Львівський державний університет безпеки життєдіяльності

Кафедра інформаційних технологій
та систем електронних комунікацій

Лекція

з дисципліни: «Клієнт-серверне програмування»

на тему: «*Операційна ситема Linux: принципи роботи з файловою системою*»

ПЛАН ЛЕКЦІЇ

1. Основні функції ОС.
2. Класифікації ОС.
3. Архітектура ОС.
4. Архітектура Unix/Linux.
5. Характерні риси Linux як ОС.
6. Файлова система Linux.
7. Структура каталогів Linux.
8. Команди роботи з файлами та директоріями.
9. Доступ до файлів і каталогів.

ЛІТЕРАТУРА

1. Andrew Tanenbaum, Herbert Bos, Modern Operating Systems 4th Edition, 2008 by Pearson Education, Inc., Upper Saddle River, New Jersey, 1150p.
2. Семеренко В. П., Крилик Л. В. Операційна система Linux.- Навчальний посібник.-Вінниця: ВНТУ, 2006. - 88 с.

1. Основні функції ОС.

Серед всіх системних програм особливе місце займає операційна система (ОС) – програма (частіше набір програм), яка завантажується при включенні комп'ютера. Вона ізолює апаратне забезпечення комп'ютера від прикладних програм користувачів, які взаємодіють з комп'ютером через інтерфейси ОС.

Таким чином, ОС представляється користувачеві у вигляді розширеної або віртуальної машини, яку легше програмувати і з якою легше працювати, ніж безпосередньо з апаратурою, що становить реальну машину.

До основних функцій ОС можна віднести:

1. Прийом від користувача завдань, або команд, сформульованих на відповідній мові, і їх обробка. Команди які пов'язані, перш за все, з запуском (припиненням, зупинкою) програм, з операціями над файлами (отримати перелік файлів в поточному каталозі, створити, перейменувати, скопіювати, перемістити файл і т.п.) і інші команди.

2. Завантаження в оперативну пам'ять програм, які підлягають виконанню.

3. Розподіл пам'яті, організація віртуальної пам'яті.

4. Запуск програми (передача їй управління, в результаті чого процесор виконує програму).

5. Ідентифікація всіх програм та даних.

6. Прийом і виконання різних запитів від програм, які виконуються. Звернення здійснюється за відповідними правилами, які і визначають інтерфейс прикладного програмування (Application Programming Interface, API) цієї ОС.

7. Обслуговування всіх операцій введення-виведення (Input/Output, I/O).

8. Забезпечення роботи систем управління файлами (СУФ) та/або систем управління базами даних (СУБД), що дозволяє різко збільшити ефективність всього ПЗ.

9. Забезпечення режиму мультипрограмування, тобто організація паралельного виконання двох або більш програм на одному процесорі, що створює видимість їх одночасного виконання.

10. Планування та диспетчеризація задач відповідно до заданих стратегії та дисциплін обслуговування.

11. Організація механізмів обміну повідомленнями і даними між програмами, які виконуються.

12. Для мережевих ОС характерною є функція забезпечення взаємодії пов'язаних між собою комп'ютерів.

13. Захист однієї програми від впливу іншої, забезпечення збереження даних, захист самої ОС від програм, які виконуються під її керівництвом.

14. Аутентифікація та авторизація користувачів. Аутентифікація – процедура перевірки імені користувача і його пароля на відповідність тим значенням, які зберігаються в його профілі. Авторизація означає, що

відповідно до облікового запису користувача, який пройшов аутентифікацію, йому (і всім запитам, які він здійснює в ОС від свого імені), призначаються певні права (привілеї), що визначають дії, які він може виконувати на комп'ютері.

15. Задоволення жорстких обмежень на час відповіді в режимі реального часу (характерно для ОС реального часу).

16. Забезпечення роботи систем програмування.

17. Надання функцій відновлення системи після збою.

До найбільш важливих функцій можна віднести наступні:

1. Реалізація абстрактної машини, що дає можливість працювати кінцевому користувачеві.

2. Раціональний розподіл обчислювальних ресурсів (зовнішньої та внутрішньої пам'яті, процесорного часу).

3. Захист інформації від несанкціонованого доступу.

4. Організація функціонування та взаємодії обчислювальних процесів.

2. *Класифікації ОС.*

ОС можуть відрізнятися особливостями реалізації внутрішніх алгоритмів керування основними ресурсами комп'ютера (процесорами, пам'яттю, пристроями), особливостями використаних методів проектування, типами апаратних платформ, областями використання та іншими властивостями.

Нижче наведено класифікацію ОС по декількох найбільш основних ознаках.

Залежно від особливостей використовуваного алгоритму управління процесором, ОС ділять на:

1. Багатозадачні та однозадачні.

2. Багатокористувацькі та однокористувацькі.

3. З підтримкою багатопроцесорності та ті що працюють з однопроцесорними архітектурами.

Багатозадачні ОС відрізняються від однозадачних наявністю функцій управління поділом ресурсів, які спільно використовуються, таких як процесор, оперативна пам'ять, файли та зовнішні пристрої.

Головною відмінністю багатокористувацьких систем від систем, що підтримують тільки одного користувача, є наявність засобів захисту інформації кожного користувача від несанкціонованого доступу інших користувачів.

Не кожна багатозадачна ОС є багатокористувацькою, і не кожна однокористувацька ОС є однозадачною.

Найважливішим ресурсом є *процесорний час*. Спосіб розподілу процесорного часу між декількома одночасно існуючими в системі процесами або нитками багато в чому визначає специфіку ОС. Багатозадачність забезпечує можливість паралельної (або псевдопаралельної) обробки декількох процесів.

Серед безлічі існуючих варіантів реалізації багатозадачності можна виділити дві групи алгоритмів:

– невитісняюча багатозадачність (NetWare, Windows 3.x);

– витісняюча (витискальна) багатозадачність (Windows NT, OS/2, Unix, Mac OS).

Основною відмінністю між цими варіантами багатозадачності є ступінь централізації механізму планування процесів.

Невитісняюча багатозадачність (Non-preemptive multitasking) – це спосіб планування процесів (потоків), при якому активний процес (потік) виконується до того часу, доки він сам, за власною ініціативою, не віддасть керування планувальнику ОС для того, щоб той вибрав з черги інший, готовий до виконання процес (потік).

Витісняюча багатозадачність (Preemptive multitasking) – це вид багатозадачності, при якому ОС може тимчасово перервати поточний процес (потік) без будь-якої допомоги з його боку. Завдяки цьому, програми, які довго експлуатують процесор (“підвислі”), як правило, не порушують роботу ОС.

Справжня багатозадачність ОС можлива тільки в багатопроцесорних, кількаядерних системах, або в розподілених обчислювальних системах.

Важливою властивістю сучасних ОС є відсутність або наявність в них засобів підтримки багатопроцесорної обробки – багатопроцесування (мультипроцеси). Воно призводить до ускладнення всіх алгоритмів керування ресурсами.

