ЛЕКЦІЯ 15. ФАЙЛОВА СИСТЕМА UNIX.

Знайомство з файловою системою ext2fs (second extended filesystem)

Файлова система — це структура, за допомогою якої ядро операційної системи надає користувачам (і процесам) ресурси довгострокової пам'яті системи, тобто пам'яті на різного виду довгострокових носіях інформації — жорстких дисках, магнітних стрічках, CD-ROM і т. п.

Імена файлів в Linux можуть мати довжину до 1024 символи і складатися з будь-яких символів, крім символу з кодом 0 і символу / (слеша). Однак є ще ряд символів, які мають в оболонці shell спеціальне значення і які тому не рекомендується включати в імена. Це наступні символи: ! @ # $ & ~ % * ( ) [ ] { } ' " \ : ; > < ' пробіл.

Можна також укласти ім’я файлу або каталогу з такими символами в подвійні лапки. Наприклад, для створення каталогу з ім'ям "My old files" варто використовувати команду:

[user]$ mkdir "My old files"

Кожному файлу в Linux відповідає так званий "індексний дескриптор" файлу, або "inode". Саме індексний дескриптор містить всю необхідну файловій системі інформацію про файл, включаючи інформацію про розташування частин файлу на носії, типі файлу й багато чого іншого. Індексні дескриптори файлів утримуються в спеціальній таблиці (inode table), що створюється при створенні файлової системи на носії.

Ім’я будь-якого файлу в Linux є не чим іншим, як посиланням на індексний дескриптор файлу. Тому кожний файл може мати, скільки завгодно різних імен. Ці імена називають ще "твердими" посиланнями. Коли ви видаляєте файл, що має кілька різних імен — твердих посилань, то фактично віддаляється тільки одне посилання — те, яке ви вказали в команді видаленя файлу. Навіть коли ви видаляєте останнє посилання, це ще може не означати видаленя вмісту файлу — якщо файл ще використовується системою або якимось додатком, то він зберігається доти, поки він не "звільниться".

Для того, щоб дати файлу (або каталогу) додаткове ім'я (створити тверде посилання), використовується команда ln у наступному форматі:

ln ім'я_існуючого_файлу нове_ім'я

Каталоги

Ієрархічну структуру каталогів звичайно ілюструють малюнком "дерева каталогів", у якому кожний каталог зображується вузлом "дерева", а файли — "листами". В MS Windows або DOS каталогова структура будується окремо для кожного фізичного носія (тобто, маємо не окреме "дерево", а цілий "ліс") і кореневий каталог кожної каталогової структури позначається якою-небудь буквою латинського алфавіту (звідси вже виникає деяке обмеження). В Linux (і UNIX взагалі) будується єдина каталогова структура для всіх носіїв, і єдиний кореневий каталог цієї структури позначається символом "/". У цю єдину каталогову структуру можна підключити будь-яке число каталогів, фізично розташованих на різних носіях (як говорять, "змонтувати файлову систему" або "змонтувати носій"). Імена каталогів будуються по тим ж правилам, що й імена

файлів. Повним ім'ям файлу (або шляхом до файлу) називається список імен вкладених друг у друга підкаталогів, що починається з кореневого каталогу й кінчається властиво ім'ям файлу.

Крім поточного каталогу для кожного користувача визначений ще його "домашній каталог" — каталог, у якому користувач має усі права: може створювати й видаляти файли, міняти права доступу до них і т.д. У каталоговій структурі Linux домашні каталоги користувачів звичайно розміщаються в каталозі /home і мають імена, що збігаються з ім'ям користувача. Наприклад, /home/jim. Кожний користувач може звернутися до свого домашнього каталогу за допомогою значка ~, тобто, наприклад, користувач jim може
звернутися до каталогу /home/jim/doc як до ~/doc. Коли користувач входить у систему, то поточним каталогом стає домашній каталог даного користувача.

Для зміни поточного каталогу служить команда cd. Як параметр цій команді треба вказати повний або відносний шлях до того каталогу, що ви хочете зробити поточним. Поняття повного шляху вже було пояснено, а поняття відносного шляху вимагає додаткового пояснення. Відносним шляхом називається перерахування тих каталогів, які потрібно пройти в "дереві каталогів", щоб перейти від поточного каталогу до якогось іншого каталогу (назвемо його цільовим). Якщо цільовий каталог, тобто каталог, що ви хочете зробити поточним, розташований нижче поточного в структурі каталогів, то зробити це просто: ви вказуєте спочатку підкаталог поточного каталогу, потім підкаталог того каталогу й т.д., аж до ім'я цільового каталогу. Якщо ж цільовий каталог розташований вище в каталоговій структурі, або взагалі на іншій "галузі" дерева, то ситуація трохи складніше. Звичайно, можна було б користуватися повним шляхом, але тоді прийде записувати дуже довгі маршрути.

