



Дисципліна «Операційні системи»

Лабораторна робота №7

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 аналізу компонент ОС у віртуальній файловій системі

та придбання навичок з керування правами доступу до файлової системи *Ext* в ОС *Linux*.

1 Теоретичні відомості

1.1 Типи файлових систем як коротенька історія розвитку ОС

Відомо, що назви більшості перших ОС для персональних комп'ютерів, наприклад, *Apple DOS*, *MS-DOS*, містили слово *Disk OS* – 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 (системами):

- файл – це набір даних, до якого можна звертатися за іменем;
- система керування файлами забезпечує виконання над файлами базових *CRUD*-операцій:
 - створення файлу (*Create*);
 - перегляд змісту файлу (*Read*);
 - зміна змісту файлу (*Update*);
 - видалення файлу (*Delete*).

Наведемо перелік типів 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*;
- *FAT (File Allocation Table)* – файлова система для *DOS*, яка містить таблицю розміщення файлів у вигляді ланцюжка кластерів (логічне розбиття диска у вигляді 1, 2, 4, 8, 16 та більше 512-байтних секторів) різного розміру, наприклад:

- *FAT8* містила $2^8 = 256$ елементів таблиці, коли max розмір диска (файлу) = 256

х 8Кб = 2 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* (альтернатива ОС *Windows* 2.0-3.0) з наступними властивостями:

○ диск ділиться на сектори розміром 512 байт з 4-х байтовим номером сектора та *max* розміром диску $2^{32} \cdot 512 = 2$ терабайти;

○ зайнятість секторів визначає бітова мапа (значення 1 – зайнято); ○ підтримка списків керування доступом (*Access Control List, ACL*);

□ *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* з наступними властивостями:

○ бітова мапа описує зайнятість кластерів на диску (2, 4, 6, ... секторів); ○ керування метаданими (дані про дані), які зберігаються у головній файловій таблиці (*Master File Table, MFT*) з рядками як файлами, а стовпцями як атрибутами файлів;

○ журналювання операцій з файлами для підвищення надійності;

□ *Ext* (*Extended File System*) – 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* – розвиток файлової системи *Ext2* із журналюванням операцій з файлами та *max* розміром файлу до 2^{40} байтів, розміром диску до 2^{45} байт; ○ *Ext4* – розвиток файлової системи *Ext3* з механізмом запису файлів у безперервні ділянки блоків (екстенти) для зменшення фрагментації (розміщення кластерів зі змістом одного файлу у різних частинах

диску) та підвищення продуктивності з *max* розміром файлу до 2^{44} байтів та розміром диску до 2^{60} байт.

Перші файлові системи були проскими, без ієрархії підпорядкування файлів.

Більшість сучасних файлових систем є ієрархічними – мають ієрархічну структуру у вигляді дерева:

- дерево починається з кореневого файлу – (*root* – вершини дерева), який для різних ОС може мати різні умовні позначення:

- для файлових систем *FAT* та *NTFS* це том з назвами-літерами *C:*, *D:*, *E:*, ... ○

- для файлових систем *NFS*, *Ext* назва – це символ слеш /

- якщо файл розглядається як сховище інших файлів, тоді він стає каталогом;

- файли повинні мати унікальну назву в межах одного каталогу.

Наявність системних структур опису файлів, наприклад, метаданих, вимагає додаткових системних операцій, які супроводжують *CRUD*-операції над файлом (*Create*, *Read*, *Update*, *Delete*), через дві фази:

- фаза виконання операцій;

- фаза оновлення системних структур, наприклад, маркування видалених блоків.

Якщо між цими фазами виникне збій в роботі ОС, тоді файлова система перейде у неузгоджений стан, у якому файлова система вже може працювати із помилками.

Відмовостійкість ОС – це властивість ОС системи, яка дозволяє їй продовжувати правильно працювати після збою. Тому для запобігання виникнення неузгодженого стану використовується механізм журналування який:

- зберігає інформацію про усі операції модифікації блоків на диску у спеціальному файлу-журналі (*log*-файл) черз виконання швидких операції додавання нових записів у кінець файлу;

- до файл-журналу додаються записи із описом операцій модифікації блоків перед їх виконання операцій модифікації блоків на диску та оновленням метаданих;

- у разі виникнення збою ОС виконується порівняння записів файл-журналу із описом метаданих та, за необхідністю, оновлення метаданих та відповідних блоків на диску.

Абстрагування роботи файлової системи формує декілька рівнів абстракцій:

- логічний (високий) рівень абстракції файлової системи описує інтерфейс програмування застосунків (*Application Program Interface*, *API*) для виконання *CRUID* операцій з файлами;

- фізичний (нижній) рівень абстракції файлової системи реалізується через:

- спеціальні програми-драйвери, які реалізують *API*, враховуючи фізичні особливості пристроїв зберігання;
- програми пришвидшення роботи через буферизацію даних в оперативній пам'яті та швидший пошук вільних блоків на диску;
- журналювання операцій модифікації блоків;
- віртуальний рівень (необ'язковий) абстракції файлової системи, який:
 - підтримує роботу з декількома типами файлових систем (*FAT32*, *NTFS*, *Ext*);
 - представляє роботу процесів ОС у вигляді файлів;

Перші файлові системи були локальними – забезпечували доступ до файлів з носіїв на локальному комп'ютері.

Мережеві файлові системи надають доступу до файлів комп'ютерів у комп'ютерній мережі.

Розподілені файлові системи забезпечують прозорість роботи з файлами, які розташовані на носіях різних комп'ютерів у комп'ютерній мережі.

1.2 Особливості файлової структури *Unix*-подібних ОС

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 1.