В системі з багатопроцесорною архітектурою багатопроцесорні ОС можуть класифікуватися за способом організації обчислювального процесу на:

– асиметричні ОС;

– симетричні ОС.

Асиметрична ОС цілком виконується тільки на одному з процесорів системи, розподіляючи прикладні завдання по іншим процесорам.

Симетрична ОС повністю децентралізована та використовує весь пул процесорів, поділяючи їх між системними і прикладними завданнями.

В основі новітніх версій сучасних ОС лежить архітектура симетричної багатопроцесорної обробки (Symmetric Multiprocessing, SMP), яка дозволяє значно скоротити час обробки поставлених завдань.

Багатозадачні ОС підрозділяються на три типи відповідно до критеріїв ефективності, які були використані при їх розробці:

1. Системи пакетної обробки.

2. Системи поділу часу.

3. Системи реального часу.

Системи пакетної обробки призначалися для вирішення завдань, в основному, обчислювального характеру, які не потребують швидкого отримання результатів. Головною метою та критерієм ефективності систем пакетної обробки є максимальна пропускна здатність, тобто рішення

максимального числа завдань в одиницю часу. Для досягнення цієї мети в таких системах використовується наступна схема функціонування.

На початку роботи формується пакет завдань. Кожне завдання містить вимоги до системних ресурсів. З цього пакету завдань формується мультипрограмна суміш, тобто безліч завдань, які будуть одночасно виконуватися. Для одночасного виконання вибираються завдання, які відрізняються вимогами до ресурсів, так, щоб забезпечувалася збалансоване завантаження всіх пристроїв обчислювальної машини.

Наприклад, в мультипрограмній суміші бажано одночасна присутність обчислювальних задач та задач з інтенсивним введенням-виведенням (I/O). Таким чином, вибір нового завдання з пакету завдань залежить від внутрішньої ситуації, що складається в системі, тобто вибирається "вигідне" завдання.

Недолік систем пакетної обробки – низька ефективність роботи користувача.

Системи поділу часу покликані виправити основний недолік систем пакетної обробки – ізоляцію користувача-програміста від процесу виконання його завдань. Кожному користувачеві системи поділу часу надається термінал, з якого він може вести діалог зі своєю програмою. Так як в системах поділу часу кожній задачі виділяється тільки квант процесорного часу, жодна задача не займає процесор надовго, і час відповіді виявляється прийнятним. Якщо квант обраний досить невеликим, то у всіх користувачів, що одночасно працюють на одній і тій же машині, складається враження, що кожен з них одноосібно використовує машину. Такі системи мають меншу пропускну спроможність, ніж системи пакетної обробки, так як на виконання приймається кожна запущена користувачем задача, а не та, яка "вигідна" системі, і, крім того, є накладні витрати обчислювальної потужності на більш часте переключення процесора з задачі на задачу.

Критерієм ефективності систем поділу часу є не максимальна пропускна здатність, а зручність та ефективність роботи користувача.

Системи реального часу застосовуються для керування різними технічними об'єктами, такими, наприклад, як верстат, супутник, наукова експериментальна установка або технологічними процесами, такими, як гальванічна лінія, доменний процес і т.п. У всіх цих випадках існує гранично допустимий час, протягом якого повинна бути виконана та чи інша програма, що керує об'єктом, в іншому випадку може статися аварія.

Критерієм ефективності для таких систем є їх здатність витримувати заздалегідь задані інтервали часу між запуском програми й одержанням результату (керувального впливу). Цей час називається часом реакції системи, а відповідна властивість системи – реактивністю.

Для цих систем мультипрограмна суміш являє собою фіксований набір заздалегідь розроблених програм, а вибір програми на виконання здійснюється виходячи з поточного стану об'єкта або відповідно до розкладу планових робіт.

3. *Архітектура ОС.*

Найбільш загальним підходом до структуризації ОС є поділ усіх її модулів на дві групи:

1. ядро – модулі, що виконують основні функції ОС;
2. модулі, що виконують допоміжні функції ОС.

Обчислювальну систему, що працює під управлінням ОС на основі ядра, можна розглядати як систему, що складається з трьох ієрархічно розташованих шарів: апаратура, ядро та утиліти з системними програмами та бібліотеками (рис. 1).

Рис. 1 Тришарова схема обчислювальної системи

Кожен шар обслуговує розміщений вище шар, виконуючи для нього певний набір функцій, які утворюють міжшаровий інтерфейс.

Модулі ядра виконують такі базові функції ОС, як управління процесами, пам'яттю, пристроями I/O і т.п.

До складу ядра входять функції, які вирішують внутрішньосистемні задачі організації обчислювального процесу, такі як перемикання контекстів виконання, завантаження/вивантаження сторінок пам'яті, обробка переривань. Ці функції не доступні для програм користувачів.

Інший клас функцій ядра служить для підтримки програм, створюючи для них так зване прикладне програмне середовище. Додатки можуть звертатися до ядра з запитами – системними викликами – для виконання тих чи інших дій, наприклад, для відкриття та читання файлу, виведення графічної інформації на монітор, отримання системного часу і т.п. Функції ядра, які можуть викликатися програмами, утворюють інтерфейс прикладного програмування (Application Programming Interface) – API.

Функції, що виконуються модулями ядра, є функціями ОС, які найбільш часто використовують і тому швидкість їх виконання визначає продуктивність всієї системи в цілому.

Для забезпечення високої швидкості роботи ОС всі модулі ядра або велика їх частина постійно знаходяться в оперативній пам'яті (оперативному запам’ятовуючому пристрої, ОЗП), тобто є резидентними.

Допоміжні модулі ОС звичайно підрозділяються на наступні групи:

– утиліти – програми, які вирішують окремі завдання управління і супроводу комп'ютерної системи, такі, наприклад, як програми стиснення дискового простору, архівування даних і ін.;

– системні програми обробки – текстові або графічні редактори, компілятори, компонувальники, відладчики;

– програми надання користувачу додаткових послуг – спеціальний варіант інтерфейсу користувача, калькулятор, ігри і т.д.

– бібліотеки процедур різного призначення, які спрощують розробку програм, наприклад бібліотека математичних функцій, функцій I/O і т.д.

Режими роботи ядра ОС

Для надійного управління ходом виконання програм, ОС повинна мати по відношенню до програм певні привілеї. Інакше некоректно працююча програма може втрутитися в роботу ОС і, наприклад, зруйнувати частину її кодів.

Забезпечити привілеї ОС неможливо без спеціальних засобів апаратної підтримки. Апаратура комп'ютера повинна підтримувати як мінімум два режими роботи:

– режим користувача (user mode);

– привілейований режим, який також називають режимом ядра (kernel mode), або режимом супервізора (supervisor mode).

