



Тема: «Основи керування файлами у файловій системі *Unix*-подібних ОС»

Викладач: Олександр А. Блажко,

доцент кафедри ІС Одеської політехніки, blazhko@ieee.org

Мета роботи: придбання навичок керування правами доступу до файлової системи *Ext* в ОС *Linux*.

План

1 Теоретичні відомості	1
2 Завдання до виконання	27

1 Теоретичні відомості

1.1 Сучасна термінальна взаємодія у комп'ютерних мережах

1.1.1 Історія термінальних пристроїв

Сьогодні у всіх студентів є електронний комп'ютер, оформлений як великий настільний комп'ютер, середнього розміру ноутбук, розміром з книгу планшетний комп'ютер або смартфон. Ми господарі цих комп'ютерів і використовуємо їх одноосібно.

Але чи завжди ми їх використовуємо на 100% їх можливостей? У період часу, коли ми не використовуємо комп'ютер, чи можемо ми його дати в оренду іншим людям? Але не віддати його в тимчасове користування фізично, щоб після завершення терміну оренди нервувати, намагаючись знайти нечесного орендаря, який не бажає повертати комп'ютер. Чи можна віддати в оренду можливості комп'ютера, орендувати самі послуги, що надаються комп'ютером, дистанційно орендувавши сам комп'ютер? Споживачі комп'ютера про це стали замислюватися ще понад 60 років тому, коли для доступу до одного дорогого комп'ютера стояла черга з тисяч споживачів.

Ще в другій половині 19 століття з'явився Телетайп (англ. *Teletype*, скорочено - *TTY*) - електромеханічна друкарська машина, яка використовувалася для передачі між двома абонентами текстових повідомлень через найпростіший електричний канал (зазвичай по парі проводів). Тому для вирішення завдання оренди комп'ютера були придумані термінальні пристрої, що дозволяють багатьом користувачам одночасно працювати з одним комп'ютером, який називається «хост» (*host* - господар) та керує *INPUT/OUTPUT*-потокami даних цих терміналів.

На рисунку 1 показано приклади телетайпів які виконували роль термінальних пристроїв для віддаленого доступу до великих комп'ютерів.



(a) Термінал *IBM 2741* як приклад друкарського телетайпу



(b) 15-тирічний Біл Гейтс, працюючи за терміналом, «уважно слухає», можливо, коментарі свого старшого товариша Полана Алена ☺



(c) Дисплей терміналу *IBM 2260* як приклад «скляного» телетайпу



(d) Приклад терміналу для віддаленого доступу до хост-комп'ютера

Рис. 1 – Приклади телетайпів які виконували роль термінальних пристроїв для віддаленого доступу до великих комп'ютерів.

Термінальний пристрій складався з мінімум трьох компонент: пристрій введення, наприклад, клавіатура, пристрій виведення, наприклад, дисплей, і комунікаційної пристрій для зв'язку через комп'ютерну мережу з комп'ютером-хостом.

Багатокористувацький термінальний пристрій дозволяє організувати на базі одного комп'ютера кілька незалежних місць – терміналів з можливістю одночасної роботи користувачів.

Спершу у кожного типу комп'ютера-хоста були свої типи термінальних пристроїв – «дурні» термінали (англ. *Dumb terminal*). Пізніше вони були замінені на інтелектуальні термінали (англ. *Intelligent terminal*), які можна було програмувати за допомогою завантаження команд з перфострічки емуляторів терміналів, налаштовуючи їх на роботу із заданим комп'ютером-хостом.

1.1.2 Термінальні пристрої *Unix*-подібних ОС

З появою ОС *Unix* багато програм, що використовують текстовий інтерфейс командного рядка, були спочатку розроблені для роботи лише через термінал. У більшості сучасних комп'ютерів, які не використовують спеціалізовані термінали, робота терміналу емулюється засобами ОС для сумісності зі старими програмами.

Для емуляції використовується псевдотермінал (англ. *Pseudo terminal*, або *PTY*) у вигляді пари псевдопристроїв, один з яких, другорядний, емулює справжній текстовий термінал, а другий, головний, надає засоби, за допомогою яких емулятор терміналу контролює процеси.

На рисунку 2 наведено приклад фрагменту екрану з декількома одночасно запущеними в ОС *Windows 10* псевдотерміналами. Але ці термінали пасивні і не можуть обмінюватися повідомленнями один з одним, тому що ОС більш орієнтована на роботу одного користувача, а не декількох, хоча ця ОС дозволяє перемикатися між роботою декількох користувачів, які увійшли до неї.

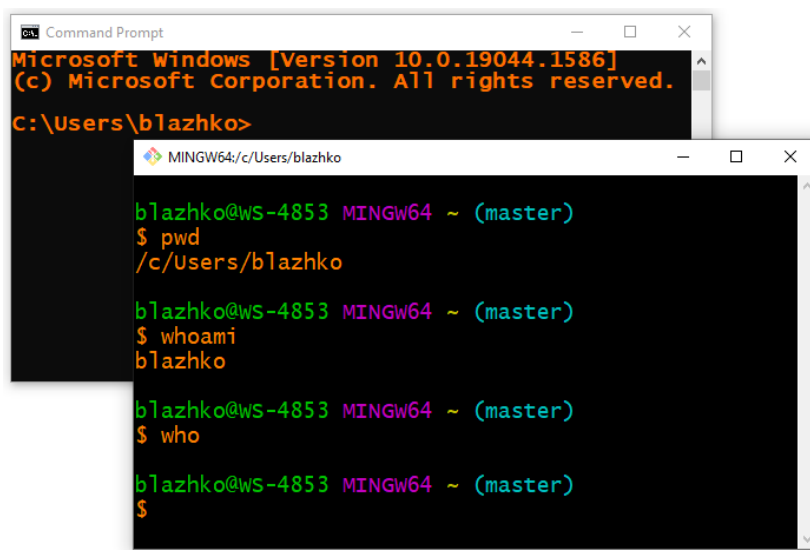


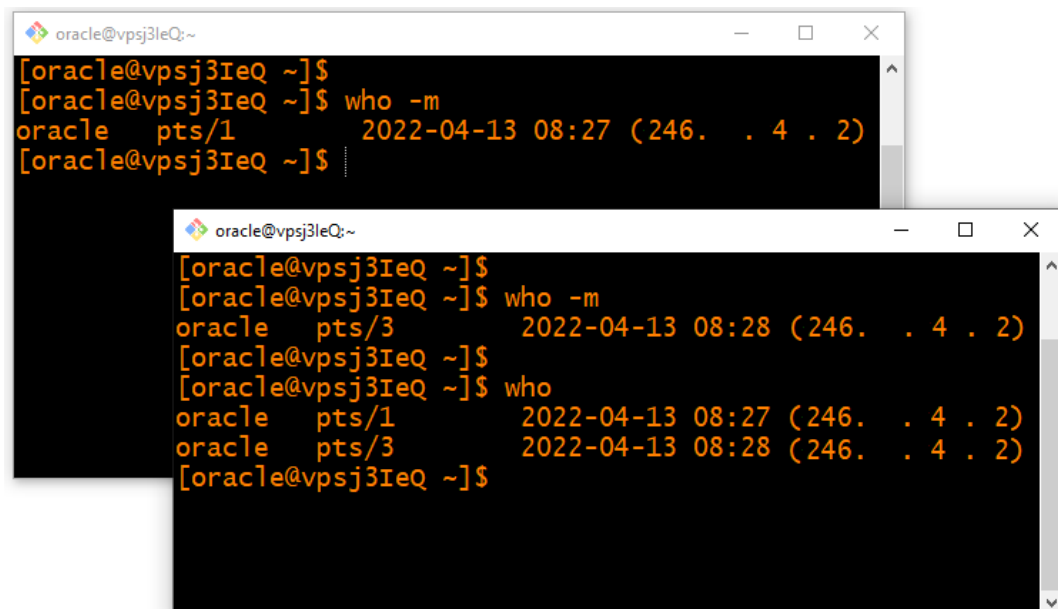
Рис 2 – Приклад фрагменту екрану з декількома одночасно запущеними в ОС *Windows 10* псевдотерміналами

Але справжні багатокористувальницькі ОС, наприклад, будь-які *Unix*-подібні ОС, пропонують користувачам псевдотермінали, які можуть взаємодіяти один з одним.

На рисунку 2 наведено приклад фрагменту екранів роботи двох псевдотерміналів, які були одночасно запущені в *Unix*-подібній ОС. Представлено роботу команди *who*, яка показує перелік псевдотерміналів, з якими взаємодіють користувачі.

