**Лекція № 5**

**2.1. Файлова система Linux**

На самому початку освоєння операційної системи Linux користувачам які мають досвід роботи з операційною системою Microsoft Windows може показатися досить дивним позначення файлів та каталогів у ній. Тому, перед вивченням системи Linux, спочатку необхідно розглянути як вона влаштована.

Першою відмінністю є те, що в назвах шляхів файлів і каталогів Linux не використовуються назви дисків на відміну від Windows, де вони обов’язково присутні. Наприклад, шлях **C:\Users\Gleb\Documents\ test.doc** в ОС Windows вказує, що файл **test.doc** знаходиться в каталозі **Documents**, який у свою чергу є безпосередньо у каталозі **Gleb**. Каталог **Gleb** – у каталозі **Users**, який знаходиться у розділі жорсткого диска, якому присвоєна літера C (зазвичай таким є перший розділ жорсткого диска на персональному комп’ютері).

За шляхом до файлу в Windows можна завжди дізнатися, в якому фізичному розділі жорсткого диска знаходиться відповідний файл. Наприклад, якщо файл потрібно зберегти на USB-накопичувачі, то цей пристрій у системі може бути позначений буквами диска **F**, **G**, **H** і т. п. Після вибору відповідного накопичувача у вкладці «Мій комп’ютер» буде автоматично відкритий доступ до файлів та каталогів, що містяться в ньому. Шляхи до них завжди починатимуться літерою відповідного накопичувача. Наприклад, **F:\test.doc**, означає, що файл **test.doc** знаходить у кореневому каталозі диска, який позначається літерою **J**.

Фундаментальна відмінність Linux від Windows у позначенні шляхів до файлів та каталогів полягає у тому, що Linux все зберігає у єдиній структурі каталогів та називається віртуальним каталогом. Віртуальний каталог визначає шляхи до файлів, які містяться на всіх пристроях, що встановлені на персональному комп’ютері, об’єднуючи їх **у єдину структуру каталогів.**

Структура віртуального каталогу Linux складається з єдиного основного каталогу, який називається кореневим і позначається єдиним символом **/**. Каталоги та файли, що знаходяться у ньому, перераховуються з урахуванням шляху до каталогу, що використовується для їх досягнення, аналогічно тому, як це відбувається в системі Windows, тільки на відміну від Windows у якості символу розділювача назв каталогів у шляху замість **\** використовується **/**.

Як приклад, розглянемо шлях до файлу в системі Linux: **/home/ gleb/Documents/test.doc**. Такий шлях вказує тільки на те, що файл **test.doc** знаходиться у каталозі **Documents**, який розташований у каталозі **gleb**, що міститься в каталозі **home**, який у свою чергу знаходиться у кореневому каталозі (на це вказує перший символ **/** у цьому шляху). Легко помітити, що розглянутий вище шлях не вказує, на якому фізичному диску персонального комп’ютера зберігається цей файл.

Складність, пов’язана з використанням віртуального каталогу Linux, полягає саме в тому, що в нього входять структури каталогів усіх запам’ятовуючих пристроїв на комп’ютері. Перший жорсткий диск, встановлений у персональному комп’ютері з ОС Linux, називається **кореневим диском**, який містить ядро віртуального каталогу і від якого виходять усі інші частини віртуального каталогу. На кореневому диску система Linux створює спеціальні каталоги, що називаються точками монтування.

Точками монтування є каталоги у віртуальному каталозі, за якими закріплюються додаткові запам’ятовуючі пристрої, монтування яких у віртуальному каталозі призводить до появи файлів і каталогів, що зберігаються у них, під точками монтування у каталогах монтування без урахування того, що вони фактично можуть перебувати на інших дисках.

На кореневому диску часто фізично розташовуються системні файли, тоді як файли користувача можуть зберігатись на іншому диску (рис. 2.1). Усе залежить від того, скільки розділів жорсткого диска було створено під час інсталяції ОС Linux.

У прикладі, який показаний на рис. 2.1, на персональному комп’ютері є два жорстких диски. Один пов’язаний з коренем віртуального каталогу, що позначено окремим символом **/**. Інший можуть бути змонтований в певних місцях віртуальної структури каталогів. У нашому прикладі другий жорсткий диск змонтований в місці розташування **/home**, яке є місцем, знаходження каталогів, призначених для користувачів системи.



**/**



**/bi**



**/etc**



**/usr**



**/var**



**/debianuser**



**/gleb**



**/kirey**



**/kio**



**/home**

**Рис. 2.1.** Файлова архітектура системи Linux

Структуру файлової системи Linux було розроблено на основі файлової архітектури UNIX. На жаль, файлова архітектура UNIX за минулі роки у зв’язку з появою різновидів цієї операційної системи втратила колишню однаковість. Іноді може показатися, що не можна знайти такі дві системи UNIX або Linux, у яких застосовувалася б повністю однакова структура файлової системи. Однак передбачено кілька загальноприйнятих назв каталогів, які призначені для виконання приблизно одних і тих же функцій. У таблиці 2.1 перераховані деякі з найбільш широко застосовуваними назвами підкаталогів віртуального каталогу Linux.

