Лабораторна робота 3

Тема: Файлова система. Процеси та керування ними.

Мета: Навчитись працювати з файлами та процесами.

1. Файлова система Linux

Файлова система Linux — це структурована колекція файлів на диску або розділі. Розділ - це сегмент пам'яті і містить певні дані. У нашій машині можуть бути різні розділи пам'яті. Як правило, кожен розділ містить файлову систему.

Комп’ютерна система загального призначення повинна систематично зберігати дані, щоб ми могли легко отримати доступ до файлів за менший час. Він зберігає дані на жорстких дисках (HDD) або на якомусь еквівалентному типі зберігання. Нижче наведено причини підтримки файлової системи:

В першу чергу комп'ютер зберігає дані в сховище RAM; він може втратити дані, якщо його вимкнути. Однак існує енергонезалежна оперативна пам’ять (Flash RAM і SSD), яка доступна для підтримки даних після перерви живлення.

Зберігання даних є перевагою на жорстких дисках порівняно зі стандартною RAM, оскільки RAM коштує більше, ніж місце на диску. Витрати на жорсткі диски поступово падають порівняно з RAM.

Файлова система Linux містить такі розділи:

- Кореневий каталог (/)
- Певний формат зберігання даних (EXT3, EXT4, BTRFS, XFS і так далі)
- Розділ або логічний том із певною файловою системою.

Файлова система Linux має ієрархічну файлову структуру, оскільки містить кореневий каталог та його підкаталоги. До всіх інших каталогів можна отримати доступ з кореневого каталогу. Розділ зазвичай має лише одну файлову систему, але він може мати більше однієї файлової системи.

Файлова система розроблена таким чином, щоб вона могла керувати та забезпечувати простір для енергонезалежних даних. Усі файлові системи вимагали простору імен, який є методологією імен і організаційною методологією. Простір імен визначає процес іменування, довжину імені файлу або підмножину символів, які можна використовувати для імені файлу. Він також визначає логічну структуру файлів у сегменті пам'яті, наприклад використання каталогів для організації конкретних файлів. Після опису простору імен необхідно визначити опис метаданих для цього конкретного файлу.

Структура даних повинна підтримувати ієрархічну структуру каталогів; ця структура використовується для опису доступного та використовуваного дискового простору для конкретного блоку. Він також містить інші відомості про файли, такі як розмір файлу, дата і час створення, оновлення та останньої зміни.

Крім того, він зберігає додаткову інформацію про розділ диска, таку як розділи та томи.

Розширені дані та структури, які вони представляють, містять інформацію про файлову систему, що зберігається на диску; він відрізняється і не залежить від метаданих файлової системи.

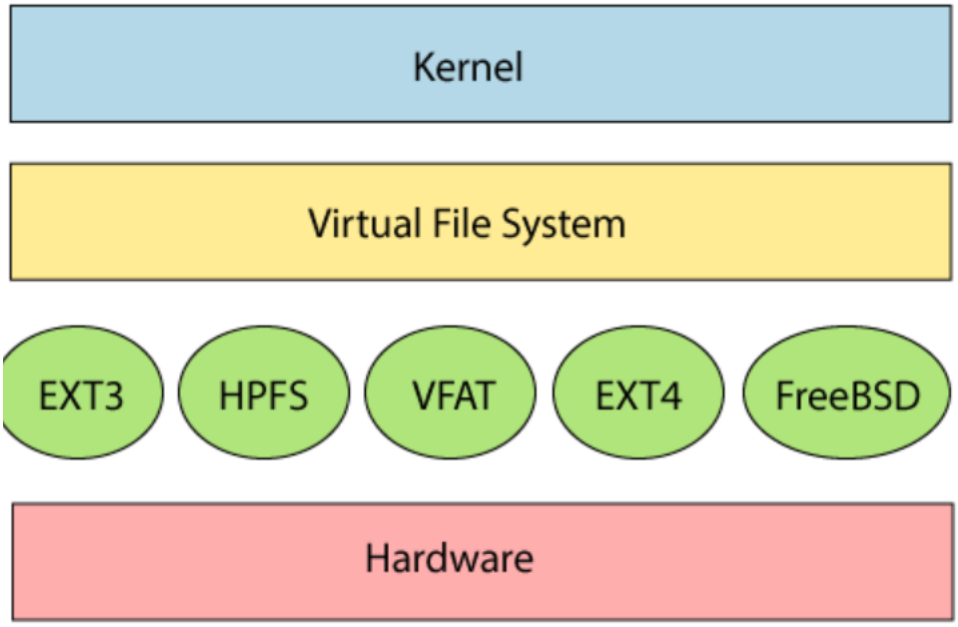


Рис.1

Файлова система вимагає API (інтерфейс програмного забезпечення) для доступу до викликів функцій для взаємодії з компонентами файлової системи, такими як файли та каталоги. API полегшує виконання таких завдань, як створення, видалення та копіювання файлів. Це полегшує алгоритм, який визначає розташування файлів у файловій системі.

Перші дві частини даної файлової системи разом називають віртуальною файловою системою Linux . Він надає єдиний набір команд для ядра та розробників для доступу до файлової системи. Ця віртуальна файлова система потребує спеціального системного драйвера для надання інтерфейсу файловій системі.

У Linux файлова система створює деревовидну структуру. Усі файли впорядковані у вигляді дерева та його гілок. Найвищий каталог називається кореневим каталогом (/) . До всіх інших каталогів у Linux можна отримати доступ з кореневого каталогу.