Ці труднощі переборюються в такий спосіб. Для кожного каталогу (крім кореневого) у дереві каталогів однозначно визначений "батьківський каталог". У кожному каталозі є два особливих записи. Одна з них позначається просто крапкою і є вказівкою на цей самий каталог, а другий запис, позначуваний двома крапками, — покажчик на батьківський каталог. Ці імена із двох крапок і використовуються для запису відносних
шляхів. Щоб зробити поточний батьківський каталог, досить дати команду

[user]$ cd ..

А щоб перейти по дереву каталогів на два "поверхи" нагору, звідки спуститися в підкаталог kat1/kat2 треба дати команду

[user]$ cd ../../kat1/kat2

Типи файлів

У попередніх розділах ми розглянули два типи файлів: звичайні файли й каталоги. Але в Linux існує ще кілька типів файлів. З ними ми познайомимося в цьому розділі. Як уже було сказано, з погляду операційної системи файл являє собою просто потік байтів. Такий підхід дозволяє поширити концепцію файлу на фізичні пристрої й деякі інші об'єкти. Це дозволяє спростити організацію даних і обмін ними, тому що аналогічним образом здійснюється запис даних у файл, передача їх на фізичні пристрої й обмін даними між процесами. У всіх

цих випадках використовується той самий підхід, заснований на ідеї байтового потоку. Тому поряд зі звичайними файлами й каталогами, файлами з погляду Linux є також:

1. файли фізичних пристроїв;
2. іменовані канали (named pipes);
3. гнізда (sockets);
4. символічні посилання (symlinks).

Файли фізичних пристроїв

Як уже говорилося, з погляду ОС Linux, всі пристрої, що підключаються до комп'ютера, (жорсткі диски, дискети, термінал, принтер, модем і т.д.), представляються файлами. Якщо, наприклад, треба вивести на екран якусь інформацію, то система як би робить запис у файл /dev/tty01.

Фізичні пристрої бувають двох типів: символьними (або байт-орієнтованими) і блоковими (або блок-орієнтованими). Розходження між ними полягає в тому, як виробляється зчитування й запис інформації в ці пристрої. Взаємодія із символьними пристроями виробляється посимвольно, у режимі потоку байтів. До таких пристроїв ставляться, наприклад, термінали. На блок-орієнтованих пристроях інформація записується (і, відповідно, зчитується) блоками. Прикладом пристроїв цього типу є жорсткі диски. На диск неможливо записати або зчитати з його один байт: обмін з диском виконується тільки блоками.

Іменовані канали (pipes)

Ще один тип спеціальних файлів – іменовані канали, або буфери FIFO (First In - First Out). Файли цього типу служать в основному для того, щоб організувати обмін даними між різними додатками (pipe переводиться з англійського як труба).

Канал — це дуже зручний і широко застосовуваний засіб обміну інформацією між процесами. Усе, що один процес поміщає в канал, інший може звідти прочитати. Якщо два процеси, що обмінюються інформацією, породжені тим самим батьківським процесом (а так найчастіше й відбувається), канал може бути неіменованим. У противному випадку потрібно створити іменований канал, що можна зробити за допомогою програми mkfifo. При цьому файл іменованого каналу бере участь тільки в ініціації обміну даними.

Доменні гнізда (sockets)

Гнізда — це з'єднання між процесами, які дозволяють їм взаємодіяти, не підпадаючи під вплив інших процесів. Взагалі гнізда (і взаємодія програм за допомогою гнізд) відіграють дуже важливу роль у всіх Unix-Системах, включаючи й Linux: вони є ключовим поняттям TCP/IP і відповідно на них цілком будується Інтернет. Однак з погляду файлової системи гнізда практично нічим не відрізняються від іменованих каналів: це просто мітки, що дозволяють зв'язати кілька програм. Після того як зв'язок установлений, спілкування програм відбувається без участі файлу гнізда: дані передаються ядром ОС безпосередньо від однієї програми до іншої.

Незважаючи на те, що інші процеси можуть бачити файли гнізд як елементи каталогу, процеси, що не беруть участь у даному конкретному з'єднанні, не можуть здійснювати над файлами гнізд операції читання/запису. Серед стандартних засобів, що використовують гнізда - система X Window, система друку й система syslog.

