“Київський фаховий коледж зв’язку”

Циклова комісія комп’ютерної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №10**

з дисципліни “Операційні системи”:

**Тема: “Зміна власників і прав доступу до файлів в Linux. Спеціальні каталоги та файли в Linux”**

Виконали студенти

групи РПЗ-13А та РПЗ-13Б

Eleven Two Zeroes:

Vlad Sapozhnyk

Max Karpenko

Dmytro Onufriiev

Київ 2024

Робота студентів групи РПЗ-13А та РПЗ-13Б Eleven Two Zeroes: Vlad Sapozhnyk, Max Karpenko and Dmytro Onufriiev.

**Мета роботи:**

1. Отримання практичних навиків роботи з командною оболонкою Bash.
2. Знайомство з базовими діями при зміні власників файлів, .прав доступу до файлів
3. Знайомство з спеціальними каталогами та файлами в Linux.

**Матеріальне забезпечення занять:**

1. ЕОМ типу IBM PC.
2. ОС сімейства Windows та віртуальна машина Virtual Box (Oracle).
3. ОС GNU/Linux (будь-який дистрибутив).
4. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux

**Завдання для попередньої підготовки: *Created by Vlad Sapozhnyk***

1. **Яке призначення команди id?**

The id command is a versatile tool used to display information about the current user or a specified user, including their user ID (UID), group ID (GID), and group memberships. It's a fundamental command for understanding user privileges and group associations in Unix-like operating systems.

1. **Як переглянути які права доступу має власник файлу?**

To view the permissions for the owner of a file, you can use the ls command along with the -l (long listing) option and the file path. The owner's permissions will be represented by the first three characters of the permission string.

*ls -l [file path]*

1. **\*Як змінити власника групи?**

To change the owner of a group in Linux, you can use the chgrp command followed by the new owner's username and the group name. For example, to change the owner of the group "sales" to the user "john", you would use the following command:

*sudo chgrp john sales*

1. **\*Як можна переглянути у терміналі який тип поточного файлу? Наведіть приклади для різних типів файлів**

*file [file path]*

1. **\*\*Для чого використовуються дозволи Setuid та Setgid?**

Setuid and Setgid are special permissions in Unix-like systems that allow a program to temporarily run with the privileges of a specific user or group, typically the root user or a group with administrative privileges. These permissions are used to grant programs the necessary access to perform certain tasks that require elevated privileges, without giving the program's actual user those same privileges.

1. **\*\*Для чого в системі потрібен так званий “липкий біт” (Sticky Bit). Наведіть приклади коли цей дозвіл доцільно використовувати.**

A "Sticky Bit" is a special permission on a file or directory in UNIX-like operating systems such as Linux. This permission has a special meaning and is intended to provide additional security and control over access to files and directories.

The main purposes of the sticky bit:

* Prevent users who are not the file owner or administrator from deleting files: When the sticky bit is set to a directory, it prevents users who are not the owner of the file or the system administrator from deleting files from that directory. This prevents other users from accidentally deleting important files.
* Ensure security in a shared directory: If several users have access to a shared directory and they store their files there, a sticky bit can prevent accidental deletion of other people's files.

Examples of using a sticky bit:

* Temporary directories that are accessible to all users: For example, a system temporary directory (/tmp) can have a sticky bit set to prevent users who are not the owner or system administrator from deleting files. This allows users to safely work in this directory without worrying about accidentally deleting files by other users.
* Shared directories for storing important files: For example, a shared directory where users store important documents or files may have a sticky bit installed to prevent other users from accidentally deleting files.

In these cases, a sticky bit helps to increase security and control over access to files and directories in shared environments where many users have access to the same resources.

**Glossary of terms**

| **Name of term** | **Explanation** |
| --- | --- |
| File ownership | A file property that determines which user and which group has access rights to a file. |
| User owner | The user who owns the file and determines the access rights of users to this file. |
| Group owner | The group to which the file belongs and determines the access rights to the file for all users belonging to this group. |
| Administrative privileges | Administrator privileges required to perform certain actions in the system, such as changing file ownership or changing access rights. |
| UID (User ID) | A unique user identifier in the system that is used to determine the owner of a file. |
| GID (Group ID) | A unique group identifier in the system that is used to determine the ownership group of a file. |
| /etc/passwd file | A text file on Unix-like systems that contains information about system users, including their identifiers and home directories. |
| id command | A command in Unix-like systems that displays information about a user and group, including the UID, username, GID of the main group, and other groups to which the user belongs. |