Ось деякі ключові особливості файлової системи Linux:

- Зазначення шляхів: Linux не використовує зворотну косу риску (\) для розділення компонентів; він використовує пряму косу риску (/) як альтернативу. Наприклад, як і в Windows, дані можуть зберігатися в C:\ My Documents\ Work, тоді як у Linux вони будуть зберігатися в /home/ My Document/ Work.
- Розділи, каталоги та диски: Linux не використовує літери дисків для організації диска, як це робить Windows. У Linux ми не можемо визначити, чи ми звертаємося до розділу, мережевого пристрою чи «звичайного» каталогу та диска.
- Чутливість до регістру: файлова система Linux чутлива до регістру. Він розрізняє імена файлів у нижньому та верхньому регістрі. Наприклад, існує різниця між test.txt і Test.txt в Linux. Це правило також застосовується до каталогів і команд Linux.
- Розширення файлів: у Linux файл може мати розширення '.txt', але необов'язково, щоб файл мав розширення. Під час роботи з Shell це створює деякі проблеми для новачків, щоб розрізняти файли та каталоги. Якщо ми використовуємо графічний файловий менеджер, він символізує файли та папки.
- Приховані файли: Linux розрізняє стандартні файли та приховані файли, в основному файли конфігурації приховані в ОС Linux. Зазвичай нам не потрібно відкривати або читати приховані файли. Приховані файли в Linux представлені крапкою (.) перед іменем файлу (наприклад, .ignore). Щоб отримати доступ до файлів, нам потрібно змінити вигляд у файловому менеджері або використовувати певну команду в оболонці.

2. Взаємодія з файловою системою за допомогою терміналу

- **Створення файлу**

```
spongebob@spongebob-VirtualBox:~$ touch filebytouch.txt
spongebob@spongebob-VirtualBox:~$ echo > filebyecho.txt
spongebob@spongebob-VirtualBox:~$ echo "some text" > filewithtext.txt
spongebob@spongebob-VirtualBox:~$ nano filebynano.txt
```

Рис.2

Для створення файлу є три основні способи:

1) Використання команди *touch*

Синтаксис: *touch filename* OR *touch /pathtofile/filename*

2) Використання переадресації STDOUT (standart output)

Для цього можна використати команду *echo* та «перенаправити» фактично ніщо, тобто ми переадресовуємо пустий рядок у файл. За рахунок того, що

файлу filebyecho.txt до цього не існувало, його буде створено.

! рекомендується вказувати тип файлу при його створенні. Системі це ніяк не допоможе, вона самостійно визначає тип, проте користувачам це полегшує роботу з системою.

Тим же способом ми можемо одночасно створити файл та записати щось у нього, як на рис.2: *echo "some text" > filewithtext.txt*

3) Використання редактора nano

Останній з представлених тут способів – із застосуванням редактора nano.

Проте, цей спосіб варто використовувати лише коли ви хочете щось почати записувати одразу після створення файлу. У іншому випадку цей спосіб ірраціональний – пустим файл просто не створиться.

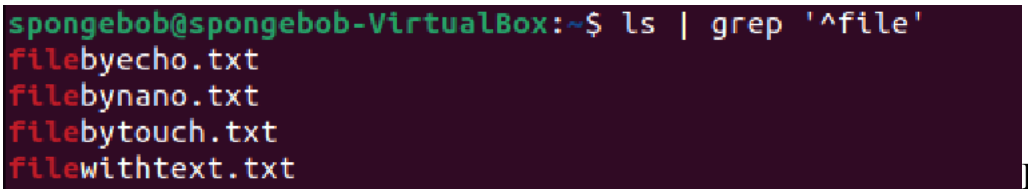


Рис.3

Останнім, на рис.3 зроблено перевірку наявності всіх чотирьох файлів.

Тут застосовано piping та wildcards. Познайомимось з ними ближче:

- **Wildcards**

Wildcards – це символи, що полегшують роботу з текстом, файлами і т.д.

- ^

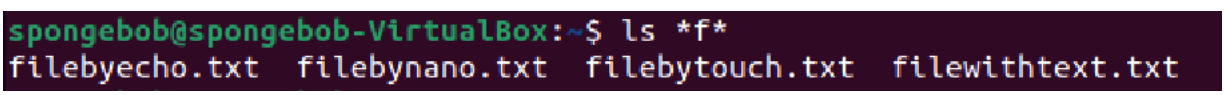
Наприклад, на рисунку 3 застосовано один із таких символів ^

Він дозволяє вказати, що перед словом file не повинно нічого бути, тобто назва файлів повинна починатись з цього слова. Результат виводу підтверджує це твердження.

- *

Наступним знаком буде *

Цей символ дозволяє замінити його будь-якими символами, а також будь-якою їх кількістю в результаті пошуку. Спробуємо вивести всі файли, що містять в собі букву f:



- ?

