“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 БІКС-13

Команда JRSY: Андрущик П.С   
Бурбан Д.Ю.  
Перевірив викладач

Сушанова В.С.

Київ 2024

**The goal of the work: (робила студентка Андрущик Поліна)**1. Getting hands-on skills with the Bash shell.  
2. Getting to know the basic actions when changing file owners, file access rights  
3. Introduction to special directories and files in Linux.  
  
**1. \*Read the short theoretical information for the laboratory work and make a small dictionary of basic English terms for the assignment of commands and their parameters.**

|  |  |
| --- | --- |
| **Term in English** | **Term in Ukrainian** |
| User Owner | Користувач, якому належить файл. За замовчуванням користувач, який створює файл, стає його власником. Право власності користувача можна змінити, але зазвичай потрібні права адміністратора. Право власності пов’язане з ідентифікатором користувача (UID). |
| Group Owner | Кожен файл також має власника групи. За замовчуванням власником групи стає основна група користувача, який створює файл. Користувачі можуть змінити групу власників файлів, якими вони володіють, на будь-яку групу, до якої вони належать. Право власності на групу пов’язано з ідентифікатором групи (GID). |
| UID | Ідентифікатор користувача. Числовий ідентифікатор, пов’язаний з обліковим записом користувача. |
| GID | ID групи. Числовий ідентифікатор, пов’язаний з обліковим записом групи. |
| Sticky bit | Спеціальний дозвіл для каталогів, щоб гарантувати, що лише власник файлу може видалити його, навіть якщо інші мають дозвіл на запис до каталогу. |

**On the basis of the considered material, answer the following questions:  
1. What is the purpose of the id command?**The id command is used to retrieve information about a user's identity, including a unique user identifier (UID), username, group identifier (GID), and group membership.  
**2. How to see what access rights the owner of a file has?**To view the access rights of a file's owner, you can use the ls -l command, which will display detailed information about the file, including access rights, owner, and owner's group.  
**3. \*How to change the group owner?**To change the owner of a group, use the chown command, specifying the new group owner and the file or directory to which you want to apply the changes.   
sudo chown :new\_owner\_group file\_or\_directory **4. \*How can you view the current file type in the terminal? Give examples for different types of files**To view the current file type in the terminal, you can use the file command.For a text file:  
file example.txt  
  
For an executable file (program):  
file my\_program  
  
For the catalog:  
file my\_directory  
**5. \*\*What are Setuid and Setgid permissions used for?**Setuid (Set User ID) and Setgid (Set Group ID) permissions are used to change the execution context of a program or access files depending on the owner of the file. When a program has Setuid permission, it runs with the rights of the owner of the file, not the user who launched it. This can be useful for applications that require special privileges, such as access to system resources. Setgid allows a program or file to use the rights of the owner's group instead of the group of the user who launched the program. This can be useful for group work on files when users belong to different groups.  
**6. \*\*Why does the system need the so-called “sticky bit” (Sticky Bit)? Give examples of when this permission should be used.**Sticky Bit is used in the system to prohibit the deletion or renaming of files except for the file owner, directory owner, and system administrator. This is especially useful in situations where multiple users have access to shared directories or files, and you want to ensure that each user can create, modify, or delete their own files in those directories, but cannot delete or modify other users' files.

For example, consider a situation where a system administrator creates a shared directory to which different users have access. To prevent accidental or intentional deletion or renaming of files by other users, the administrator can set the sticky bit on this directory. Therefore, while users can create, modify, or delete their files in this directory, they will not be able to delete or rename the files of other users who also have access to this directory.

Using the sticky bit allows you to preserve the confidentiality and security of files in shared directories, to avoid accidental or intentional deletion of files by other users, and to ensure appropriate management of access to system resources.  
  
**The main positions of the course of work:(робив Бурбан Данило)**

1. Initial work in CLI mode in Linux OS of the Linux family:

Start the VirtualBox virtual machine, select CentOS and run it. Log in to the system as user: CentOS, password for login: reverse (if you run LR in 401 aud.) and launch the terminal.

