“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”**

Виконав(ла/ли)

студент(ка/и)

групи: КСМ-23а

The Awkward Turtles:

Когут Б.М.,

Михайленко О.О.,

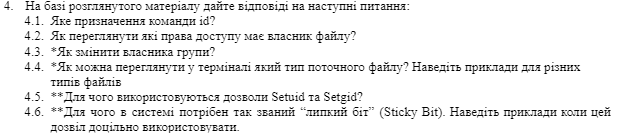
Трощинський Я.П.

Перевірила викладач

Сушанова В.С.

Київ 2024

**Виконав Когут Богдан**

****

1. What is the purpose of the id command?
2. How can you check what access rights the file owner has?
3. How to change the group owner?
4. How can you determine the type of the current file in the terminal? Provide examples for different file types.
5. What are the purposes of the Setuid and Setgid permissions?
6. Why is the "Sticky Bit" needed in the system? Provide examples when this permission is appropriate to use.
7. **Purpose of id:**  
   Displays user ID, group ID, and group memberships.  
   Example: id username.
8. **Check access rights:**  
   Use ls -l file.txt. The first three characters (rw-) show the owner's permissions.
9. **Change group owner:**  
   Use chgrp group\_name file.txt. For directories, add -R for recursive changes.
10. **Determine file type:**  
    Use file file\_name. Examples:

Text: ASCII text.

Script: Bourne-Again shell script.

Executable: ELF 64-bit executable.

1. **Setuid and Setgid:**

**Setuid:** Run file with owner's permissions (e.g., passwd).

**Setgid:** Run file or inherit directory group permissions.

1. **Sticky Bit:**  
   Restricts file deletion to the owner. Example: /tmp. Add with chmod +t dir\_name. Check with ls -ld.

****

Create a table of commands learned in step 2 of the workflow for NDG Linux Essentials: **Lab 17: Ownership and Permissions** and **Lab 18: Special Directories and Files** in the following format.

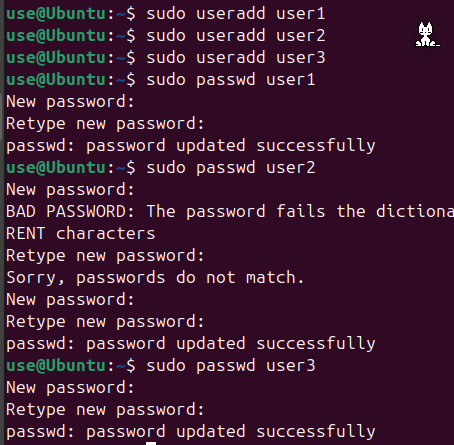
**Lab 17:**

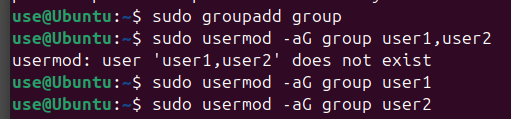
|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Command** |  |  | | --- | |  | | **Purpose and Functionality** |
| chmod | Modifies the file permissions using symbolic or numeric modes. Example: chmod a+x file (gives everyone execute permission). |
| chown | Changes the ownership of a file or directory to a specified user or group. |
| chgrp | Changes the group ownership of a file or directory. |
| mkdir | Creates new directories. Example: mkdir newdir (creates a directory named newdir). |
| ls -l | Lists files and directories with detailed information, including ownership and permissions. |
| ls | Lists files and directories in the current directory. |
| -d | Indicates that the file is a directory. Example: ls -d \*/ lists only directories. |
| - | Indicates a regular file. Example: ls will show regular files with -. |
| l | Indicates a symbolic link. Example: ls -l shows symbolic links with l. |
| b | Indicates a block device file. Example: ls -l shows block device files with b. |
| c | Indicates a character device file. Example: ls -l shows character device files with c. |
| p | Indicates a pipe file. Example: ls -l shows pipe files with p. |
| s | Indicates a socket file. Example: ls -l shows socket files with s. |
| -a | Lists all files, including hidden files (those starting with .). Example: ls -a lists all files. |
| -la | Lists all files with detailed information, including hidden files. Example: ls -la lists all files with details. |
| -d | Lists only directories. Example: ls -d \*/ lists only directories. |
| chmod a+x file | Gives everyone execute permission on the file. |
| chmod g-w file | Removes write permission for group owners on the file. |
| chmod go+r file | Adds read permission for the group owner and others on the file. |
| chmod o=rwx | Sets the permissions for others to read, write, and execute on the file. |
| date | Displays or sets the system date and time. |
| stat | Displays detailed information about a file, including permissions and ownership. |
| -R | Recursively applies changes to directories and files within them. Example: chmod **-R 755 dir** applies the permission change to all files in the directory dir. |