Підвищення стійкості ОС, яке забезпечується переходом ядра в привілейований режим, досягається за рахунок деякого уповільнення виконання системних викликів. Системний виклик привілейованого ядра ініціює перемикання процесора з призначеного для користувача режиму в привілейований, а при поверненні до програми – назад в режим користувача (рис. 3.2). У всіх типах процесорів через додаткову дворазову затримку перемикання перехід на процедуру зі зміною режиму виконується повільніше, ніж виклик процедури без зміни режиму.

Архітектура ОС, заснована на привілейованому ядрі та програмах для режиму користувача, стала, по суті, класичною. Її використовує багато ОС, наприклад, різні версії Unix, VAX VMS, IBM OS/390, OS/2, і з певними модифікаціями – Windows NT.

Розрізняють такі типи архітектур ядер ОС:

1. Монолітне ядро (BSD UNIX, Linux, MS-DOS, KolibriOS і ін.).

2. Модульне ядро.

3. Мікроядро (Windows CE, Symbian OS, OpenVMS, Mach, QNX, AIX, Minix та ін.).

4. Екзоядро.

5. Наноядро (KeyKOS).

6. Гібридне ядро (деякі варіанти Unix/Linux, лінійка Windows NT).

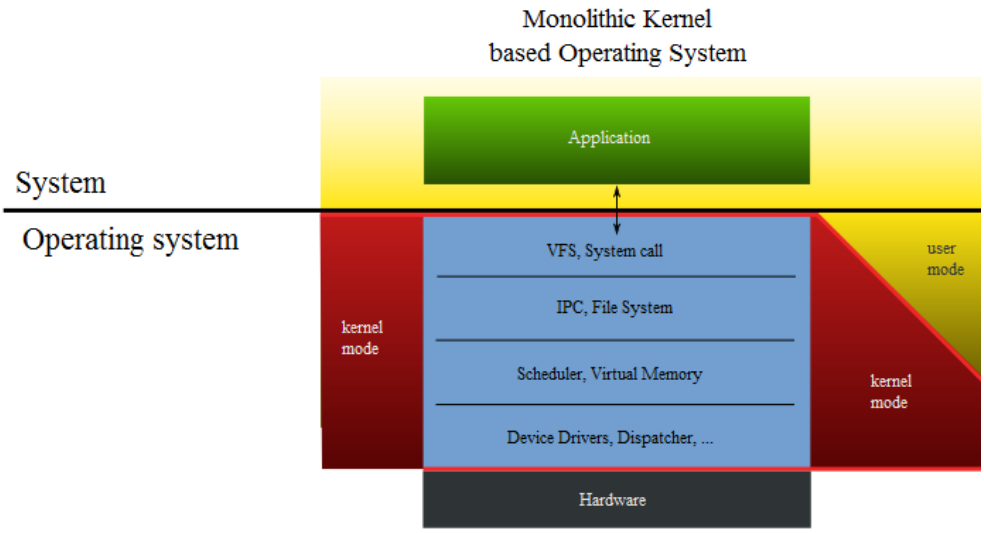

Рис. 2 Структура ОС на базі монолітного ядра

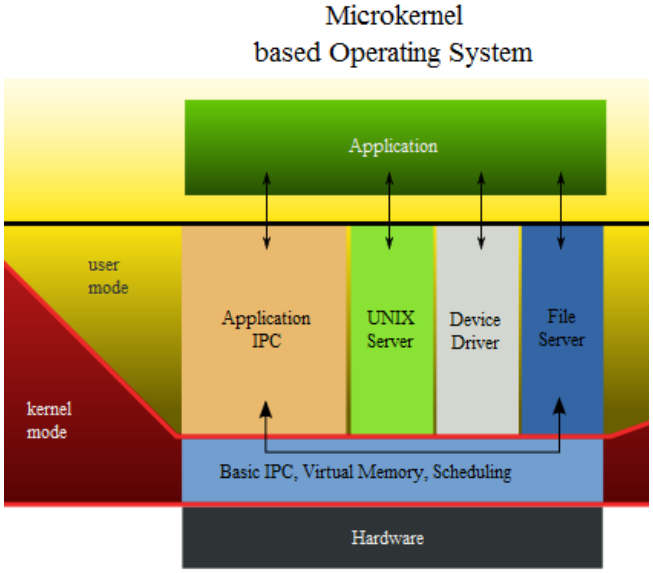

Рис. 3. Структура ОС на базі мікроядра

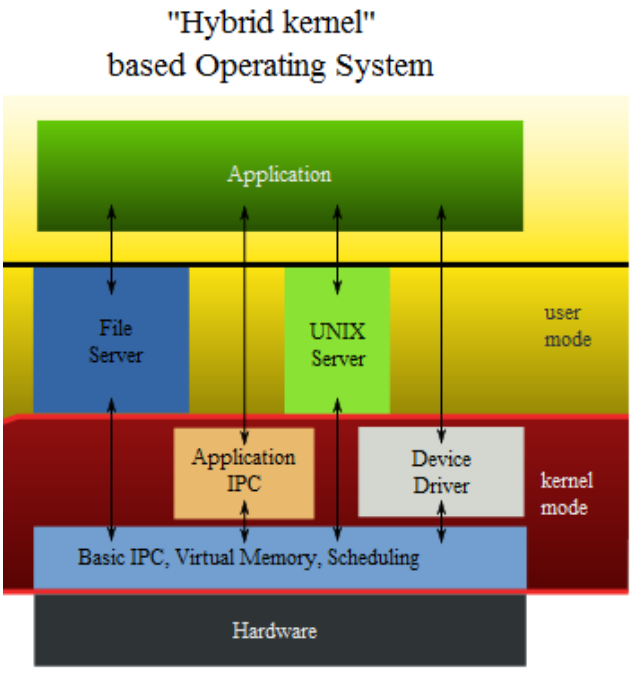

Рис. 4. Структура ОС на базі гібридного ядра

ОС на базі монолітного ядра

Організація ОС на монолітному ядрі є найпоширенішим та найстарішим способом – ОС працює як єдина програма в режимі ядра, написана у вигляді набору процедур. Служби, системні виклики, що надаються ОС, запитуються шляхом переміщення параметрів в чітко визначене місце, наприклад, в стек. Потім виконується спеціальна інструкція ОС, що перемикає машину з призначеного для користувача режиму в режим ядра і передає управління ОС. ОС витягує параметри і визначає, який системний виклик повинен бути виконаний. Після цього ОС переміщається по індексу в таблиці на рядок стека, що містить покажчик на процедуру, яка виконує системний виклик.

Мікроядерний підхід

ОС на базі мікроядра мають високу надійність за рахунок розбиття ОС на невеликі модулі. Тільки один з них – мікроядро – запускається в режимі ядра (привілейованому режимі), а всі інші запускаються у вигляді відносно слабко наділених повноваженнями звичайних процесів користувачів. Якщо, наприклад, в драйвері будь-якого пристрою відбудеться помилка, то вона не призведе до повного зависання комп'ютера, тільки викличе неможливість доступу до цього пристрою.