Start the Ubuntu\_PC virtual machine (if you are doing the LR tasks through the netacad academy)

1. Start your Linux operating system (if you're on your own PC and have it installed) and launch a terminal.

| Command | Purpose and Functionality |
| --- | --- |
| chgrp | Changes the group ownership of a file or directory. |
| chown | Changes the owner of a file or directory. |
| mkdir | Creates a new directory (folder). |
| ls | Lists the contents of the current directory or another specified directory. |
| chmod | Changes the access permissions of a file or directory. |
| stat | Displays information about the status of a file or directory, such as the last modification date and time, access rights, owner, etc. |
| passwd | Allows changing a user's password. |
| wall | Sends a message to all users currently logged into the system. |
| ln | Creates a link (either symbolic or hard) to a file or directory. |
| rm | Removes (deletes) a file or directory. When used with specific options, it can also recursively delete the contents of a directory or even prompt for confirmation. |

1. Perform the following practical tasks in the terminal following actions (show screenshots):

create three new users;

create a new user group, add two of the three created users there;

create a new file that will be available for reading, editing and execution by the owner of the file, for example, the simplest script script;

for users in the owner group, grant view and execute permissions (without edit permission) to this file;

prohibit access to this file for other users;

\*do similar actions for directories - create a directory that will be accessible to all three users, create a directory that will be accessible only to the owner, create a directory that users of the owner group can view, but not edit;

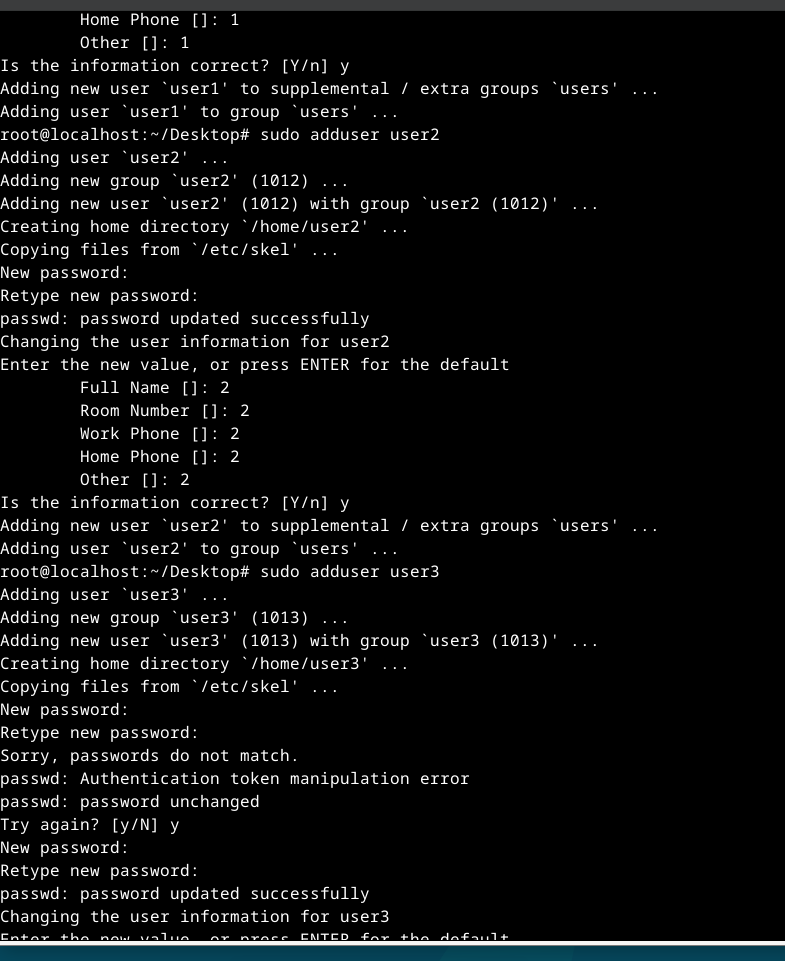
\*create an empty file called emptyfile using the touch emptyfile command. Now "reset" the permissions for the file with chmod 000 emptyfile. What happens if you change the permissions on emptyfile by passing only one value to chmod in numeric mode, for example chmod 4 emptyfile? What if we use two numbers, for example chmod 44 emptyfile? What can we learn about how chmod reads a numeric value?

