“Київський фаховий коледж зв’язку”

Циклова комісія комп’ютерної та програмної 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 КСМ-23а

Команда

Bald-Hedgehogs:

Сарапин Я.О.,

Кошкін І.О.,

Кочубей С.С.

Перевірила викладач

Сушанова В.С.

Київ 2024

Робота студент(а/ки/ів) групи КСМ-23а Команда Bald-Hedgehogs: Сарапин Я.О., Кошкін І.О., Кочубей С.С.

**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: (Kochubei S. 9v)**

1. **Словник термінів**

* **Ownership** – власність, прив’язка файлу до конкретного користувача і групи.
* **Permissions** – дозволи, права доступу до файлу або каталогу (читання, запис, виконання).
* **Chmod** – команда для зміни прав доступу до файлів і директорій.
* **Chown** – команда для зміни власника файлу або групи.
* **Chgrp** – команда для зміни групи файлу або каталогу.
* **Setuid** – спеціальний біт дозволу, який дозволяє виконувати файл від імені власника.
* **Setgid** – спеціальний біт дозволу, який дозволяє виконувати файл від імені групи.
* **Sticky Bit** – спеціальний біт дозволу, який запобігає видаленню файлів у загальній директорії користувачами, що не є власниками файлів.
* **Symbolic Link** – символічне посилання, яке вказує на інший файл або директорію.
* **Hard Link** – жорстке посилання, яке фізично вказує на той самий файл у файловій системі.

4.1. Призначення команди **id**  
Команда **id** використовується для відображення інформації про користувача, включаючи **UID** (ідентифікатор користувача), **GID** (ідентифікатор основної групи) та групи, до яких належить користувач.

4.2. Як переглянути, які права доступу має власник файлу?  
Для перегляду прав доступу власника файлу використовується команда **ls -l.** Вивід показує тип файлу, права власника, групи та інших користувачів.

4.3. Як змінити власника групи?  
Команда **chgrp** змінює групу файлу або каталогу.  
Команда **chown** також може змінити власника групи, якщо використовується з параметром **:group.**

4.4. Як можна переглянути у терміналі, який тип поточного файлу?  
Для визначення типу файлу використовується команда **file.** Вона виводить інформацію про тип вказаного файлу.

4.5. Для чого використовуються дозволи **Setuid** та **Setgid**?

**Setuid** дозволяє виконувати файл із привілеями власника файлу, навіть якщо користувач є іншим. Це корисно для виконання програм, які вимагають підвищених привілеїв (наприклад, **passwd**).

**Setgid** діє аналогічно, але надає права групи файлу. Для каталогів це означає, що всі файли, створені всередині, успадковують групу каталогу.

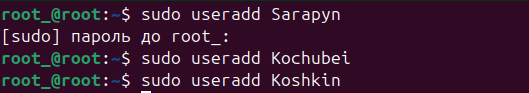
4.6. Для чого потрібен “липкий біт” (**Sticky Bit**)?  
**Sticky Bit** використовується для запобігання видаленню або перейменуванню файлів у загальних директоріях (наприклад, **/tmp**) користувачами, які не є власниками цих файлів.

Після встановлення **Sticky Bit** у загальній директорії лише власник файлу або адміністратор може видаляти свої файли.

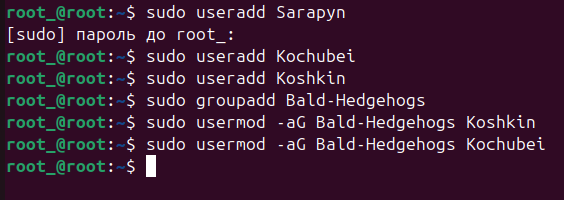
**Хід роботи**: (Sarapyn)