**Lab 18:**

|  |  |
| --- | --- |
| **Command** | **Purpose and Functionality** |
| -d | Lists only directories when used with ls. Example: ls -d \*/ lists only directories. |
| ls | Lists files and directories in the current directory. |
| -l | Displays detailed information about files and directories, including permissions, ownership, and more. Example: ls -l shows details about files. |
| passwd | Changes the user password or updates the password for the current user. |
| wall | Sends a message to all users currently logged into the system. Example: wall "System maintenance in 10 minutes" sends a message to all users. |
| -i | Used with commands like rm to prompt for confirmation before deletion. Example: rm -i file asks for confirmation before removing the file. |
| ln | Creates hard and symbolic links to files or directories. Example: ln file link creates a hard link to the file. ln -s file link creates a symbolic link. |
| rm | Deletes files or directories. Example: rm file deletes a file, rm -r dir deletes a directory. |
| -s | Creates a symbolic (soft) link to a file or directory when used with ln. Example: ln -s file link creates a symbolic link to the file. |

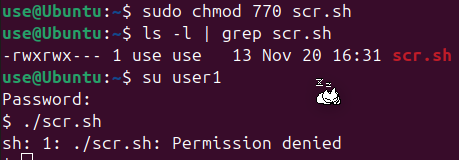
**Виконав Трощинський Ярослав**

* 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;

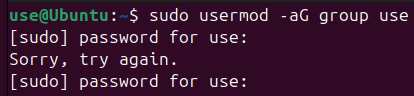


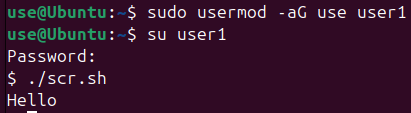
* створіть новий файл, який буде доступний на зчитування, редагування та виконання власником файлу, наприклад найпростіший скриптовий сценарій;



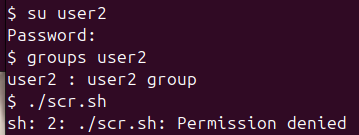


* для користувачів групи власника надайте дозволи на перегляд та виконання (без дозволу на редагування) цього файлу;

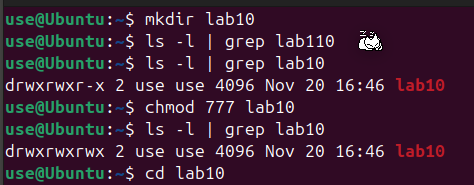


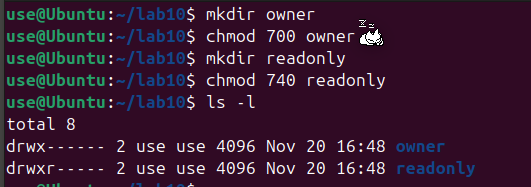


* для інших користувачів заборонити доступ до цього файлу;

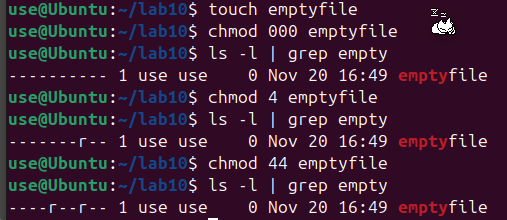


* \*подібні дії виконайте для директорій - створіть директорію, яка буде доступна для всіх трьох користувачів, створіть директорію, яку буде доступна тільки для власника, створіть директорію, яку користувачі групи власника зможуть переглядати, але не редагувати;



