**"КИЇВСЬКИЙ ФАХОВИЙ КОЛЕДЖ ЗВ’ЯЗКУ"**

**Лабораторна робота 10**

з дисципліни «Операційні системи»

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

файли в Linux”

Виконали: студенти **3** курсу,

групи **КСМ-13А**

**Засенко Олександр**

(прізвище та ініціали)

**Дзюбенко Дмитро**

(прізвище та ініціали)

**Сторожук Костянтин**

(прізвище та ініціали)

Київ  2023

**ЗМІСТ**

Мета роботи……………………………………………………………...………...3

Хід роботи…………………………………………………………………………6

Контрольні запитання……………………………………………………………15

Conclusions………………………………………………………………………..17

**Мета роботи:**

1. Отримання практичних навиків роботи з командною оболонкою Bash.

2. Знайомство з базовими діями при зміні власників файлів.

3. Знайомство з базовими діями при зміні прав доступу до файлів

4. Знайомство з спеціальними каталогами та файлами в Linux.

**Матеріальне забезпечення занять**

1. ЕОМ типу IBM PC.

2. ОС сімейства Windows (Windows 7).

3. Віртуальна машина – Virtual Box (Oracle).

4. Операційна система GNU/Linux – CentOS.

5. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux

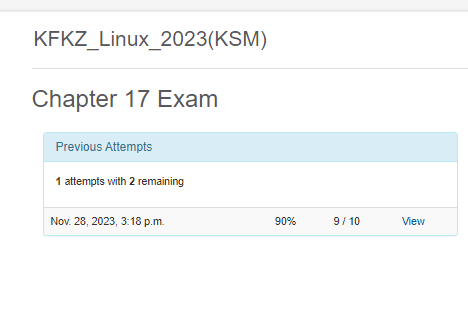
**Завдання для попередньої підготовки**

***Готував матеріал студент Zasenko***

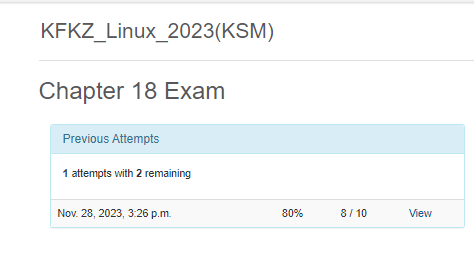
Невеликий словник базових англійських термінів з питань призначення команд та їх параметрів.

|  |  |
| --- | --- |
| File Ownership | By default, users own the files that they create. While this ownership can be changed, this function requires  administrative privileges. Although most commands usually show the user owner as a name, the operating system is  associating the user ownership with the UID for that username. |
| Changing Groups | If you know that the file you are about to create should belong to a group different from your current primary  group, then you can use the newgrp command to change your current primary group. |
| Setuid | When the setuid permission is set on an executable binary file (a program) the binary file is run as the owner  of the file, not as the user who executed it. This permission is set on a handful of system utilities so that they can be  run by normal users, but executed with the permissions of root, providing access to system files that the normal user  doesn&#39;t normally have access to. |
| Setgid | The setgid permission is similar to setuid, but it makes use of the group owner permissions. There are two  forms of setgid permissions: setgid on a file and setgid on a directory. The behavior of setgid depends on whether it is  set on a file or directory. |

**Пройдіть тестування у курсі NDG Linux Essentials за такими темами:**



4.1 Purpose of the id command:



The id command displays information about the user's user ID (UID), user group (GID), and additional user groups.

4.2 How to view the access rights of the file owner:

Use the ls -l command and pay attention to the first three characters of the output, which indicate the file owner's permissions.

4.3. How to change the owner of a group:

Use the chown command with the -R option to change the group owner for all files in a directory. Example:

chown -R new\_owner:new\_group /path\_to\_directory\_or\_file

4.4. How to view the type of the current file:

Use the file command. For example:

file text\_file.txt

file program\_file

file directory

4.5. Setuid and Setgid permissions:

Setuid (SUID): Allows the program to run as the owner of the file, regardless of who started the program.

Setgid (SGID): Applies to directories and allows new files in that directory to inherit the group from the parent directory.

4.6. "Sticky Bit:

Sticky Bit is applied to directories and makes it impossible for users other than the file owner or administrator to delete or rename files. It is typically used in directories such as /tmp where multiple users can create files, and the goal is to prevent other users from accidentally deleting or modifying other users' files.

**Хід роботи**

***Готував матеріал студент Dziubenko***

1. Початкова робота в CLI-режимі в Linux ОС сімейства Linux:

1.1. 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.