Знак питання має ту ж функцію що й *, проте на його місці може стояти будь-який ОДИН символ:

```
spongebob@spongebob-VirtualBox:~$ touch ab abc abcd
spongebob@spongebob-VirtualBox:~$ ls
ab    Desktop    filebyecho.txt   filewithtext.txt  Public     Videos
abc   Documents  filebynano.txt   Music             snap
abcd  Downloads  filebytouch.txt  Pictures          Templates
spongebob@spongebob-VirtualBox:~$ ls a?
ab
```

- [a-b]

Ми також можемо використовувати уточнення частини символів, що мають бути в результатах:

```
spongebob@spongebob-VirtualBox:~$ touch a b c d e f
spongebob@spongebob-VirtualBox:~$ ls
a     b        Documents  filebyecho.txt    Music     Templates
ab    c        Downloads  filebynano.txt    Pictures  Videos
abc   d        e          filebytouch.txt   Public
abcd  Desktop  f          filewithtext.txt  snap
spongebob@spongebob-VirtualBox:~$ ls [a-c]
a  b  c
```

```
spongebob@spongebob-VirtualBox:~$ touch 1 2 3 4 5
spongebob@spongebob-VirtualBox:~$ ls
1  5    abcd  Desktop    f                filewithtext.txt  snap
2  a    b     Documents  filebyecho.txt   Music             Templates
3  ab   c     Downloads  filebynano.txt   Pictures          Videos
4  abc  d     e          filebytouch.txt  Public
spongebob@spongebob-VirtualBox:~$ ls [3-5]
3  4  5
```

```
spongebob@spongebob-VirtualBox:~$ ls [1-2,4-5]
1  2  4  5
```

- **Керування виконанням команд**
- Незалежне виконання команд

Для незалежного виконання команд можна або прописувати кожну окремо, щоразу натискаючи Enter, або ж використовуючи крапку з комою:
touch file.txt; mv file.txt Desktop/; ls; echo "----------"; ls Desktop/

```
spongebob@spongebob-VirtualBox:~$ touch file.txt; mv file.txt Desktop/; ls ; ec
ho "----------" ; ls Desktop/
Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates
----------
file.txt
```

Проаналізуємо цей рядок:

Спочатку ми створюємо файл file.txt вже відомим нам методом

Далі ми його переносимо за допомогою команди mv у папку робочого столу

Опісля відображаємо вміст директорії, в якій ми зараз знаходимось, тобто домашній директорії користувача

Потім виводимо пунктирну лінію(це зроблено лише для зручності, смислового навантаження окрім читабельності виводу нема)

Останньою ми виконуємо команду виводу вмісту папки робочого столу, щоб показати, що файл перенесено

Всі згадані команди будуть розглянуті трішки пізніше.

- Залежне виконання команд

Для цього використовується згаданий раніше пайпінг/логічне «або» - |

З його допомогою вивід попередньої команди передається в наступну, за умови успішного виконання попередньої:

```
spongebob@spongebob-VirtualBox:~$ touch file.txt | mv file.txt Documents/
spongebob@spongebob-VirtualBox:~$ touchh file.txt | mv file.txt Documents/
Command 'touchh' not found, did you mean:
  command 'touch' from deb coreutils (8.32-4ubuntu2)
Try: sudo apt install <deb name>
```

Як бачимо, перша команда виконалась без помилок у виводі, навідміну від другої, де першу команду було прописано неправильно.

При використанні незалежного виконання помилка б не завадила виконанню наступної

Є ще один спосіб створення залежності команд, уже згаданий раніше – переадресація. Наведу ще один її приклад:

```
spongebob@spongebob-VirtualBox:~$ echo text > file.txt
spongebob@spongebob-VirtualBox:~$ cat file.txt
text
spongebob@spongebob-VirtualBox:~$ cat file.txt > newfile.txt
spongebob@spongebob-VirtualBox:~$ cat newfile.txt
text
```

Проаналізуємо:

Було створено файл file.txt з одночасним записом у нього слова text

Потім вміст файлу виведено в термінал за допомогою команди cat

Потім цей же вивід переадресовано у новий файл, що доводить вивід вмісту нового файлу у термінал

- **Стандартні потоки**

У лінуксі є три основних потоки: STDIN(0), STOUT(1), STDERR(2)

STDOUT – дефолтний для терміналу

Тому коли ми прописували команду *echo > file.tx*, ми переадресовували якраз потік виводу з номером 1, тобто з тим же успіхом можна було прописати *echo 1> file.tx*

Потік 0 – потік вводу, тобто потік інформації, що надходить від користувача, наприклад

Потік 2 – потік помилок, тобто при виконанні якоїсь команди, ми можемо зберігати помилки до окремого фалу, щоб потім їх, наприклад проаналізувати:

```
spongebob@spongebob-VirtualBox:~$ cat nonexistingfile.txt 2> errors.txt
spongebob@spongebob-VirtualBox:~$ cat errors.txt
cat: nonexistingfile.txt: No such file or directory
spongebob@spongebob-VirtualBox:~$ cat nonexistingfile.txt
cat: nonexistingfile.txt: No such file or directory
```

В цьому виводі ми спробували вивести вміст неіснуючого файлу та переадресували помилки у файл errors.txt

Далі ми вивели вміст файлу помилок та спробували вивести вміст неіснуючого файлу на екран, щоб показати, що помилка та ж і перенаправлення потоку помилок спрацювало успішно.

- **Навігація файловою системою**
- **pwd**

Для того, щоб орієнтуватись в системі, необхідно в першу чергу знати, де ви знаходитесь зараз

По-перше ваше розташування завжди вказане справа від імені домену(spongebob-VirtualBox).

```
@spongebob-VirtualBox:~$
```

Протягом трьох лабораторних всі команди були опрацьовані в домашній директорії юзера, вказаного зліва від @:

```
spongebob@s
```

Ця синя хвилька ~ якраз і є позначенням домашньої директорії, проте у неї є і повний шлях. Для того, щоб побачити його, скористаємось командою *pwd*(**P**rint **W**orking **D**irectory):

```
spongebob@spongebob-VirtualBox:~$ pwd
/home/spongebob
```

- **cd**

Для того, щоб «блукати» по папкам, потрібно скористатись командою *cd*(**c**hange **d**irectory)

Для переходу в іншу папку потрібно прописати:

```
spongebob@spongebob-VirtualBox:~$ cd Desktop/
spongebob@spongebob-VirtualBox:~/Desktop$
```

Як бачимо і після домену розташування змінилось.