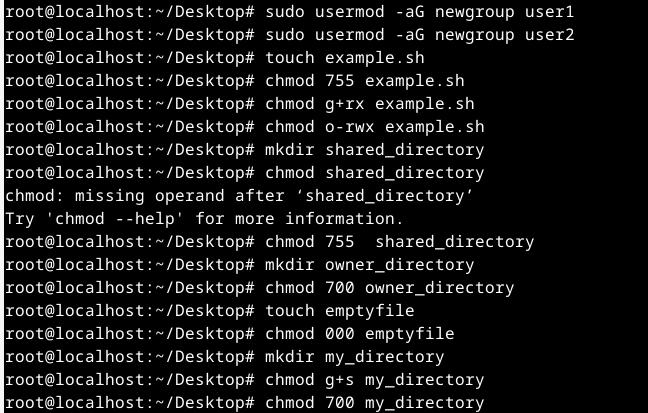
\*\*create a directory called where all files will automatically belong to your user group and can only be deleted by the user who created them?

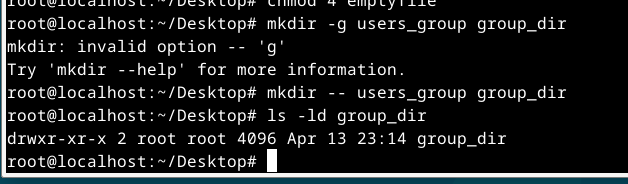
\*\*under each user, create one new file and a hard and symbolic link to it;

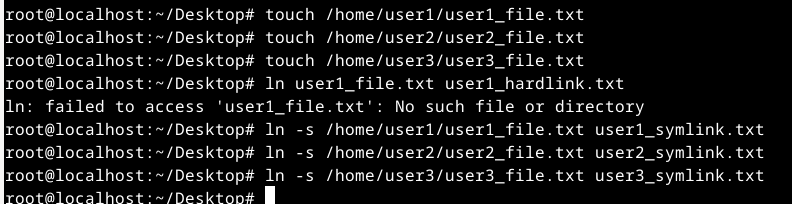
\*\*try other users to view these files;

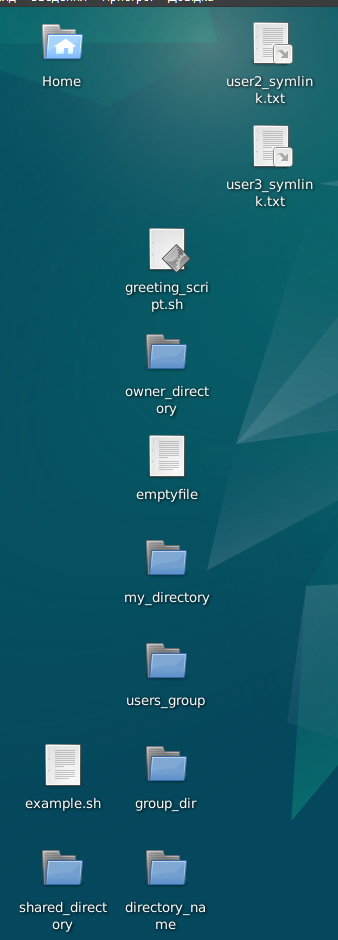
\*\*try other users to delete these files, draw conclusions.











**Answers to control questions:**

**1. Give examples of changing access rights by symbolic method (Symbolic** Method)?  
Changing access rights using the symbolic method (Symbolic Method) is used to modify access rights to files and directories using symbols representing actions (add, delete, install) and access rights (read, write, execute). Here are some examples:

**Adding access rights:**

Adding read access by all users to the file: example.txt   
**chmod +r example.txt**

**Remove access rights:**

Removing write access for a group in the example.txt file:  
**chmod g-w example.txt**  
  
**Setting specific access rights:**

Setting execute permission for owner, read and write permission for group and others for the script.sh file:  
**chmod u+x,g+rw,o+rw script.sh**  
**Changing access rights for a directory and applying recursively:**

Adding write access to the docs/ directory and all its subdirectories and files:  
**chmod -R +w docs/**

**2. Give examples of changing access rights by numerical method (numeric method, octal method)?**chmod 644 example.txt  
chmod 755 docs/  
chmod 755 script.sh  
chmod -R 755 docs/

**3. What is the purpose of the umask command?**The `umask` command is used to set or display the default access mask value for created files and directories. This mask specifies access rights that will be automatically subtracted from full access rights when creating new files and directories.

For example, if the current value of the umask is `0022`, this means that any full set of permissions (for example, `777` for a directory or `666` for a file) will be stripped of "everyone" write permissions, i.e., the created files will have permissions `755` for directories and `644` for files.