**Хід роботи. *Created by Max Karpenko***

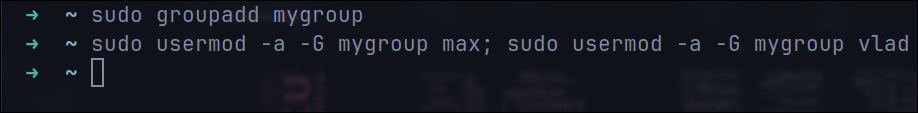
* 1. Початкова робота в CLI-режимі в Linux ОС сімейства Linux:
  2. Запустіть віртуальну машину VirtualBox, оберіть CentOS та запустіть її. Виконайте вхід в систему під користувачем: CentOS, пароль для входу: reverse ***(якщо виконуєте ЛР у 401 ауд.)*** та зпустіть термінал.
  3. Запустіть віртуальну машину Ubuntu\_PC ***(якщо виконуєте завдання ЛР через академію netacad)***
  4. Запустіть свою операційну систему сімейства Linux ***(якщо працюєте на власному ПК та її встановили)*** та запустіть термінал.
  5. Опрацюйте всі приклади команд, що представлені у лабораторних роботах курсу ***NDG Linux Essentials: Lab 17: Ownership and Permissions*** та ***Lab 18: Special Directories and Files.*** Створіть таблицю команд вивчених у п.2 ходу роботи у наступному вигляді:

| Назва команди | Її призначення та функціональність |
| --- | --- |
| ls -l | Відображає розширену інформацію про файли та директорії, включно з правами доступу, власником та групою. |
| chmod | Змінює права доступу до файлів або директорій, визначаючи, хто може читати, писати або виконувати файли. |
| chown | Змінює власника файлу або директорії, дозволяючи контролювати, хто має доступ до ресурсів. |
| chgrp | Змінює групу власності для файлів або директорій, що впливає на доступ до цих ресурсів залежно від групи користувача. |
| umask | Встановлює початкові права доступу для новостворених файлів і директорій в сесії користувача. |
| sudo | Дозволяє виконувати команди з правами суперкористувача, важлива для адміністрування системи. |
| find | Використовується для пошуку файлів в директорійній структурі на основі різних критеріїв, включаючи права доступу. |
| chmod +s | Встановлює біт setuid або setgid на виконуваних файлах, що дозволяє виконувати файл з правами власника файла або групи. |
| ln -s | Створює символьне посилання на файл або директорію, уможливлюючи легкий доступ до ресурсів через альтернативний шлях. |
| touch | Cтворює порожній файл або оновлює часові мітки існуючого файлу, часто використовується для управління файлами. |

* 1. **Виконайте наступні практичні завдання у терміналі наступні дії (продемонструвати скріншоти):**
* **створіть трьох нових користувачів;**

****

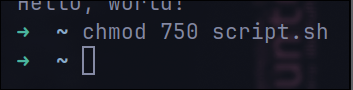
* **створіть нову групу користувачів, туди додайте двох, з трьох створених користувачів;**

****

* **створіть новий файл, який буде доступний на зчитування, редагування та виконання власником файлу, наприклад найпростіший скриптовий сценарій;**

****

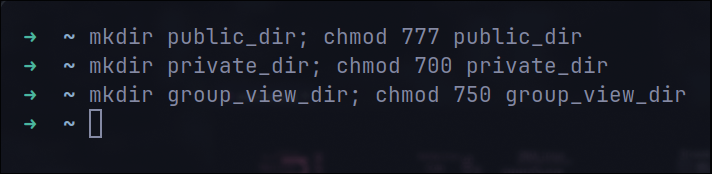
* **для користувачів групи власника надайте дозволи на перегляд та виконання (без дозволу на редагування) цього файлу;**

****

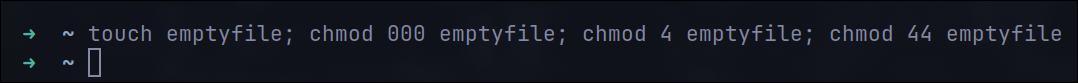
* **для інших користувачів заборонити доступ до цього файлу;**

The command above has already denied access to the file.