Наприклад, команда *who* з опцією *-m* виводить наступні дані:

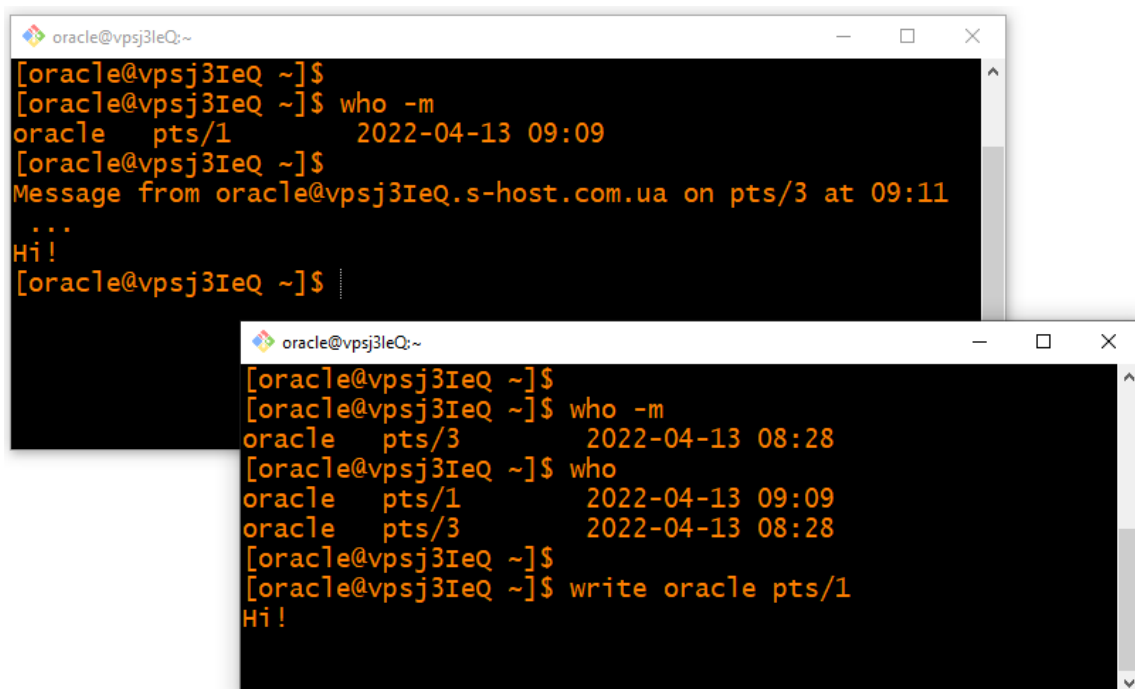
ім'я користувача (*oracle*), унікальна назва псевдотерміналу (*PTS/1*), дата та час початку роботи користувача з псевдотерміналом, а також *IP*-адреса комп'ютера, з якої користувач встановив з'єднання із сервером. Виконання команди *who* без опцій виводить на екран дані про всі псевдотермінали, які запущено на хост-комп'ютері.



```
oracle@vpsj3IeQ:~  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$ who -m  
oracle pts/1 2022-04-13 08:27 (246. . 4 . 2)  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$ who -m  
oracle pts/3 2022-04-13 08:28 (246. . 4 . 2)  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$ who  
oracle pts/1 2022-04-13 08:27 (246. . 4 . 2)  
oracle pts/3 2022-04-13 08:28 (246. . 4 . 2)  
[oracle@vpsj3IeQ ~]$
```

Рис 2 – Приклад фрагменту екрану з декількома одночасно запущеними в *Unix*-подібній ОС псевдотерміналами

Для показу простої взаємодії між різними псевдотерміналами можна використати команду *write*, яка надсилає повідомлення від одного терміналу до іншого, як це показано на рисунку 3.



```
oracle@vpsj3IeQ:~  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$ who -m  
oracle pts/1 2022-04-13 09:09  
[oracle@vpsj3IeQ ~]$  
Message from oracle@vpsj3IeQ.s-host.com.ua on pts/3 at 09:11  
...  
Hi!  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$ who -m  
oracle pts/3 2022-04-13 08:28  
[oracle@vpsj3IeQ ~]$ who  
oracle pts/1 2022-04-13 09:09  
oracle pts/3 2022-04-13 08:28  
[oracle@vpsj3IeQ ~]$  
[oracle@vpsj3IeQ ~]$ write oracle pts/1  
Hi!
```

Рис 3 – Приклад фрагменту екрану з обміном повідомлень між псевдотерміналами

Для виконання команди *write* необхідно вказати ім'я користувача та назву його псевдотерміналу. Як видно з рисунку 3, користувач *oracle*, знаходячись у псевдотерміналі *PTS/3*, надіслав повідомлення «Hi!» користувачу *oracle*, тобто самому собі, але на інший псевдотермінал *PTS/1*. Для виходу з програми достатньо вказати комбінацію *Ctrl+C*

Для керування доступом будь-яких програм, які дозволяють передавати повідомлення на псевдотермінал, використовується команда *msg [y | n]*, де *y* – дозвіл на відправку повідомлень, *n* – заборона, наприклад, *msg n*

1.1.3 Налаштування віддаленого доступу до терміналу ОС *Linux*

Багато років розробники і споживачі термінальних систем не турбувалися про безпеку даних і процесів. Тільки для зручності адміністрування користувачі повинні були проходити процеси ідентифікації, вводячи свої унікальні ідентифікатори (*login name*), підтверджуючи їх процесом аутентифікації, наприклад, через надавання секретної фрази – пароллю. Для забезпечення віддаленого доступу багато років використовувався мережевий протокол *Telnet* (англ. *Teletype Network*) як алгоритм взаємодії між терміналами і комп'ютером-хостом. У протоколі не передбачено використання ні шифрування, ні перевірки достовірності даних, і при бажанні стороння людина, зловмисник, може легко перехопити конфіденційні дані, основним з яких є пароль користувача.

Коли споживачів стали хвилювати питаннями мережевої безпеки, *telnet* був замінений на безпечні протоколи, наприклад, на мережевий протокол *SSH* (*Secure SHell* – «безпечна оболонка»).

Для встановлення з терміналу комп'ютера-клієнта віддаленого доступу до комп'ютера-сервера (хост-комп'ютер, *host*) необхідне мережеве з'єднання між комп'ютерами.

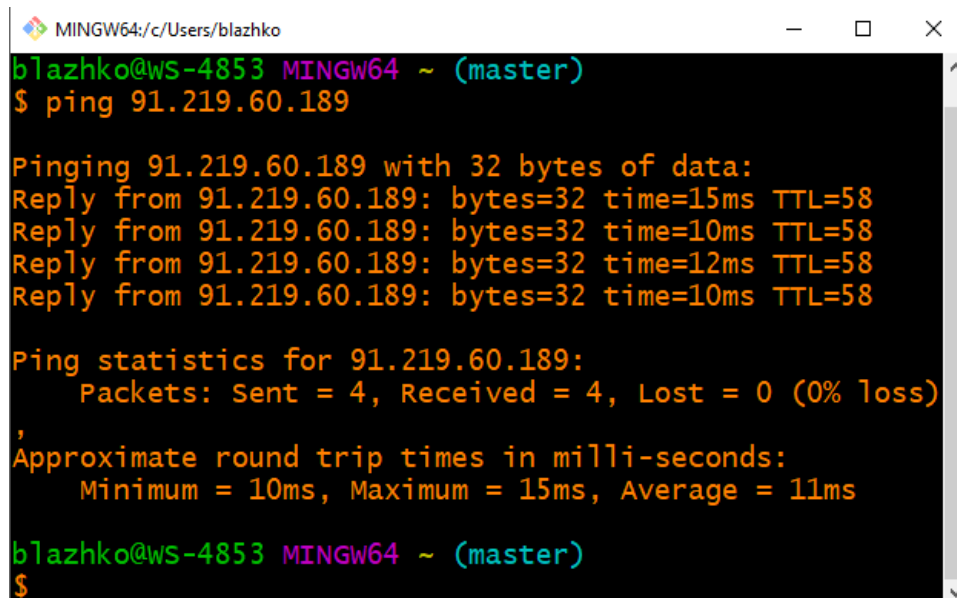
Для перевірки доступності віддаленого комп'ютера в мережі часто використовується команда *PING*:

```
ping мережева_адреса
```

Команда *ping* відправляє запити (англ. *Echo-Request*) до комп'ютера з вказаною *IP*-адресою у мережі і фіксує відповіді (англ. *Echo-Reply*). Час між відправленням запиту й одержанням відповіді дозволяє визначати двосторонні затримки у маршруті та частоту втрати пакетів, тобто визначати завантаженість каналів передачі даних і проміжних пристроїв. Для припинення роботи команди можна натиснути традиційну комбінацію клавіш *Ctrl+C*

На рисунку 4 наведено приклад команди *ping*, яка перевіряє мережеву доступність комп'ютера з *IP*-адресою = 91.219.60.189. Команда відправила чотири мережеві пакети та

отримала відповіді підтвердження наявності комп'ютера у мережі з середньою затримкою 11 мілісекунд.



```
blazhko@ws-4853 MINGW64 ~ (master)
$ ping 91.219.60.189

Pinging 91.219.60.189 with 32 bytes of data:
Reply from 91.219.60.189: bytes=32 time=15ms TTL=58
Reply from 91.219.60.189: bytes=32 time=10ms TTL=58
Reply from 91.219.60.189: bytes=32 time=12ms TTL=58
Reply from 91.219.60.189: bytes=32 time=10ms TTL=58

Ping statistics for 91.219.60.189:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
    ,
    Approximate round trip times in milli-seconds:
        Minimum = 10ms, Maximum = 15ms, Average = 11ms

blazhko@ws-4853 MINGW64 ~ (master)
$
```

Рис 4 – Приклад команди *ping*, яка перевіряє мережеву доступність комп'ютера з IP-адресою = 91.219.60.189

В оболонці *Bash* для віддаленого доступу з одного терміналу до іншого, віддаленого терміналу використовується команда *SSH* – *Secure Shell* (англ. *Secure SHell* - «безпечна оболонка»), яка використовує *SSH*-протокол:

```
ssh ім'я_користувача@мережева_адреса_серверу
```

Наприклад, для з'єднання користувача з ім'ям *user1* до серверу з IP= 46.175.148.116, необхідно виконати команду:

```
ssh user1@ 46.175.148.116
```

Відомо, що при першому встановленні з'єднання за *SSH*-протоколом відбувається обмін відкритими ключами та перевірка їх надійності.