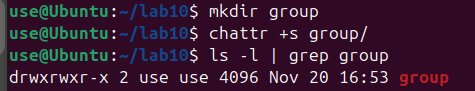


* \*створіть порожній файл під назвою emptyfile за допомогою команди touch emptyfile. Тепер “обнуліть” дозволи для файлу з chmod 000 emptyfile. Що станеться, якщо змінити дозволи для emptyfile, передавши лише одне значення для chmod у числовому режимі, наприклад, chmod 4 emptyfile? Що буде, якщо ми використаємо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

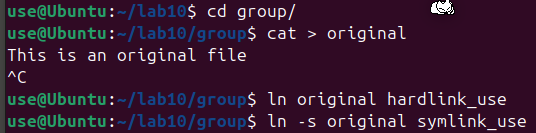


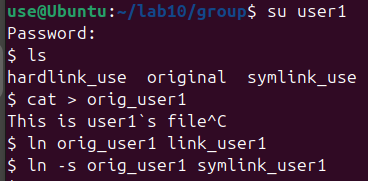
Command chmod reading numbers similar to us, it replaces empty places with 0, so if we insert **“chmod 4 file”**, terminal will read it like **“chmod 004 file”**

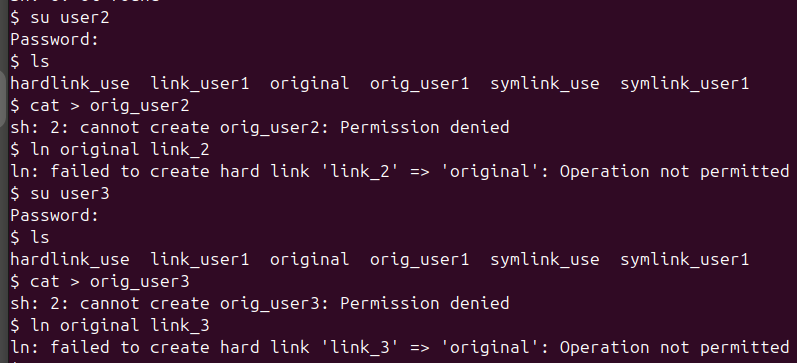
* \*\*створіть каталог під назвою, де всі файли автоматично будуть належати Вашій групі користувачів і можуть бути видалені лише користувачем, який їх створив?



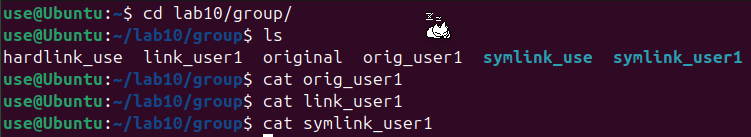
* \*\*під кожним користувачем створіть по одному новому файлу, та жорстке та символічне посилання на нього;

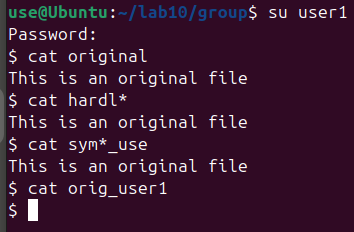


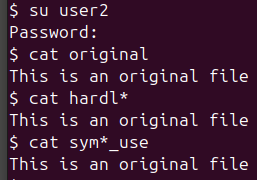




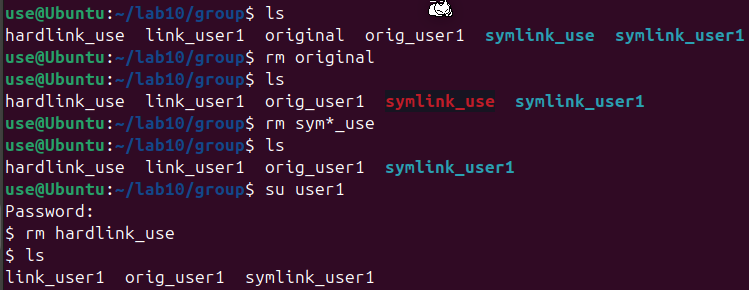
* \*\*спробуйте іншими користувачами переглянути ці файли;

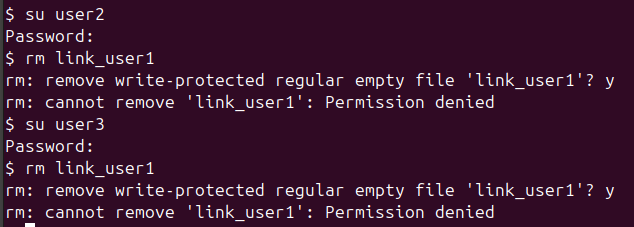






* \*\*спробуйте іншими користувачами видалити ці файли, зробіть висновки.