Символічні посилання

Жорстке посилання є просто ще одним ім'ям для вихідного файлу. Воно прописується в індексному дескрипторі вихідного файлу. Після створення жорсткого посилання неможливо розрізнити, де вихідне ім'я файлу, а де посилання. Якщо ви видаляєте один із цих файлів (точніше одне із цих імен), то файл ще зберігається на диску (поки в нього є хоч одне ім'я-посилання).

Дуже важко розрізнити первісне ім'я файлу й пізніше створені тверді посилання на нього. Тому жорсткі посилання застосовуються там, де відслідковувати розходження й не потрібно. Одне із застосувань подібних посилань полягає в тому, щоб запобігти можливості випадкового видаленя файлу.

Особливістю жорстких посилань є те, що вони прямо вказують на номер індексного дескриптора, а, отже, такі імена можуть вказувати тільки на файли усередині тої ж самої файлової системи (тобто, на тому ж самому носії, на якому перебуває каталог, що містить це ім'я).

Але в Linux є інший тип посилань, так звані символічні посилання. Ці посилання теж можуть розглядатися як додаткові імена файлів, але в той же час вони представляються окремими файлами — файлами типу символічних посилань. На відміну від жорстикх посилань символічні посилання можуть указувати на файли, розташовані в іншій файловій системі, наприклад, на носії, що монтується, або навіть на іншому комп'ютері. Якщо вихідний файл вилучений, символічне посилання не віддаляється, але стає марним. Використовуйте символічні посилання в тих випадках, коли хочете уникнути плутанини, пов'язаної із застосуванням жорстких посилань.

Створення будь-якого посилання зовні подібно копіюванню файлу, але фактично як вихідне ім'я файлу, так і посилання вказують на той самий реальний файл на диску. Тому, наприклад, якщо ви внесли зміни у файл, звернувшись до нього під одним ім'ям, ви виявите ці зміни й тоді, коли звернетеся до файлу по ім'ю-посиланню. Для того, щоб створити символічне посилання, використовується вже згадувана команда ln з
додатковою опцією -s:

ln -s ім'я_файлу_або_каталогу ім'я_посилання

Права доступу до файлів і каталогів

В основі механізмів розмежування доступу лежать імена користувачів і імена груп користувачів. Ви вже знаєте, що в Linux кожний користувач має унікальне ім'я, під яким він входить у систему (логується). Крім того, у системі створюється деяке число груп користувачів, причому кожний користувач може бути включений в одну або кілька груп. Створює й видаляє групи суперкористувач, він же може змінювати состав учасників тої або іншої групи.

Члени різних груп можуть мати різні права по доступі до файлів, наприклад, група адміністраторів може мати більше прав, чим група програмістів.

В індексному дескрипторі кожного файлу записане ім'я так званого власника файлу групи, що має права на цей файл. Спочатку, при створенні файлу його власником оголошується той користувач, що цей файл створив. Точніше — той користувач, від чийого ім'я запущений процес, що створює файл. Група теж призначається при створенні файлу — по ідентифікаторі групи процесу, що створює файл. Власника й групу файлу можна поміняти в ході подальшої роботи за допомогою команд chown і chgrp
(докладніше про неї буде сказано трохи пізніше).

Тепер давайте ще раз виконаємо команду ls -l. Але задамо їй як додатковий параметр ім'я конкретного файлу, наприклад, файлу, що задає саму команду ls. (Звернете, до речі, увага на цю можливість команди ls -l — одержати інформацію про конкретний файл, а не про всі файли каталогу відразу).

[user]$ ls -l /bin/ls

-rwxr-xr-x 1 root root 49940 Sep 12 1999 /bin/ls

Ви бачите, що в цьому випадку власником файлу є користувач root і група root. Але нас зараз у виводі цієї команди більше цікавить перше поле, що визначає тип файлу й права доступу до файлу. Це поле в наведеному прикладі представлено ланцюжком символів -rwxr-xr-x. Ці символи можна умовно розділити на 4 групи.

Перша група, що складається з єдиного символу, визначає тип файлу. Цей символ відповідно до можливих типів файлів, розглянутими в попередньому розділі, може приймати такі значення:

- \- — звичайний файл;
- d — каталог;
- b — файл блокового пристрою;
- c — файл символьного пристрою;
- s — доменне гніздо (socket);
- p — іменований канал (pipe);
- l — символічне посилання (link).