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

- системні утиліти – каталог */bin*
- системні програмні бібліотеки – каталог */lib*
- програмні бібліотеки користувачів – */usr/lib*
- конфігураційні файли – каталог */etc*
- каталоги користувачів – каталог */home*
- каталоги пристроїв – каталог */dev*, наприклад, термінали – */dev/tty*, каталоги псевдотерміналів – */dev/pts/*, розділ *hard*-диску – */dev/sda1*, *floppy*-диск – */dev/fd*, *CD-ROM* – */dev/cdrom*, файли *INPUT/OUTPUT*-потоків потоків */dev/stdin*, */dev/stdout*;
- каталог віртуальної файлової системи – каталог */proc*

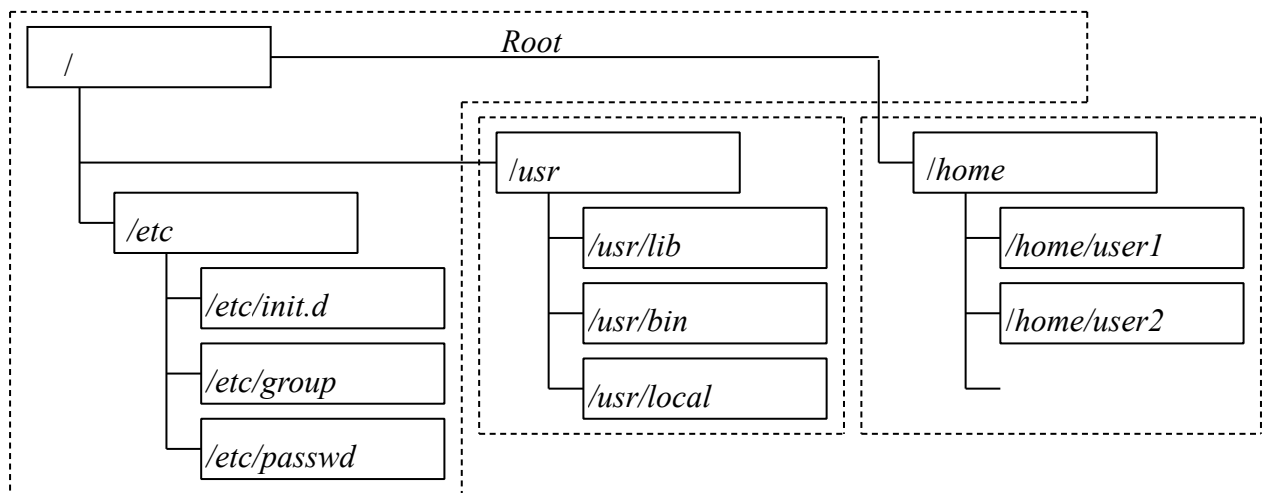


Рис. 1 – Фрагмент структури простору імен ієрархії файлової системи *Ext*

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.

Для перегляду дерева каталогів в ОС *Linux* є утиліта *tree*. Але у версії *Ubuntu 14* ця утиліта недоступна. Роботу утиліти можна швидко реалізувати через відомий конвеєрний ланцюжок трансформації команд *find*, *tr* та *sed*:

```
find . -type d | tr '/' ';' | \ sed -  
e 's/[^;]*;/|____/g;s/____|/|/g'
```

Запропонований ланцюжок виконує наступні дії:

- команда *find* надає перелік назв каталогів у поточному каталозі
- команда *tr* замінює символ `\` на `;` для того, щоб наступна команда *sed* могла легко його обробляти, бо символ `\` вже є мета-символом для команди *sed*;
- в команді *sed* послідовно виконує дві операції: ○ додавання до назв каталогів послідовності псевдосимволів імітації зв'язків між елементами дерева;
○ видалення зайвих псевдосимволів зв'язків.

1.2 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 файла *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 в межах *ssh*-протоколу існує утиліта *scp* (*Secure Copy*) – команда копіювання файлу *SourceFile*:

□ з локальної файлової системи до файлової системи віддаленої ОС:

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

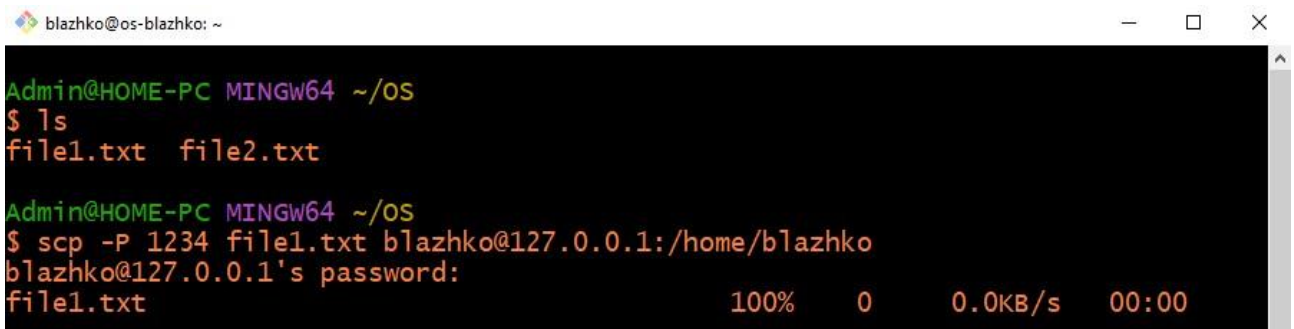
□

з файлової системи відделеної ОС до локальної файлової системи:

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

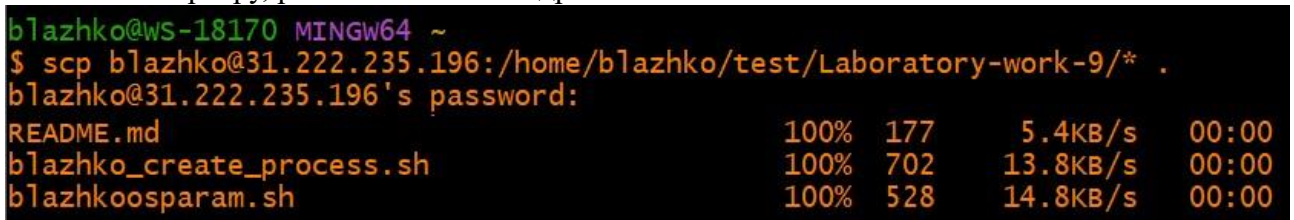
У вказаних командах назва файлу *SourceFile* це повний шлях до скопійованого файлу.