Табл. 2.1. Загальноприйняті назви каталогів Linux

|  |  |
| --- | --- |
| **Каталог** | **Призначення** |
| **/** | Кореневий каталог, у якому зазвичай не прийнято розміщувати будьякі файли |
| **/bin** | Каталог двійкових файлів, у якому зберігається велика кількість програм для користувача рівня, що відносяться до категорії утиліт GNU |
| **/boot** | Каталог, у якому зберігаються файли, необхідні для завантаження системи |
| **/dev** | Каталог, у якому Linux створює спеціальні файли пристроїв |
| **/etc** | Каталог файлів конфігурації системи |
| **/home** | Вихідний каталог, у якому Linux створює домашні каталоги користувачів |

*Закінчення табл. 2.1.*

|  |  |
| --- | --- |
| **/lib** | Каталог, у якому зберігаються файли бібліотек системи і програмних застосунків |
| **/media** | Каталог, який є загальним місцем для точок монтування, що використовуються для змінних носіїв |
| **/mnt** | У старих версіях відігравав роль каталогу **/media**. Зараз практично не використовується |
| **/opt** | Додатковий каталог, який часто служить для зберігання додаткових пакетів програм |
| **/root** | Домашній каталог користувача **root** (варто звернути увагу на те, що він розташовується окремо від домашніх каталогів інших користувачів) |
| **/sbin** | Системний каталог двійкових файлів, у якому зберігається багато адміністративних програм на рівні GNU |
| **/tmp** | Тимчасовий каталог, у якому можуть створюватися і знищуватися тимчасові робочі файли |
| **/usr** | Каталог програм, встановлених користувачем |
| **/var** | Каталог зі змінним вмістом, призначений для файлів, які часто змінюються, таких як файли журналів. |

Після запуску нового запрошення командного інтерпретатора відкривається сеанс з переходом у вихідний каталог користувача, який є унікальним, та призначений для роботи тільки з певним обліковим записом користувача. Під час створення облікового запису користувача система зазвичай призначає унікальний каталог виключно для цього облікового запису.

Так у рядку-запрошенні до введення команд між символами **:** та **$** після входу певного користувача у систему буде виведено символ **~**, що є псевдонімом до повного шляху до домашнього каталогу користувача відносно його самого. При вході користувача до системи, його поточним робочим каталогом і буде домашній каталог, у цьому можна переконатися, виконавши досить просту команду **pwd** (**Print Working Directory**):



**2.1.1. Перехід каталогами**

У файловій системі Linux для переходу в інший каталог у сеансі командного інтерпретатора використовується команда зміни каталогу **cd** (**Change Directory**). Синтаксис команди **cd** є відносно нескладним: **cd <destination>**

Команда **cd** може приймати єдиний аргумент – призначення, **destination**, який вказує на назву каталогу, до якого необхідно перейти. Якщо в команді **cd** не вказано призначення, здійснюється перехід у домашній каталог користувача. Таку саму дію можна виконати, використавши у якості аргументу символ **~**: **cd ~**

У цьому випадку назва каталогу, яка представлена знаком ~, що означає псевдонім домашнього каталогу користувача, але у загальному випадку може бути визначена з використанням двох різних способів:

як абсолютний шлях до каталогу; як відносний шлях до каталогу.



Абсолютний шлях до файлу точно визначає, де знаходиться каталог у структурі віртуального каталогу, починаючи з кореневого каталогу і закінчуючи назвою самого каталогу, що призводить до отримання повної його назви. Для переходу в конкретне місце розташування у файловій системі з використанням абсолютного шляху до файлу, досить вказати повну назву шляху в команді **cd**:



Відносні шляхи до файлу дозволяють вказувати шлях до необхідного файлу відносно поточного робочого каталогу, не змушуючи починати з кореня. Позначення відносного шляху до файлу починається не з символу **/**, що вказує на кореневий каталог.

Замість цього позначення відносного шляху до файлу починається або з назви каталогу (якщо відбувається перехід до каталогу, який розташований у поточному каталозі), або зі спеціального символу, який зазначає місце розташування щодо поточного каталогу. Для цього використовуються два спеціальні символи:

точка (**.**), яка позначає поточний каталог; подвійна точка (**..**), яка представляє батьківський каталог.