На рисунку 5 показано приклад такого першого встановлення з'єднання. Як видно на рисунку, програма повідомляє, що аутентичність (*authenticity*) хоста за вказаною IP-адресою не може бути встановлена. Програма-клієнт не знає, чи дійсно саме сервер зі вказаною IP-адресою, до якого вона звертається, відповідає цій програмі, а не який-небудь злоумисник. Найчастіше користувач може на свій страх та ризик вказати *yes*, вважаючи, що сервер справжній. Але для більшої гарантії рекомендується заздалегідь отримати від адміністратора серверу так званий *fingerprint*, який є хеш-значенням відкритого ключа серверу.

Також на рисунку 5 показано значення *fingerprint*, яке сервер надіслав клієнту – воно починається зі слова *SHA256* (*Secure Hash Algorithm* – алгоритм безпечного хешування даних).

```
blazhko@WS-4853 MINGW64 ~ (master)
$ ssh oracle@91.219.60.189
The authenticity of host '91.219.60.189 (91.219.60.189)' can't be established.
ED25519 key fingerprint is SHA256:Afw2LdFcWCzAwjh88ENH7dIC80FBr+tcMOW+xHoiFkk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Рис. 5 – Фрагмент екрану із прикладом першого встановлення з'єднання з віддаленим терміналом через команду *SSH*

Алгоритм *SHA256* стискає дані будь-якого розміру до фіксованого розміру 256 біт і робить це безпечно, що дозволяє:

- для будь-якого файлу створити хеш-значення та зберігати його стиснуту копію файлу;
- якщо хось захоче відробити файл, він не зможе легко його змінити, залишивши незмінним хеш-значення;
- для перевірки того, що файл не підроблено, необхідно повторно для файлу створити хеш та перевірити його з початковим значенням, створеним при створенні файлу і коли значення співпадають, тоді можна без ризику використовувати файл.

Також на рисунку 5 показано, що програма використовує алгоритм *ED25519* як новітню версію алгоритму створення та перевірки цифрового підпису *Edwards-curve Digital Signature Algorithm (EdDSA)*, який встановлюється на файли для більшого рівня підтвердження факту непідробки файлу злоумисником.

Адміністратор серверу може надати *fingerprint*, виконавши на сервері команду:

```
ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub
```

В результаті виконання команди буде показано значення *fingerprint* як хеш-значення, створене за алгоритмом безпечного хешування *SHA256* (за замовчуванням), наприклад:

```
SHA256:Afw2LdFcWCzAwjh88ENH7dIC80FBr+tcMOW+xHoiFkk
```

На рисунку 6 показано приклад виконання такої команди.

```
[oracle@vpsj3IeQ ~]$ ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:Afw2LdFcWCzAwjh88ENH7dIC80FBr+tcMOW+xHoiFkk no comment (ED25519)
```

Рис. 6 – Фрагмент екрану із прикладом команди отримання значення *fingerprint*

Необхідно скопіювати значення *fingerprint* та вказати його при першому встановленні з'єднання з сервером, як це показано на рисунку 7. Програма-клієнт збереже це значення у

файлі `./ssh/known_hosts` поточного каталогу локального користувача та завершити свою роботу за проханням серверу, як це показано на рисунку 7.

```
$ ssh oracle@91.219.60.189
The authenticity of host '91.219.60.189 (91.219.60.189)' can't be est
ablished.
ED25519 key fingerprint is SHA256:Afw2LdFcWCzAwjh88ENH7dIC80FBr+tcMOW
+xHoiFkk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
SHA256:Afw2LdFcWCzAwjh88ENH7dIC80FBr+tcMOW+xHoiFkk
Warning: Permanently added '91.219.60.189' (ED25519) to the list of k
nown hosts.
Connection closed by 91.219.60.189 port 22

blazhko@ws-4853 MINGW64 ~ (master)
```

Рис. 7 – Фрагмент екрану із прикладом першого встановлення з'єднання та визначенням значення *fingerprint*

Після повторного виконання команди `ssh` віддалена ОС вже запросить традиційний пароль, як показано на рисунку 8 для підключення до віддаленого терміналу.

```
blazhko@ws-4853 MINGW64 ~ (master)
$ ssh oracle@91.219.60.189
oracle@91.219.60.189's password:
```

Рис. 8 – Фрагмент екрану із прикладом запрошення паролю для доступу до віддаленого терміналу командою `ssh`

Використовуючи *SSH*-протокол, можна скористатися можливістю не вказувати постійно пароль при встановленні з'єднання з віддаленою ОС. Пригадаємо, що в лабораторній роботі, присвяченій безпечному доступу до *GitHub*-серверу через *SSH*-протокол необхідно було відкритий ключ користувача розмістити на сервері. Подібний процес виконується для передачі тогож відкритого ключа у спеціальний файл `./ssh/authorized_keys` у домашньому каталозі користувача на віддаленій ОС. Для цього використовується наступна команда:

```
ssh-copy-id ім`я_користувача@мережева_адреса_серверу
```

Наприклад, для передачі відкритого ключа у домашній каталог користувача з ім`ям користувача *oracle* до серверу з *IP=91.219.60.189*, необхідно виконати команду:

```
ssh-copy-id oracle@91.219.60.189
```

На рисунку 9 наведено приклад виконання вказаної команди.


```

blazhko@WS-4853 MINGW64 ~ (master)
$ ssh-copy-id oracle@91.219.60.189
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/c/Users/blazhko/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
oracle@91.219.60.189's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'oracle@91.219.60.189'"
and check to make sure that only the key(s) you wanted were added.

blazhko@WS-4853 MINGW64 ~ (master)

```

Рис. 9 – Фрагмент екрану із прикладом виконання команди *ssh-copy-id* передачі відкритого ключа у домашній каталог користувача *oracle* до серверу з *IP=91.219.60.189*

Як вказано у останньому абзаці повідомлень виконаної команди, в подальшому необхідно повторно встановити з'єднання з віддаленим терміналом *Linux*-серверу, в процесі чого ОС вже не буде пропонувати вказувати пароль користувача.

Реалізація можливості з'єднатися з віддаленим терміналом *Linux*-серверу дозволяє спростити процеси, які вимагають виконання множини операцій обміну файлами. Але слід пам'ятати, що таке спрощення повинно бути винятком із правил, а не правилом, тому що цим може скористатися зловмисник, який без вашого дозволу якимось чином зможе отримати доступ до вашого локального комп'ютера, а потім без обмежень отримає доступ до віддаленої ОС. Тому рекомендується після завершення активний дій з обміном файлами з віддаленим сервером відключати режим безпарольного *SSH*-з'єднання. Для цього достатньо очистити (видалити) файл *./ssh/authorized_keys* у домашньому каталозі користувача на віддаленій ОС, наприклад, для видалення файлу з домашнього каталогу користувача *blazhko* необхідно виконати наступну команду:

```
rm /home/blazhko/.ssh/authorized_keys
```

1.1.4 Початок роботи у терміналі ОС *Linux*

Після успішного підключення до віддаленого терміналу ОС *Linux* можна починати роботу, виконуючи традиційні команди оболонки *Bash* та утиліти командного рядку, як показано на рисунку 10.