За замовчуванням команда *scp* використовує порт 22, тому, якщо, використовується інший порт, наприклад, 1234, тоді необхідно вказати його значення через прапорець *-P*, як показано на рисунку 2(a).



```
blazhko@os-blazhko: ~  
Admin@HOME-PC MINGW64 ~/OS  
$ ls  
file1.txt file2.txt  
Admin@HOME-PC MINGW64 ~/OS  
$ scp -P 1234 file1.txt blazhko@127.0.0.1:/home/blazhko  
blazhko@127.0.0.1's password:  
file1.txt 100% 0 0.0KB/s 00:00
```

(a) копіювання файлу *file1.txt* з локального каталогу до файлової системи віддаленого серверу, розташованого за адресою 127.0.0.1 з каталогом */home/blazhko*



```
blazhko@ws-18170 MINGW64 ~  
$ scp blazhko@31.222.235.196:/home/blazhko/test/Laboratory-work-9/* .  
blazhko@31.222.235.196's password:  
README.md 100% 177 5.4KB/s 00:00  
blazhko_create_process.sh 100% 702 13.8KB/s 00:00  
blazhkoosparam.sh 100% 528 14.8KB/s 00:00
```

(б) копіювання до локального каталогу всіх файлів каталогу */home/blazhko/test/Laboratorywork-9/* файлової системи віддаленого серверу, розташованого за адресою 31.222.235.196 Рис. 2 – Приклади копіювання файлів через утиліту *scp*

Щоб не вказувати шлях до домашнього каталогу користувача, можна використовувати його псевдонім як символ *~* (тільда).

1.3 Особливості опису файлів у файловій системі *EXT*

Файл типу каталог можна розглядати як список елементів з коротким описом файлів, які розміщено у каталозі та містять:

- назва файлу, яка необхідна користувачеві для подальшої роботи з файлом;
- посилання на детальний опис файлу, необхідний ОС для подальшої роботи з файлом.

Починаючи з першої файлової системи *Unix* детальний опис файлу має свій індексний дескриптор (*index node - inode*), який описує властивості файлу, наприклад:

- розмір файлу;
- ідентифікатор (ID) пристрою, де розташований файл;
- ID користувача-власника файлу, який зберігається в файлі */etc/passwd*;

□

□ ID групи користувачів власника, яка зберігається в файлі */etc/group*; тип файлу;

□ повноваження доступу, які надаються власнику, групі та всім іншим користувачам;

□ повноваження на читання, модифікацію і виконання для власника, групи та всіх інших користувачів;

□ час останнього доступу і зміни файлу;

□ лічильник кількості жорстких посилань на файл;

□ показники на блоки (кластери) диска, в яких розміщений файл;

□ ім'я каталогу або блочного пристрою, де розташований файл;

□ кількість блоків на диску, наприклад, кластерів, які займає файл.

Всі індексні дескриптори зберігаються у спеціальній області на диску – таблиці індексних дескрипторів (*Inode Table*).

Коли користувач намагається отримати доступ до файлу, тоді:

□ в файлі типу каталог шукається елемент з назвою файлу та 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 змінено атрибути файлу або його зміст;

□ створено (*Birth*) - зарезервовано для відображення початкової дати створення файлу.

На рисунку 3 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. 3 – Результат виконання команди `stat /etc/passwd`

Також значення *inode* можна отримати для файлів через команду `ls` із прапорцем `-li`:

```
ls -li
```

Вказана команда із вкащаним прапорцем виведе на екран список файлів, як він зберігається у файлі типу каталог.

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 4 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. 4 – Приклади результатів роботи команди для різних типів файлів

1.4 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 5 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. 5 – 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 файлової 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 символічного зв'язку встановлюються всі повноваження доступу; Для створення символічного зв'язку використовується команда *ln* з опцією *-s* :

ln -s початковий_файл файл-зв'язок

На рисунку 6 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. 6 – Приклад створення та аналізу символічного зв'язку

Також з рисунку 6 видно, що в другому стовпчику для символічного зв'язку з'явилася літера *l*, яка визначає особливий тип файлу, чого не було із жорстким зв'язком.

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

Для видалення файлів-зв'язків можна використати звичайну команду *rm*. Але є інша команда видалення з назвою, пов'язаною із посиланнями – *unlink*.

1.3 Керування повноваженнями доступу до файлів

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

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

- 1) повноваження користувача як власника файлу - того, хто його створив;
- 2) повноваження групи користувачів, які об'єднано з урахуванням якихось загальних

виконуваних ними задач або повноважень;

3) повноваження всіх інших користувачів як повноваження за замовчуванням.

Повноваження *власника* вище повноважень *групи*, а повноваження групи вище повноважень *за замовчуванням*.

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

Для визначення власника файлу та отримання переліку повноваження 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);
- ☐ повноваження 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 використовується всіма файлами в даному каталозі.

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

```
chgrp ім'я_власника-групи список_файлів
```

Наприклад, наступна команда встановить групу користувачів *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 повноваженнями доступу.

Unix-подібні ОС підтримують три типи повноваження доступу – повноваженнями виконувати операції:

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

Можна надати повноваження виконувати кожну 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 виконувати відповідну операцію позначається символом дефісу. В таблиці 2 наведено приклади коментарів до опису повноважень.

Таблиця 2 – Приклад коментарів до опису повноважень

Повноваження власникакористувача 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

Повноваження доступу можуть бути змінені тільки власником файлу або

адміністратором системи з використанням команди *chmod* з аргументами як показано на рисунку 7:

- 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 повноваженнями: r, w, x
- перелік файлів, для яких визначаються повноваження.