Наприклад, для переходу в каталог bin, який знаходиться у каталозі **/usr**, при цьому каталог **/usr** є поточний робочий каталог, достатньо виконати наступну команду:



Таку ж саму дію можна виконати, позначивши на початку шляху поточний робочий каталог (символ «**.**»):



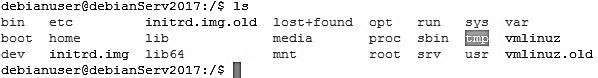
Символ подвійної крапки стає надзвичайно зручним, якщо потрібно виконати перехід за ієрархією каталогів. Наприклад, якщо поточним є каталог **/usr/bin** а потрібно перейти у каталог **/usr/etc**, можна виконати наступну команду:



Символ подвійної крапки забезпечує перехід вгору на один рівень, а потім частина відносного шляху **/src** дозволяє знову перейти на більш низький рівень, але вже в каталог **src**. Для переміщення структурою каталогів можна не один раз використовувати символ подвійної точки, а скільки буде потрібно. Наприклад, для переходу в каталог /etc з каталогу **/usr/src**, можна ввести наступне:



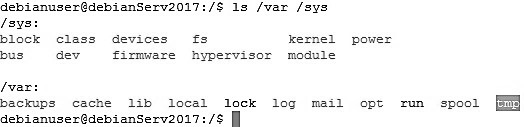
Одна з найбільш важливих функцій командного інтерпретатора полягає у можливості перегляду вмісту каталогів. Інструментом, який виконує цю дію, є команда **ls**, назва якої є скороченням від слова list (список). Найбільш простим застосуванням команди **ls** є її виконання без жодного ключа та аргументу. У цьому випадку команда виведе вміст поточного каталогу в алфавітному порядку.



У зазначеному прикладі показане виведення вмісту кореневого каталогу, оскільки на момент виконання команди **ls** саме кореневий каталог був поточним робочим. Якщо ж виконати команду **ls** з аргументом, то буде виведений вміст каталогу, шлях до якого (абсолютний або відносний) був зазначений у якості цього аргументу, навіть якщо користувач у поточний момент часу не знаходиться у відповідному каталозі. У наступному прикладі виводиться вміст каталогу **/var**, який наразі не є поточним робочим каталогом:



Крім того, у команді **ls** можна вказувати більш ніж один аргумент. У цьому випадку буде почергово виведений вміст каталогів, які вказані у якості аргументів. Наступний приклад демонструє виведення вмісту каталогів **/var** та **/sys**:



У результаті виконання можна помітити цікаву особливість, що при застосуванні більше ніж одного аргументу в команді **ls**, вона виводить вміст усіх вказаних у якості аргументів каталогів, але у зворотному порядку.

У свою чергу, якщо команда **ls** виконується без жодного ключа, вона виводить вміст каталогів у найбільш простому вигляді: в алфавітному порядку в декілька стовпчиків. Зазвичай це може бути зовсім не те, що потрібно користувачеві.

У таблиці 2.2 перерахована частина найбільш широко застосовуваних ключів, команди **ls**, за допомогою яких можна отримати необхідні дані. Досить багато однобуквених ключів мають свої аналоги серед повнослівних ключів.

Табл. 2.2. Часто застосовувані ключі команди ls

|  |  |  |
| --- | --- | --- |
| **Однобуквений ключ** | **Повнослівний ключ** | **Опис** |
| ***-a*** | ***--all*** | Виводить усі файли та каталоги, включаючи приховані, назви яких починаються з точки |
| ***-F*** | ***--classify*** | Виводить вміст каталогів, приєднуючи до записів спеціальні символи, що позначають певні типи файлів |
| ***-s*** | ***--size*** | Окрім назв файлів та каталогів, також виводить перед ними розмір відповідного файлу, або каталогу в блоках |
| ***-r*** | ***--reverse*** | Виводить вміст каталогів у зворотному порядку сортування, режим якого вказаний за допомогою відповідного ключа (за замовчуванням – за алфавітом) |
| ***-S*** | ***--sort=size*** | Виводить вміст каталогу відсортованого за розмірами файлів |
| ***-t*** | ***--sort=time*** | Виводить вміст каталогу відсортованого за часом модифікації файлів |
| ***-l*** |  | Виводить вміст каталогу з найбільш детальною інформацією (довгий формат) |
| ***-h*** | ***--humanreadable*** | У сукупності з ключем **-l** виводить розмір у більш придатному вигляді, використовуючи позначення K – для кілобайтів, M – для мегабайтів і G – для гігабайтів |
| ***-i*** | ***--inode*** | У сукупності з ключем **-l** у якості першого стовпчика виводить індексний номер (так званий індексний вузол) для кожного файлу. |

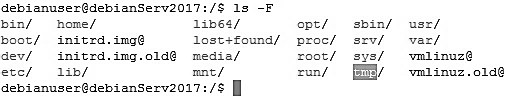
Тепер розглянемо найбільш часто застосовувані ключі детальніше. Ключ **-F** є так званим класифікатором, що дозволяє сформувати виведення, в якому простіше відрізнити звичайні файли від каталогів, символічних посилань та виконуваних файлів. Застосування цього ключа під час виведення вмісту каталогу додає до назви кожного елемента, якщо він не представляє звичайний файл, спеціальний символ, у якості якого може бути:

***/*** – для каталогу;

***@*** – для символічного посилання;

***\**** – для виконуваного файлу, який представляє собою.

У наступному прикладі представлено виведення вмісту кореневого каталогу з використанням ключа-класифікатора



З іншого боку, застосування команди **ls** в основній формі виводить вміст каталогу, показуючи не всі файли. У системі Linux інформація щодо конфігурації достатньо часто зберігається у прихованих файлах і каталогах, назви яких починаються з точки. Такі файли не з’являються в створюваному за замовчуванням лістингу **ls**. Для відображення прихованих файлів поряд зі звичайними файлами і каталогами потрібно використати ключ **-a**. Наступний приклад виводить вміст кореневого каталогу, включаючи приховані файли:



Як можна побачити, прихованим каталогом у кореневому є каталог ***.cache***. Крім того у якості прихованих каталогів у будь-якому каталозі зберігаються дескриптори, що вказують на поточний (***.***) та батьківський (***..***) каталоги. Хоча для кореневого каталогу дескриптор «***..***» практично ніякої ролі не відіграє, оскільки він не має батьківського каталогу (точніше він сам для себе і є батьківським каталогом), для всіх інших каталогів цей дескриптор має істотне значення.

Використання ключа ***-l*** з командою ***ls*** кардинально відрізняє виведення вмісту каталогів, від тих виведень, що вже були розглянуті раніше. Цей ключ дозволяє отримати так званий довгий формат лістингу, в якому наведена докладна інформація щодо кожного елементу в каталозі.

У кожному рядку лістингу в довгому форматі містяться відомості про різні елементи певного каталогу. Такий лістинг, окрім назви елемента, показує іншу корисну інформацію і являє собою інформацію у структурованому вигляді, де:

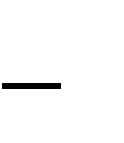
* **1-й стовпчик** – права доступу до файлів/каталогів (буде розглянуто у наступних розділах);

**2-й стовпчик** – кількість жорстких посилань на файл;



**3-й стовпчик** – ім’я користувача власника файлу;

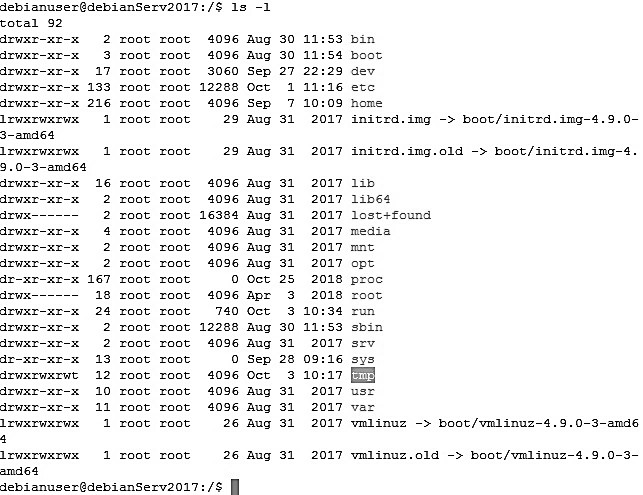
* **4-й стовпчик** – назва основної групи користувачів, до якої відноситься власник файлу;

 **5-й стовпчик** – розмір файлу в байтах (при використанні ключа ***-h*** виводяться додаткові позначення);

**6–8 стовпчики** – дата та час останньої модифікації файлу; **9-й стовпчик** – назва файлу.



У наступному прикладі представлено виведення вмісту кореневого каталогу в довгому форматі.



Також можна звернути увагу на те, що виведення символічних посилань у довгому форматі достатньо відрізняється від виведення всіх інших елементів. Окрім назви файлу, що є символічним посиланням, також представлена його назва, на який вказує це символічне посилання.

**2.1.2. Команди обробки файлів та каталогів**

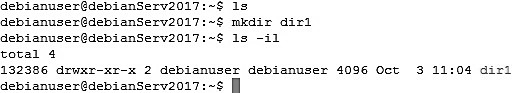
У командного інтерпретатора **bash** є велика кількість команд для маніпулювання файлами у файловій системі Linux. Далі будуть розглянуті команди, які застосовуються для роботи з файлами і каталогами з інтерфейсу командного рядка в усіх ситуаціях, коли потрібно здійснювати їх обробку.