* **\*подібні дії виконайте для директорій - створіть директорію, яка буде доступна для всіх трьох користувачів, створіть директорію, яку буде доступна тільки для власника, створіть директорію, яку користувачі групи власника зможуть переглядати, але не редагувати;**

****

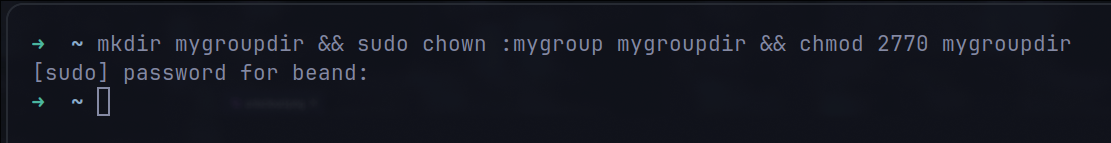
* **\*створіть порожній файл під назвою emptyfile за допомогою команди touch emptyfile. Тепер “обнуліть” дозволи для файлу з chmod 000 emptyfile. Що станеться, якщо змінити дозволи для emptyfile, передавши лише одне значення для chmod у числовому режимі, наприклад, chmod 4 emptyfile? Що буде, якщо ми використаємо два числа, наприклад chmod 44 emptyfile? Що ми можемо дізнатися про те, як chmod зчитує числове значення?**

****

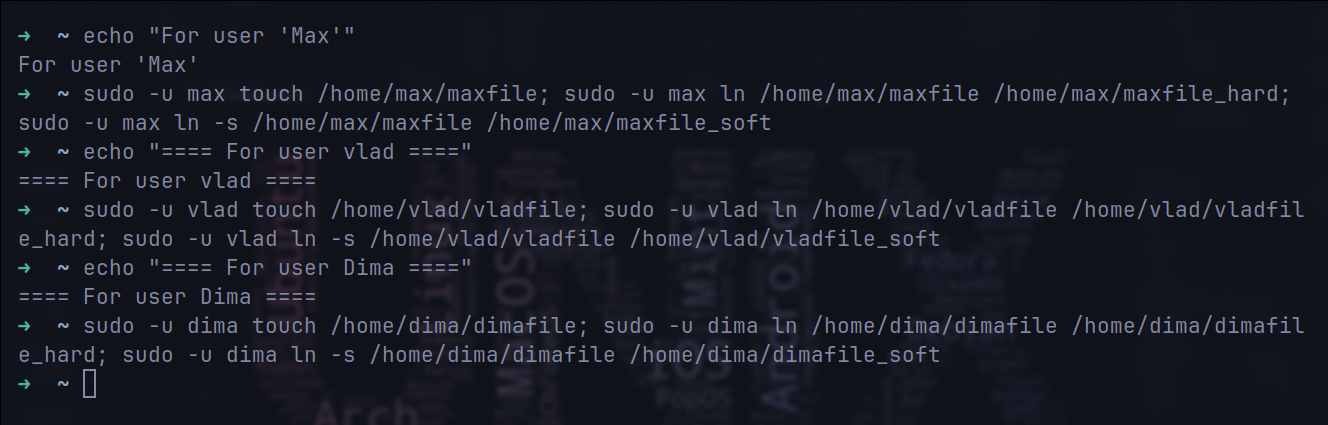
**How chmod reads a numeric value:**

chmod interprets numeric arguments as octal numbers. If fewer than three digits are specified, the missing digits are treated as leading zeros.

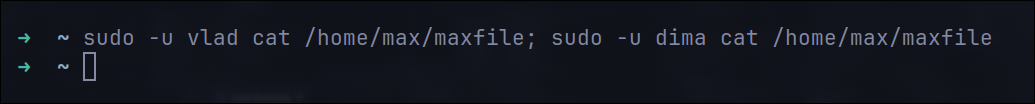
* **The first digit** from left to right corresponds to the owner's rights, the second to the group's rights, and the third to the rights of other users.
* **Each digit can be a sum:** 4 (read), 2 (write), and 1 (execute).
* **\*\*створіть каталог під назвою, де всі файли автоматично будуть належати Вашій групі користувачів і можуть бути видалені лише користувачем, який їх створив?**

****

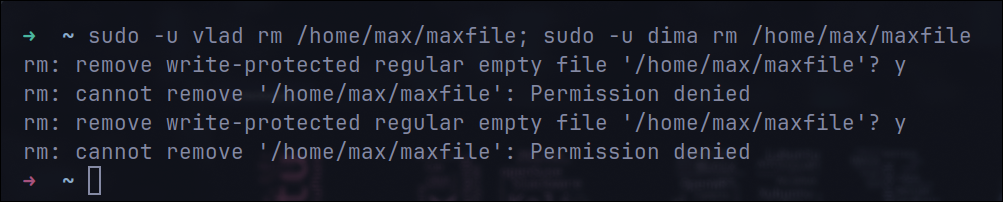
* **\*\*під кожним користувачем створіть по одному новому файлу, та жорстке та символічне посилання на нього;**