Для повернення на одну папку вище:

```
spongebob@spongebob-VirtualBox:~/Desktop$ cd ..
spongebob@spongebob-VirtualBox:~$
```

Тепер ми знову в домашній папці

Для повернення на декілька папок вище:

Припустимо, що ми набагато глибше у файловій системі. Наприклад, у папці /etc/apt/

```
spongebob@spongebob-VirtualBox:/tmp$ cd /etc/apt/
spongebob@spongebob-VirtualBox:/etc/apt$ cd ../../
spongebob@spongebob-VirtualBox:/$
```

Також цим способом можна користуватись для переходу до іншої папки:

```
spongebob@spongebob-VirtualBox:/$ cd /etc/apt/
spongebob@spongebob-VirtualBox:/etc/apt$ cd ../../tmp/
spongebob@spongebob-VirtualBox:/tmp$
```

Отже, думаю, ви вже здогадались, що горизонтальна двокрапка означає вихід на рівень вище.

Для повернення в домашню папку юзера потрібно прописати наступне:

cd ~

```
spongebob@spongebob-VirtualBox:/tmp$ cd ~
spongebob@spongebob-VirtualBox:~$
```

- **ls**

ls(list) дозволяє відобразити файли поточної папки або іншої, якщо вона вказана:

```
spongebob@spongebob-VirtualBox:~$ ls
cat      Documents  errors.txt  Music       Pictures  snap       Videos
Desktop  Downloads  file.txt    newfile.txt  Public    Templates
spongebob@spongebob-VirtualBox:~$ ls Desktop/
file.txt
```

У команд також є поняття опцій, або світчів(switch), що дозволяють спеціалізувати вивід відповідно до потреб

-a (all) виводить всі файли, включно з прихованими(такі файли в назві мають крапку попереду):

```
spongebob@spongebob-VirtualBox:~$ ls -a
.               cat         file.txt  newfile.txt  .sudo_as_admin_successful
..              .config     .gnupg    Pictures     Templates
.bash_history   Desktop     .lesshst  .profile     .thunderbird
.bash_logout    Documents   .local    Public       Videos
.bashrc         Downloads   .mozilla  snap
.cache          errors.txt  Music     .ssh
```

-l виводить у вигляді списку:

```
spongebob@spongebob-VirtualBox:~$ ls -l
total 52
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:50 cat
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 01:38 Desktop
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 01:48 Documents
drwxr-xr-x 3 spongebob spongebob 4096 тра  5 19:16 Downloads
-rw-rw-r-- 1 spongebob spongebob   52 тра 13 01:59 errors.txt
-rw-rw-r-- 1 spongebob spongebob    1 тра 13 01:56 file.txt
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Music
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:51 newfile.txt
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Pictures
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Public
drwx------ 3 spongebob spongebob 4096 тра  5 19:15 snap
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Templates
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Videos
```

Цей вивід та його значення вже були згадані в попередній лабораторній.

Опції можна поєднувати. Наприклад:

```
spongebob@spongebob-VirtualBox:~$ ls -la Desktop/
total 8
drwxr-xr-x  2 spongebob spongebob 4096 тра 13 01:38 .
drwxr-x--- 18 spongebob spongebob 4096 тра 13 01:59 ..
-rw-rw-r--  1 spongebob spongebob    0 тра 13 01:38 file.txt
```

- Повні та залежні шляхи

Повний шлях – це шлях, що вказується з будь-якої точки системи, тобто наприклад у папці /etc/ppp я можу отримати вміст папки /usr/bin:

```
spongebob@spongebob-VirtualBox:~$ cd /etc/ppp/
spongebob@spongebob-VirtualBox:/etc/ppp$ ls /usr/bin/
'['                                  mountpoint
 aa-enabled                          mousetweaks
 aa-exec                             mscompress
 aa-features-abi                     msexpand
 aconnect                            mt
 acpi_listen                         mt-gnu
 add-apt-repository                  mtr
 addpart                             mtr-packet
```

Залежний шлях – шлях, до якого є доступ з поточного положення. Наприклад будучи в папці /etc, для переходу до папки /ppp, мені неотрібно прописувати її повний шлях /etc/ppp:

```
spongebob@spongebob-VirtualBox:~$ cd /etc/
spongebob@spongebob-VirtualBox:/etc$ cd ppp/
spongebob@spongebob-VirtualBox:/etc/ppp$
```

Звичайно можна постійно користуватись повними шляхами, але це не завжди зручно та викликає лишні проблеми та затрати часу.

- **Керування файлами**
- Копіювання

Щоб скопіювати файл використовується команда cp (**co**p**y**)

Синтаксис: *cp /path/to/file /new/path/to/file*

Наприклад:

```
spongebob@spongebob-VirtualBox:~$ ls Desktop/
file.txt
spongebob@spongebob-VirtualBox:~$ ls Documents/
spongebob@spongebob-VirtualBox:~$ cp file.txt Documents/
spongebob@spongebob-VirtualBox:~$ ls Documents/
file.txt
```