**Створення каталогів**

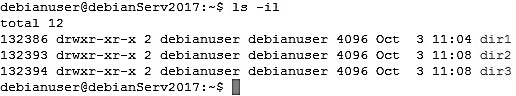
Для створення каталогів призначена команда **mkdir**, яка має наступний синтаксис:

***mkdir <каталог ...>***

де ***каталог ...*** – одна або декілька назв каталогів, вказаних через пробіл. Усі каталоги створюються у поточному робочому каталозі. **Приклади** ***mkdir dir1* –** створює один каталог під назвою dir1;



***mkdir dir2 dir3* –** створює два каталоги з назвами dir2 та dir3.

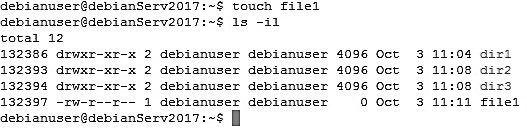


Використання ключа -p призведе до створення ланцюжка каталогів, якщо вказаний відповідний шлях. Наприклад:

***mkdir -p dir1/dir2*** – створить каталог ***dir2***, створивши до цього і його батьківський каталог ***dir1***, який буде розміщений у поточному робочому каталозі.

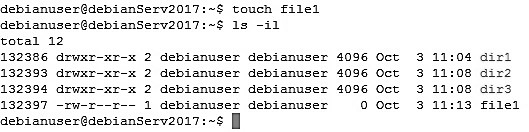
**Створення файлів**

Час від часу виникають ситуації, у яких потрібно створити порожній файл. У подібних випадках за відсутності необхідного файлу можна використовувати команду ***touch***, що дозволить легко його створити.



Команда ***touch*** створює новий файл з вказаною назвою, а власником файлу призначає ім’я поточного користувача. У розглянутій команді ***ls*** використовувалися ключі ***-il***, тому перший запис в лістингу показує номер індексного вузла, присвоєний файлу. Кожен файл у файловій системі Linux має унікальний номер індексного вузла.

Насправді створення порожнього файлу командою ***touch*** не є основним призначенням цієї команди, а тільки її побічним ефектом, який має місце, якщо у якості аргументу команди була вказана назва неіснуючого файлу. Основним призначенням команди ***touch*** є зміна часу останньої модифікації файлу. Наприклад, застосуємо цю команду ще раз, але до файлу, який до цього вже був створений:



Як можна помітити, під час повторного застосування команди touch для файлу file1 час його останньої зміни був оновлений, а саме збільшений на дві хвилини. Таким чином ми ніби «доторкнулися» до файлу.

**Копіювання файлів**

Для копіювання файлів і каталогів у Linux призначена команда ***cp***, яка означає скорочення від слова **CoPy**. Така команда повинна мати як мінімум 2 аргументи. У цьому випадку перший аргумент є джерелом копіювання, а другий – призначенням.

Розглянемо цю форму команди більш детально. Команда ***cp*** у цьому випадку буде мати наступний синтаксис: ***cp <елемент1> <елемент2>***,

де ***елемент1*** є джерелом копіювання і може бути файлом або каталогом, який повинен обов’язково існувати. Для копіювання каталогу необхідно застосувати ключ -r, який означає рекурсивне копіювання каталогу разом з його підкаталогами.

Натомість ***елемент2*** може бути файлом або каталогом. Система визначає, чим саме є цей аргумент та чи існує він взагалі.

Якщо елементу із зазначеною назвою не існує, то у випадку копіювання файлу система створить у поточному робочому файлі новий з відповідною до другого аргументу назвою.

Але якщо елемент з цією назвою існує і він є файлом, система перезапише його, тобто вся інформація, що зберігалася у ньому до цього, буде втрачена. У цьому випадку застосовувати команду копіювання варто досить обережно.

Якщо елемент із заданою назвою існує і він є каталогом, система скопіює елемент, вказаний у якості першого аргументу (файл або каталог) у нього.

Команда ***cp*** буде діяти зовсім іншим чином, якщо матиме три або більше аргументи. У цьому випадку команда буде мати наступний синтаксис: ***cp <елемент...> <каталог>***,

де ***елемент...*** – список елементів-джерел копіювання, якими можуть бути як файли, так і каталоги, але якщо серед джерел є хоча б один каталог, потрібно застосувати ключ ***-r***; ***каталог*** – каталог-призначення, який повинен обов’язково існувати, інакше команда буде виконана з помилкою.

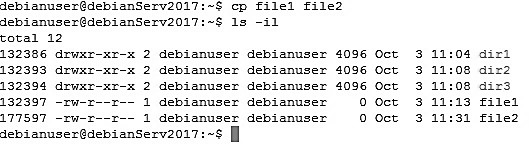
Як і у більшості команд оболонки ***bash*** у команди ***cp*** є різноманітні ключі для вирішення багатьох різних задач, основні з яких представлені у таблиці 2.3.

