

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:

To set up a task schedule using the cron scheduler in Ubuntu Linux, you first need to open the crontab file for editing. Use the command:

crontab -e

Then add the necessary lines for your tasks. Note that the path to the executable file or command may vary in your case, and you must set the correct paths and parameters.

1. Run the task at 8 am and 6:30 pm each day:

0 8 \* \* \* /path\_to\_your\_command\_or\_script

30 18 \* \* \* /path to your command or script

2. Performing the task twice a day (for example, at 9 am and 8:30 pm):

0 9,20 \* \* \* /path\_to\_your\_team\_or\_script

3. Perform the task only on weekdays between 8 and 18 hours:

0 8-18 \* \* 1-5 /path\_to\_your\_command\_or\_script

In this example, 1-5 represents working days from Monday to Friday.

4. Run the task once a year, for example, on the first day of the year (January 1):

0 0 1 1 \* /path\_to\_your\_command\_or\_script

5. Execute the task once a month, for example, on the first day of the month:

0 0 1 1 \* /path\_to\_your\_command\_or\_script

6. Run the task once a day at 3:30 am:

30 3 \* \* \* /path\_to\_your\_command\_or\_script

7. Performing the task every hour:

0 \* \* \* /path\_to\_your\_command\_or\_script

8. Execute the task at startup (after reboot):

Add your task to `/etc/rc.local` before the line `exit 0`.

For example:

/path\_to\_your\_command\_or\_script &

*Готував матеріал студент*

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