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

Рис. 7 – Аргументи команди *chmod*

Приклади використання команди *chmod* наведено у таблиці 3.

Таблиця 3 – Приклади використання команди *chmod*

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

Права визначаються бітами у масці повноважень доступу:

- '0' – відсутність відповідного повноваження;
- '1' – присутність відповідного повноваження;
- три біти визначають 3-бітне двійкове число;
- 3-бітне 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 символьного опису прав, числового двійкового та числового десятинного.

Таблиця 4 – Приклади символьного опису прав, числового двійкового та числового десяткового

Тип опису повноважень	Власник	Група користувачів	Інші користувачі (за замовчуванням)
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 8 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. 8 – Демонстрація результатів виконання команди зміни повноважень на файлі символічного зв'язку

Для швидкого перегляду інформації про каталоги та файли можна використати файловий менеджер *mc* із псевдографічним інтерфейсом. Для його інсталяції необхідно виконати команду:

```
sudo apt install mc
```

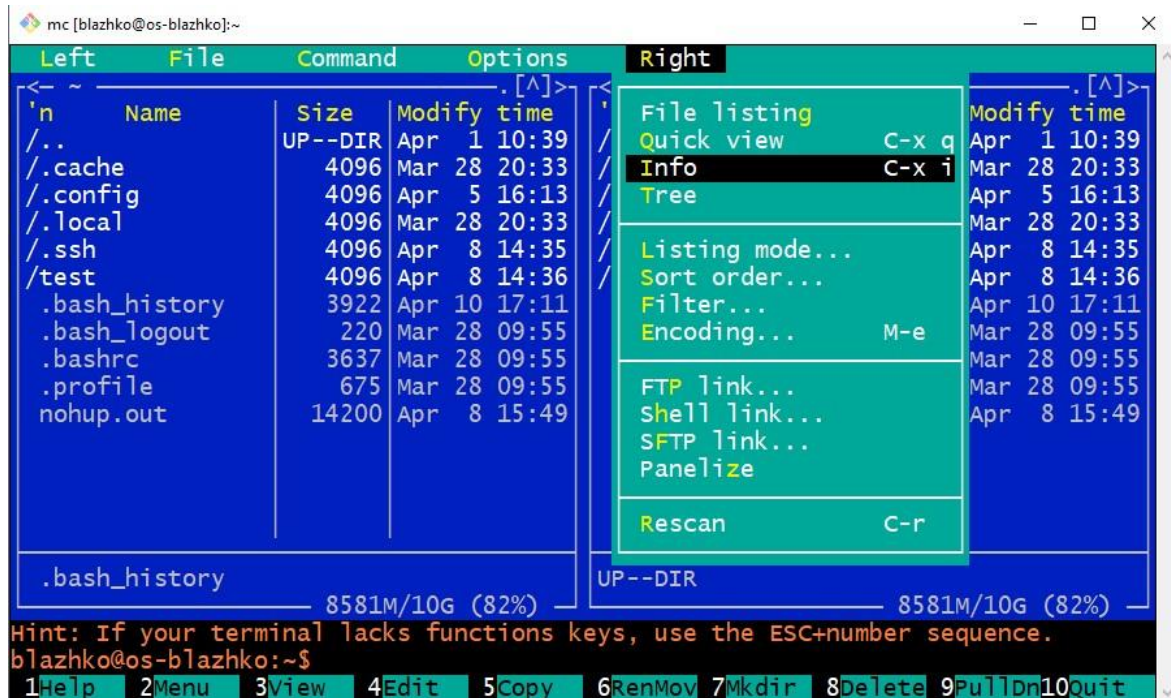
Файловий менеджер запускається командою *mc*, відповідно, надаючи доступ до команд у двох режимах:

□ через пункти меню:

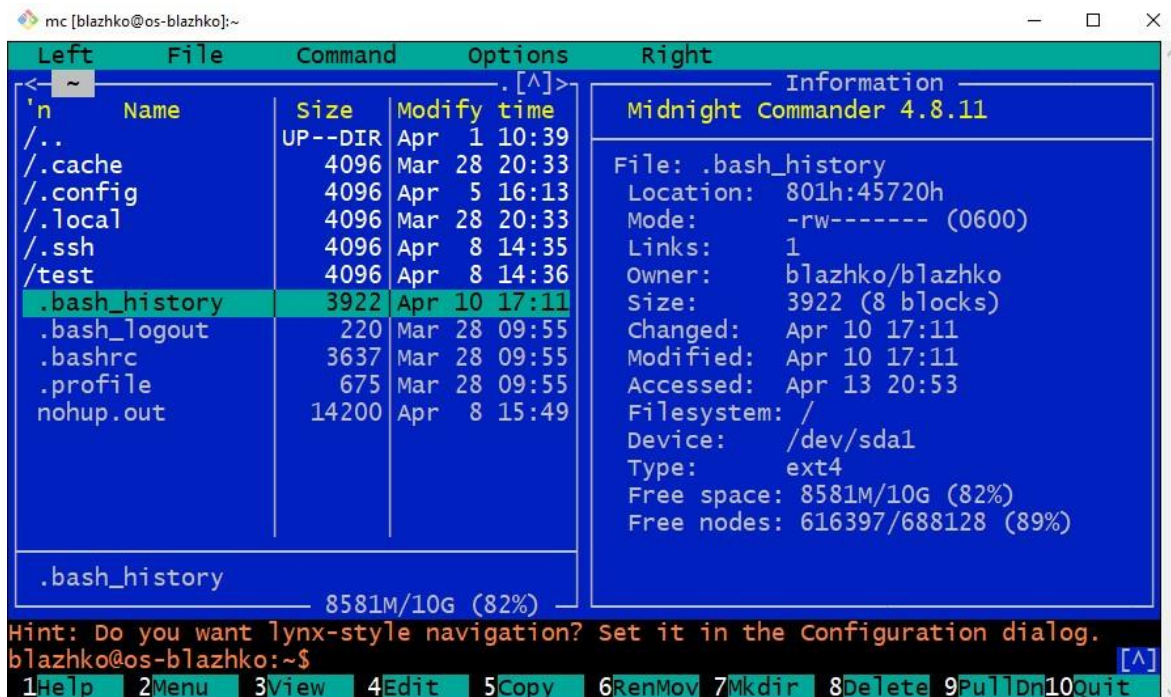
- нижній перелік швидких команд через клавіші *F1-F10*;
- верхнє дворівневе меню, доступ до якого виконується через клавішу *F9*;

□ через курсор миші.