Табл. 2.3. Основні ключі команди cp

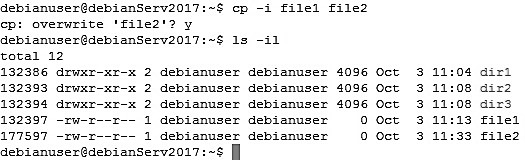
|  |  |
| --- | --- |
| **Ключ** | **Опис** |
| ***-i*** | Інтерактивний режим команди ***cp***: якщо у результаті виконання команди файл може бути перезаписаний, перед її виконанням виводиться додаткове питання для користувача перезаписати файл чи ні |
| ***-r*** | Копіювання каталогів разом з їх вмістом (рекурсивне копіювання) |
| ***-p*** | Копіювання зі збереженням атрибутів файлів (права доступу, власник, група власників і т. д.), якщо це можливо |
| ***-a*** | Запис файлів у архів зі збереженням їх атрибутів |
| ***-u*** | Режим оновлення: копіювання вихідного файлу, тільки якщо він має більш нові значення дати і часу, порівняно з цільовим файлом |
| ***-v*** | Застосування режиму докладного виведення з поясненням того, що відбувається. |

**Приклади**

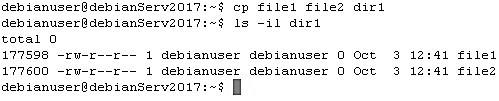
***cp file1 file2*** – копіювання файлу ***file1*** у файл ***file2***. Якщо ***file2*** існує, у ньому буде міститися перезаписана інформація, що збережена у файлі ***file1***. Якщо файл ***file2*** відсутній, то він буде створений.



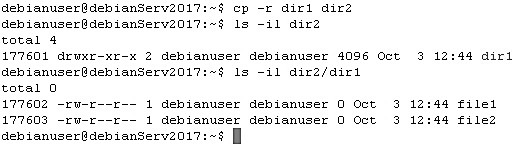
***cp -i file1 file2*** – те саме, що і вище, але якщо файл ***file2*** існує, у користувача буде запитано підтвердження перед перезаписом файлу.



***cp file1 file2 dir1*** – копіювання файлів ***file1*** і ***file2*** у каталог ***dir1***, який обов’язково повинен існувати.



***cp -r dir1 dir2*** – копіювання каталогу ***dir1*** зі всім його вмістом у каталог ***dir2***. Якщо каталог ***dir2*** не існує, він буде створений і заповнений вмістом каталогу ***dir1***.



**Переміщення та перейменування файлів**

Команда **mv** (скорочення від **MoVe**)виконує операції переміщення та перейменування файлів залежно від особливостей використання. З точки зору Linux, переміщення та перейменування є тією ж самою операцією, перейменування являє собою особливий випадок переміщення, коли один файл переміщується в інший. Команда ***mv*** використовується майже так само, як команда ***cp*** та може використовувати практично ті ж самі ключі, однак існують певні нюанси:

***mv <файл1> <файл2>* –** перейменовує файл ***файл1*** у ***файл2***, не

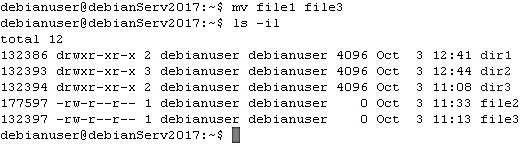
змінюючи індексного дескриптора файлу;

***mv <каталог1> <каталог2> –*** якщо ***каталог2*** існує то ***каталог1*** переміщується разом зі всім його вмістом у ***каталог2***, інакше ***каталог1*** перейменовується у ***каталог2***; ***mv <елемент...> <каталог>* –** переміщує один або більше елементів, якими можуть бути файли і каталоги, з поточного робочого каталогу в ***каталог***, який обов’язково має існувати.

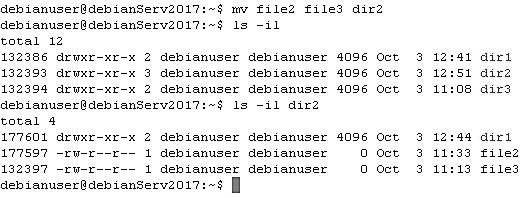
Варто звернути увагу на те, що команда ***mv*** на відміну від команди ***cp*** не підтримує ключ ***-r***.

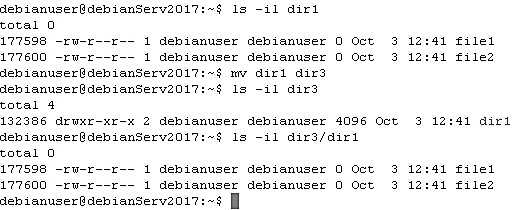
**Приклади**

***mv file1 file3*** –перейменування файлу ***file1*** у ***file3***. Якщо ***file3*** існує, він буде замінений на новий файл ***file1***, якщо ні, буде створений. У будь-якому випадку з’явиться новий файл ***file3***;



***mv -i file1 file3*** – те саме, що і вище, але якщо файл ***file3*** існує, у користувача буде запитано підтвердження перед перезаписом файлу; ***mv file2 file3 dir2*** – переміщення файлів ***file2*** і ***file3*** у каталог ***dir2***, який повинен існувати;

 ***mv dir1 dir2*** – переміщення каталогу ***dir1*** зі всім його вмістом у каталог ***dir2***. Якщо каталог ***dir2*** не існує, відбудеться перейменування каталогу ***dir1*** у ***dir2***.