****

* **\*\*спробуйте іншими користувачами переглянути ці файли;**

****

* **\*\*спробуйте іншими користувачами видалити ці файли, зробіть висновки.**

****

**This shows that:**

* **Directory permissions on maxfile restrict users' actions -** Even if they can view files in the directory, permissions on the /home/max directory likely restrict write permissions, which is necessary to delete files.
* **Managing permissions is critical to protecting files -** Setting the proper permissions helps protect file data from improper modification or deletion, especially on multi-user systems.

**Контрольнi запитання: *Created by Dmytro Onufriiev***

1. **Наведіть приклади зміни прав доступу символічним методом (Symbolic Method)?**

The symbolic method of changing permissions in Linux allows you to change permissions on files or directories by specifying symbolic representations that reflect the current permissions and the operations to be performed. Below are some examples of how to use the symbolic method to change permissions:

Adding write permissions for the owner of a file:

*chmod u+w file.txt*

In this example, u corresponds to the file owner, + indicates adding a right, and w indicates a write right.

Removing execution rights for a group and other users:

*chmod go-x file.txt*

In this example, go corresponds to the group and other users, - indicates removal of the right, and x indicates the right to execute.

Setting the same rights for the owner, group, and other users:

*chmod a=rw file.txt*

In this example, a indicates all users, = indicates setting specific rights, and rw sets read and write permissions for all users.

1. **Наведіть приклади зміни прав доступу числовим методом (numeric method, octal method)?**

In the numeric method, also known as the octal or octal method, access rights are represented by a three-digit number, where each digit corresponds to rights for the owner, group, and other users. Each digit encodes a set of rights in octal notation. Let's look at a few examples:

Setting read and write permissions for the owner, and read permissions for the group and other users:

*chmod 644 file.txt*

In this example:

Permissions for the owner: 6, which corresponds to read and write permissions.

Permissions for the group: 4, which corresponds to read permission.

Permissions for other users: 4, which also corresponds to read permission.

Setting full rights for the owner, but restricting rights for the group and other users:

*chmod 700 file.txt*

In this example:

Permissions for the owner: 7, which corresponds to full rights (read, write, execute).

Permissions for the group and other users: 0, which corresponds to no rights.

Granting all users execution rights to execute the file:

*chmod 755 script.sh*

In this example:

Permissions for the owner: 7, which corresponds to full rights (read, write, execute).

Permissions for the group and other users: 5, which corresponds to read and execute rights.

1. **Яке призначення команди umask?**

The umask command in Linux is used to set or display the default permission mask for newly created files and directories in the current shell environment.

When a user creates a new file or directory, the default permissions are used. These permissions are calculated using the current permissions and the umask mask. Each bit of the umask mask indicates which bit of the permissions will not be set for newly created files or directories.

1. **Порівняйте жорсткі та символічні посилання?**

Hard links and symbolic links are two different mechanisms in UNIX-like operating systems for creating links to files or directories. Here is a comparison of them:

Hard links:

A hard link is an alternate name for a file or directory that refers to the same narrow file system node as the original.

They cannot point to directories.

They can only be used on the same file system.

If the original file is deleted, the hard link still points to the same content.

Changes made to the hard link or the original are reflected in both.

They have the same inode and file number as the original.

Symbolic links:

A symbolic link is a special file containing a path or URL that points to another file or directory.

They can point to any file or directory, including directories.

They can cross file system boundaries.

If the original file is deleted, the symbolic link becomes invalid (broken).

Changes in the original are reflected in the symbolic link, but changes in the link do not affect the original.

They do not have the same inode as the original, but only contain a link to another file.

Both types of links have their uses depending on the needs of the user and the requirements of the particular situation.

1. **\*Чи можна виконати файл, для якого є права на виконання, але не встановлені права на читання (--x)? Поясніть.**

For example, a file that has execute (--x) permissions but no read permissions can be executed.

Execute permissions allow you to run an executable or execute a command, and read permissions allow you to read the contents of the file. If you don't have read permissions, you can't read or copy the contents of the file, but that doesn't prevent you from running the file if you have execute permissions.