1.2. Запустіть віртуальну машину Ubuntu\_PC (якщо виконуєте завдання ЛР через академію netacad)

1.3. Запустіть свою операційну систему сімейства Linux (якщо працюєте на власному ПК та її встановили) та запустіть термінал.

1. Опрацюйте всі приклади команд, що представлені у лабораторних роботах курсу NDG Linux

Essentials:

- Lab 17: Ownership and Permissions

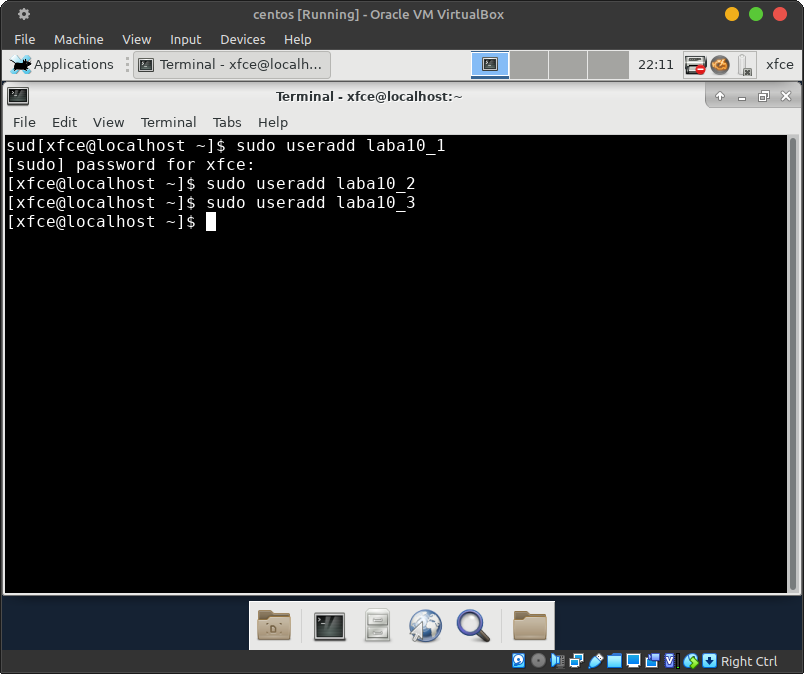
- Lab 18: Special Directories and Files

1. Створіть таблицю команд вивчених у п.2 ходу роботи у наступному вигляді:

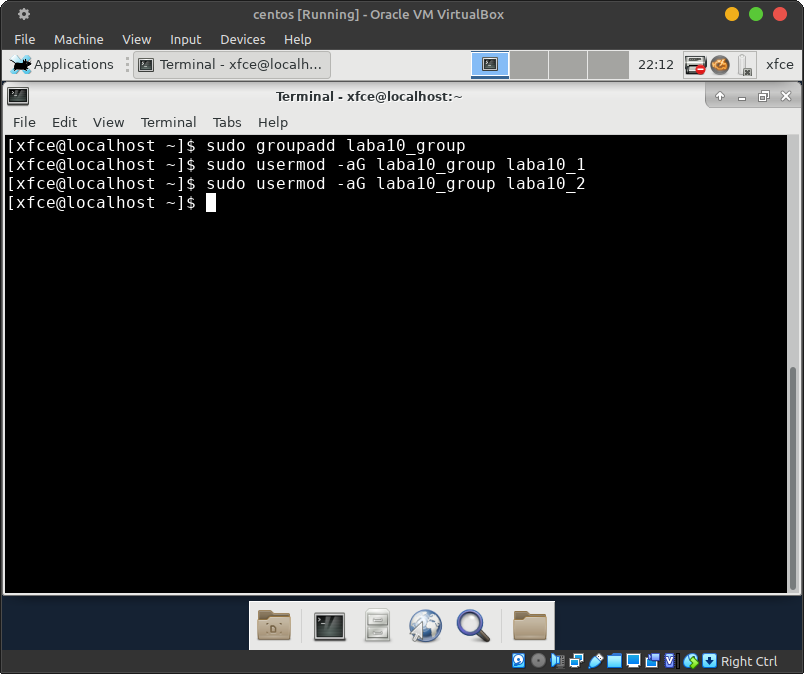
|  |  |
| --- | --- |
| Назва команди | Її призначення та функціональність |
| cd | Перехід до директорії |
| mkdir | Створення каталогу |
| touch | Створення пустого текстового документа |
| ls | Вивід на екран вміст каталогу |
| chmod | Додавання повноважень файлу |
| echo | Вивід на екран повідомлення користувачу |
| ./test.sh | Запуск виповнюваного скрипта |
| stat | Вивід статистики файлу |
| chown | Зміна користувача або групи на супер-користувача(чів) |
| chgrp | Зміна групи користувачів всіх файлів на директорію структури |
| ln | Створення посилання на файл або директорію |
| rm | Видалення файлу або каталогу |