**Видалення файлів та каталогів**

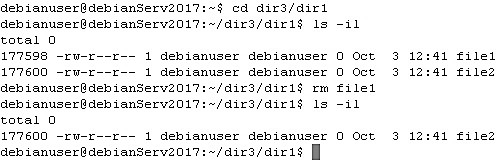
Для видалення файлів та каталогів використовується команда ***rm*** (скорочення від **ReMove**), яка має наступний синтаксис: ***rm <елемент...>***де ***елемент*** – це один або декілька файлів чи каталогів.

Основні ключі, що підтримує команда ***rm***, наведені у таблиці 2.4.

Табл. 2.4. Параметри команди rm

|  |  |
| --- | --- |
| **Ключ** | **Опис** |
| ***-r*** | Видалення каталогів разом з їх вмістом. Цей ключ є обов’язковим для видалення каталогів |
| ***-i*** | Інтерактивний режим: запитує у користувача підтвердження перед видаленням існуючого файлу, а інакше просто видаляє вказані файли та каталоги |
| ***-f*** | Виконання видалення у будь-якому разі. Зазначений ключ ігнорує відсутні файли і ключ ***-і***, якщо той вказаний разом з ним. Виконання команди з цим ключем у деяких випадках може бути достатньо небезпечним |
| ***-v*** | Виведення в процесі видалення інформаційних повідомлень. |

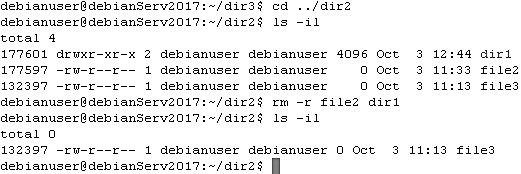
**Приклади** ***rm file1*** –видалення файлу ***file1***;



***rm -i file1*** – те саме, але перед видаленням ***file1*** буде

виведено запрошення підтвердження у користувача; ***rm -r file2 dir1*** – видалення файлу ***file2*** і каталогу ***dir1***

разом з його вмістом;



***rm -rf file2 dir1*** – те саме, але якщо файл ***file2*** або каталог ***dir1*** не існують, команда просто продовжує роботу, не виводячи ніяких повідомлень.

Також існує команда ***rmdir***, що призначена для видалення каталогів, але при цьому вони мають бути порожніми.

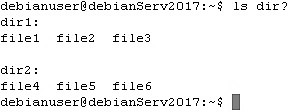
**Групові символи**

Оскільки назви файлів та каталогів всюди використовуються у командній оболонці, існують спеціальні символи, за допомогою яких можна швидко визначити цілі групи назв файлів. Такі символи називаються **груповими символами** або **символами підстановки** та дозволяють вибирати назви файлів за шаблоном.

Більшість групових символів буде розглянуто у розділі посібника щодо регулярних виразів, наразі розглянемо два основні символи, які використовуються у командах, яким у якості аргументів передаються назви файлів або каталогів. Такими символами є:

***\**** – символ, який заміщує від нуля до довільної їх кількості; ***?*** – символ, який заміщує від нуля до одного будь-якого символу.

Наприклад, у домашньому каталозі користувача ***debianuser*** знаходяться каталоги, що мають назви ***dir1*** та ***dir2*** відповідно. Для того, щоб подивитися почергово вміст цих каталогів необов’язково вказувати їх у якості аргументів команди ***ls***, можна задати тільки один аргумент з груповим символом. ***ls dir?***



Як бачимо з наведеного прикладу, команда ***ls*** дійсно вивела вміст каталогів ***dir1*** та ***dir2***. Можна також зазначити, що у цьому випадку виконання команди ***ls dir\**** призведе до такого самого результату.

Розглянемо інший приклад. Створимо у каталозі ***dir2*** файли з назвами ***file40*** та ***file400***, а потім виконаємо команду ***mv file4\* ../dir1***.



Таким чином, з вищенаведеного прикладу можна побачити, що у каталог ***dir1*** були переміщені файли ***file4***, ***file40*** та ***file400***, які задовольняють вказаному шаблону ***file4\****.

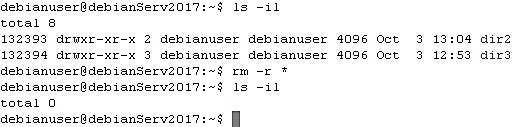
Використовуючи групові символи, можна видалити весь вміст поточного каталогу. Однак:

***rm \**** – видалить весь вміст поточного каталогу, якщо у ньому

знаходяться тільки файли і немає жодного каталогу; ***rm -r \**** – видалить весь вміст поточного каталогу, включаючи

файли і каталоги, на що вказує ключ ***-r***.