When you execute an executable file, the operating system loads the file's program code into memory and executes it without accessing its contents externally. So, if you have execute permission, you can run the file regardless of whether you have read permission.

It is important to note, however, that if the executable file directs its output to the standard output (stdout) and you do not have read permission, you will not be able to view that output in the console.

1. **\*Якщо ми змінюємо права доступу та дозволи в поточній сесії чи будуть вони збережені в наступній?**

Changes to access rights and permissions made in the current session will not be saved in subsequent sessions. These changes are only temporary and limited to the time area while the user remains in the current session.

In order to save changes to access rights and permissions for permanence, they need to be made in the appropriate configuration files or automation scripts that will be executed at system startup or on certain events.

1. **\*Чи є якийсь шаблон, яким система користується щодо прав та доступів при створенні нових файлів. Як можна змінити права дозволу за замовчуванням?**

Yes, the system uses the umask mask to set default permissions when creating new files and directories. The umask mask determines which access rights will be missing for new files and directories.

1. **\*Яким чином можна створити жорстке посилання? В яких ситуаціях їх доцільно використовувати?**

To create a hard link on Linux, use the ln command with the -s option:

*ln -s <target> <link>*

Where <target> is the path to the original file or directory, and <link> is the path to the new hard link you want to create.

For example, if you have a file called file.txt in the /home/user directory and you want to create a hard link to it in the current directory called link\_to\_file.txt, you can run the following command:

*ln /home/user/file.txt link\_to\_file.txt*

Hard links are useful in the following situations:

Creating short, convenient paths to files or directories: They allow you to create convenient paths to frequently used files or directories without having to copy or move files.

Supports compatibility with different file systems: Hard links can point to files within the same file system as the original, making them useful for creating links to files on the same partitions or disks.

Use in scripts and programs: Hard links can be useful for creating links to files that are used in programs or scripts where the file path may change or be long.

1. **\*Яким чином можна створити символічне посилання? В яких ситуаціях їх доцільно використовувати?**

To create a symbolic link on Linux, use the ln command with the -s option:

*ln -s <target> <link>*

Where <target> is the path to the original file or directory, and <link> is the path to the new symbolic link you want to create.

For example, if you have a file called file.txt in the /home/user directory and you want to create a symbolic link to it in the current directory called link\_to\_file.txt, you can run the following command:

*ln -s /home/user/file.txt link\_to\_file.txt*

Symbolic links are useful in the following situations:

Creating links to files or directories elsewhere in the file system: They allow you to create links to files or directories in different parts of the file system, including different disks or partitions.

Create convenient or short links to files or directories: Symbolic links allow you to create short, convenient paths to files or directories, regardless of their actual location.

Use in scripts and programs: Symbolic links can be useful for creating links to files that are used in programs or scripts where the file path may change or be long.

1. **\*\*Уявіть, що програмі потрібно створити одноразовий тимчасовий файл, який більше ніколи не знадобиться після закриття програми. Який правильний каталог для створення цього файлу?**

To create a one-time temporary file that will never be needed again after the program is closed, you can use the system's temporary files directory. On most UNIX-like systems, this directory is usually located in /tmp.

1. **\*\*Є файл оригінал та для нього створено два посилання - символічне та жорстке. Що відбудеться з іншими файлами, якщо видалити:**

* **файл оригінал;**
* **символічне посилання;**
* **жорстке посилання.**

When you delete different elements - the original file, a symbolic link, or a hard link - the consequences are different.

Deleting the original file:

If you delete the original file, it will be removed from the file system, but the hard link and symbolic link will remain. However, they will point to a non-existent file, so trying to access them may result in a “file not found” error.

Deleting a symbolic link:

If you delete a symbolic link, it will simply be deleted, and the original file and hard link will remain unchanged. Deleting a symbolic link doesn't affect the original file or any other links to it.

Deleting a hard link:

If you delete a hard link, the original file and all other links to it (including symbolic links) remain unchanged. A hard link is simply a different name pointing to the same file system node as the original file. Thus, deleting a hard link will not affect the file itself or other links to it.

In each case, deleting one of the items will not automatically delete the other items.

**Висновок:**

During this group work, we gained practical skills in working with the Bash command shell, which is an important component in the administration and operation of the Linux operating system. We learned how to change file owners and permissions, which are important aspects of file management and security in the Linux environment.

In addition, we also explored special directories and files in Linux, such as /tmp and /dev, and understood their purpose and features. This will allow us to better understand the structure of the Linux file system and use it more effectively for various tasks.