|  |  |
| --- | --- |
| Команда | її призначення |
| chmod | змінює права доступу до файлів, директорій |
| chown | змінює власника файлу або директорії |
| chgrp | змінює групу, до якої належить файл |
| ls -l | список файлів з детальною інформацією |
| ls -d | список директорій |
| chmod a+x file | дає користувачам права на виконання файлу |
| chmod g-w file | забирає права на запис для групи власників файлу |
| chmod go+r file | додає право на читання для групи власників та інших користувачів. |
| chmod o=rwx | встановлює права для інших користувачів на читання, запис і виконання файлу. |
| stat | показує детальну інформацію про файл, включаючи права доступу та власність |
| passwd | змінює пароль користувача |
| ln | створює жорсткі та символічні посилання на файли або каталоги |

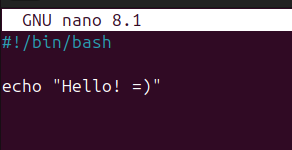
First of all I created 3 new users with command useradd:

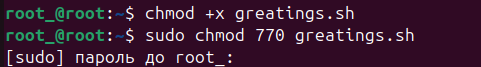


Next step I create new group and add its members:

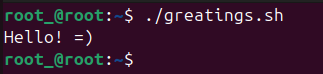


After that I create simple bush script in nano redactor:

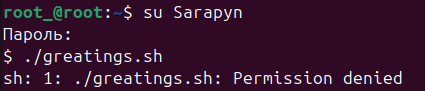




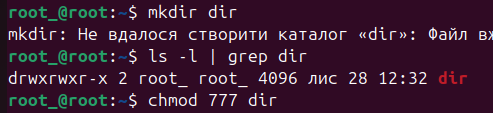
For users in the owner group, grant view and execute permissions to the file:

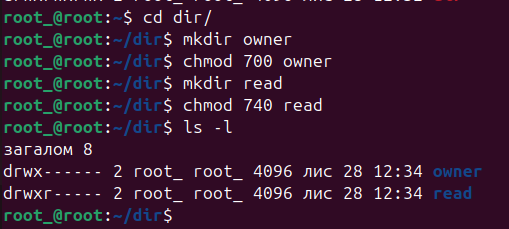


For other users forbid view and execute permissions to the file:

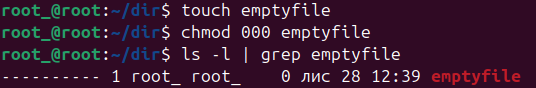


Create a directory accessible to all three users Create a directory accessible only to the owner Create a directory that users in the owner group can view but not edit:

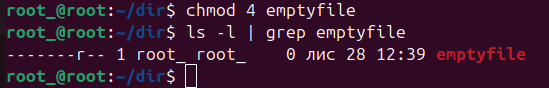




Than I create empty file and reset the permissions for the file with chmod 000 emptyfile:

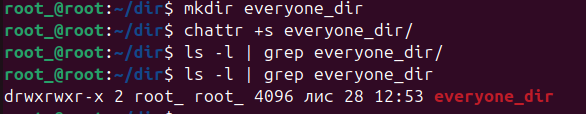


And change permissions with command chmod 4:

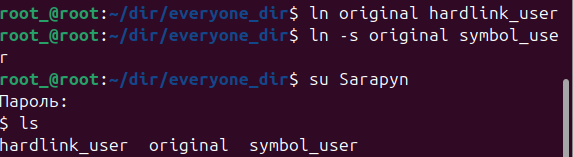


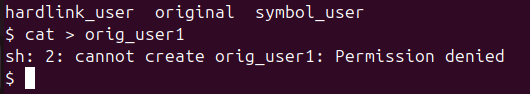
The chmod command interprets numbers in a way similar to how we do, filling in empty places with zeros. So, if you enter chmod 4 file, the terminal processes it as chmod 004 file.