Наступний приклад видаляє весь вміст домашнього каталогу користувача ***debianuser***.



Однак не можна застосовувати групові символи у наступних випадках:

 у команді ***cd***, оскільки неможливо перейти одночасно у декілька

каталогів одразу; у командах ***mkdir*** та ***touch***;

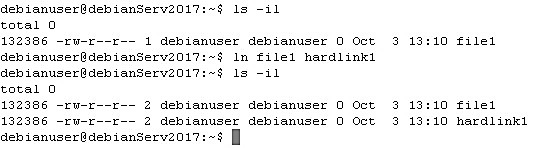


у командах ***cd*** та ***mv*** у якості призначення (але можна у якості джерела).

**Створення посилань**

У файловій системі можна створити посилання на файли, які бувають двох типів: жорсткі та символічні. Обидва типи посилань можна створити командою ***ln*** (скорочення від **LiNk**). Виконання команди без ключа дозволяє створити жорстке посилання, а з ключем ***–s*** – символічне. Синтаксис команди наступний: **ln [-s] <елемент> <посилання>**

Початковим способом створення посилань в операційній системі UNIX та похідних від неї операційних систем є жорсткі посилання. З точки зору файлової системи, жорстке посилання є нічим іншим як другою альтернативною назвою файлу. За замовчуванням кожен файл має одне жорстке посилання, яке і є його назвою. Під час створення жорсткого посилання у каталозі створюється додатковий запис, однак він має одне й те саме значення індексного дескриптора. Це можна чітко побачити у наступному прикладі.



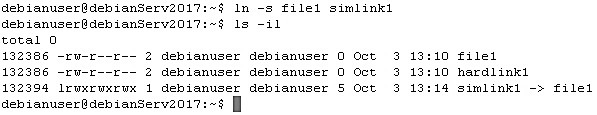
У представленому прикладі було створено жорстке посилання ***hardlink1*** на файл ***file1***. З прикладу видно, що обидві назви вказують на одне й те саме місце у пам’яті, оскільки мають один і той самий індексний дескриптор.

Жорсткі посилання мають два важливих обмеження:

 жорстке посилання не може вказувати на файл за межами власної файлової системи, це означає, що посилання не може вказувати на файл, який знаходиться в іншому розділі диска;  жорстке посилання не може вказувати на каталог.

Саме з метою подолання обмеження жорстких посилань був створений інший тип посилань – символічні посилання. Символічне посилання представляє собою спеціальний файл особливого типу, в якому записаний текстовий вказівник на файл або каталог. Тим самим, можна провести аналогію з ярликами в операційній системі Windows.

Наступний приклад демонструє створення символічного посилання ***simlink1*** на файл ***file1***:

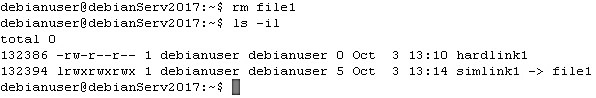


У прикладі видно, по-перше, що індексний дескриптор символічного посилання відрізняється від індексного дескриптора файлу, що вказує на те, що це є абсолютно іншим файлом. По-друге, символічне посилання відрізняється від оригінального файлу своїм розміром. У цьому випадку воно має розмір 5 байт, хоча оригінальний файл має розмір 0 байт, оскільки є порожнім. Це пояснюється тим, що у посиланні зберігається назва файлу-оригіналу, яка і визначає розмір посилання. Так, дійсно назва ***file1*** складається з 5 символів, кожен з яких займає 1 байт. По-третє, у довгому форматі виведення змісту каталогу виведення назви символічного посилання істотно відрізняється від інших елементів файлової системи, яке має формат ***<ім‟я\_ посилання> -> <ім‟я\_файлу-оригіналу>***. До того ж символічні посилання виводяться іншим кольором (колір морської хвилі).

Відмінності між жорсткими та символічними посиланнями можна також побачити під час видалення файлів, на які вони вказують. При видаленні файлу, який має жорсткі посилання, насправді він буде мати на одну назву менше, а сам залишиться збереженим у пам’яті допоки не будуть видалені всі жорсткі посилання на нього.

Щодо видалення файлу, який має символічне посилання, то він дійсно буде видалений, але посилання залишається, проте не буде вказувати нінащо. Про такі посилання кажуть, що вони «биті». У багатьох реалізаціях команда ***ls*** виділяє биті посилання кольором, наприклад, червоним, щоб звернути на них увагу (Увага! Усі приклади представлені інверсними кольорами).

Наступний приклад демонструє все сказане вище:



Стосовно видалення файлу ***file1***, який мав як жорстке, так і символічне посилання, насправді він видалений не буде, оскільки залишатиметься жорстке посилання ***hardlink1***. Однак символічне посилання ***simlink1*** перестане працювати, оскільки воно зберігатиме назву ***file1***, а не ***hardlink1***.