Функції ОС більш високого рівня виконують спеціалізовані компоненти – сервери, що працюють в режимі користувача. При такій побудові ОС працює більш повільно, так як часто виконуються переходи між привілейованим режимом і режимом користувача, однак система виходить більш гнучкою – її функції можна модифікувати, модифікуючи сервери режиму користувача. Крім того, сервери добре захищені один від одного, як і будь-які призначені для користувача процеси. Найбільш часто цей принцип побудови закладений в ОС реального часу, що працюють в промислових пристроях, авіоніки і військовій техніці.

4. *Архітектура Unix/Linux.*

Дворівнева модель системи представлена на рис. 5.

Рис. 5 Спрощена модель системи Unix

У центрі знаходиться ядро системи (kernel). Ядро безпосередньо взаємодіє з апаратною частиною комп'ютера, ізолюючи прикладні програми від її власної архітектури. Ядро має набір послуг, що надаються прикладним

програмам. До послуг ядра відносяться операції введення/виведення (відкриття, читання, запису та управління файлами), створення та управління процесами, їх синхронізації і взаємодії між процесами. Ядро розподіляє пам'ять і забезпечує доступ до файлів та периферійних пристроїв. Всі програми запитують послуги ядра за допомогою системних викликів. Структурна схема ядра представлена на рис. 6.

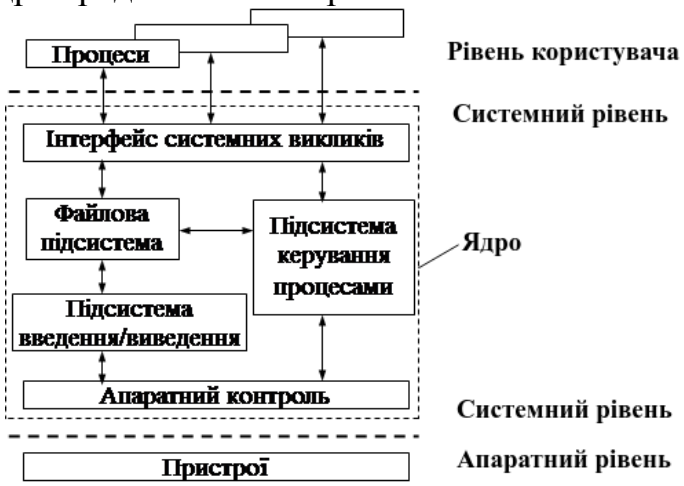

Рис. 6. Внутрішня структура ядра Unix

Другий рівень в моделі системи Unix складають програму та завдання, як системні, що визначають функціональність системи, так і прикладні, що забезпечують інтерфейс користувача Unix. Однак, незважаючи на зовнішню різнорідність програм, схеми їх взаємодії з ядром однакові.

Стан системи, в якому знаходиться ядро, і область пам'яті, в якій знаходиться ядро, разом називаються *простором ядра* (або режимом ядра, kernel-space). Відповідно, призначені для користувача програми виконуються в просторах завдань (режим користувача, режим завдань, user-space). Програмам користувача доступним є лише деяка підмножина машинних ресурсів, вони не можуть виконувати деякі системні функції, безпосередньо звертатися до апаратури і робити інші заборонені операції. При виконанні програмного коду ядра система знаходиться в просторі (режимі) ядра, на відміну від нормального виконання призначених для користувача програм, яке відбувається в режимі завдання.

Прикладні програми, що працюють в системі, взаємодіють з ядром за допомогою інтерфейсу *системних викликів* (system call) (рис. 7).

Коли прикладна програма виконує системний виклик, то кажуть, що ядро виконує роботу від імені прикладної програми. Більш того, говорять, що прикладна програма виконує системний виклик в просторі ядра, а ядро виконується в контексті процесу. Такий тип взаємодії, коли прикладна програма звертається до ядра через інтерфейс системних викликів, є фундаментальним способом виконання завдань.

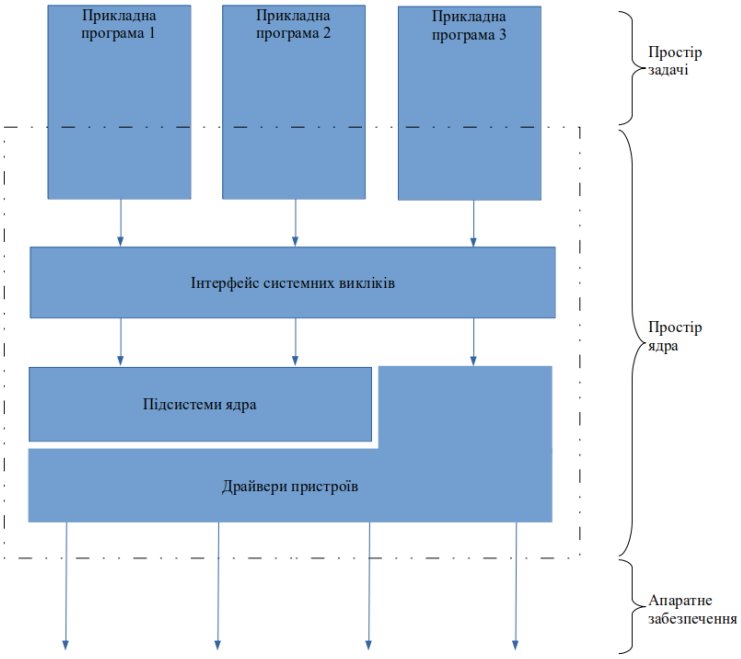

Рис. 7. Взаємодія між прикладними програмами, ядром та апаратним забезпеченням

Загальну архітектуру ОС на базі ядра GNU/Linux часто розглядають як піраміду (рис. 8).

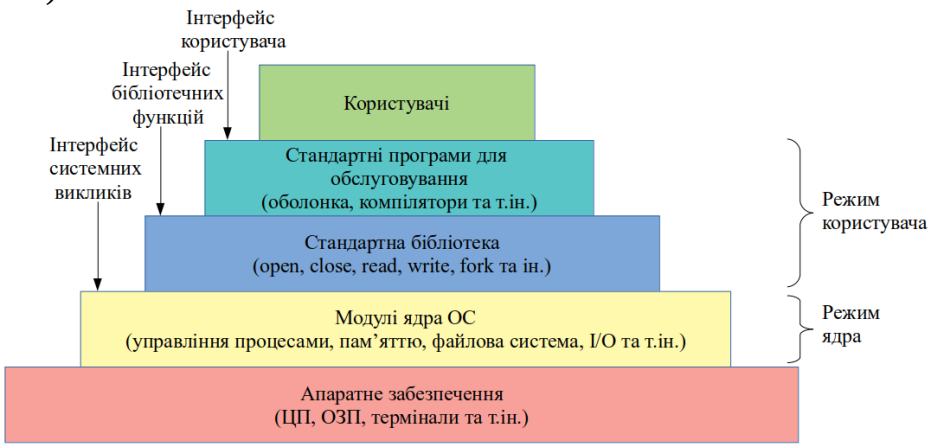

Рис. 8. Рівні ОС на базі ядра GNU/Linux

5. *Характерні риси Linux як ОС.*

Linux – це операційна система, яка створена на основі загальновідомої системи Unix. Датою народження Linux є 1991 рік. Саме цього року фінський студент Лінус Торвальдс написав невелику системну програму, що дозволяла лише керувати процесами та основною пам'яттю комп'ютера, і звернувся до всіх програмістів із закликом продовжити його роботу. Зусиллями багатьох ентузіастів зі всього світу вже через декілька місяців була створена закінчена операційна система сімейства Unix. Сьогодні Linux стоїть в одному ряду з найпотужнішими операційними системами і продовжує далі розвиватись і розширювати свої функціональні можливості. Жодна серйозна фірма програмного профілю не може ігнорувати цю операційну систему і тому більшість програмних пакетів мають свої версії і для Linux.