1. Виконайте наступні практичні завдання у терміналі наступні дії (продемонструвати скріншоти):

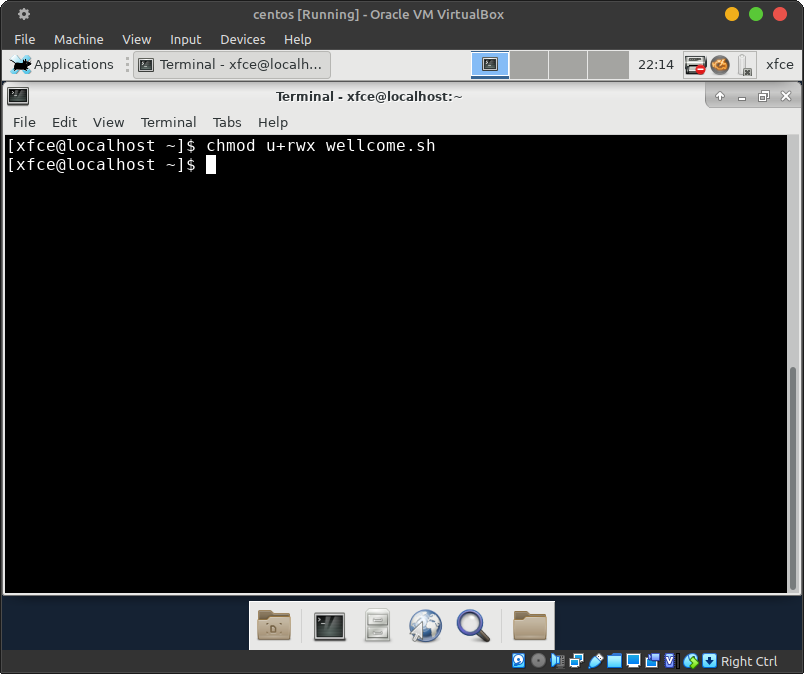
* створіть трьох нових користувачів;



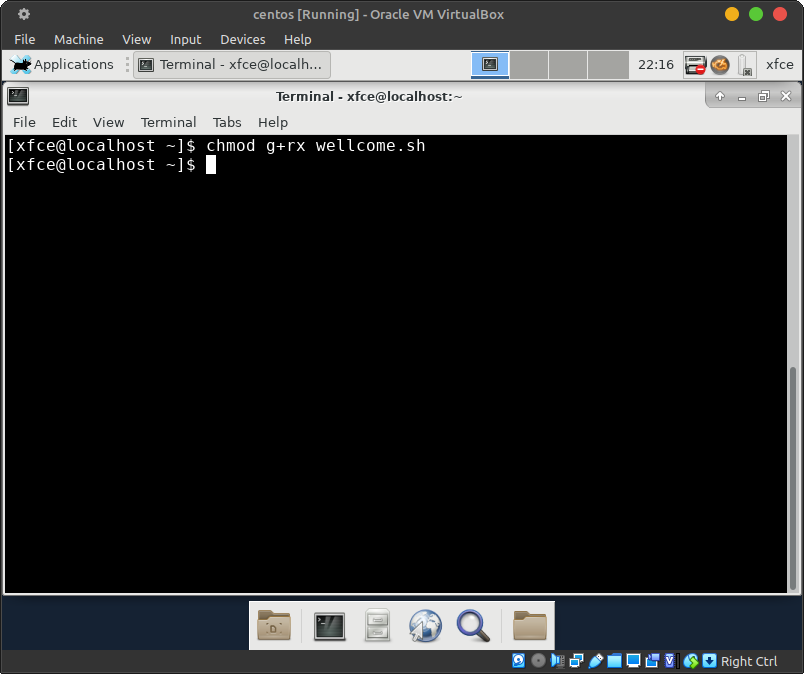
* створіть нову групу користувачів, туди додайте двох, з трьох створених користувачів;