На рисунку 9 наведено приклади екранних форм файлового менеджера.



(а) приклад екрану з доступом до верхнього меню (клавіша F9)



(б) приклад екрану, на якому у частині екрану справа надаються властивості обраного файлу у частині екрану зліва

Рис. 9 – Приклади екранних форм файлового менеджера.

1.4 Робота з «темним» каталогом» файлової системи *Ext*

Багато комбінацій повноваження rwx можуть працювати незалежно, наприклад, повноваження r та x працюють незалежно, повноваження x не потребують повноваження r для каталогу, і навпаки. Поєднуючи деякі повноваження, можна домогтися цікавих ефектів, наприклад, створення «темних» каталогів, файли яких доступні тільки в тому випадку, якщо користувач заздалегідь знає їх імена, оскільки отримання списку файлів в таких каталогах заборонено. Цей спосіб, до речі, використовується при створенні публічних архівів у мережі, коли деякі розділи архіву можуть використовувати лише «посвячені» користувачі, які знають про наявність того чи іншого файлу в каталозі.

Розглянемо наступний приклад.

Нехай існує файл-каталог *files*, який містить три звичайні файли *f1*, *f2*, *f3* із характеристиками, наведеними на рисунку 10.

```
[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. 10 – Фрагмент характеристик файлів, отриманий командами *ls -l*

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

Для відбирання повноважень на виконання файлу-каталогу *files* необхідно виконати команду:

```
chmod -x files
```

Після виконання команди повторне читання каталогу призведе до помилок, показаних на рисунку 11, 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. 11 – 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 *x* файлу-каталогу. Але відсутність повноважень на операцію читання *r* не дозволяє переглянути будь-яку характеристику про файл.

Як видно на рисунку 12, після відбирання повноважень на операцію читання *r* та надання повноважень на операцію виконання *x* вже неможливо переглянути зміст каталогу, але в нього ще можливо перейти через команду *cd*.

Наведена комбінація повноважень на файл-каталог, коли можна виконувати файл-каталог, але не можна читати з файлу-каталогу, призводить до ефекту створення «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. 12 – Приклад створення «темного» каталогу як результати роботи команд відбирання прав на читання та надання прав на виконання

На рисунку 13 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. 13 – 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 14 наведено результати роботи команд в каталозі після відібрання у користувача повноважень 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. 14 – Результати роботи команд в каталозі без прав на запис

1.5 Особливості віртуальної файлової системи

Раніше вже визначалося, що файлові системи *Unix*-подібних ОС активно використовують віртуалізацію. Наприклад віртуальна файлова система **Procfs** дозволяє отримати доступ до інформації з ядра ОС про системні процеси. Зазвичай вона розташовується у каталозі */proc*. *Procfs* створює дворівневе представлення опису процесів:

- на верхньому рівні процеси розглядаються як віртуальні каталоги, назви яких співпадають зі значенням *PID* процесів;
- на нижньому рівні процеси розглядаються як віртуальні файли з описом різних апаратних компонент комп'ютера та ОС, наприклад:
 - характеристики оперативної пам'яті - */proc/meminfo*
 - характеристики процесора - */proc/cpuinfo*
 - характеристики *swap*-файлу (файлу підкачки) - */proc/swaps*
 - характеристики ядра ОС - */proc/version*

1.5.1 Аналіз характеристик пам'яті у файлі */proc/meminfo*

Оперативна пам'ять представлена у файлі наступними характеристиками:

- *MemTotal* – доступний обсяг пам'яті, при цьому частина фізично доступної пам'яті резервується під час запуску ОС і не входить в зазначений тут обсяг;
- *MemFree* – невикористаний обсяг пам'яті, доступний для виділення процесам;
- *MemAvailable* – доступний обсяг пам'яті без врахування розділів або файлів підкачки;

□ *Buffers* – об'єм пам'яті, зайнятий під зберіганням даних, які очікують операції запису на диск, при цьому буфер дозволяє програмам продовжувати виконання свого завдання, не чекаючи моменту, коли дані будуть фізично записані на диск;

□ *Cached* – об'єм пам'яті, зайнятий під кеш читання сторінок з диску (файли, директорії, файли блокових пристроїв, дані, які стосуються механізму *IPC*, дані процесів рівня користувача, скинутих в область підкачки), але не включає в себе *SwapCached*;

□ *SwapCached* – об'єм пам'яті, який одного разу був розміщений в область підкачки *swap*, але потім перенесений назад у пам'ять, однак дані все ще присутні в *swap*, і при необхідності цей обсяг пам'яті може бути знову звільнений без витрачання ресурсів на операції введення/виводу.

□ *Active* – об'єм пам'яті, зайнятий найбільш часто використовуваними сторінками пам'яті, коли ці сторінки пам'яті активно використовуються процесами і звільнятимуться тільки в разі крайньої необхідності;

□ *SwapTotal* – загальний об'єм області підкачки (як в розділі підкачки, так і в окремих *swap*-файлах, якщо вони використовуються);

□ *SwapFree* – вільний об'єм в області підкачки (як в розділі підкачки, так і в окремих *swap*-файлах, якщо вони використовуються);

□ *VmallocTotal* – загальний розмір віртуальної пам'яті;

□ *Dirty* – розмір пам'яті, яка чекає на запис на *hard*-диск;

□ *PageTables* – розмір пам'яті, який виділено для таблиць сторінок.

На рисунку 15 представлено результат перегляду змісту файлу */proc/meminfo* зі всіма характеристиками пам'яті.

MemTotal:	1006936 kB	KernelStack:	5824 kB
MemFree:	72744 kB	PageTables:	23804 kB
MemAvailable:	455820 kB	NFS_Unstable:	0 kB
Buffers:	181184 kB	Bounce:	0 kB
Cached:	302056 kB	WritebackTmp:	0 kB
SwapCached:	3184 kB	CommitLimit:	2599592 kB
Active:	438388 kB	Committed_AS:	1558116 kB
Inactive:	227748 kB	VmallocTotal:	34359738367 kB
Active(anon):	110988 kB	VmallocUsed:	0 kB
Inactive(anon):	101156 kB	VmallocChunk:	0 kB
Active(file):	327400 kB	HardwareCorrupted:	0 kB
Inactive(file):	126592 kB	AnonHugePages:	28672 kB
Unevictable:	3652 kB	CmaTotal:	0 kB
Mlocked:	3652 kB	CmaFree:	0 kB
SwapTotal:	2096124 kB	HugePages_Total:	0
SwapFree:	1956808 kB	HugePages_Free:	0
Dirty:	124 kB	HugePages_Rsvd:	0
Writeback:	0 kB	HugePages_Surp:	0
AnonPages:	184548 kB	Hugepagesize:	2048 kB
Mapped:	50732 kB	DirectMap4k:	118720 kB
Shmem:	26824 kB	DirectMap2M:	929792 kB
Slab:	163204 kB		
SReclaimable:	112996 kB		
SUnreclaim:	50208 kB		

Рис. 15 – Приклад змісту файлу */proc/meminfo*

1.5.2 Аналіз характеристик процесора у файлі */proc/cpuinfo*

Характеристики процесора представлено такими важливими параметрами:

- *cpu family* – тип процесора, наприклад, 6; для архітектури x86 це число треба поставити перед «86», наприклад, 686, 586, 486 або 386;
- *model* – модель процесора в межах типу процесора, наприклад, 63;
- *model name* – текстова назва моделі процесора, наприклад, *Intel(R) Xeon(R)*;
- *cpu MHz* – швидкість процесора у мегагерцах, наприклад, 2599;
- *cache size* – розмір кешу пам'яті другого рівня, наприклад, 30720 KB;
- *cpu cores* – кількість ядер процесора, наприклад, 1;
- *cpuid level* – максимальний набір параметрів, який можна безпечно використовувати для запиту інформації в процесора архітектури x86;
- *bogomips* – тестова швидкість процесора, яка визначається під час завантаження ОС *Linux* з метою відкалібрування внутрішньої затримки під виконання холостого циклу, але не може бути використана для порівняння потужності різних процесорів.

На рисунку 16 представлено результат перегляду змісту файлу */proc/cpuinfo* зі всіма характеристиками процесора.