```
[oracle@vpsj3IeQ ~]$ whoami
oracle
[oracle@vpsj3IeQ ~]$ pwd
/home/oracle
[oracle@vpsj3IeQ ~]$ uname -a
Linux vpsj3IeQ.s-host.com.ua 3.10.0-1160.24.1.el7.x86_64
x86_64 x86_64 x86_64 GNU/Linux
[oracle@vpsj3IeQ ~]$
```

Рис. 10 – Фрагмент екрану із прикладом початку роботи у віддаленому терміналу

Як багато користувальницьке середовище, ОС повинна забезпечувати зберігання інформації про її користувачів за рахунок ведення так званих профілів користувачів або облікових записів користувачів. У файловій системі створюється база даних користувачів, їх паролів і профілів доступу до локальних ресурсів обчислювальної системи.

В *Unix*-подібних ОС файл з обліковими записами користувачів розміщено за повним шляхом */etc/passwd*

Структура профілю користувача у файлі */etc/passwd* містить наступні стовпчики, розділені символом двокрапка:

- унікальне ім'я користувача;
- унікальний ідентифікатор користувача (*UID - user ID*) як число, з яким ОС легше працювати ніж зі строковими даними -імені користувача;
- ідентифікатор основної групи (*GID - group ID*), в яку входить користувач;
- реальне ім'я користувача;
- початковий (домашній) каталог користувача;
- назва оболонки командного рядку за замовчуванням.

Для керування профілями користувачів потрібен рівень адміністратора ОС.

Звичайний користувач може лише змінювати свій пароль наступною командою:

passwd

Але при створенні нового паролю, традиційно, користувач повинен вказати свій поточний пароль.

В *Unix*-подібних ОС файл з описом груп, до яких можуть належати користувачі та процеси, розміщено за повним шляхом */etc/group*

Структура файлу */etc/group* містить наступні стовпчики, розділені символом двокрапка:

- унікальне ім'я групи;
- ідентифікатор групи (*GID - group ID*);
- перелік імен користувачів, які належать до групи.

Файл */etc/group* дозволяє кожному користувачу належати до різних груп, хоча у нього лише одна основна група, описана у файлі */etc/passwd*

1.2 Особливості файлової структури *Unix*-подібних ОС

1.2.1 Коротенька історія розвитку файлових систем

Відомо, що назви більшості перших ОС для персональних комп'ютерів, наприклад, *Apple DOS*, *MS-DOS*, містили слово *Disk* – дискова, вказуючи на важливість для будь-якої ОС мати ефективну підсистему довгострокового зберігання даних.

Після впровадження у великі ОС механізму абстрагування периферійних пристроїв у вигляді файлів різного призначення, підсистеми зберігання даних стали називатися файловими підсистемами, в якій файл – це набір даних, до якого можна звертатися за іменем, а керування файлами забезпечувалося через виконання над ними базових команд створення, перегляду, зміни та видалення.

Наведемо перелік дискових файлових систем за хронологією їх появи:

- *FS (File System)* - перша файлова система для перших ОС *Unix*;
- *UFS (Unix File System)* - файлова система, створена для ОС сімейства *Unix BSD*;
- *FAT8 (File Allocation Table)* – перша версія файлової системи для *DOS*, яка містить таблицю розміщення файлів у вигляді ланцюжка кластерів (логічне розбиття диску у вигляді 1, 2, 4, 8, 16 та більше 512-байтних секторів) та містить $2^8 = 256$ елементів таблиці, коли *max* розмір диска (файлу) = $256 \times 8\text{Кб} = 2\text{ Мб}$ для 16-секторних кластерів (8Кб);
- *FAT12* - файлова система для ОС *MS DOS 1.0-2.0* *max* розміром таблиці 2^{12} елементів;
- *FAT16* - файлова система для ОС *MS DOS 3.0* та *Windows 1.0-3.0* *max* розміром таблиці 2^{16} елементів;
- *FAT32* – файлова система для ОС *Windows 95* та *max* розміром таблиці 2^{32} елементів;
- *HPFS (High Performance File System)* – файлова система для ОС *IBM OS/2*, як альтернативи ОС *Windows 2.0-3.0*;
- *HFS (Hierarchical File System)* - файлова система ОС *Mac OS* з *max* розміром таблиці 2^{16} елементів;
- *HSF+* - розвиток файлової системи *HFS* з *max* розміром таблиці 2^{32} елементів та підтримкою файлів з кодуванням *Unicode* для імен файлів і каталогів;
- *NTFS (New Technology File System)* - файлова система для ОС *Windows NT*, створена на основі *HPFS*;
- *Ext (Extended File System)* - перша файлова система для ОС *Linux*, створена на основі файлової системи ОС *Minix* для розширення обмеження розміру файлу до 2Гб та довжини імені файлу до 256 символів;

- *Ext2* – розвиток файлової системи *Ext* ОС *Linux* з підвищеною швидкістю роботи та керування доступу до файлів у вигляді *ACL*-списків;
- *Ext3* – варіант файлової системи *Ext2* з журналюванням з передбаченим записом деяких даних, який дозволяє відновити файлову систему при збоях в роботі комп'ютера, та *max* розміром файлу до 2^{40} байтів, розміром диску до 2^{45} байт;
- *Ext4* – розвиток файлової системи *Ext3* з механізмом запису файлів в безперервні ділянки блоків (екстенти) для зменшення фрагментації (розміщення кластерів зі змістом одного файлу в різних частинах диску) та підвищення продуктивності *max* розміром файлу до 2^{44} байтів та розміром диску до 2^{60} байт.

1.2.2 Обґрунтований вибір розміру кластеру для FAT

Основною проблемою *FAT* є неекономне використання диску при зберіганні великої кількості маленьких файлів, коли розмір файлів значно менше за розмір кластеру.

Припустимо, що використовується файлова система *FAT8*, в якій зберігається множина файлів з розмірами у кілобайтах: 1, 2, 3, 2, 1, 4, 2 Кб. Легко підрахувати, що середній розмір файлів = 2.1 Кб

Для забезпечення максимальної економії місця на диску необхідно використати кластер мінімального розміру = 2 сектори (1 Кб). В середньому, один файл буде займати 3 комірки *FAT*-таблиці, бронюючи 3Кб на диску. На диску, в середньому, можна буде розмістити $= 256/3 = 85.3 = 85$ файлів із середнім значенням коефіцієнту ефективності використання місця на диску $= 2.1/3 = 70\%$

Для підвищення економії місця у *FAT*-таблиці можна використати кластер середнього розміру = 8 секторів (4 Кб). При цьому, в середньому, один файл буде займати 1 комірку *FAT*-таблиці, бронюючи 4Кб на диску. На диску, в середньому, можна буде розмістити $= 256/4 = 64$ файлів із середнім значенням коефіцієнту ефективності використання місця на диску $= 2.1/4 = 52\%$

Для забезпечення максимальної економії місця у *FAT*-таблиці необхідно використати кластер максимального розміру, наприклад, 16 секторів (8 Кб). При цьому, в середньому, один файл також буде займати 1 комірку *FAT*-таблиці, але вже бронюючи 8Кб на диску. На диску, в середньому, можна буде також розмістити $= 256/8 = 32$ файлів, але вже із середнім значенням коефіцієнту ефективності використання місця на диску $= 2.1/8 = 26\%$

1.2.3 Структура файлової системи *Ext*

В класичній ОС *UNIX* та її нащадках, наприклад, ОС *Linux*, файлові системи виглядають як гілки та листя єдиного дерева. Дерево починається з кореневого каталогу (*ROOT* - вершини дерева), який починається з символу слеш /, як показано на рисунку 10.

Всі *UNIX*-подібні ОС мають приблизно однакову структуру дерева каталогів:

- системні утиліти - каталог */bin*
- системні бібліотеки – каталог */lib*,
- конфігураційні файли - каталог */etc*
- каталоги користувачів - каталог */home*

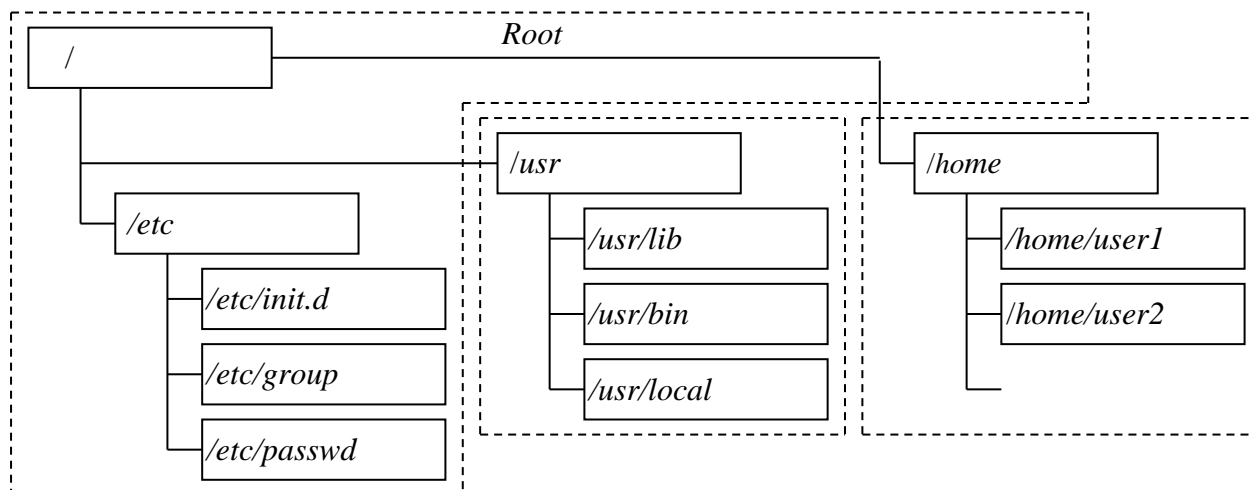


Рис. 10 - Фрагмент структури простору імен ієрархії файлової системи *Ext*

Основні команди керування файловою системою традиційно виконуються через оболонку командного рядку, наприклад, оболонку *Bash*.

В файловій системі *Ext*, як і в більшості систем, файли мають наступні типи:

- звичайний файл:
 - текстовий файл;
 - бінарний файл з об'єктними кодами;
 - спеціальний файл блокового або символьного пристрою;
 - іменованний канал;
 - сокет – файл, призначений для взаємодії між процесами, наприклад, для доступу до комп'ютерної мережі *TCP/IP*;
 - файл-зв'язок або посилання на інші файли;
- каталог як спеціальний файл, що містить інформацію про набір інших файлів.

1.2.4 Віддалене копіювання файлів через командний рядок

Відомо, що в *UNIX*-подібних ОС для копіювання файлів використовується команда *cp* (*copy*):

```
cp SourceFile TargetFile
```

Наприклад, для копіювання файлу *File1*, який розташовано у каталозі */dir1*, у файл *File2*, який розташовано у каталозі */dir2*, необхідно виконати:

```
cp /dir1/File1 /dir2/File2
```

Для безпечного копіювання файлів між файловими системами віддалених *UNIX*-подібних ОС існує утиліта *scp* (*Secure Copy*) - команда копіювання файлу *SourceFile* з локальної файлової системи до файлової системи віддаленої ОС:

```
scp SourceFile user@host:/directory/TargetFile
```

команда копіювання файлу *SourceFile* з файлової системи віддаленої ОС до локальної файлової системи:

```
scp user@host:/directory/SourceFile TargetFile
```

У вказаних командах назва файлу *SourceFile* має включати повний шлях до файлу, який копіюється.

1.2.5 Особливості опису файлів у файловій системі *EXT*

Починаючи з першої файлової системи *Unix* кожен файл має свій індексний дескриптор (*index node* - *inode*), що описує властивості файлу, наприклад:

- розмір файлу;
- ідентифікатор (ID) пристрою, що містить файл;
- ID користувача-власника файлу, який зберігається в файлі */etc/passwd*;
- ID групи користувачів, до якої належить власник, та який зберігається в файлі */etc/group*;
- тип файлу;
- права доступу, присвоєні власнику, групі та іншим користувачам;
- права читання, модифікації і виконання для власника, групи та інших користувачів;
- час останнього доступу і зміни файлу;
- лічильник кількості жорстких посилань на файл;
- покажчики на блоки (кластери) диска, в яких розміщений файл;
- ім'я каталогу або блочного пристрою, де розташований файл;
- кількість блоків, займаних файлом.

Коли користувач намагається отримати до файлу доступ, тоді:

- на ім'я файлу в таблиці визначається відповідний йому номер *inode*
- за номером *inode* відбувається звернення до *Inode table* і зчитується дескриптор файлу.

Для отримання мета-опису файлу у вигляді дескриптору файлу можна використати команду *stat*, яка надає наступні подробиці:

- файл (*File*) - шлях до файлу, за яким показується інформація;
- розмір (*Size*) - розмір файлу в байтах;
- блоків (*Blocks*) - кількість блоків файлової системи, зайнятих файлом;
- блок (*IO Block*) - розмір блоку файлової системи в байтах;
- тип файлу (текстовий, бінарний та інші);
- пристрій (*Device*) - ідентифікатор пристрою, наприклад, жорсткий диск, на якому збережений файл;
- *Inode* - унікальний номер *Inode* цього файлу;
- посилання (*Links*) - кількість жорстких посилань на цей файл;
- доступ (*Access*) - права доступу до файлу;
- *Uid* - ідентифікатор і ім'я користувача-власника файлу;
- *Gid* - ідентифікатор і ім'я групи файлу;
- доступ (*Access*) - час останнього доступу до файлу;
- модифіковано (*Modify*) - час коли в останній раз змінювався зміст файлу;
- змінено (*Change*) - час, коли в останній раз змінювалися атрибути файлу або зміст файлу;
- створено (*Birth*) - зарезервовано для відображення початкової дати створення файлу, але поки ще не реалізовано.

На рисунку 11 показано результат виконання команди *stat /etc/passwd*

```
[oracle@vpsj3IeQ ~]$ stat /etc/passwd
  File: '/etc/passwd'
  Size: 2015          Blocks: 8          IO Block: 4096   regular file
Device: fd01h/64769d Inode: 12234        Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Access: 2022-04-13 16:58:01.455624935 +0300
Modify: 2021-12-22 15:57:32.785217787 +0200
Change: 2021-12-22 15:57:32.786217812 +0200
 Birth: -
```

Рис. 11 – Результат виконання команди *stat /etc/passwd*

Додатково значення *inode* можна отримати для всіх файлів каталогу через команду *ls* з опцією *-li*:

```
ls -li
```

Для швидкого визначення типу файлу без перегляду його змісту можна використати команду *file*, яка розглядає початкову частину файлу і використовує спеціальні тести. Контекстно-залежні або позиційно-залежні тести порівнюють різні місця розташування в файлі з текстовою базою даних, яка містить сигнатури у вигляді послідовності байтів як

«візитна картка» якогось типу файлів. Ця база даних міститься у файлі під назвою *magic*, який зазвичай розташований в */etc/magic* або в */usr/share/file/magic*.

На рисунку 12 наведено приклади результатів роботи команди *file* для типів файлів:

- 1) текстовий пустий файл;
- 2) текстовий файл;
- 3) текстовий скриптовий файл;
- 4) бінарний файл-зображення;
- 5) бінарний аудіо-файл;
- 6) бінарний відео-файл;
- 7) спеціальний текстовий doc-файл;
- 8) текстовий html-файл;
- 9) бінарний pdf-файл

```
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ ls
bash_example.sh  file.jpg  file.mp4  OS_lab_1.doc  OS_lab_1.pdf
file_empty.txt   file.mp3  file.txt  OS_lab_1.html
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file file_empty.txt
file_empty.txt: empty
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file file.txt
file.txt: ASCII text
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file bash_example.sh
bash_example.sh: Bourne-Again shell script, UTF-8 Unicode text executable, with
CRLF line terminators
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file file.jpg
file.jpg: JPEG image data, JFIF standard 1.01
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file file.mp3
file.mp3: Audio file with ID3 version 2.3.0MPEG ADTS, layer III, v1, 64 kbps, 4
4.1 kHz, Monaural
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file file.mp4
file.mp4: ISO Media, MPEG v4 system, version 1
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file OS_lab_1.doc
OS_lab_1.doc: Composite Document File V2 Document, Little Endian, Os: Windows, V
ersion 10.0, Code page: 1251, Title: \020\20080s\177, Author: Aleksandr, Templat
e: Normal.dotm, Last Saved By: Normal.do m, Revision Number: 148, Name of Creati
ng Application: Microsoft Office Word, Total Editing Time: 1d+07:36:00, Last Pri
nted: Mon Feb 27 04:13:00 2017, Create Time/Date: Sun Oct 1 11:02:00 2017, Last
Saved Time/Date: Mon Feb 22 07:26:00 2021, Number of Pages: 6, Number of Words:
2235, Number of Characters: 12743, Security: 0
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file OS_lab_1.html
OS_lab_1.html: HTML document, UTF-8 Unicode text, with very long lines
[oracle@vpsj3Ieq BlazhkoWorkLab5]$ file OS_lab_1.pdf
OS_lab_1.pdf: PDF document, version 1.4
[oracle@vpsj3Ieq BlazhkoWorkLab5]$
```

Рис. 12 - Приклади результатів роботи команди для різних типів файлів

1.2.6 Керування файлами-зв'язками

Багато файлових систем дають змогу задавати кілька імен для одного й того ж файлу. Такі імена називають *зв'язками (links)*.

В *Unix*-подібних ОС розрізняють:

- жорсткі зв'язки (*hard links*);
- гнучкі (символічні) зв'язки (*symbolic link*).

Жорсткий зв'язок (*hard links*) дозволяє створити для одного файлу декілька імен. Усі жорсткі зв'язки визначають одні й ті самі дані на диску, для користувача вони не відрізняються: не можна визначити, які з них були створені раніше, а які – пізніше.

Для створення жорсткого зв'язку використовується команда *ln*:

ln вихідний_файл шлях_файл-зв'язок

На рисунку 13 наведено приклад створення для файлу з назвою *file.txt* жорсткого зв'язку з назвою *file_hard_link.txt*. Аналіз значень індексних дескрипторів *inode* (перша колонка результату виконання команди *ls -li*) показує, що файл та його жорсткий зв'язок мають однакове значення *inode*. Тобто ці файли повністю однакові, а відрізняються лише назвами. Змінивши один файл, автоматично буде змінено і інший файл. Третя колонка результату виконання команди *ls -li* показує, що кожний файл має два жорсткі зв'язки.

```
[oracle@vpsj3IeQ BlazhkoWorkLab5]$ ls
bash_example.sh  file.jpg  file.mp4  OS_lab_1.doc  OS_lab_1.pdf
file_empty.txt   file.mp3  file.txt  OS_lab_1.html
[oracle@vpsj3IeQ BlazhkoWorkLab5]$
[oracle@vpsj3IeQ BlazhkoWorkLab5]$ ln file.txt file_hard_link.txt
[oracle@vpsj3IeQ BlazhkoWorkLab5]$
[oracle@vpsj3IeQ BlazhkoWorkLab5]$ ls -li
total 520
140100 -rwxr--r-- 1 oracle oinstall 4931 Apr 13 20:32 bash_example.sh
140105 -rw-r--r-- 1 oracle oinstall 0 Apr 13 20:35 file_empty.txt
140104 -rwxr-xr-- 2 oracle oinstall 65 Apr 13 20:34 file_hard_link.txt
140106 -rw-r--r-- 1 oracle oinstall 64039 Apr 13 20:44 file.jpg
140107 -rw-r--r-- 1 oracle oinstall 63027 Apr 13 20:44 file.mp3
140108 -rw-r--r-- 1 oracle oinstall 107076 Apr 13 20:44 file.mp4
140104 -rwxr-xr-- 2 oracle oinstall 65 Apr 13 20:34 file.txt
140101 -rw-r--r-- 1 oracle oinstall 105984 Apr 13 20:33 OS_lab_1.doc
140102 -rw-r--r-- 1 oracle oinstall 63471 Apr 13 20:33 OS_lab_1.html
140103 -rw-r--r-- 1 oracle oinstall 100900 Apr 13 20:33 OS_lab_1.pdf
[oracle@vpsj3IeQ BlazhkoWorkLab5]$
```

Рис. 13 – Приклад створення та аналізу жорсткого зв'язку

Жорсткі зв'язки мають певні недоліки, які обмежують їх застосування:

- не можуть бути задані для каталогів;
- усі жорсткі зв'язки одного файлу завжди мають перебувати на одному й тому самому розділі жорсткого диска (в одній файловій системі).

Символічний або гнучкий зв'язок (*symbolic link*) – зв'язок, який фізично відокремлено від файлу, на який він вказує. Фактично, це спеціальний файл, що містить повний шлях до файлу, на який вказує.

Наведемо властивості символічних зв'язків:

- при вилученні зв'язку, вихідний файл, тобто файл, на який він вказував, залишиться;

- якщо вихідний файл перемістити або вилучити, зв'язок розірветься, і доступ через нього стане неможливий, але, якщо вихідний файл потім поновити на тому самому місці, зв'язком знову можна користуватися;
- символічні зв'язки можуть вказувати на каталоги і файли, що перебувають на інших файлових системах або на іншому розділі диску.

Для створення символічного зв'язку використовується команда *ln* з опцією *-s* :

ln -s початковий_файл файл-зв'язок

На рисунку 14 наведено приклад створення для файлу з назвою *file.txt* жорсткого символічного зв'язку з назвою *file_sym_link.txt*. Аналіз значень індексних дескрипторів *inode* показує, що файл та його символічний зв'язок мають різні значення *inode*.