Users cannot remove files if they do not have a write-permission.

**Виконав Михайленко О**

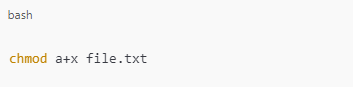
### **1. Examples of Changing Permissions Using the Symbolic Method**

The symbolic method uses characters to represent permissions:

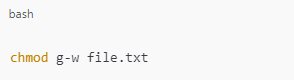
* u (user), g (group), o (others), a (all).
* + (add), - (remove), = (set).

**Examples:**

* Add execute permission for everyone:

****

Remove write permission for the group:

****

Set read-only permissions for others:

****

### **2. Examples of Changing Permissions Using the Numeric Method (Octal Method)**

The numeric method uses digits to set permissions:

* 4 — read (r).
* 2 — write (w).
* 1 — execute (x).

**Examples:**

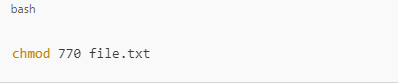
* Full access for the user, read/execute for the group and others:



Read-only access for everyone:



Deny access to others:



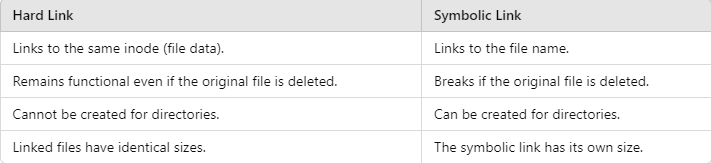
### **3. Purpose of the** umask **Command**

The umask command sets default permissions for new files and directories, restricting the allowed permissions.

* Formula: **Permissions = Maximum Permissions - umask**.
* Example: If umask = 022, then:
  + Files: 666 - 022 = 644 (read/write for the owner, read-only for others).
  + Directories: 777 - 022 = 755 (full access for the owner, read/execute for others).  
    **View the current umask value:**



4.Hard Links vs. Symbolic Links



### **5. Can a File Be Executed with** --x **Permissions but No Read (**--r**)?**

Yes, a file with execute (--x) permissions can be run, but the program must execute without being loaded into memory. However, most interpreters and compilers require read (r) access.

### **6. Are Permission Changes Retained Across Sessions?**

Yes, changes made with chmod are stored in the filesystem and persist across sessions until explicitly changed again.

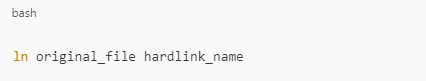
### **7. Default Permission Template and Modifying It**

By default, new files are created with 666 (files) or 777 (directories), limited by the current umask.  
**Change default permissions:**



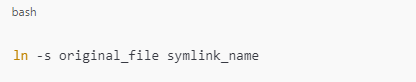
### **8. How to Create a Hard Link and Its Use Cases**

**Creation:**



### **9. How to Create a Symbolic Link and Its Use Cases**

**Creation:**



### **10. Correct Directory for Temporary Files**

Temporary files that are automatically deleted should be created in /tmp.

### **11. What Happens to Links When Files Are Deleted?**

* **Original File:**
  + Hard links remain functional as they point to the same inode.
  + Symbolic links become "broken."
* **Symbolic Link:**
  + Deleting a symbolic link does not affect the original file or other links.
* **Hard Link:**
  + Deleting a hard link reduces the link count. The file is deleted only when all hard links are removed.

### **Conclusion**

The purpose of this work was to gain practical skills in using the Bash command shell and to become familiar with fundamental operations for managing file ownership and permissions. Additionally, it involved exploring special directories and files in Linux.