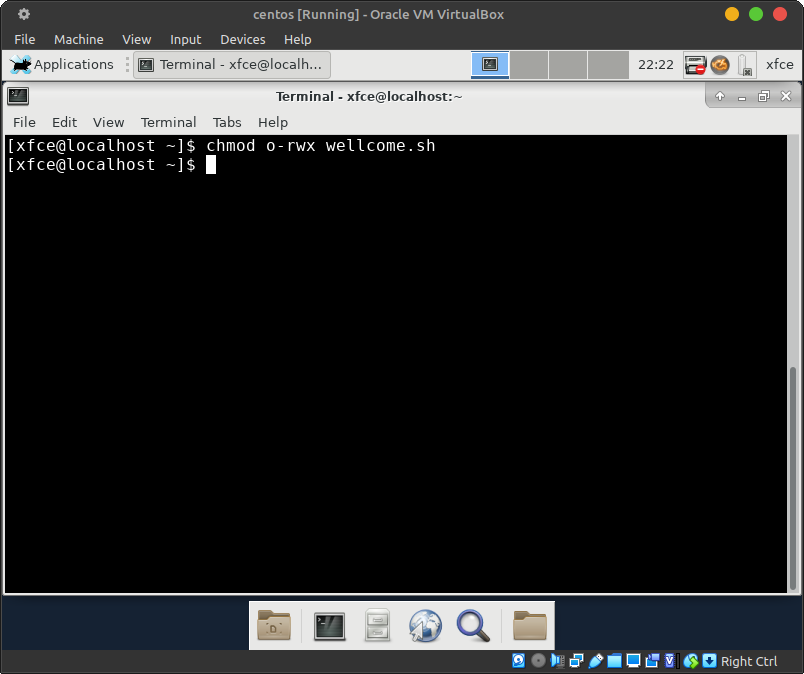
* створіть новий файл, який буде доступний на зчитування, редагування та виконання власником файлу, наприклад найпростіший скриптовий сценарій;



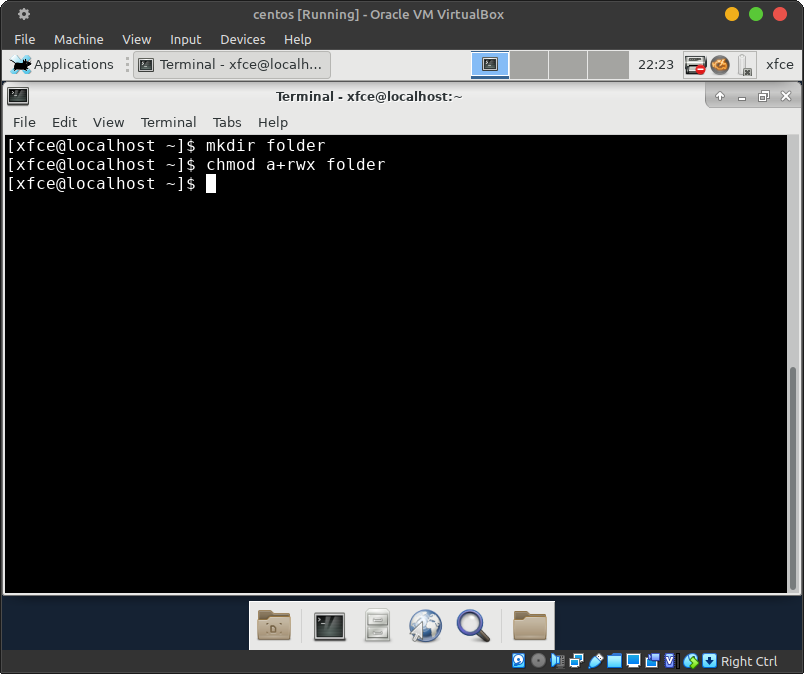
* для користувачів групи власника надайте дозволи на перегляд та виконання (без дозволу на редагування) цього файлу;