```
[oracle@vpsj3IeQ BlazhkoWorkLab5]$ ln -s file.txt file_sym_link.txt
[oracle@vpsj3IeQ BlazhkoWorkLab5]$
[oracle@vpsj3IeQ BlazhkoWorkLab5]$ ls -li
total 520
140100 -rwxr--r-- 1 oracle oinstall 4931 Apr 13 20:32 bash_example.sh
140105 -rw-r--r-- 1 oracle oinstall 0 Apr 13 20:35 file_empty.txt
140104 -rwxr-xr-- 2 oracle oinstall 126 Apr 14 12:50 file_hard_link.txt
140106 -rw-r--r-- 1 oracle oinstall 64039 Apr 13 20:44 file.jpg
140107 -rw-r--r-- 1 oracle oinstall 63027 Apr 13 20:44 file.mp3
140108 -rw-r--r-- 1 oracle oinstall 107076 Apr 13 20:44 file.mp4
140109 lrwxrwxrwx 1 oracle oinstall 8 Apr 14 13:03 file_sym_link.txt -> file.txt
140104 -rwxr-xr-- 2 oracle oinstall 126 Apr 14 12:50 file.txt
140101 -rw-r--r-- 1 oracle oinstall 105984 Apr 13 20:33 os_lab_1.doc
140102 -rw-r--r-- 1 oracle oinstall 63471 Apr 13 20:33 os_lab_1.html
140103 -rw-r--r-- 1 oracle oinstall 100900 Apr 13 20:33 os_lab_1.pdf
[oracle@vpsj3IeQ BlazhkoWorkLab5]$
```

Рис. 14 - Приклад створення та аналізу символічного зв'язку

Також з рисунку 14 видно, що в другій колонці для символічного зв'язку з'явилася літера *l*, яка визначає особливий тип файлу, чого не було із жорстким зв'язком.

Можливо, єдиним недоліком використання символічного зв'язку перед жорстким зв'язком є неможливість обмежувати права доступу на файл зв'язку, тому що вони автоматично будуть встановлені на сам файл, до якого йде зв'язок.

В таблиці 1 наведено всі можливі описи типів файлів, які визначає перший символ з вказаного стовпчика.

Таблиця 1 – Позначення типів файлів

Символ	Опис типу файла
-	Звичайний файл
<i>b</i>	Файл, пов'язаний із пристроєм блокового типу, наприклад, пристрій постійного зберігання даних
<i>c</i>	Файл, пов'язаний із пристроєм символьного типу, наприклад, термінальний пристрій з клавіатурою та дисплеєм
<i>d</i>	Файл-каталог
<i>l</i>	Файл-символічний зв'язок

1.3 Керування правами (повноваженнями) доступу до файлів

В процесі керування файлами ще в ОС *Multics*, як прообразі першої ОС *Unix*, було запропоновано обмежувати права доступу на операції з файлами з боку користувачів та процесів.

Один зі засобів визначення повноважень на доступ до файлу – це список керування доступом (*Access Control List, ACL*), який визначає:

- 1) права користувача як власника файлу - того, хто його створив;
- 2) права групи користувачів, які об'єднано з урахуванням якихось загальних виконуваних ними задач або повноважень;
- 3) права всіх інших користувачів як права за замовчуванням.

Права *власника* вище прав *групи*, а права *групи* вище прав *за замовчуванням*.

Користувач може належати до декількох груп одночасно, але файл завжди належить тільки одній групі.

Для визначення власника файлу та отримання переліку прав доступу до файлу використовується команда *ls* з опцією *-l*, яка формує список файлів у псевдотабличному виді:

```
ls -l
```

Окрім імені файлу команда із вказаною опцією виводить:

- тип файлу (перший символ у першій колонці);
- права доступу до файлу (наступні символи у першій колонці);
- кількість посилань на файл (друга колонка):
 - для звичайного файлу - кількість жостких зв'язків;
 - файлу-каталогу - кількість файлів, або каталогів, які входять до змісту каталогу;
- ім'я користувача-власника (третя колонка);

- ім'я групи користувачів (четверта колонка);
- розмір файлу в байтах (п'ята колонка);
- час останньої модифікації файлу, при цьому для файлів з часом більше ніж 6 місяців тому міститься рік замість часу дня.

Кожен каталог зі списком вмісту передуює рядком *'total blocks'*, де *blocks* - загальний дисковий простір у блоках, наприклад, кластерах, які використовуються всіма файлами в даному каталозі.

Власником знову створеного файлу є користувач, що створив файл. Для зміни власника файлу використовується команда:

```
Chown ім'я_власника-користувача список_файлів
```

Наприклад, наступна команда встановить користувача *haupt* власником файлів *client.c* та *server.c*:

```
chown haupt client.c server.c
```

Для зміни власника групи використовується команда:

```
Chgrp ім'я_власника-групи список_файлів
```

Наприклад, наступна команда встановить групу *admin* власником всіх файлів поточного каталогу:

```
chgrp admin *
```

Володіння файлом визначає той набір операцій, що користувач може зробити з файлом. Деякі з них, такі як зміна власника файлу, здійснює тільки власник або адміністратор ОС. Інші операції, такі як читання, запис або виконання, додатково контролюються правами доступу.

Unix-подібні ОС підтримують три типи прав доступу - повноважень виконувати операції:

- читання (*Read*);
- модифікації (*Write*);
- виконання (*eXecute*).

Можна надати повноваження виконувати кожен операцію користувачу, який належить одному з трьох типів користувачів:

- власник – *user*;
- група користувачів, які входять до власника групи – *group*;
- всі інші користувачі – *others*.

В результаті комбінації повноважень з типами користувачів можна отримати 9-ть можливих повноважень на виконання операцій різними типами користувачів.

Повноваження можна швидко переглянути у другому стовпчику результати виконання команди `ls -l` за винятком першого символу, що позначає тип файлу.

Наявність повноважень виконувати операцію позначається відповідним символом (*r*, *w*, *x*), а відсутність повноважень втконувати відповідну операцію позначається символом дефіс. В таблиці 2 наведено приклади коментарів до опису повноважень.

Таблиця 2 – Приклад коментарів до опису повноважень

Повноваження власника-користувача файлу	Повноваження власника групи	Повноваження інших користувачів
<i>rwx</i>	<i>r-x</i>	<i>r--</i>
1) Дозволяється читати, 2)Дозволяється модифікувати 3) Дозволяється виконувати	1) Дозволяється читати 2) Не дозволяється модифікувати 3) Дозволяється виконувати	1) Дозволяється читати 2) Не дозволяється модифікувати 3) Не дозволяється виконувати

Права доступу можуть бути змінені тільки власником файлу або адміністратором системи з використанням команди `chmod` з аргументами як показано на рисунку 15:

- вказівка на тип доступу користувача:
'u' – user, 'g' – group, 'o' – others, 'a' – all, всі типи користувачів;
- вказівка на дію, яку необхідно виконати командою:
'+' – додати, '-' – видалити, '=' – присвоїти;
- вказівка на операцію з відповідними повновження: *r, w, x*
- перелік файлів, для яких визначаються повноваження.

$$chmod \begin{bmatrix} u \\ g \\ o \\ a \end{bmatrix} \begin{bmatrix} + \\ - \\ = \end{bmatrix} \begin{bmatrix} r \\ w \\ x \end{bmatrix} file1 file2$$

Рис. 15 – Аргументи команди `chmod`

Приклади використання команди `chmod` наведено у таблиці 3.

Таблиця 3 - Приклади використання команди *chmod*

Команда	Опис
<code>\$ chmod a+w text</code>	Надати право на модифікацію для всіх типів користувачів
<code>\$ chmod go=r text</code>	Встановити право на читання для всіх типів користувачів, за винятком власника
<code>\$ chmod g+x-w autorun</code>	1) додати для групи право на виконання файлу 2) зняти право на модифікацію
<code>\$ chmod u+w,og+r-w t1 t2</code>	1) додати право модифікації для власника, 2) додати право на читання для групи та інших користувачів, 3) відключити право на модифікацію для всіх користувачів, крім власника

Права визначаються бітами в масці повноважень доступу:

- '0' – відсутність відповідного повноваження;
- '1' – присутність відповідного повноваження;
- три біти визначають 3-розрядне двійкове число;
- 3-розрядне двійкове число перетворюється у відповідне десяткове число.

Таким чином, дві нижче вказані команди є еквівалентними:

`chmod u=rwx,g=rx,o=r *`

`chmod 754 *`

В таблиці 4 показано приклади символічного опису прав, числового двійкового та числового десяткового.

Таблиця 4 - Приклади символічного опису прав, числового двійкового та числового десяткового

Тип опису прав	Власник	Група користувачів	Інші користувачі
Символьний	r w x	r - x	r - -
Числовий-двійковий	1 1 1	1 0 1	1 0 0
Числовий-десятковий	7	5	4

Раніше вже говорилося, що єдиним недоліком використання символічного зв'язку перед жорстким зв'язком є неможливість обмежувати повноваження доступу на файл зв'язку, тому що вони автоматично будуть встановлені на сам файл, до якого йде зв'язок.

На рисунку 16 наведено демонструється результат зміни таких повноважень для файлу символічного зв'язку, коли після виконання операції зняття всіх повноважень на файл-зв'язок насправді знято всі повноваження на сам файл, а у файлі-зв'язку опис повноважень не змінився і дорівнює 777.