Linux функціонує практично на всіх апаратних платформах і підтримує більше типів процесорів і програмних систем, ніж будь-яка інша операційна система. Linux однаково добре працює як на персональних комп'ютерах, так і в комп'ютерних мережах. Її висока мобільність обумовлена як спадковістю від Unix, так і завдячуючи широкій підтримці багатьох програмістів. Linux має повну реалізацію мережного інтерфейсу TCP/IP, що забезпечує підключення до Internet та надання повного спектра послуг цієї всесвітньої мережі.

Linux не тільки багатозадачна операційна система, але це також і система для багатьох користувачів. Навіть на одному комп'ютері можна працювати одночасно на шести текстових консолях і одній графічній.

Варто відзначити дві характерні особливості Linux: *безкоштовність та відкритість програмного коду*. Linux поширюється за генеральною відкритою ліцензією GNU в рамках фонду вільного програмного забезпечення (Free Software Foundation), що робить цю ОС доступною всім бажаючим. Linux захищена авторським правом і не знаходиться в загальнодоступному користуванні, проте відкрита ліцензія GNU це майже те ж саме, що і передача в загальнодоступне користування. Вона складена так, що Linux залишається безкоштовною і в той же час стандартизованою системою. Існує лише один офіційний варіант ядра Linux.

Від Unix операційній системі Linux дісталися ще дві чудові особливості: вона є *розрахованою на багато користувачів і є багатозадачною системою*. Багатозадачність означає, що система може виконувати декілька завдань одночасно. Розрахований на багато користувачів режим означає, що в системі можуть одночасно працювати декілька користувачів, кожен з яких взаємодіє з нею через свій термінал. Ще одним з достоїнств цієї ОС є можливість її установки спільно з Windows на один комп'ютер.

Загальна характеристика Linux - це сучасна POSIX-Сумісна й Unix-подібна операційна система для персональних комп'ютерів і робочих станцій.

ОС Linux підтримує стандарти відкритих систем і протоколи мережі Internet і сумісна із системами Unix, DOS, MS Windows. Усі компоненти системи, включаючи вихідні тексти, поширюються з ліцензією на вільне копіювання й установку для необмеженого числа користувачів.

Характерні риси Linux:

• багатозадачність: багато програм виконуються одночасно;

• багатокористувальницький режим: багато користувачів одночасно працюють на одній і тій же машині;

• захищений режим процесора (386 protected mode);

• захист пам'яті процесу; збій програми не може викликати зависання системи;

• ощадливе завантаження: Linux зчитує з диска тільки ті частини програми, які дійсно використовуються для виконання;

• поділ сторінок по запису між екземплярами виконуваної програми. Це значить, що процеси-екземпляри програми можуть використовувати при

виконанні ту саму пам'ять. Коли такий процес намагається зробити запис в пам’ять, то 4-х кілобайтна сторінка, у яку йде запис, копіюється на вільне місце. Ця властивість збільшує швидкодію й заощаджує пам'ять;

• віртуальна пам'ять зі сторінковою організацією (тобто на диск із пам'яті витісняється не весь неактивний процес, а тільки необхідна сторінка); віртуальна пам'ять у самостійних розділах диска й/або файлах файлової системи; обсяг віртуальної пам'яті до 2 Гбайт; зміна розміру віртуальної пам'яті під час виконання програм;

• загальна пам'ять програм і дискового кешу: вся вільна пам'ять використовується для буферизації обміну з диском;

• динамічні поділювані бібліотеки, що завантажуються;

• сертифікація по стандарті POSIX.1, сумісність зі стандартами System V і BSD на рівні вихідних текстів;

• наявність вихідного тексту всіх програм, включаючи тексти ядра, драйверів, засобів розробки й додатків. Ці тексти вільно поширюються. У цей час деякими фірмами для Linux поставляється ряд комерційних програм без вихідних текстів, але все, що було вільним так і залишається вільним;

• керування завданнями в стандарті POSIX;

• емуляція співпроцесора в ядрі, тому додаток може не піклуватися про емуляцію співпроцесора. Звичайно, якщо співпроцесор у наявності, то він і використовується;

• підтримка національних алфавітів і угод, можливість додавати нові;

• множинні віртуальні консолі: на одному дисплеї кілька одночасних незалежних сеансів роботи, що перемикаються із клавіатури;

• підтримка ряду розповсюджених файлових систем (MINIX, Xenix, файлові системи System V); наявність власної передової файлової системи обсягом до 4 Терабайт і з іменами файлів до 255 знаків;

• прозорий доступ до розділів DOS (або OS/2 FAT): розділ DOS виглядає як частина файлової системи Linux; підтримка VFAT (WNT, Windows 95);

• спеціальна файлова система UMSDOS, що дозволяє встановлювати Linux у файлову систему DOS;

• доступ (тільки читання) до файлової системи HPFS-2 OS/2 2.1;

• підтримка всіх стандартних форматів CD ROM;

• підтримка мережі TCP/IP, включаючи ftp, telnet, NFS і т.д.

6. *Файлова система Linux.*

Файлова система Linux має багато каталогів, які утворюють ієрархічну деревоподібну систему (рис.9). На початку цього дерева розташований кореневий каталог, який позначається як символ / (похила лінія). Далі розташовуються інші каталоги та файли. На відміну від операційних систем MS-DOS і Windows, тут майже всі каталоги мають стандартні імена і їх не можна вилучати або перейменовувати.