```

processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 62
model name    : Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz
stepping      : 4
microcode     : 0x1
cpu MHz       : 2399.998
cache size    : 30720 KB
physical id   : 0
siblings      : 1
core id       : 0
cpu cores     : 1
apicid        : 0
initial apicid : 0
fpu           : yes
fpu_exception : yes
cpu_id level  : 13
wp            : yes
flags         : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
m constant_tsc arch_perfmon rep_good nopl xtopology eagerfpu pni pclmulqdq
e avx f16c rdrand hypervisor lahf_lm ssbd ibrs ibpb stibp fsgsbase tsc_adju
bogomips      : 4799.99
clflush size  : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

```

Рис. 16 – Приклад змісту файлу */proc/cpuinfo*

1.5.3 Аналіз характеристик *swap*-файлу в файлі */proc/swaps*

Стан файлу підкачки представлено наступними важливими параметрами:

- *Filename* – шлях до файлу підкачки;
- *Size* – загальний розмір у байтах;
- *Used* – загальний розмір використаної пам'яті.

На рисунку 17 представлено приклад змісту файлу */proc/swaps*

Filename	Type	Size	Used	Priority
/swapfile	file	4194300	1282048	-2

Рис. 17 – Приклад змісту файлу */proc/swaps*

1.5.4 Аналіз характеристик ядра ОС у файлі */proc/version*

Характеристики встановленої ОС *Linux* представлено параметрами:

- *Linux version* - версія ядра;
- *gcc version* - версія компілятора мови програмування C.

На рисунку 18 представлено приклад змісту файлу */proc/version*

```

Linux version 3.10.0-1127.19.1.el7.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 4.8.5 20150623
(Red Hat 4.8.5-39) (GCC) ) #1 SMP Tue Aug 25 17:23:54 UTC 2020

```

Рис. 18 – Приклад змісту файлу */proc/version*

1.6 Первинний аналіз файлової системи Unix-подібних ОС

Для первинного аналізу файлової системи Unix-подібних ОС можна використати утиліта *df* (*disk free*):

```
df [прапорці] [каталог]
```

Популярні прапорці:

-*T* – інформація про тип файлової системи;

-*i* – інформація про розміри файлових систем не у байтах, а у кількості *inode*.

На рисунку 19 наведено приклади виконання утиліти *df*.