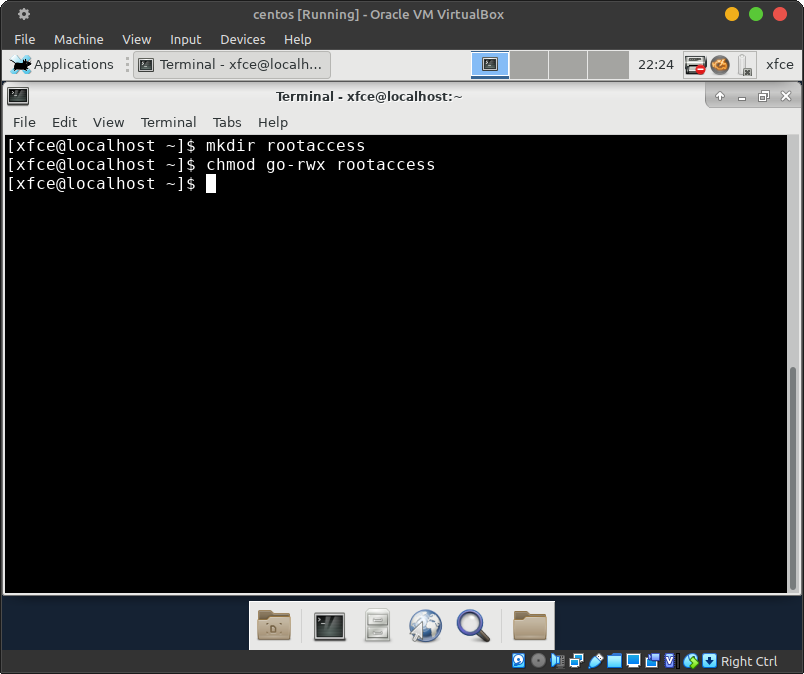


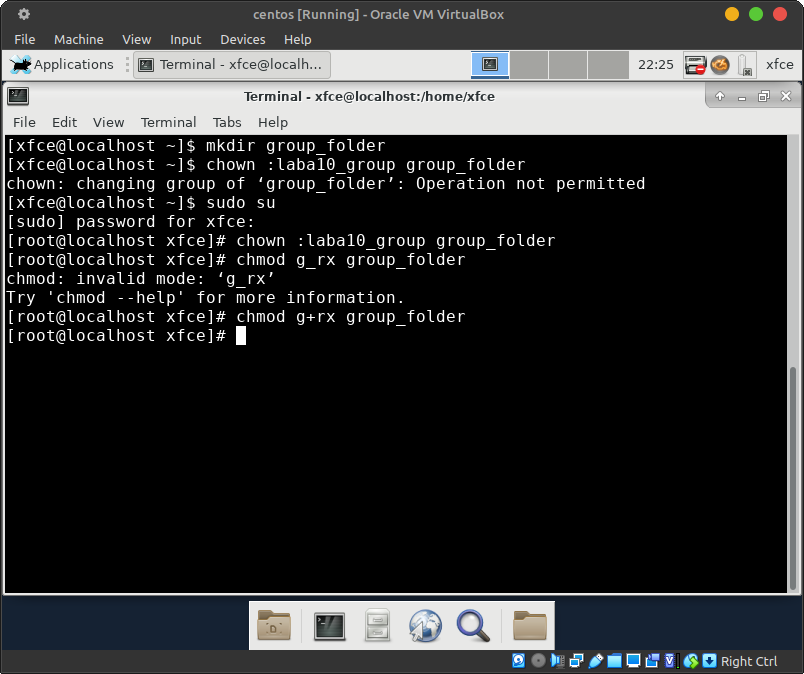
* для інших користувачів заборонити доступ до цього файлу;



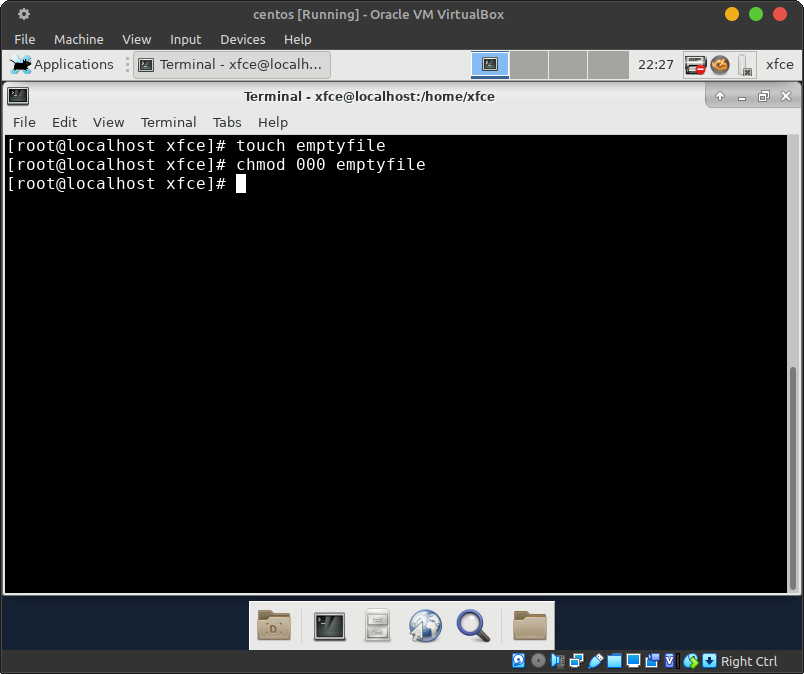
* подібні дії виконайте для директорій - створіть директорію, яка буде доступна для всіх трьох користувачів, створіть директорію, яку буде доступна тільки для власника, створіть директорію, яку користувачі групи власника зможуть переглядати, але не редагувати;