Далі йдуть три групи по три символи, які й визначають права доступу до файлу відповідно для власника файлу, для групи користувачів, що зіставлена даному файлу, і для всіх інших користувачів системи. У нашім прикладі права доступу для власника визначені як rwx, що означає, що власник (root) має право читати файл (r), робити запис у цей файл (w), і запускати файл на виконання (x). Заміна кожного із цих символів прочерком буде означати, що користувач втрачає відповідного права. У тім же прикладі ми бачимо, що всі інші користувачі (включаючи й тих, які ввійшли в групу root) позбавлені права запису в цей файл, тобто не можуть файл редагувати й взагалі якось змінювати.

Взагалі кажучи, права доступу й інформація про тип файлу в UNIX-Системах зберігаються в індексних дескрипторах в окремій структурі, що складається із двох байтів, тобто з 16 біт (це природно, адже комп'ютер оперує бітами, а не символами r, w, x). Чотири біти із цих 16-ти відведені для кодованого запису про тип файлу. Наступні три біти задають особливі властивості файлів, що виконуються, про які ми скажемо трохи пізніше. І, нарешті, 9 біт, що

залишилися визначають права доступу до файлу. Ці 9 біт розділяються на 3 групи по трьох біта. Перші три біти задають права користувача, наступні три біти — права групи, останні 3 біти визначають права всіх інших користувачів (тобто всіх користувачів, за винятком власника файлу й групи файлу).

При цьому, якщо відповідний біт має значення 1, то право надається, а якщо він дорівнює 0, то право не надається. У символьній формі запису прав одиниця заміняється відповідним символом (r, w або x), а 0 представляється прочерком.

Право на читання (r) файлу означає, що користувач може переглядати вміст файлу за допомогою різних команд перегляду, наприклад, командою more або за допомогою будь-якого текстового редактора. Але, зробивши зміні змісту файлу в текстовому редакторі, ви не зможете зберегти зміни у файлі на диску, якщо не маєте права на запис (w) у цей файл. Право на виконання (x) означає, що ви можете завантажити файл в пам’ять й спробувати запустити його на виконання як програму, що виконується як. Звичайно, якщо в дійсності файл не є програмою (або скріптом shell), те запустити цей файл на виконання не вдасться, але, з іншого боку, навіть якщо файл дійсно є програмою, але право на виконання для нього не встановлене, то він теж не запуститься.

Для зміни прав доступу до файлу використовується команда chmod. Її можна використовувати у двох варіантах. У першому варіанті ви повинні явно вказати, кому яке
право даєте або кого цього права позбавляєте:

[user]$ chmod wXp ім'я-файлу

де замість символу w підставляється

- або символ u (тобто користувач, що є власником);
- або g (група);
- або o (всі користувачі, що не входять у групу, який належить даний файл);
- або a (всі користувачі системи, тобто й власник, і група, і всі інші). Замість X ставиться:
- або + (надаємо право);
- або – (лишаєм відповідного права);
- або = (установити зазначені права замість наявних),

Замість p — символ, що позначає відповідне право:

- r (читання);
- w (запис);
- x (виконання).

От кілька прикладів використання команди chmod:

[user]$ chmod a+x file_name

надає всім користувачам системи право на виконання даного файлу.

[user]$ chmod go-rw file_name

видаляє право на читання й запис для всіх, крім власника файлу.

[user]$ chmod ugo+rwx file_name

дає всім права на читання, запис і виконання.

Команди для роботи з файлами й каталогами

У попередніх розділах ми вже згадували деякі команди для роботи з файлами й каталогами: pwd, cd, ls, ln, chmod. У цьому розділі розглянемо (дуже коротко) ще
деякі команди, що використовуються досить часто.

Команди *chown* і *chgrp*

Ці команди служать для зміни власника файлу й групи файлу. Виконувати зміну власника може тільки суперкористувач, зміну групи може виконати сам власник файлу або суперкористувач. Для того, щоб мати право змінити групу, власник повинен додатково бути членом тої групи, який він хоче дати права на даний файл. Формат цих двох команд аналогічний:

[root]# chown vasja ім'я-файлу [root]#
chgrp usersgrp ім'я-файлу

Команда *mkdir*

Команда mkdir дозволяє створити підкаталог у поточному каталозі. Як аргумент цій команді треба дати ім'я створюваного каталогу. У знову створеному каталозі автоматично створюються два записи: ‘**.**’ (посилання на цей самий каталог) і ‘**..**’ (посилання на батьківський каталог). Щоб створити підкаталог, ви повинні мати в поточному каталозі право запису. Можна створити підкаталог не в поточному, а в якімсь іншому каталозі, але
тоді необхідно вказати шлях до створюваного каталогу:

[user]$ mkdir /home/kos/book/glava5/part1

Команда mkdir може використовуватися з наступними опціями:

1. -m mode — задає режим доступу для нового каталогу (наприклад, -m 755);
2. -p — створювати зазначені проміжні каталоги (якщо вони не існують).