- Переміщення

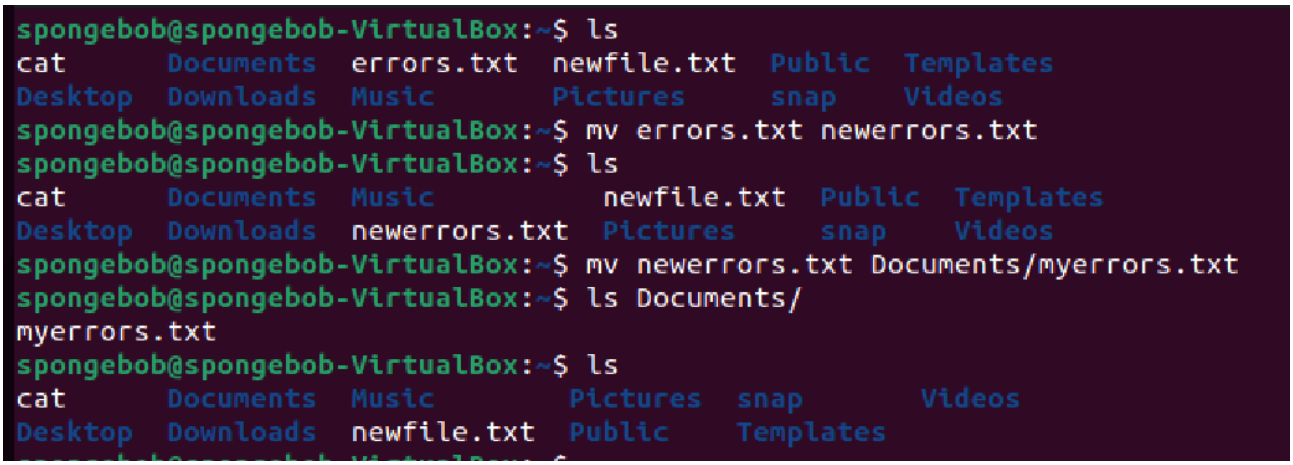
Для переміщення використовується команда mv (**m**o**v**e)

Синтаксис: *mv /path/to/file /new/path/to/file*

Наприклад:

```
spongebob@spongebob-VirtualBox:~$ ls Documents/
file.txt
spongebob@spongebob-VirtualBox:~$ mv Documents/file.txt Music/
spongebob@spongebob-VirtualBox:~$ ls Music/
file.txt
spongebob@spongebob-VirtualBox:~$ ls Documents/
```

За допомогою цієї команди також можна перейменувати файл перенісши його, або залишивши там же:

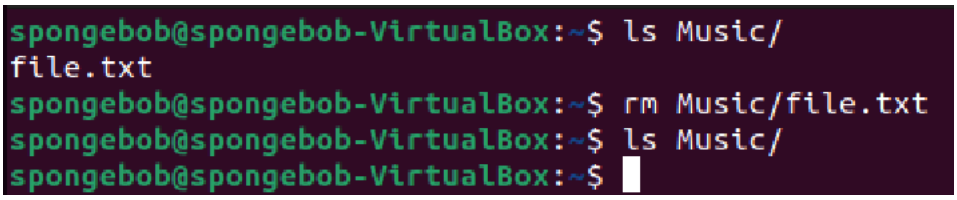


- Видалення

Щоб видалити файл, використовуються команда rm (**rem**ove)

Синтаксис: *rm /path/to/file*

Наприклад:



- Символічні посилання

Символічні посилання (symlink) бувають двох видів: softlink та hardlink

Softlink схожа до ярлика. Її вміст синхронізований з оригіналом файлу і якщо видалити оригінал, зникне й так званий «ярлик», що на нього посилався.

Hardlink же – самостійна копія. Проте, від звичайної копії її відрізняє синхронізованість з оригіналом, тобто якщо змінити вміст оригіналу, зміниться й посилальний файл. Відмінність її від «м'якої» копії полягає в тому, що при видаленні оригіналу копія нікуди не зникне.

Для створення копії існує команда ln (**lin**k)

По дефолту ln має режим створення hardlink. Поекспериментуємо з нашим файлом:

```
spongebob@spongebob-VirtualBox:~$ ls
cat      Documents  errors.txt  Music       Pictures  snap       Videos
Desktop  Downloads  file.txt    newfile.txt  Public    Templates
spongebob@spongebob-VirtualBox:~$ ln file.txt Documents/
spongebob@spongebob-VirtualBox:~$ ls -l
total 52
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:50 cat
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 01:38 Desktop
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 02:38 Documents
drwxr-xr-x 3 spongebob spongebob 4096 тра  5 19:16 Downloads
-rw-rw-r-- 1 spongebob spongebob   52 тра 13 01:59 errors.txt
-rw-rw-r-- 2 spongebob spongebob    1 тра 13 01:56 file.txt
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 02:29 Music
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:51 newfile.txt
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Pictures
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Public
drwx------ 3 spongebob spongebob 4096 тра  5 19:15 snap
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Templates
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Videos
Help
spongebob@spongebob-VirtualBox:~$ ls -l Documents/
total 4
-rw-rw-r-- 2 spongebob spongebob 1 тра 13 01:56 file.txt
```

Посилання створено. Тепер доведемо твердження щодо синхронності та незалежності:

```
spongebob@spongebob-VirtualBox:~$ echo some text > file.txt
spongebob@spongebob-VirtualBox:~$ cat file.txt
some text
spongebob@spongebob-VirtualBox:~$ cat Documents/file.txt
some text
```

Вміст однаковий

Спробуємо видалити оригінал:

```
spongebob@spongebob-VirtualBox:~$ rm file.txt
spongebob@spongebob-VirtualBox:~$ ls Documents/
file.txt
spongebob@spongebob-VirtualBox:~$ cat Documents/file.txt
some text
spongebob@spongebob-VirtualBox:~$ ls
cat      Documents  errors.txt  newfile.txt  Public  Templates
Desktop  Downloads  Music       Pictures     snap    Videos
```

Як бачимо, після видалення, посилання не лише збереглось, а й зберегло вміст оригіналу.