```
blazhko@os-blazhko:~$ df -T
Filesystem      Type      1k-blocks    Used Available Use% Mounted on
udev            devtmpfs   1018996        4    1018992   1% /dev
tmpfs           tmpfs      206252        944    205308   1% /run
/dev/sda1       ext4      10705168 1571364    8566972  16% /
none            tmpfs         4            0         4   0% /sys/fs/cgroup
none            tmpfs        5120          0        5120   0% /run/lock
none            tmpfs     1031240          0    1031240   0% /run/shm
none            tmpfs     102400          0    102400   0% /run/user
```

(а) Приклад виконання утиліти з додатковим стовпчиком *Type* – тип файлової сичтеми

```
blazhko@os-blazhko:~$ df /home/blazhko/
Filesystem      1k-blocks    Used Available Use% Mounted on
/dev/sda1       10705168 1386024    8752312  14% /
```

(б) Приклад виконання утиліти з інформацією про один зазначений каталог

```
blazhko@os-blazhko:~$ df -i
Filesystem      Inodes IUsed IFree IUse% Mounted on
udev            213576    444 213132    1% /dev
tmpfs           219699    423 219276    1% /run
/dev/sda1       688128 71758 616370   11% /
none            219699      3 219696    1% /sys/fs/cgroup
none            219699      1 219698    1% /run/lock
none            219699      1 219698    1% /run/shm
none            219699      2 219697    1% /run/user
```

(в) Приклад виконання утиліти з інформацією про розмір файлової системи у кількості *inode*

Рис.19 – Приклади виконання утиліти *df*

1.7 Керування обмеженням використання ресурсів ОС

Якщо сервер з *ОС Linux* використовується багатьма користувачами, тоді можливі ситуації, коли програми одного користувача помилково почнуть активно використовувати ресурси сервера, наприклад, оперативну пам'ять або потужність процесора, обмежуючи роботу інших користувачів. Тому важливо контролювати всі процеси, обмежуючи дії користувачів. Таке керування може бути застосовано на трьох рівнях:

- глобальний рівень – для всіх користувачів одночасно;
- груповий рівень – для окремих груп користувачів;
- користувацький рівень – для окремих користувачів.

Для перегляду значення поточних обмежень та для тимчасової зміни цих значень використовується команда:

```
ulimit [options] [user-name]
```

Опція *user-name* визначає ім'я користувача, для якого описується обмеження. Якщо *user-name* не визначено, тоді параметри визначаються для поточного користувача.

Деякі приклади значень для *options*:

- *a* – отримати значення всіх параметрів;

- f – максимальний розмір файлу (Кб);
- n – максимальна кількість відкритих файлових дескрипторів;
- t – максимальний кількість хвилин використання процесора;
- u – максимальна кількість процесів;
- v – максимальний об'єм віртуальної пам'яті (Кб).

На рисунку 20 наведено результат виконання команди `ulimit -a`

```
blazhko@os-blazhko:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 15921
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 15921
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

Рис. 20 – Результат виконання команди `ulimit -a`

В таблиці 1 наведено приклади команд `ulimit`.

Таблиця 5 – Приклади команд `ulimit`

№	Приклад команди	Приклад операції, яка порушує обмеження	Приклад повідомлення про помилку щодо порушення обмеження
1	<code>ulimit -f 10</code>	<code>cp /bin/bash .</code>	<i>File size limit exceeded (core dumped)</i>
2	<code>ulimit -n 2</code>	<code>ls</code>	<i>-bash: start_pipeline: pgrp pipe: Too many open files</i>
3	<code>ulimit -u 1</code>	<code>ls</code>	<i>-bash: fork: retry: No child processes</i>
4	<code>ulimit -v 1</code>	<code>ls</code>	<i>Segmentation fault</i>

На рисунку 21 наведено приклади виконання трьох команд встановлення обмежень, приклади операції, які порушують обмеження та приклад повідомлень про помилку щодо порушення обмежень.

```

[blazhko@vps6iefe ~]$ ulimit -f 10
[blazhko@vps6iefe ~]$ cp /bin/bash .
File size limit exceeded (core dumped)
[blazhko@vps6iefe ~]$
[blazhko@vps6iefe ~]$ ulimit -n 2
[blazhko@vps6iefe ~]$ ls
-bash: start_pipeline: pgrp pipe: Too many open files
ls: error while loading shared libraries: libselinux.so.1: cannot open s
hared object file: Error 24
[blazhko@vps6iefe ~]$ ulimit -t 1
[blazhko@vps6iefe ~]$ while [ true ]; do (( x++ )); done
Connection to 46.175.148.116 closed.

```

Рис. 21 – Приклади виконання трьох команд встановлення обмежень

2 Завдання до виконання

2.1 Налаштування *Git*-репозиторію та *GitHub*-репозиторію

Для забезпечення звітності з рішень лабораторної роботи необхідно виконати наступні дії:

- 1) встановити з'єднання з віртуальною ОС, яка використовувалася у попередній лабораторній роботі;
- 2) перейти до каталогу *Git*-репозиторія;
- 3) створити нову *Git*-гілку з назвою «*Laboratory-work-7*»;
- 4) перейти до роботи зі створеною гілкою;
- 5) створити каталог з назвою «*Laboratory-work-7*»;
- 6) перейти до каталогу «*Laboratory-work-7*»;
- 7) в каталозі «*Laboratory-work-7*» 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;
- 8) виконати операції з оновлення *GitHub*-репозиторію змінами *Git*-репозиторія через послідовність *Git*-команд *add*, *commit* із коментарем «*Changed by Local Git*» та *push*;
- 9) на веб-сервісі *GitHub* перейти до створеної гілки «*Laboratory-work-7*»;
- 10) перейти до каталогу «*Laboratory-work7*» та розпочати процес редагування файлу *README.md* ;
- 11) в подальшому за результатами рішень кожного наступного розділу завдань до файлу *README.md* додавати рядки як заголовки 3-го рівня *Markdown*-форматування з назвами розділу, а також знімки екрані та підписи до знімків екранів з описом пунктів завдань.

2.2 Аналіз типів файлів

2.2.1 Запустити на локальному комп'ютері ще один термінал *Git Bash* та перейти до *Git*-репозиторію із файлами *WebAR*-застосунку лабораторної роботи №2.

2.2.2 Скопіювати 5-ть файлів на віддалений *Linux*-сервер до каталогу «*Laboratorywork-7*» *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.

2.2.3 Знаходячись на віддаленому *Linux*-сервері, перейти до каталогу «*Laboratorywork-7*», в якому вже знаходяться 5-ть скопійованих раніше файлів.