Команда *cat*

Команда cat часто використовується для створення файлів (хоча можна скористатися й командою touch). По команді cat на стандартний вивід (тобто на екран) виводиться вміст зазначеного файлу (або декількох файлів, якщо їхні імена послідовно задати як аргументи команди). Якщо вивід команди cat перенаправляти у файл, то можна одержати копію якогось файлу:

[user]$ cat file1 > file2

Первісне призначення команди cat саме й припускало перенапрямок виводу, тому
що ця команда створена для конкатенації, тобто об'єднання декількох файлів в один:

[user]$ cat file1 file2 ... file > new-file

Саме можливості перенапрямку вводу й виводу цієї команди й використовуються для створення нових файлів. Для цього на вхід команди cat направляють дані зі
стандартного ввводу (тобто із клавіатури), а вивід команди — у новий файл:

[user]$ cat > newfile

Після того, як ви надрукуєте все, що хочете, натисніть комбінацію клавіш <Ctrl>+<D> або <Ctrl>+<C>, і все, що ви ввели, буде записане в newfile. Звичайно, у
такий спосіб створюються, в основному, короткі текстові файли.

Команда *cp*

Хоча для копіювання файлів іноді користуються командою cat, але в Linux існує для цього спеціальна команда cp. Її можна застосовувати в одній із двох форм:

[user]$ cp [options] source destination
[user]$ cp [options] source_directory new_directory

У першому випадку файл або каталог source копіюється, відповідно, у файл або каталог destination, а в другому випадку файли, що знаходяться в каталозі source_directory копіюються в каталог new_directory. Для копіювання треба мати права на читання файлів, які копіюються, і права на запис у каталог, у який виробляється копіювання.

Якщо в якості цільового вказується існуючий файл, то його вміст буде затерто, тому при копіюванні треба дотримувати обережності. Втім, можна використовувати команду cp опцією -i, тоді перед перезаписом існуючого файлу буде запитуватися підтвердження (дуже рекомендую вам завжди використовувати цю опцію).

У команди cp є ще кілька корисних опцій (табл. 15.1).

Таблиця 15.1. Основні опції команди cp

Опція	Значення
-p	Зберігає час модифікації файлу й максимально можливі повноваження. Без цієї опції для нового файлу задаються повноваження, що відповідають повноваженням користувача, що запустив команду
-R або -r	Якщо source - каталог, то копіюється як він, так і всі вхідні в нього підкаталоги, тобто зберігається вихідна форма дерева каталогів
-d	Якщо задати цю опцію, то символічні посилання будуть залишатися посиланнями (а інакше замість посилання копіюється файл, на який дається посилання)
-f	Перезаписувати файли при копіюванні (якщо такі вже є) без додаткових попереджень

Команда *mv*

Якщо вам необхідно не скопіювати, а перемістити файл із одного каталогу в інший, ви можете скористатися командою mv. Синтаксис цієї команди аналогічний синтаксису команди cp. Більше того, вона спочатку копіює файл (або каталог), а тільки потім видаляє вихідний файл (каталог). І опції в неї такі ж, як в cp.

Команда mv може використовуватися не тільки для переміщення, але й для перейменування файлів і каталогів (тобто переміщення їх усередині одного каталогу). Для цього треба просто задати як аргументи старе й нове ім’я файлу:

[user]$ mv oldname newname

Але команда mv не дозволяє перейменувати відразу кілька файлів (використовуючи шаблон ім'я), так що команда mv *.xxx *.yyy не буде працювати.

При використанні команди mv, також як і при використанні cp, не забувайте застосовувати опцію -i для того, щоб одержати попередження, коли файл буде перезаписуватися.