```
[oracle@vpsj3IeQ files]$ ls -l
total 4
d--x--x--x 2 oracle oinstall 4096 Apr 14 15:46 files
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ln -s files/ files_sym_link
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l
total 4
d--x--x--x 2 oracle oinstall 4096 Apr 14 15:46 files
lrwxrwxrwx 1 oracle oinstall  6 Apr 14 16:14 files_sym_link -> files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ chmod 000 files_sym_link/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l
total 4
d----- 2 oracle oinstall 4096 Apr 14 15:46 files
lrwxrwxrwx 1 oracle oinstall  6 Apr 14 16:14 files_sym_link -> files/
[oracle@vpsj3IeQ files]$
```

Рис. 16 – Демонстрація результатів виконання команди зміни повноважень на файлі символічного зв'язку

1.2.7. Робота з «темним» каталогом»

Для каталогів система трактує операції читання й модифікації відмінно від інших файлів. Відомо, що каталог це також файл, але спеціальний. Право на виконання операції читання каталогу дозволяє одержати імена (і тільки імена) файлів, що перебувають у даному каталозі як файлі. Щоб одержати додаткову інформацію про файли каталогу, наприклад, докладний лістинг команди *ls -l*, системі необхідно переглянути метадані файлів, що вимагає прав на операцію виконання при доступі до каталогу.

Розглянемо наступний приклад.

Нехай існує файл-каталог *files*, який містить три звичайні файли *f1*, *f2*, *f3*, характеристики яких наведено на рисунку 17.

```
[oracle@vpsj3IeQ files]$ ls -l
total 4
drwxr-xr-x 2 oracle oinstall 4096 Apr 14 14:57 files
[oracle@vpsj3IeQ files]$ ls -l files/
total 0
-rw-r--r-- 1 oracle oinstall 0 Apr 14 14:52 f1
-rw-r--r-- 1 oracle oinstall 0 Apr 14 14:52 f2
-rw-r--r-- 1 oracle oinstall 0 Apr 14 14:52 f3
[oracle@vpsj3IeQ files]$
```

Рис. 17 – Фрагмент характеристик файлів, отриманий командами *ls -l*

Як видно з результату виконання першої команди *ls -l* всі типи користувачів мають права на виконання і читання файлу-каталогу *files*.

Для відбирання прав на виконання файлу-каталогу *files* необхідно виконати команду:
chmod -x files

Після виконання команди повторне читання каталогу призведе до помилок, показаних на рисунку 18, тому що:

- у користувача є лише можливість отримати тільки імена файлів у відповідності з наявністю повноважень на читання каталогу;
- у користувача немає можливості отримати опис файлів, бо у користувача відсутні повноваження на виконання каталогу, про що свідчить помилка «*Permission denied*» - у дозволі відмовлено.

```
[oracle@vpsj3IeQ files]$ chmod -x files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l
total 4
drw-r--r-- 2 oracle oinstall 4096 Apr 14 14:57 files
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l files/
ls: cannot access files/f1: Permission denied
ls: cannot access files/f3: Permission denied
ls: cannot access files/f2: Permission denied
total 0
-????????? ? ? ? ? ? f1
-????????? ? ? ? ? ? f2
-????????? ? ? ? ? ? f3
[oracle@vpsj3IeQ files]$
```

Рис. 18 – Фрагмент характеристик файлів з помилками обмеження доступу на отримання всіх характеристик файлів

Щоб зробити каталог поточним за допомогою команди *cd*, також потрібно мати повноваження на операцію виконання файлу-каталогу. Але відсутність повноважень на операцію читання не дозволяє переглянути будь-яку характеристику про файл.

Як видно на рисунку 19 після відбирання повноважень на операцію читання та надання повноважень на операцію виконання вже неможливо переглянути зміст каталогу, але в нього ще можливо перейти через команду *cd*.

Наведена комбінація прав на файл-каталог, коли можна виконувати файл-каталог, але не можна читати з файлу-каталогу, призводить до ефекту створення так званого «темного» каталогу: файли «темного» каталогу будуть доступні користувачу тільки в тому випадку, якщо користувач заздалегідь знає їхні імена, оскільки неможливо одержати назви цих файлів.

```
[oracle@vpsj3IeQ files]$ chmod +x files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ chmod -r files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l
total 4
d-wx--x--x 2 oracle oinstall 4096 Apr 14 14:57 files
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l files/
ls: cannot open directory files/: Permission denied
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ cd files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls
ls: cannot open directory .: Permission denied
[oracle@vpsj3IeQ files]$ |
```

Рис. 19 – Приклад створення «темного» каталогу як результати роботи команд відбирання прав на читання та надання прав на виконання

На рисунку 20 наведено приклади команд, які показують, що знаходячись у «темному» каталозі:

- не можна отримувати зміст каталогу зі списком імен файлів;
- можна отримувати опис файлів, якщо заздалегідь знати імена файлів;
- можна створювати нові файли.

```

[oracle@vpsj3IeQ files]$ ls -l
total 4
d-wx--x--x 2 oracle oinstall 4096 Apr 14 15:46 files
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ cd files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls
ls: cannot open directory .: Permission denied
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l f*
ls: cannot access f*: No such file or directory
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l f1
-rw-r--r-- 1 oracle oinstall 0 Apr 14 14:52 f1
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l f2
-rw-r--r-- 1 oracle oinstall 0 Apr 14 14:52 f2
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l f3
-rw-r--r-- 1 oracle oinstall 0 Apr 14 14:52 f3
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls > f4
ls: cannot open directory .: Permission denied
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l f4
-rw-r--r-- 1 oracle oinstall 0 Apr 14 15:50 f4
[oracle@vpsj3IeQ files]$

```

Рис. 20 – Результати роботи команд в «темному» каталозі

Для видалення, створення або зміни імені файлу досить мати право на виконання операції запису в каталог, у якому він знаходиться. При цьому не враховуються права доступу самого файлу. Для того, щоб видалити деякі файли з каталогу, необов'язково мати які-небудь права доступу до цього файлу, важливо лише мати право на запис для каталогу, у якому він використовується.

На рисунку 21 наведено результати роботи команд в каталозі після відібрання у користувача права на операцію запису.