2.2.4 Для кожного із вказаних файлів виконайте команду визначення їх типів.

2.2.5 Для кожного із вказаних файлів виконайте команду перегляду мета-опису файлу у вигляді дескриптору файлу.

2.2.6 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*

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

2.2.8 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

Знаходячись у каталозі «*Laboratory-work-7*» на віддаленому *Linux*-сервері, виконати наступні завдання.

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

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

- ☐ варіант взяти зі стовпчика «Повноваження доступу 1» таблиці 6;
- ☐ у таблиці вказано лише повноваження, які необхідно встановити, та не вказано повноваження, які необхідно зняти;

□ тип файлу не повинен протирічити визначеним повноваженням, наприклад, якщо використовується звичайний текстовий файл, тоді він не може мати повноваження на виконання;

2.3.3 Перевірити обмеження повноважень 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 повноваження доступу до файлу з назвою *транслітація_вашого_прізвища + hard_link_2*:

□ варіант взяти зі стовпчика «Повноваження доступу 2» таблиці 6;
□ у таблиці вказано лише повноваження, які необхідно встановити, та не вказано повноваження, які необхідно зняти;

□ тип файлу не повинен протирічити визначеним правам, наприклад, якщо використовується звичайний текстовий файл, тоді він не може мати повноваження на виконання;

2.3.5 Перевірити обмеження повноважень доступу до файлу з попереднього пункту завдання, виконавши відповідні команди роботи з файлом, наприклад, якщо заборонено читання, тоді переглянути файлу 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 повноваженнями 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 повноважень 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 6 – Варіанти завдань для повноважень доступу

№ варіанта	Повноваження доступу 1			Повноваження 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 Огляд віртуальної файлової системи

2.4.1 Виконати *sed*-команду або ланцюжок конвеєра команд *grep* та *sed*, який :

- отримує рядок з файлу віртуальної файлової системи *Procfs* у відповідності із варіантом, представленим у стовпчику «Параметр пам'яті або процесору» таблиці 7;
- виводить на екран коротке україномовне повідомлення з урахуванням параметру, наприклад, для параметру *MemTotal* виводиться повідомлення «Доступний обсяг пам'яті»;
- виводить на екран значення параметру після україномовного повідомлення.

Таблиця 7 – Варіанти завдань для визначення параметрів пам'яті або процесору

№ варіанту	Параметр пам'яті/процесору	№ варіанту	Параметр пам'яті/процесору
1	<i>MemTotal</i>	16	<i>cache size</i>
2	<i>MemFree</i>	17	<i>cpu cores</i>
3	<i>MemAvailable</i>	18	<i>Dirty</i>
4	<i>Buffers</i>	19	<i>bogomips</i>
5	<i>Cached</i>	20	<i>PageTables</i>
6	<i>SwapCached</i>	21	<i>MemTotal</i>
7	<i>Active</i>	22	<i>MemFree</i>
8	<i>SwapTotal</i>	23	<i>MemAvailable</i>
9	<i>SwapFree</i>	24	<i>Buffers</i>
10	<i>VmallocTotal</i>	25	<i>Cached</i>
11	<i>vendor_id</i>	26	<i>SwapCached</i>
12	<i>cpu family</i>	27	<i>Active</i>

13	<i>model</i>	28	<i>SwapTotal</i>
14	<i>model name</i>	29	<i>SwapFree</i>
15	<i>cpu MHz</i>		

2.5 Керування обмеженням використання ресурсів ОС

2.5.1 Встановити з'єднання із віртуальною машиною, яка створена у розділі 2.1., через *SSH*-команду від імені користувача, створеного під час інсталяції віртуальної ОС.

2.5.2 Перевірити роботу команди *ulimit* за всіма прикладами з таблиці 5 (підрозділ 1.7), показавши на екрані:

- ☐ приклад кожної команди;
- ☐ приклад операції, яка порушує обмеження;
- ☐ приклад повідомлення про помилку щодо порушення обмеження.

Після виконання кожного приклада рекомендується закрити псевдотермінал та повторно з'єднатися із віртуальною ОС.

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

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

2.6.2 Оновити *Git*-репозиторій змінами нової гілки «*Laboratory-work-7*» з *GitHub* репозиторію.

2.6.3 Оновити *GitHub*-репозиторій змінами нової гілки «*Laboratory-work-7*» *Git* репозиторію, враховуючи файли, які було створено у попередніх розділах.

2.6.4 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*.

2.7 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

Контрольні запитання

1. Яку структуру має файлова система *Unix*-подібних ОС ?
2. Чим файл-каталог відрізняється від звичайних файлів ?
3. Чим команда *scr* відрізняється від команди *cr* ?
4. Що таке таблиця індексних дескрипторів ?
5. Що міститься у першому стовпчику таблиці, яка є результатом виконання команди *ls -i* ?
6. Для чого використовується жорсткий зв'язок файлу ?
7. Для чого використовується м'який зв'язок файлу ?
8. Які недоліки жорстких зв'язків файлів у порівнянні із м'якими зв'язками ?
9. На що вказують перша літера *d* та літера *l* у першому стовпчику таблиці, отриманому як результат команди *ls -i* ?
10. Що таке *ACL* ?
11. Чим повноваження власника відрізняються від повноважень групи ?
12. Чим повноваження групи відрізняються від повноважень за замовчуванням ?
13. Розшифруйте наявність літер *r*, *w*, *x* в описі повноважень файлу ?
14. Чим тип опису повноважень
15. Чим символічний тип опису повноважень файлу відрізняється від числового-двійкового ?
16. Чим числовий-двійковий тип опису повноважень файлу відрізняється від числового-десятькового ?
17. Що таке «темний каталог» ?
18. Що дозволяє віртуальна файлова система ?
19. Наведіть два приклади ресурсів, на використання яких можна встановити обмеження ?