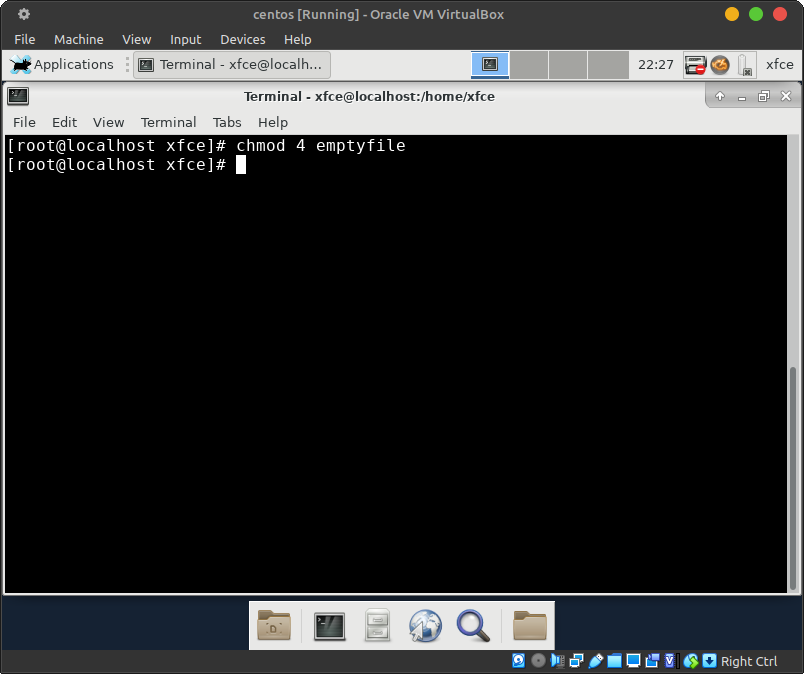




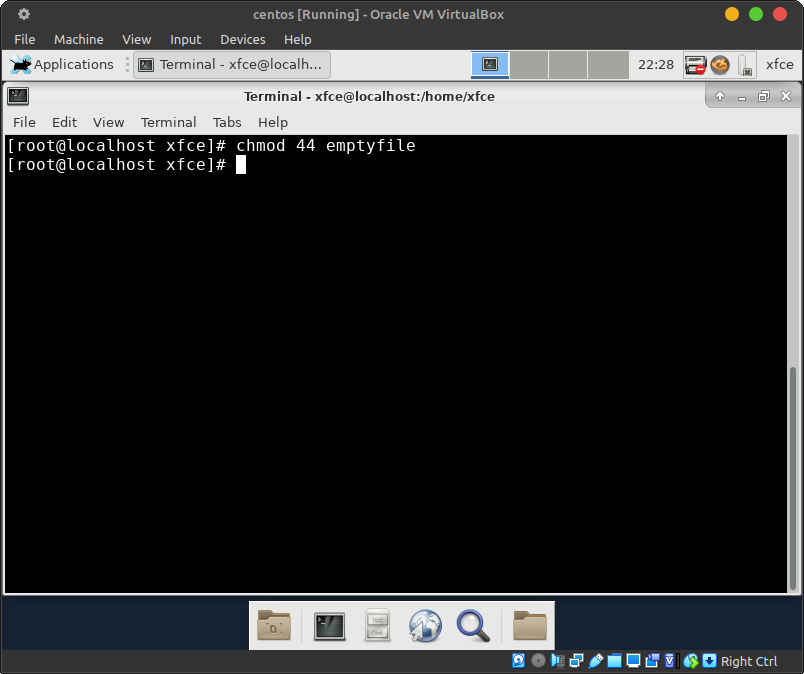


* створіть порожній файл під назвою emptyfile за допомогою команди touch emptyfile. Тепер “обнуліть” дозволи для файлу з chmod 000 emptyfile. Що станеться, якщо змінити дозволи для emptyfile, передавши лише одне значення для chmod у числовому режимі, наприклад, chmod 4 emptyfile? Що буде, якщо ми використаємо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