Тепер Спробуємо попрацювати з softlink

Для його створення потрібно використати опцію –s (soft)

```
spongebob@spongebob-VirtualBox:~$ ln -s Documents/file.txt ~
spongebob@spongebob-VirtualBox:~$ ls -l
total 48
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:50 cat
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 01:38 Desktop
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 02:38 Documents
drwxr-xr-x 3 spongebob spongebob 4096 тра  5 19:16 Downloads
-rw-rw-r-- 1 spongebob spongebob   52 тра 13 01:59 errors.txt
lrwxrwxrwx 1 spongebob spongebob   18 тра 13 02:43 file.txt -> Documents/file.t
xt
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 02:29 Music
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:51 newfile.txt
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Pictures
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Public
drwx------ 3 spongebob spongebob 4096 тра  5 19:15 snap
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Templates
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Videos
spongebob@spongebob-VirtualBox:~$ cat Documents/file.txt
some text
spongebob@spongebob-VirtualBox:~$ cat file.txt
some text
```

Стрілочка -> у виводі показує, що файл у домашній директорії зв'язаний з оригіналом у папці Документи. Вміст також збігається.

Спробуємо видалити оригінал:

```
spongebob@spongebob-VirtualBox:~$ rm Documents/file.txt
spongebob@spongebob-VirtualBox:~$ ls Documents/
spongebob@spongebob-VirtualBox:~$ ls -l
total 48
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:50 cat
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 01:38 Desktop
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 02:44 Documents
drwxr-xr-x 3 spongebob spongebob 4096 тра  5 19:16 Downloads
-rw-rw-r-- 1 spongebob spongebob   52 тра 13 01:59 errors.txt
lrwxrwxrwx 1 spongebob spongebob   18 тра 13 02:43 file.txt -> Documents/file.t
xt
drwxr-xr-x 2 spongebob spongebob 4096 тра 13 02:29 Music
-rw-rw-r-- 1 spongebob spongebob    5 тра 13 01:51 newfile.txt
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Pictures
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Public
drwx------ 3 spongebob spongebob 4096 тра  5 19:15 snap
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Templates
drwxr-xr-x 2 spongebob spongebob 4096 тра  4 18:55 Videos
spongebob@spongebob-VirtualBox:~$ cat file.txt
cat: file.txt: No such file or directory
spongebob@spongebob-VirtualBox:~$
```

Після видалення файл підсвічує червоним та при спробі відображення його вмісту, отримали помилку про його відсутність.

- Тип файлу

Для цього існує команда *file*:

```
spongebob@spongebob-VirtualBox:~$ ls
cat      Documents  Music         Pictures  snap       Videos
Desktop  Downloads  newfile.txt   Public    Templates
spongebob@spongebob-VirtualBox:~$ file newfile.txt
newfile.txt: ASCII text
spongebob@spongebob-VirtualBox:~$ file Music/
Music/: directory
```

- Знайти файл

Команда *find*

Синтаксис: *find –switches*

-user для власника

-name(case sensitive)

-iname(case insensitive)

-type f/d/l/s

-size 1M/G

-o логічне або

Приклад:

```
spongebob@spongebob-VirtualBox:~$ find -name newfile.txt -type f
./newfile.txt
```

- Створити папку

Команда *mkdir*

```
spongebob@spongebob-VirtualBox:~$ ls
Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates
spongebob@spongebob-VirtualBox:~$ mkdir NewDirectory
spongebob@spongebob-VirtualBox:~$ ls
Desktop    Downloads  NewDirectory  Public  Templates
Documents  Music      Pictures      snap    Videos
```

- Видалити папку

Є два способи:

Видалення за допомогою *rmdir /path/to/dir*

```
spongebob@spongebob-VirtualBox:~$ ls
Desktop    Downloads  NewDirectory  Public  Templates
Documents  Music      Pictures      snap    Videos
spongebob@spongebob-VirtualBox:~$ rmdir NewDirectory/
spongebob@spongebob-VirtualBox:~$ ls
Desktop    Downloads  Pictures  snap       Videos
Documents  Music      Public    Templates
```

Або ж видалити за допомогою рекурсивного видалення: *rm –r /path/to/dir*

Рекурсивне видалення видалить папку з усіма підпапками та файлами:

```
spongebob@spongebob-VirtualBox:~$ mkdir NewDirectory
spongebob@spongebob-VirtualBox:~$ ls
Desktop     Downloads  NewDirectory  Public  Templates
Documents   Music      Pictures      snap    Videos
spongebob@spongebob-VirtualBox:~$ rm -r NewDirectory/
spongebob@spongebob-VirtualBox:~$ ls
Desktop     Downloads  Pictures  snap       Videos
Documents   Music      Public    Templates
```

3. Керування процесами

- Показати процеси

Для того, щоб показати процеси, можна скористатись командою *ps*:

```
spongebob@spongebob-VirtualBox:~$ ps
    PID TTY          TIME CMD
   4146 pts/0    00:00:00 bash
   4663 pts/0    00:00:00 ps
```