Команди *rm* і *rmdir*

Для видалення непотрібних файлів і каталогів в Linux служать команди rm (видаляє файли) і rmdir (видаляє порожній каталог). Для того, щоб скористався цими командами, ви повинні мати право запису в каталозі, у якому розташовані видаляються файли, що, або каталоги. При цьому повноваження на зміну самих файлів не обов'язкові. Якщо хочете перед видаленням файлу одержати додатковий запит на підтвердження операції, використовуйте опцію -i. Якщо ви спробуєте використовувати команду rm (без усяких опцій) для видалення каталогу, то буде видане повідомлення, що це каталог, і видаленя не відбудеться. Для видаленя каталогу треба видалити в ньому всі файли, після чого видалити сам каталог за допомогою команди rmdir. Однак можна видалити й непустий каталог з усіма вхідними в нього підкаталогами й файлами, якщо використовувати команду rm з опцією -r. Якщо ви дасте команду rm *, то видалите всі файли в поточному каталозі. Підкаталоги при цьому не вийдуть. Для видаленя як файлів, так і підкаталогів поточного каталогу треба теж скористатися опцією -r. Однак завжди пам’ятайте, що в Linux немає команди відновлення файлів після їхнього видаленя!

Команди *more* і *less*

Команда cat дозволяє вивести на стандартний вивід (на екран) зміст будь-якого файлу, однак вона використовується для цих цілей дуже рідко, хіба що для виводу дуже невеликих по обсягу файлів. Справа в тому, що вміст великого файлу миттєво проскакує на екрані, і користувач бачить тільки останні рядки файлу. Тому cat використовується в основному по її прямому призначенню — для конкатенації файлів, а для перегляду вмісту файлів (звичайно, текстових) використовуються команди more і less (або текстові редактори).

Команда-фільтр more виводить уміст файлу на екран окремими сторінками, розміром саме в цілий екран. Для того, щоб побачити наступну сторінку, треба нажати на клавішу пробілу. Натискання на клавішу <Enter> приводить до зсуву на один рядок. Крім клавіш пробілу й <Enter> у режимі паузи ще деякі клавіші діють як керуючі (наприклад, клавіша <B> повертає вас на один екран назад), але ми тут не будемо приводити повного їхнього переліку, як і переліку опцій команди. Вам для початку треба ще тільки запам'ятати, що вийти з режиму перегляду можна за допомогою клавіші <Q>, тому що якщо ви цього не знаєте, то вам доведеться довго натискати пробіл, поки ви не доберетеся до кінця довгого файлу. Про всі опції команди more ви можете прочитати в інтерактивному керівництві man або info.

Утиліта less, розроблена в рамках проекту GNU, містить всі функції й команди керування виводом, наявні в програмі more, і деякі додаткові, наприклад, дозволяє
використовувати клавіші керування курсором (<Стрілка нагору>, <Стрілка вниз>, <PgUp>, <PgDown>) для переміщення по тексту. Згадаєте, ми вже говорили про це, коли розглядали інтерактивну підказку man.

Команди more і less дозволяють робити пошук підрядка в переглядається файлі, що, причому команда less дозволяє робити пошук як у прямому, так і в зворотньому
напрямку. Для організації пошуку рядка символів string треба набрати в командному рядку програми в нижній частині екрана (там, де двокрапка) /string. Якщо рядок, що шукається, буде знайдений, буде відображений відповідний шматок тексту, причому знайдений рядок буде перебувати в самому верху екрана.

Команда *find* і символи шаблонів для імен файлів

Ще однією часто використовуваною командою для роботи з файлами в Linux є команда пошуку потрібного файлу find. Команда find може шукати файли по ім'ю,
розміру, даті створення або модифікації й деяким іншим критеріям. Загальний синтаксис команди find має такий вигляд:

find [список_каталогів] критерій_пошуку

Порівняння файлів

Linux також має інструменти для порівняння файлів. Найпростіший з них — команда cmp. Ця команда просто порівнює вміст двох файлів
по-байтно:

[user]$ cmp file1 file2

Якщо файли повністю збігаються, вона мовча закінчує свою роботу (відбувається повернення до командного рядка без яких- небудь додаткових повідомлень), а якщо файли розрізняються, видаються номер рядка й номер байта в рядку, де має місце перше розходження. Звичайно, інформації, що видається командою cmp, обмаль для того, щоб прийняти, наприклад, рішення про те, який із двох файлів нам більше важливий. Тому варто скористатися командою diff для одержання повного звіту про те, які ж розходження є в файлах. Для одержання звіту досить указати команді, які саме файли порівнювати:

[user]$ diff paper.old paper.new

Звіт про виявлені розходження буде виданий на стандартний вихід. Звісно, його краще перенаправляти у файл:

[user]$ diff paper.old paper.new >paper.diff