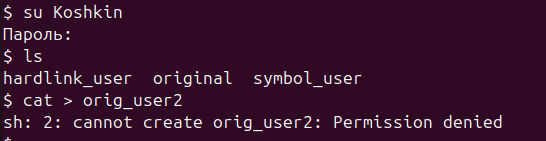
I created a directory and set it up so that all files in it are automatically owned by my user group and can only be deleted by those who created them:



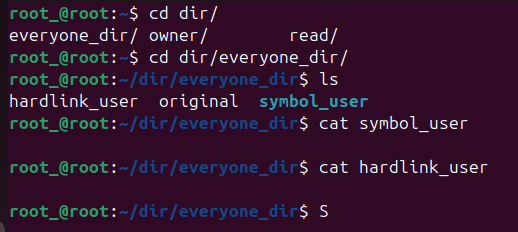
Next step I created one new file under each user, and a hard and symbolic link to it:







And I try to view this file with root-user:



Conclusion: I may have made a mistake somewhere, but users could not view this file except as root.

**Контрольны запитання:**(Koshkin I)

### **1. Приклади зміни прав доступу символічним методом.**

The symbolic method uses letters to represent access rights: r (read), w (write), x (execute). For each category of user (owner, group, other), a set of rights is set. Example: chmod u+x,g+rw file.txt

### **2. Приклади зміни прав доступу числовим методом.**

The numeric method uses an octal number to represent access rights. The first digit defines the owner's rights, the second digit defines the group's rights, and the third digit defines other rights. Each digit consists of three bits representing read, write, and execute permissions. **Приклад:** chmod 755 file.txt

### **3. Призначення команди umask**

The umask command sets a mask that determines which permissions will be subtracted from the standard permissions when creating new files or directories.

### **4. Порівняння жорстких та символічних посилань**

* **Жорстке посилання:** This is one name for the same file. They have the same inode and cannot exceed the file system boundaries.
* **Символічне посилання:** This is a pointer to another file or directory. They may exceed the limits of the file system and may point to non-existent files.

### **5. Чи можна виконати файл без прав на читання?**

**No, read permissions are required to execute a file. This is because executing a file involves reading its contents.**

### **6. Збереження прав доступу при зміні сесії**

Changes to access rights made in the current session are saved to disk and will be valid for subsequent sessions.

### **7. Шаблон прав доступу при створенні нових файлів**

When creating new files, the system uses the umask mask to determine access rights. To change the default permissions, change the umask value in your shell.

### **8. Створення жорсткого посилання**

* **Команда:** ln source\_file target\_file
* **Використання:** Useful for creating multiple names for a single file when they all need to have the same access rights.

### **9. Створення символічного посилання**

* **Команда:** ln -s source\_file target\_file
* **Використання:** Useful for creating shortcuts to files or directories, organizing the file system, and creating links to files in other file systems.

### **10. Каталог для тимчасових файлів**

The /tmp directory is usually used for temporary files that need to be deleted after the program is finished.

### **11. Видалення файлів та посилань**

* **Видалення оригінального файлу:** All hard links will become invalid, and symbolic links will point to a non-existent file.
* **Видалення символічного посилання:** Does not affect the original file or hard links.
* **Видалення жорсткого посилання:** Does not affect the original file or other hard links as long as at least one exists.

**Висновки: (Kochubei S. 9v)**

In this practical session, we explored the fundamental concepts of file ownership and permissions in Linux, along with the usage of special directories and files. Using the Bash command line, we learned how to navigate the file system, create and manage directories and files, and effectively utilize commands such as chmod, chown, and chgrp to modify permissions and ownership.

We gained hands-on experience in setting specific access rights for users, groups, and others, ensuring controlled access to sensitive files. Additionally, we worked with symbolic and hard links, understanding their differences and practical applications.

Furthermore, we explored special directories like /tmp and /var/tmp and examined system-critical files such as /etc/shadow and /usr/bin/passwd. This allowed us to better understand their roles in the system’s security and operations.

By the end of this session, we not only enhanced our skills in managing file permissions but also developed a deeper understanding of how Linux handles file security and system integrity through its permission system and special directories.