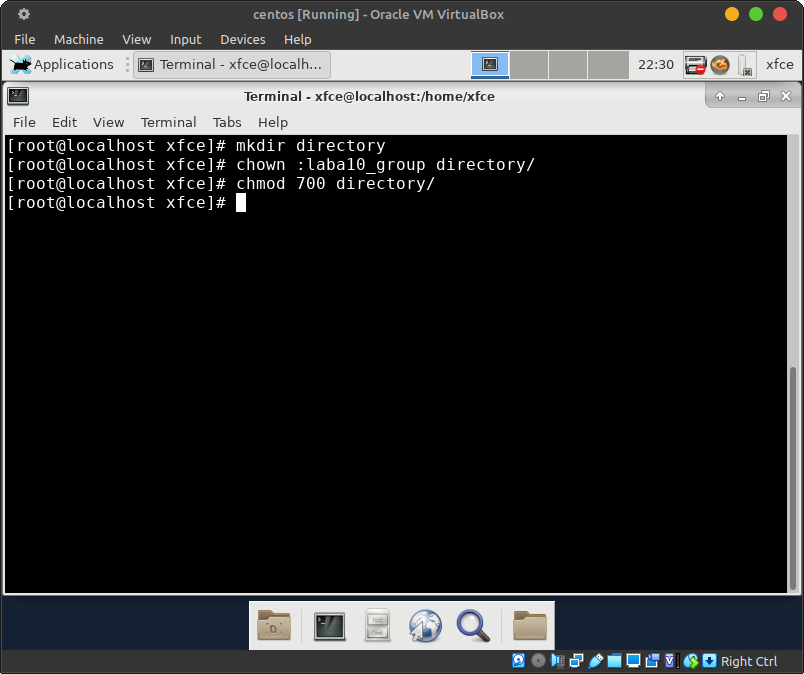


This will give the file owner read access, and other permissions will be disabled for all users.

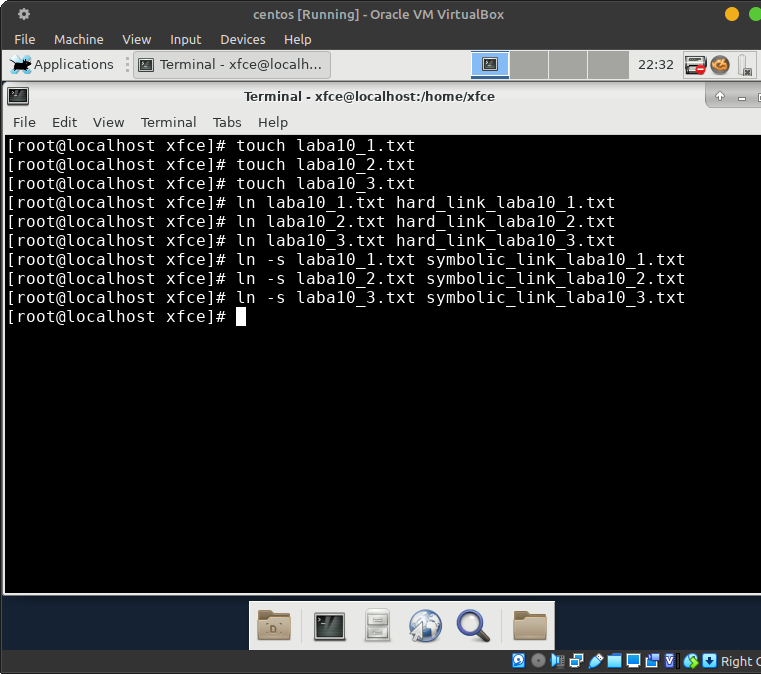


It will also grant read permissions to both the owner and the group. But these permissions for other users will remain zero.

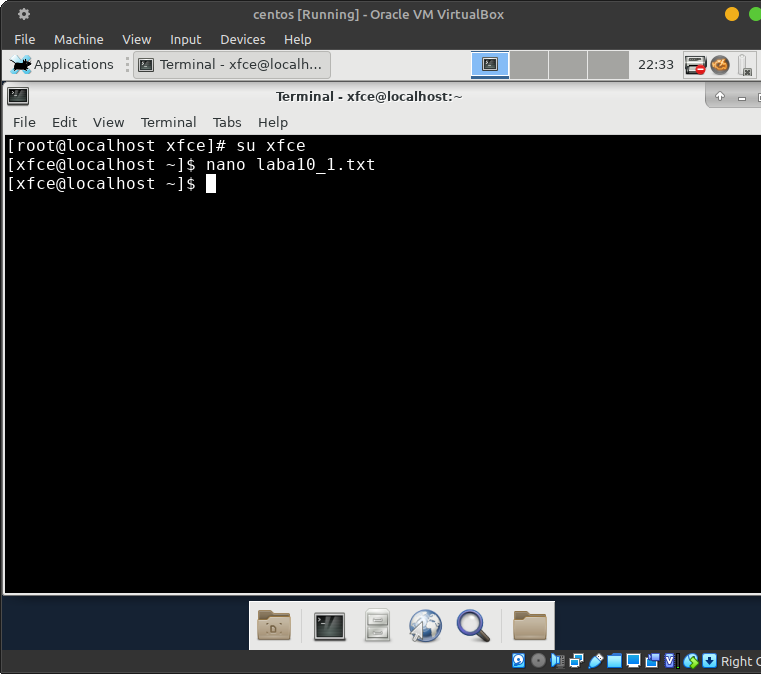
* створіть каталог під назвою, де всі файли автоматично будуть належати Вашій групі користувачів і можуть бути видалені лише користувачем, який їх створив?