Або *top*:

```
spongebob@spongebob-VirtualBox:~$ top
```

```
top - 02:54:50 up  2:38,  1 user,  load average: 0,00, 0,00, 0,00
Tasks: 175 total,   1 running, 174 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,7 us,  0,0 sy,  0,0 ni, 99,3 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   2911,0 total,   1399,9 free,    603,1 used,    907,9 buff/cache
MiB Swap:    923,2 total,    923,2 free,      0,0 used.   2126,0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1434 spongeb+  20   0 3741580 338424 121696 S   0,3  11,4   1:03.18 gnome-+
   4512 root      20   0       0      0      0 I   0,3   0,0   0:00.44 kworke+
      1 root      20   0  165644  12052   7636 S   0,0   0,4   0:01.35 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthrea+
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_pa+
      6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworke+
      9 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_per+
     10 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_ta+
     11 root      20   0       0      0      0 S   0,0   0,0   0:00.00 rcu_ta+
     12 root      20   0       0      0      0 S   0,0   0,0   0:00.21 ksofti+
     13 root      20   0       0      0      0 I   0,0   0,0   0:00.65 rcu_sc+
     14 root      rt   0       0      0      0 S   0,0   0,0   0:00.08 migrat+
     15 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_i+
     16 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
     17 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kdevtm+
     18 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
     19 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 inet_f+
     20 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kauditd
     21 root      20   0       0      0      0 S   0,0   0,0   0:00.00 khungt+
     22 root      20   0       0      0      0 S   0,0   0,0   0:00.00 oom_re+
     23 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 writeb+
```

Результати цієї команди можуть здатись абсолютно незрозумілими.

Поки ми зупинимо цю команду, натиснувши Ctrl+C

Ми розберемось, як використати її, в наступному кроці:

- Зупинити процес

Давайте запустимо наступний процес:

```
spongebob@spongebob-VirtualBox:~$ ping localhost > /dev/null &
```

& автоматично посилає процес на задній план, тож термінал готовий до наступних команд. Те ж саме відбудеться, якщо після запуску команди ми натиснемо Ctrl+Z

Переглянути процеси бекграунду можна за допомогою команди *jobs*

```
spongebob@spongebob-VirtualBox:~$ ping localhost > /dev/null &
[1] 4686
spongebob@spongebob-VirtualBox:~$ jobs
[1]+  Running                 ping localhost > /dev/null &
```

Повернути його можна за допомогою команди *fg %n*

```
spongebob@spongebob-VirtualBox:~$ fg %1
ping localhost > /dev/null
^Z
[1]+  Stopped                 ping localhost > /dev/null
```

Я одразу натисну Ctrl+Z щоб повернути процес в бекграунд

Щоб зупинити його є декілька способів. Для того, щоб розглянути всі, я продублюю команду:

```
spongebob@spongebob-VirtualBox:~$ ping localhost > /dev/null &
[2] 4689
spongebob@spongebob-VirtualBox:~$ ping localhost > /dev/null &
[3] 4690
spongebob@spongebob-VirtualBox:~$ jobs
[1]+  Stopped                 ping localhost > /dev/null
[2]   Running                 ping localhost > /dev/null &
[3]-  Running                 ping localhost > /dev/null &
```

Перший спосіб – команда *kill %n*

```
spongebob@spongebob-VirtualBox:~$ kill %1

[1]+  Stopped                 ping localhost > /dev/null
spongebob@spongebob-VirtualBox:~$ jobs
[1]+  Terminated              ping localhost > /dev/null
[2]   Running                 ping localhost > /dev/null &
[3]-  Running                 ping localhost > /dev/null &
```

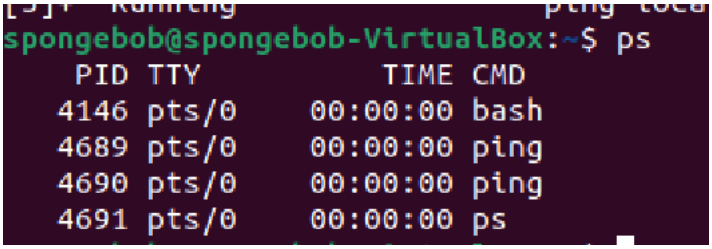
Чекаємо повної зупинки (від декількох секунд до кількох хвилин). Подивимось ще раз:

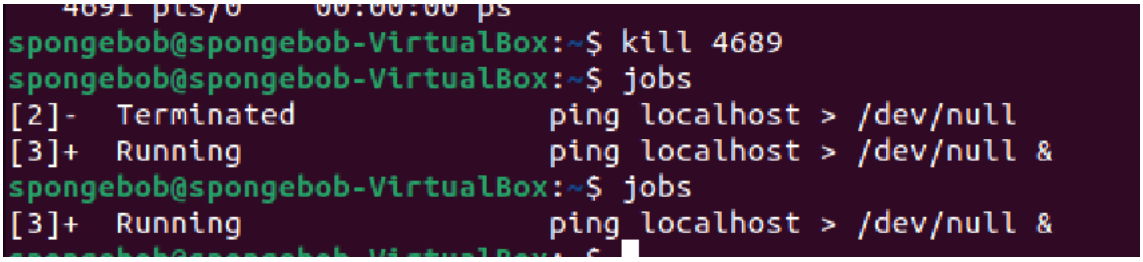
```
spongebob@spongebob-VirtualBox:~$ jobs
[2]-  Running                 ping localhost > /dev/null &
[3]+  Running                 ping localhost > /dev/null &
```