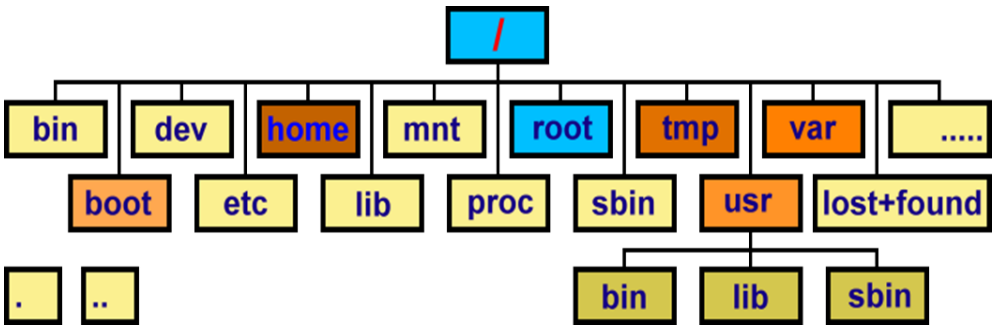

Рис. 9 Структура каталогів

Каталог X, що входить до складу каталога більш високого рівня Y, є по відношенню до нього підкаталогом, а сам каталог Y – батьківським каталогом для X, або надкаталогом.

Для найменування каталогів першого рівня, тобто підкаталогів кореневого каталога, необхідно спочатку вказати ім'я кореневого каталога (символ /), а потім – власне ім'я підкаталога, наприклад:

/bin

Для найменування каталогів наступних рівнів необхідно послідовно вказати імена всіх каталогів по дереву каталогів, починаючи із кореневого каталога. Ці імена розділяються між собою символом /. Наприклад:

/home/user/tom

Каталог, з яким в даний момент працює користувач, називається поточним. Він позначається символом *. * (точка). Для найменування в поточному каталозі безпосереднього батьківського каталога можна скористатись також символами *. . * (дві точки).

Файл з точки зору операційної системи є найбільшою сукупністю даних, з якою можна виконувати різні стандартні операції: копіювання, перейменування, вилучення і т.д. З позицій користувача файл – це поіменована область на диску або іншому машинному носії даних. В файлах можуть зберігатись тексти програм, документи тощо.

В Linux поняття файла є *універсальним*. Всі фізичні пристрої (дисплей, принтер, клавіатура та ін) представляються як спеціальні файли, які зберігаються в каталозі /dev. До речі, каталоги в Linux розглядаються як третя різновидність файлів, оскільки вони теж займають деяке місце на диску.

Ім'я файла може бути довільним, однак має задовольняти деякі обмеження. По-перше, імена файлів не можуть включати в себе пропуски. По-друге, не рекомендується включати до складу імені такі символи:

/ * \? " ' ^! @ $% & () { } []:; < >

По-третє, довжина імені файла не повинна перевищувати 256 символів. Не варто також забувати, що розрізняють великі та малі букви алфавіту.

Позначення кожного файла може складатись із двох частин, які розділяються точкою: основного імені файла і розширення імені файла. Розширення імені файла використовувати не обов'язково, але бажано, оскільки воно вказує на тип файла. Наприклад,

*.txt - текстовий файл;

*.bin - бінарний, тобто виконуваний, файл;

*.c - програма мовою Сі.

Для звертання до файлів поточного каталогу достатньо вказати лише ім’я файлу та розширення імені файлу (якщо воно існує). В загальному випадку для звертання до файлу із каталогу X, необхідно в імені файлу вказувати весь шлях по дереву каталогів, починаючи від кореневого каталога і до каталога X. В такому випадку матимемо абсолютне складове ім’я файла. Наприклад, абсолютне складове ім’я файла /Home/user/file.txt позначає файл file.txt, що знаходиться в каталозі user, який є підкаталогом каталога першого рівня /Home.

Якщо необхідний каталог X знаходиться по дереву каталогів нижче від поточного каталога, тоді при звертанні до такого файла із поточного каталога в імені файла допускається вказувати лише шлях від поточного каталогу і до каталогу У. В цьому випадку матимемо відносне складове ім’я файла.

При виконанні однакових операцій із групою файлів можна використовувати для них узагальнені імена. Символ * позначає будь-яку кількість будь-яких символів в основному імені файла або в розширенні імені файла. Ім’я файла із символами * по суті буде позначати не один файл, а групу файлів. Наприклад, запис *.txt буде позначати всі текстові файли поточного каталога, а запис M*.c позначатиме всі програми мовою С, імена яких розпочинаються із букви М.

Символ ? позначає лише один довільний символ або відсутність одного символу в основному імені файла або в розширення імені файла. Наприклад, запис file?.txt може бути узагальненою формою позначення всіх файлів, основне ім’я яких включає 4 або 5 символів і розпочинаються словом file.

7. *Структура каталогів Linux*.

Розглянемо призначення основних каталогів першого рівня.

Каталог */root*. Він є робочим каталогом суперкористувача. Після реєстрації суперкористувач попадає саме в цей каталог.

Каталог */home*. Цей каталог використовується для зберігання даних користувачів. В ньому створюються підкаталоги для кожного користувача під тим іменем, під яким реєструється користувач на початку сеанса роботи (login). Тільки в свому каталозі (а також в каталозі /tmp) рядовий користувач може створювати нові підкаталоги і файли.

Каталог */boot*. В ньому містяться файли, що використовуються при початковому завантаженні операційної системи (ОС).

Каталоги */bin і /sbin*. В цих каталогах містяться: системні утиліти і бінарні (тобто виконувальні) файли, оболонки, файли багатьох зовнішніх команд, редактори та т.п. Головною відмінністю між програмами, що зберігаються в згаданих каталогах є те, що програми з каталогу */sbin* можуть бути виконані лише суперкористувачем.

Каталог */lib*. В цьому каталозі знаходяться загальні системні бібліотеки. В одному з підкаталогів каталога /lib знаходиться ядро Linux.

Каталог */dev*. Тут знаходяться файли, які представляють системні пристрої (термінали, принтери, вінчестери і т.п.).

Каталог */usr*. Цей каталог призначено для зберігання файлів, які не змінюються та спільно використовуються різними користувачами. В підкаталогах */usr/bin і /usr/sbin* міститься велика кількість програм, які за своїми функціями подібні до каталогів */bin* та */sbin*. Підкаталог /usr/share містить дані, які переносяться між комп'ютерами з різними операційними системами.

Каталог */mnt*. В цьому каталозі знаходаться підкаталоги, що використовуються як точки монтування для інших файлових систем. До підкаталога /mnt/windows підключається файлова система ОС Windows, до підкаталогу /mnt/floppy підключаються дискети і т.д. Імена цих підкаталогів можуть бути змінені.

Каталог */etc*. Цей каталог використовується для зберігання конфігураційних файлів ОС. Серед множини підкаталогів дуже важливим є підкаталог /etc/X11, де зберігаються файли конфігурації системи X Window, а також підкаталог /etc/rc.d, де міститься сценарій початкового завантаження Linux.

Каталог */opt*. В цьому каталозі інсталюються додаткові пакети програм.

Каталог */var*. Тут зберігаються файли, вміст яких часто змінюється. Найважливіші такі підкаталоги:

/var/log – для зберігання системних журналів;

/var/mail – для організації поштових скриньок користувачів; /var/spool – для організації буферних черг для принтера, пошти і т.п.

Каталог */lost+found*. Цей каталог призначений для зберігання пошкоджених даних, які можуть з'явитися після перевірки файлової системи Linux.