The main purpose of the `umask` command is to provide a certain level of default security for new files and directories, reducing the chances of careless granting of broad permissions.

**4. Compare hard and symbolic links?**The main difference is that hard links point to the same file system node and represent the actual file data, while symbolic links point to a file name and can be created for files in different locations on the system.

**5. \*Is it possible to execute a file that has execute permissions but no read permissions set (--x)? Explain.**Yes, a file that is set to execute but not read (--x) can be executed on Unix-like systems.

Read permissions are responsible for being able to view the contents of the file. Execute rights allow you to run an executable file as a program. If you have execute permission on a file, you can execute it even if you don't have read permission, because you can use the file name to run the program, but you won't be able to view its contents.

So, if a file has --x permissions, you can execute it, but you won't be able to read its contents. This right is useful for running programs that are available to you but should not be visible to your user.

**6. \*If we change access rights and permissions in the current session, will they be saved in the next session?.**No, changes to access rights and permissions made in the current session will not be saved in subsequent sessions. Ending the current session or restarting the system will undo all temporary changes and the files or directories will revert to their original access rights and permissions.

To make changes permanent, you need to use commands such as chmod or chown with options that change file or directory permissions. Such changes will be made directly to the file system and will be applied in subsequent sessions.

**7. \*Is there some pattern that the system uses for rights and accesses when creating new files. How can I change the default permissions?**Many Unix-like operating systems have the concept of a default permission mask, which defines the access rights that will be applied to new files and directories when they are created. This mask is known as umask (user file-creation mask).

Usually umask is set in system configuration files (eg /etc/profile, /etc/bashrc) and can also be configured per user in their ~/.bashrc file or `~ ~/.bash\_profile.  
  
The umask value specifies which permissions will not be granted by default when creating a new file or directory. It is usually expressed as an octal number that represents the combination of access rights that will not be granted. For example, 022 means that group and others will have the write right removed from the write rights.

**8. \*How can I create a hard link? In what situations is it advisable to use them?**Hard links can be created using the «ln» command in the terminal.  
Hard links are a useful tool in various situations:  
-working with backups  
-working with shared data  
-ensuring access to files using different names  
-working with applications that expect a file with a specific name

**9. \*How can I create a symbolic link? In what situations is it advisable to use them?**A symbolic link can be created using the ln -s command in the terminal. This is a short path that points to a file or directory on the file system. This is useful for creating alternative file paths, for working with large directory trees, or for convenient access to files from different locations on the system.

**10. \*\*Imagine that an application needs to create a one-time temporary file that will never be needed again after the application is closed. What is the correct directory to create this file?**The correct directory to create a one-time temporary file that will not be needed after the program is closed is the directory designated for temporary system files. On many Unix-like systems, this is usually the /tmp or /var/tmp directory. Using these directories allows you to automatically clean up temporary files after the application closes or the system restarts, ensuring efficient use of disk space.

**11. \*\*There is an original file and two links have been created for it - symbolic and hard. What will happen to other files if you delete:  
-original file;  
-symbolic link;  
-hard link  
  
Symbolic link:** the symbolic link will remain, but it will point to a file that does not exist. Attempting to access a symbolic link will result in an error.

**Hard link:** a hard link will also remain and will continue to point to the same file, even after the original file is deleted. This way, the data pointed to by the hard link will remain available. However, the file will be considered deleted and will not actually exist.

**When deleting a symbolic or hard link:**-nothing will happen to the original file, it will remain intact.-if you delete a symbolic link, it will simply be removed from the file system. The original file and hard link will remain intact and independent. **-**if you delete a hard link, only the link itself will be deleted, leaving the original file and symbolic link intact.  
 **Conclusions based on the results of the work:**During the laboratory work, the process of changing owners and access rights to files in the Linux operating system was investigated. Learned basic commands such as chown and chmod that allow you to change file owners and their permissions accordingly. Special file permissions, such as setuid, setgid, and sticky bit, were also considered, which allow certain functionality to be implemented, such as running a program with owner or group rights, automatically assigning access rights for new files in directories, and more.

In addition, the specifics of working with special directories and files in Linux, such as /tmp and /var/tmp, which are designed to temporarily store data and automatically delete them after the program is terminated or the system is restarted, were considered.

Therefore, the laboratory work made it possible to gain an understanding of the process of managing owners and access rights to files in Linux, as well as to get acquainted with the features of working with special directories and files, which will help in further tasks related to system administration.