Інший спосіб – *kill PID*

PID – process ID

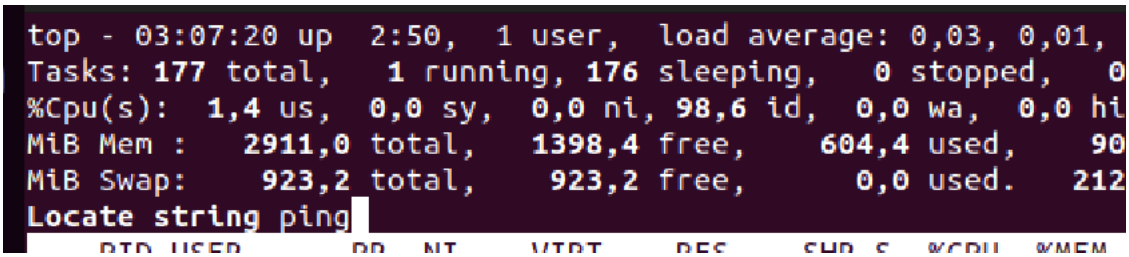
Подивитись PID можна у виводі команди ps

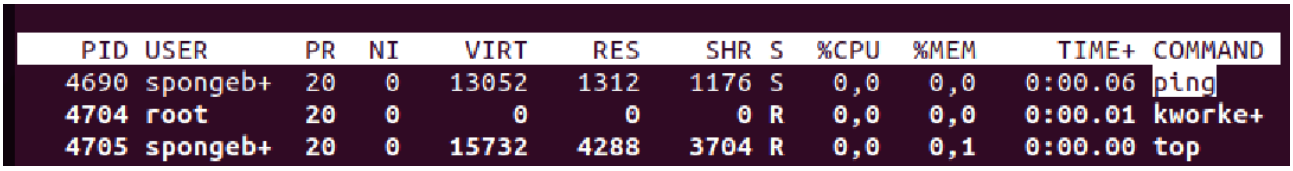




Тепер повернемось до команди top:

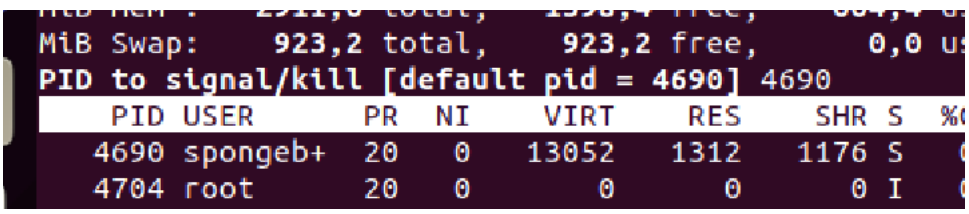
Натискаємо Shift+L та прописуємо потрібен нам процес (ping):

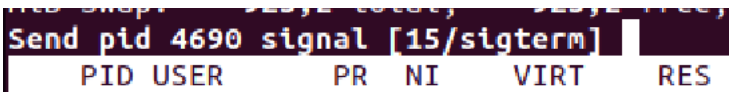


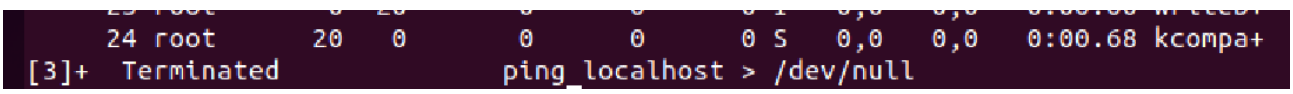


Якщо процес не знайдено натисність Ctrl+C та запустіть ще раз

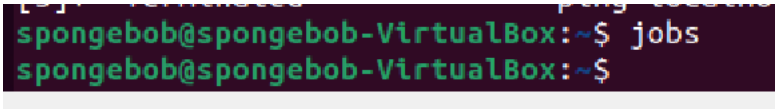
Тепер видалимо процес: прописуємо k та після цього наш PID – 4690 та натискаємо Enter двічі:







Перевіримо:



Все спрацювало

Завдання до лабораторної роботи:

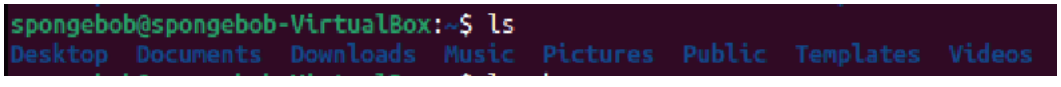
1. Прописати одним рядком: створення папки, перехід до неї, створення файлу з текстом у ньому, вихід з папки, видалення папки
2. Створити файл з вашим ім’ям та прізвищем, створити softlink, hardlink, довести їх властивості (синхронізованість вмісту та залежність/незалежність від оригіналу)
3. Створити який-небудь процес, відмінний від показаного в лабораторній. Відправити його в бекграунд, видалити його одним зі способів. Довести успішність видалення
4. Спровокувати помилку виконання команди, перенаправивши помилку у новий файл. Перейменувати файл, перенісши його до іншої папки
5. Створити файл з порядкóвим записом кількох слів. Вивести його вміст, використавши wildcards

Контрольні питання:

1. Який шлях до кореневої папки?
2. Чи є у лінуксі аналог кошика інших ОС? Вкажіть шлях та в чому їх різниця.
3. Коли варто застосовувати залежні шляхи? Чому?
4. Які є види посилань?
5. Чому папку потрібно видаляти рекурсивно?
6. Чим пайпінг відрізняється від прописання команд через крапку з комою?
7. Скільки стандартних потоків є у лінуксі?
8. Який номер у потоку помилок?
9. Що таке PID? Чому він унікальний?
10. Яка різниця між кореневою та домашньою папкою?

Вимоги до звіту:

1. Скріншоти повинні бути коротко прокоментовані.
2. На скріншотах повинно бути видно: прописану команду, результат її виконання та юзера, як на прикладі:



3. До кожного завдання має бути попередньо прописана умова та подальше пояснення результатів.