Каталог */tmp*. Тут зберігаються тимчасові файли системи і користувачів.

Каталог */auto*. Цей каталог використовується для конфігурування пристроїв з метою автоматичного знаходження і монтування носіїв інформації в момент їх встановлення.

8. *Команди роботи з файлами та директоріями.*

Вичерпну інформацію про формат будь-якої команди Linux можна отримати за допомогою довідкової команди *man*:

man<імя команди>

pwd	Виводить поточний шлях
ls	Виводить список файлів і каталогів по порядку
ls-laX	Виводить форматований список всіх файлів і директорій, включаючи приховані
cd	Перехід в домашню теку
cd /home	Перехід в директорію /home
cd ..	Перехід в каталог рівнем вище того, в якому зараз перебуваєте

cd ../..	Перехід в каталог двома рівнями вище того, в якому зараз перебуваєте
cd -	Перехід в каталог у якому ви перебували до переходу в поточний каталог
touch /home/primer2	Створення порожнього файлу /home/primer2
cat /home/primer2	Показати вміст файлу /home/primer2
tail /var/log/messages	Виводить кінець файлу. Зручно при роботі з великими файлами
head /var/log/messages	Виводить перші рядки файлу
nano /home/primer2	Редагувати файл /home/primer2
sudo tee-a /home/primer2	Додавання до кінця файлу "OP" в файл /home/primer2
cp /home/Mut@NT/primer.txt /home/primer.txt	Копіює /home/Mut@NT/primer.tx у home/primer.txt
ln-s /home/Mut@NT/primer.txt /home/primer	Створює символічне посилання /home/primer до файлу /home/Mut@NT/primer.txt
mkdir /home/Mut@NT/OP	Створення директорії з ім'ям OP
rmdir /home/Mut@NT/OP	Видалення директорії з ім'ям OP
rm-rf /home/Mut@NT/OP	Видалення директорії з вкладеними файлами
cp-la /dir1 /dir2	Копіювання директорій
mv /dir1 /dir2	Перейменування директорії
du-sh /home/Mut@NT/	Виводить на екран розмір заданої директорії. Можна використовувати для визначення розміру файлів (кількість блоків диска, зайнятих кожним файлів у вашому поточному каталозі)
tree	Показує деревоподібний список файлів і тек у вашому поточному каталозі. Також підраховує їх кількість. залежно від кількості файлів підрахунок файлів може зайняти деякий час
dir	Показує вміст вашого поточного каталогу в алфавітному порядку і з урахуванням регістру назв
df	Виводить в консолі кількість зайнятого та вільного місця на жорсткому диску для кожного каталога системи
locate primer	Пошук всіх файлів з ім'ям primer

Перегляд каталогу

Перегляд вмісту каталогу здійснюється командою ls. Формат команди:

ls [–опції] [<Каталог>] [<Файл1> <...>]

При відсутності всіх опцій та параметрів можна отримати список всіх файлів і підкаталогів поточного каталогу у мінімальному форматі. Мінімальний формат передбачає видачу лише імен файлів та підкаталогів. Розширений формат дозволяє розрізнити між собою файли та підкаталоги (після імен останніх ставиться похила лінія “/”).

Повний формат передбачає видачу для кожного файла або підкаталога інформації в такій послідовності:

тип (“d” - підкаталог, “-“ - файл, “b” - блочний пристрій, “c” - символьний пристрій);

права доступу (для користувача-власника, групи користувачів, всіх інших);

користувач;

група користувачів;

обсяг пам’яті в байтах;

дата створення;

час створення;

ім’я.

Основні опції:

al – видача списку файлів та підкаталогів у повному форматі;

F – видача списку файлів та підкаталогів у розширеному форматі;

R – видача списку файлів, каталогів і всієї ієрархії підкаталогів у мінімальному форматі;

[<Каталог>] визначаєтой каталог, для якого визначається список файлів та підкаталогів. За замовчуванням видається інформація про поточний каталог.

[<Файл1> <…>] визначає файл або файли, про які видається інформація.

Приклад 1. Видача імен файлів та підкаталогів поточного каталогу у мінімальному форматі:

ls

Приклад 2. Видача інформації про каталог /home/user/mydir у розширеному форматі:

ls –F /home/user/mydir

Приклад 3. Видача інформації про файл file1.txt у повному форматі:

ls –al file1.txt

Створення каталогу

Для створення нових каталогів (підкаталогів) використовується команда *mkdir*. Формат команди:

mkdir [–опції] [<Каталог1> <…>]

Основні опції:

m <права> – задання прав доступу для створюваного каталога (в символьному вигляді або у вигляді числа у восьмеричній системі числення).

Приклад 1. Створити каталог *mydir1* із наданням прав доступу за замовчуванням:

mkdir mydir1

Приклад 2. Створити каталог *mydir2* із забороною запису та вилучення із нього файлів для всіх, за винятком користувача-власника:

mkdir –m 755 mydir2

Зміна поточного каталогу

Для переходу із поточного каталогу в інший каталог використовується команда cd. Формат команди:

cd [<Каталог>]

Приклад 1. Перейти в каталог */home/user/dir1/mydir*:

cd /home/user/dir1/mydir

Приклад 2. Перейти у батьківський каталог, тобто на один рівень вище по ієрархії каталогів:

cd..

Приклад 3. Перейти у домашній каталог користувача із будь-якого каталога:

cd

Знищення каталогу

Для знищення каталогів (підкаталогів) використовується команда rmdir. Формат команди:

rmdir [–опції] [<Каталог1> <…>]

За замовчуванням знищуються лише пусті каталоги та підкаталоги.

Основні опції:

p – знищення всіх підкаталогів того каталога, який ліквідовується.

Приклад 1. Знищити каталог mydir:

rmdir mydir

Приклад 2. Знищити каталог mydir та його пусті підкаталоги mydir1 та mydir2:

rmdir - p mydir/mydir1 mydir/mydir2

Створення файла

Створити файл можна за допомогою багатофункціональної команди cat. Формат команди для виконання цієї задачі:

cat > <file>

Приклад 1. Для створення нового текстового файла необхідно спочатку виконати команду

cat > file1.txt

Далі вводиться необхідний текст. Для повернення в командний режим необхідно натиснути клавіші <Ctrl><d>.

Для додання нових даних в кінець цього файла треба виконати аналогічні дії, але з іншою командою:

cat >> file1.txt

Копіювання файлів

Для копіювання файлів використовується команда ср. Формат команди:

ср [–опції] <Файл1> … < ФайлN> <Файл>|<Каталог>

В результаті виконання команди відбувається копіювання одного файла <Файл1> у новий файл <Файл> або кількох файлів <Файл1> … < ФайлN> в каталог <Каталог>.

Основні опції:

i – видача запиту на підтвердження заміни існуючого файла.