```

[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ chmod -w files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls -l
total 4
d--x--x--x 2 oracle oinstall 4096 Apr 14 15:46 files
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ cd files/
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ ls > f5
-bash: f5: Permission denied
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ rm f1
rm: cannot remove 'f1': Permission denied
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$ mv f1 f6
mv: cannot move 'f1' to 'f6': Permission denied
[oracle@vpsj3IeQ files]$
[oracle@vpsj3IeQ files]$

```

Рис. 21 – Результати роботи команд в каталозі без прав на запис

Окремих прав на видалення файлу не існує, є тільки операція видалення імені – *unlink*.

2 Завдання до виконання

2.1 Робота з віддаленим *Linux*-сервером через термінальний режим роботи

Запустити на локальному комп'ютері псевдотермінал *Git Bash* та виконати наступні завдання.

2.1.1 Перевірити наявність мережевого з'єднання між вашим локальним комп'ютером та віддаленим сервером *Linux*, який має мережеву адресу з *IP* = 46.175.148.116

2.1.2 За вказаною мережевою адресою встановити з'єднання з віддаленим псевдотерміналом на *Linux* сервері, використовуючи логін та пароль, наданий вам лектором.

2.1.3 Отримати на екран значення поточного каталогу користувача.

2.1.4 Знаходячись у поточному каталозі користувача, налаштувати *Git*-параметри (ім'я облікового запису та *E-mail*), як це було зроблено в лабораторній роботі №2.

2.1.5 Для безпечного клонування з *GitHub*-репозиторію необхідно налаштувати роботу *SSH*-протоколу для вашого користувача, тому необхідно виконати генерацію *SSH*-ключів, як це було зроблено в лабораторній роботі №2.

2.1.6 У попередньому пункті було створено каталог *.ssh* та файли з новими *SSH*-ключами. Але для продовження використання *GitHub*-репозиторію рекомендується виконати дії за одним із двох варіантів:

1) використовувати ключі, які вже було створено у лабораторній роботі №2, тому для цього треба запустити на локальному комп'ютері ще один термінал *Git Bash* та скопіювати файли з відкритим та закритим *SSH*-ключами з каталогу *.ssh*, який знаходиться у поточному каталозі користувача локального комп'ютера, до каталогу *.ssh*, який знаходиться у поточному каталозі користувача на віддаленому *Linux*-сервері.

2) використовувати ключі, які створено у пункті 2.1.5, тому на веб-сервісі *GitHub* за посиланням <https://github.com/settings/keys> необхідно зареєструвати ще один відкритий *SSH*-ключ, як це було виконано у лабораторній роботі №2.

2.1.7 Виконати безпечне клонування *GitHub*-репозиторію, який використовувався вами для надання звітності з виконання рішень, створивши *Git*-репозиторій. Якщо виникне помилка, перевірте правильність скопійованих *SSH*-ключів.

2.1.8 Перейти до каталогу *Git*-репозиторія та:

- створити нову *Git*-гілку з назвою «*Laboratory-work-5*»;
- перейти до роботи зі створеною гілкою;
- створити каталог з назвою «*Laboratory-work-5*»;
- перейти до каталогу «*Laboratory-work-5*»;

- в каталозі «*Laboratory-work-5*» створити файл *README.md* та додати до файлу рядок тексту із темою лабораторної роботи «*Основи керування файлами у файловій системі Unix-подібних ОС*» як заголовок 2-го рівня *Markdown*-форматування;
- виконати операції з оновлення *GitHub*-репозиторію змінами *Git*-репозиторія через послідовність *Git*-команд *add*, *commit* із коментарем «*Changed by Local Git*» та *push*;
- на веб-сервісі *GitHub* перейти до створеної гілки «*Laboratory-work-5*»;
- перейти до каталогу «*Laboratory-work-5*» та розпочати процес редагування файлу *README.md* ;
- у файлі *README.md* додати рядок у вигляді заголовку 3-го рівня *Markdown*-форматування: «*1 Робота з віддаленим Linux-сервером через термінальний режим роботи*»;
- в подальшому за результатами рішень кожного наступного пункту завдання до файлу *README.md* додавати знімки екрану та підписи до знімків екранів з описом завдань.

2.1.9 Змінити пароль вашого облікового запису користувача ОС *Linux*.

2.1.10 Завершити роботу у віддаленому псевдотерміналі через команду *exit* та, знаходячись на локальному комп'ютері, виконати команду з налаштування встановлення з'єднання із терміналом ОС *Linux* без необхідності введення паролю користувача.

2.1.11 Виконати повторне з'єднання з віддаленим псевдотерміналом ОС *Linux* без необхідності введення паролю користувача.

2.1.12 Отримати на екран з файлу */etc/password* один рядок з описом облікового запису вашого користувача, а з файлу */etc/group* – рядок з описом вашої основної групи.

2.1.13 Виконати додаткове з'єднання з віддаленим терміналом ОС *Linux*, та розташувати на екрані псевдотермінали поряд одним з одним.

2.1.14 Отримати на екран список активних псевдотерміналів лише вашого користувача

2.1.15 Виконати команду передачі повідомлення на один із ваших псевдотерміналів, вказавшу рядок з вашим прізвищем та ім'ям, після чого завершити роботу з командою внесення.

2.1.16 В одному з псевдотерміналів виконати команду блокування повідомлень та повторити попереднє завдання.

2.1.17 Запустити на локальному комп'ютері ще один термінал *Git Bash*, перейти до *Git*-репозиторію із файлами *WebAR*-застосунку лабораторної роботи №2 та скопіювати 5-ть файлів на віддалений *Linux*-сервер до каталогу «*Laboratory-work-5*» *Git*-репозиторія, враховуючи наступні типи по одному файлу:

- текстовий файл, наприклад *README.md*;

- текстовий *html*-файл, наприклад, *index.html*;
- бінарний *pdf*-файл з вашим *WebAR*-буклетом;
- будь-який бінарний файл-зображення;
- будь-який бінарний аудіо-файл.

2.1.18 Вимкнути режим безпарольного *SSH*-з'єднання з віддаленим сервером

2.2 Аналіз типів файлів

Документуючі рішення цього розділу, у файлі *README.md* необхідно вказати наступний текстовий рядок у вигляді заголовку 3-го рівня *Markdown*-форматування:

«2 Аналіз типів файлів». В подальшому за результатами рішень кожного пункту завдання до файлу *README.md* додати знімки екрану та підписи до знімків екранів з описом завдань.

Знаходячись у каталозі «*Laboratory-work-5*» *Git*-репозиторія на віддаленому *Linux*-сервері, виконати наступні завдання, враховуючи п'ять скопійованих раніше файлів.

2.2.1 Для кожного з вказаних файлів виконайте команду визначення їх типів.

2.2.2 Для кожного з вказаних файлів виконайте команду перегляду мета-опису файлу у вигляді дескриптору файлу.

2.2.3 Для *html*-файлу створити два жорсткі зв'язки з назвами *транслітація_вашого_прізвища + hard_link_1*, *транслітація_вашого_прізвища + hard_link_2*, наприклад, *Blazhko_hard_link_1* та *Blazhko_hard_link_2*

2.2.4 Для *html*-файлу створити файл символічного зв'язку з назвою *транслітація_вашого_прізвища + sym_link_1*, наприклад, *Blazhko_sym_link_1*

2.2.5 Вивести на екран у псевдо табличному форматі дані про створені файли-зв'язки, в якому серед стовпчиків буде стовпчик зі значеннями *inode* та стовпчик зі значеннями кількості зв'язків.

2.3 Керування правами доступу до файлів

Документуючі рішення цього розділу, у файлі *README.md* необхідно вказати наступний текстовий рядок у вигляді заголовку 3-го рівня *Markdown*-форматування:

«3 Керування правами доступу до файлів». В подальшому за результатами рішень кожного пункту завдання до файлу *README.md* додати знімки екрану та підписи до знімків екранів, які містять опис завдань.

Продовжуючи роботу у каталозі «*Laboratory-work-5*» *Git*-репозиторія на віддаленому *Linux*-сервері, виконати наступні завдання.

2.3.1 Переглянути права доступу до створених файлів жорсткого та символічного зв'язків.

2.3.2 Надати символічні права доступу до файлу з назвою *транслітація_вашого_прізвища + hard_link_1*:

- варіант взяти з колонки «Права доступу 1» таблиці 5;
- у таблиці вказано лише права, які необхідно встановити, та не вказано права, які необхідно зняти;
- тип файлу не повинен протирічити визначеним правам, наприклад, якщо використовується звичайний текстовий файл, тоді він не може мати права на виконання;

2.3.3 Перевірити обмеження прав доступу до файлу з попереднього пункту завдання, виконавши відповідні команди роботи з файлом, наприклад, якщо заборонено читання, тоді переглянути файл, або, якщо заборонено редагування, тоді виконати редагування файлу.

2.3.4 Надати числові десяткові права доступу до файлу з назвою *транслітація_вашого_прізвища + hard_link_2*:

- варіант взяти з колонки «Права доступу 2» таблиці 5;
- у таблиці вказано лише права, які необхідно встановити, та не вказано права, які необхідно зняти;
- тип файлу не повинен протирічити визначеним правам, наприклад, якщо використовується звичайний текстовий файл, тоді він не може мати права на виконання;

2.3.5 Перевірити обмеження прав доступу до файлу з попереднього пункту завдання, виконавши відповідні команди роботи з файлом, наприклад, якщо заборонено читання, тоді переглянути файл або, якщо заборонено редагування, тоді виконати редагування файлу.

2.3.6 Повторити попереднє завдання але вже для файлу символічного зв'язку, перевіривши можливість такої зміни, після чого проаналізувати, що сталося з правами для файлу-зв'язку та файлу, на який цей зв'язок посилається.

2.3.7 Створити новий каталог з назвою «*Dark Directory*» та скопіювати до каталогу один текстовий файл.

2.3.8 Перетворити каталог «*Dark Directory*» на «темний» каталог, виконавши відповідні команди зміни прав доступу до цього каталогу.

2.3.9 Отримати на екран список файлів «темного» каталогу. Для «темного» каталогу ця операція повинна завершитися помилкою.

2.3.10 Отримати на екран опис одного файлу темного каталогу за відомою вам назвою цього файлу. Для «темного» каталогу ця операція має бути успішною.

2.3.11 Скопіювати до «темного» каталогу ще один текстовий файл. Для «темного» каталогу ця операція має бути успішною.

Таблиця 5 – Варіанти завдань

№ варіанта	Права доступу 1			Права доступу 2		
	Власник	Група	Інші	Власник	Група	Інші
1	001	111	101	111	101	010
2	010	110	110	001	100	101
3	011	101	111	010	011	110
4	100	100	001	011	010	001
5	101	011	010	100	001	010
6	110	010	011	011	010	111
7	111	001	100	100	001	001
8	101	111	111	101	111	010
9	110	110	001	110	110	101
10	111	101	010	001	111	101
11	001	111	101	111	101	010
12	010	110	110	001	100	101
13	011	101	111	010	011	110
14	100	100	001	011	010	001
15	101	011	010	100	001	010
16	110	010	011	011	010	111
17	111	001	100	100	001	001
18	101	111	111	101	111	010
19	110	110	001	110	110	101
20	111	101	010	001	111	101
21	001	111	101	111	101	010
22	010	110	110	001	100	101
23	011	101	111	010	011	110
24	100	100	001	011	010	001
25	101	011	010	100	001	010
26	110	010	011	011	010	111
27	111	001	100	100	001	001
28	101	111	111	101	111	010
29	110	110	001	110	110	101

2.4 Підготовка процесу *Code Review* для надання рішень завдань лабораторної роботи на перевірку викладачем

2.4.1 На веб-сервісі *GitHub* зафіксувати зміни у файлі *README.md*

2.4.2 Скопіювати файли, які було створено у попередньому розділі, в каталог «*Laboratory-work-5*» *Git*-репозиторію.

2.4.3 Оновити *Git*-репозиторій змінами нової гілки «*Laboratory-work-5*» з *GitHub*-репозиторію.

2.4.4 Оновити *GitHub*-репозиторій змінами нової гілки «*Laboratory-work-5*» *Git*-репозиторію.

2.4.5 Виконати запит *Pull Request*.

Примітка: Увага! Не натискайте кнопку «Merge pull request»!

Це повинен зробити лише рецензент, який є вашим викладачем!

Коли рецензент-викладач перегляне ваше рішення він виконає злиття нової гілки та основної гілки, натиснувши кнопку «*Merge pull request*». Якщо рецензент знайде помилки, він повідомить про це у коментарях, які з'являться на сторінці *Pull request*.

2.5 Оцінка результатів виконання завдань лабораторної роботи

Оцінка	Умови
+3 бали	1) всі рішення відповідають завданням 2) <i>Pull Request</i> представлено не пізніше найближчої консультації після офіційного заняття із захисту лабораторної роботи
-0.5 балів за кожну помилку	в рішенні є помилка, про яку вказано в <i>Code Review</i>
-1 бал	<i>Pull Request</i> представлено пізніше дати найближчої консультації після офіційного заняття із захисту лабораторної роботи за кожний тиждень запізнення
+2 бали	1) ви увійшли до трійки перших з вашої групи, хто без помилок виконав усі завдання; 2) отримано правильну відповідь на два запитання, які стосуються призначення команд, представлених на знімках екранів