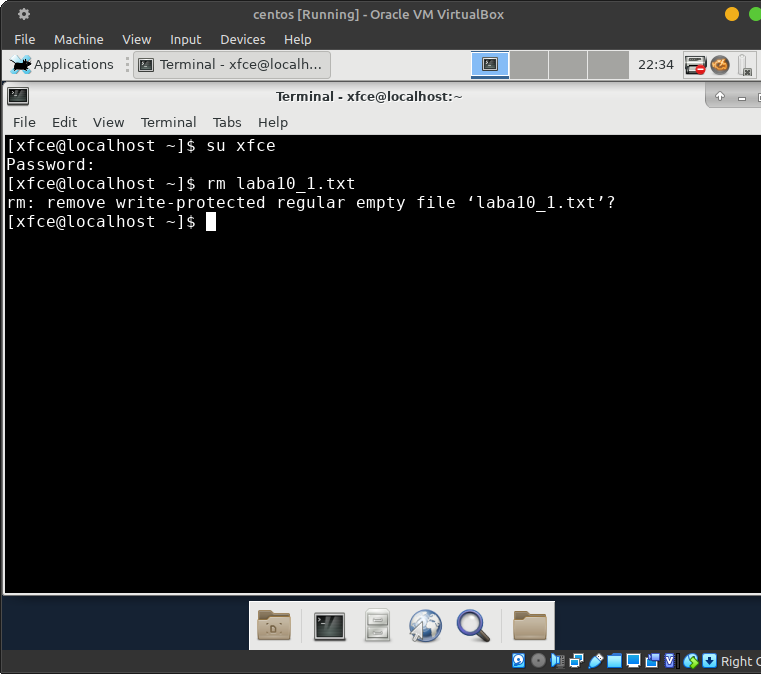
* під кожним користувачем створіть по одному новому файлу, та жорстке та символічне посилання на нього;



* спробуйте іншими користувачами переглянути ці файли;



* спробуйте іншими користувачами видалити ці файли, зробіть висновки.



In most cases, users can delete their own files and links, but cannot delete files of other users who have adequate access rights set up.

**Контрольні запитання**

***The material was prepared by a student Storozhuk***

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

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

3. No, a file with execute privileges but no read privileges cannot be executed. To execute a file, the system must read its contents in order to execute its code. If read permissions are not set, the system cannot read the contents of the file and execution is not possible.

4. The umask command allows you to set a permission mask for files created by the user. The access rights mask determines what access rights will be set for the file by default.

5. Yes, the access rights and permissions that you change in the current session will be saved in the next session. This is because access rights and permissions are stored in the file system. When you change the access rights or permissions for a file or directory, you are actually changing the value stored on the file system.

The only exception to this is access rights that you set with an access mask. A permission mask applies only to files that are created in current sessions. The access rights you set with the access rights mask will not be applied to files that are created in the next session.

6. For example, Unix-like operating systems, such as Linux, use a pattern to determine access rights to new files. This pattern is called an access rights mask (umask). An access rights mask is specified as three octal digits, each of which corresponds to one of the access rights bits:

* Read (r): Allows the user to read the contents of the file.
* Write (w): Allows the user to write the contents of the file.
* Execute (x): Allows the user to run the file for execution.

For example, a mask of 022 means that by default, files created by the user will have access rights of 666. This means that the user who created the file will have read and write permissions to the file, while other users will have read-only permissions.

7. To create a one-time temporary file that will never be needed again after the program is closed, use the /tmp directory. The /tmp directory is for temporary files created by programs.

8. A hard link is a way to refer to a file or directory without making a copy. A hard link is actually a reference to the index number of the file or directory.

9. A symbolic link is a way to refer to a file or directory by creating a copy of the file or directory path. A symbolic link is actually a reference to a path.

10. Compare hard links and symbolic links?

Common features:

* Both types of links are a way to refer to a file or directory without making a copy.
* Both types of links can be used to share a file or directory with other users.
* Both types of links can be used to create an access point to a file or directory elsewhere on the file system.

Differences:

* Hard links refer to the index number (inode) of a file or directory. This means that hard links are actually the same file or directory. Any changes made to the source file or directory will be reflected in both locations.
* Symbolic links refer to the path to a file or directory. This means that symbolic links are separate files or directories. Any changes made to the source file or directory will not be reflected in the link.

**Conclusions**

***The material was prepared by a student Storozhuk***

In this article, we've gone through the basic steps for changing file owners, file access rights, and learned about special directories and files in Linux.