Приклад 1. Створити у поточному каталозі копію файла file1.txt:

cp file1.txt file2.txt

Приклад 2. Скопіювати із поточного каталогу файли file1.txt і file2.txt у каталог /home/user/work:

cp file1.txt file2.txt /home/user/work

Переміщення (перейменування) файлів

Для переміщення файлів між різними каталогами використовується команда mv. Формат команди:

mv [–опції] <Файл1> ... < ФайлN> <Каталог>

В результаті виконання команди відбувається переміщення одного або кількох файлів <Файл1> … < ФайлN> в каталог <Каталог>. Початкові файли Файл1> … < ФайлN> при цьому знищуються.

Основні опції:

i – видача запиту на підтвердження заміни існуючого файла.

f – знищення файлів призначення, якщо вони існують, без запиту.

Приклад 1.

Перемістити із поточного каталогу /home/user файли file.txt і tom.txt у каталог /home/user/work:

mv file.txt tom.txt /home/user/work

Для перейменування файлів в межах одного каталогу використовується команда mv у форматі:

mv [–опції] <Файл1> < Файл2>

де <Файл1> - старе імя файла;

< Файл2> - нове імя файла.

Звичайно, можна переміщати файли в інші каталоги із одночасним їх перейменуванням.

Знищення файлів

Для переміщення файлів між різними каталогами використовується команда rm. Формат команди:

rm [–опції] <Файл1> ... < ФайлN>

За замовчуванням знищуються лише файли і без попереднього запиту на знищення.

Основні опції:

i – видача запиту на підтвердження знищення файла.

f – знищення каталогу і всіх його підкаталогів, в тому числі і непустих.

Приклад 1. Знищити всі текстові файли із запитом на підтвердження для кожного файла:

*rm –i *.txt*

Об’єднання файлів

Для об’єднання файлів використовується багатофункціональна команда cat. Формат команди для виконання цієї задачі:

cat [–опції] <Файл1> ... < ФайлN>

За замовчуванням файли <Файл1> … < ФайлN> об’єднуються один за другим в порядку їх запису в команді і результат видається на стандартний пристрій виведення, тобто на екран дисплея.

Основні опції:

n – здійснити нумерацію рядків об’єднаного файла.

e(E) – показати кінець кожного рядка за допомогою символу $.

Приклад 1. До файла file1.txt дописати файл file2.txt:

cat file1.txt file2.txt

Приклад 2. Об’єднати файли file1.txt і file2.txt у файл common.txt:

cat file1.txt file2.txt > common.txt

Пошук відмінностей між файлами

Для знаходження відмінностей між файлами використовується команда diff. Формат команди:

diff [–опції] <Файл1> < Файл2>

Порівнюються між собою по рядках <Файл1> і < Файл2> і видаються ті рядки обох файлів, які не однакові.

Основні опції:

a – вважати всі файли текстовими і порівнювати їх по рядках.

b – ігнорувати відмінності в кількості пропусків, табуляції і т.п.

i – ігнорувати відмінності в регістрах.

q – повідомляти лише про сам факт відмінностей без подробиць.

Приклад 1. Визначити відмінності між файлами file1.txt і file2.txt:

diff file1.txt file2.txt

Пошук у файлі за зразком

Пошук у файлі заданих ключових слів чи текстових фраз здійснюється командою grep. Формат команди:

grep [–опції] <зразок> [<файл>]

Результатом виконання команди є всі рядки із <файл>, які містять заданий <зразок>.

Основні опції:

f FILE – зразок для пошуку береться у першому рядку із файла FILE;

a – розглядати бінарні файли як текстові;

r – ігнорувати відмінності в регістрах.

Приклад 1. Знайти всі рядки у file1.txt, які містять слово “display” або “Display”:

grep –r “display” file1.txt

Перегляд файлів

Існує декілька команд для перегляду вмісту файла на екрані дисплея. Для малих за розміром файлів можна скористатись багатофункціональною командою cat. Вміст файла file1.txt на екрані дисплея можна побачити після виконання команди

cat file1.txt

Якщо вміст файла не поміщається повністю на екрані, тоді знадобиться команда more. За командою

more file1.txt

на екран дисплея буде виведено першу сторінку цього файла. Натискуючи клавішу <Enter>, можна переглянути посторінково весь вміст файла.

Переглянути посторінково текст файла можна також і за командою

less file1.txt

Важливою перевагою цієї команди є те, що можна рухатись не тільки вниз по тексту, але і повертатись назад.

За допомогою команди

head [–опції] <файл>

можна переглянути лише початок файла, а за допомогою команди

tail [–опції] <файл>

можна переглянути лише кінець цього файла.

Основні опції команд head і tail:

n c – видати на екран n символів;

n l – видати на екран n рядків;

n d – видати на екран n блоків.

Приклад 1. Видати на екран перші 5 рядків файла file1.txt:

head –5l file1.txt

Приклад 2. Видати на екран останні 40 символів файла file1. txt:

tail –40c file1.txt

9. *Доступ до файлів і каталогів*

Оскільки Linux – система для багатьох користувачів, тому для захисту файлів кожного користувача від неправильної дії інших користувачів, підтримується механізм прав доступу до файлів і каталогів. Цей механізм дозволяє кожному файлу або каталогу призначити конкретного власника.

Linux дозволяє також спільно використовувати файли кількома користувачами, які можуть об’єднатись в окрему групу користувачів. Кожен користувач є членом як мінімум однієї групи користувачів.

Права доступу до каталогів і файлів розподіляються на три типи:

- читання (read),
- запис (write),
- виконання (execute).

Ці типи прав доступу можуть бути надані трьом категоріям користувачів: власникові файла, групі користувачів або всім іншим користувачам.

Дозвіл на читання дає можливість читати вміст файлів, а у випадку каталогів – переглядати перелік імен файлів в каталозі (використовуючи, наприклад, команду 1s). Дозвіл на запис дає можливість записувати в файл і змінювати його. Для каталогів це дає право створювати в каталозі нові файли і каталоги або вилучати файли в цьому каталозі. А дозвіл на виконання надає користувачеві можливість виконати файл (як бінарні програми, так і командні файли). Дозвіл на виконання стосовно каталогів визначає можливість виконувати команди для роботи із каталогами.

Для того, щоб отримати повну інформацію про файл <file.txt>, потрібно виконати команду

ls –al file1.txt

На екрані дисплея отримаємо такий результат

- rw- r- - r- - 1 user1 dev Mar 15 10:11 file1.txt

Перший символ дозволяє розрізнити між собою файли і каталоги:

для файлів записується “-“ , а для каталогів “d”. Далі йде рядок прав доступу до цього файла з боку різноманітних категорій користувачів.

Потім виводиться кількість (1) синонімів, під якими даний файл відомий системі. Третє поле – ім’я власника файла (user1) і четверте - група (dev), до якої належить власник файла. Очевидно, що останнє поле містить ім’я файла (file1.txt), а інші поля означають дату та час створення файла.

Розглянемо детальніше права доступу до файла.

В рядку *rw- r-- r--*

перші три символи показують права доступу для власника файла, наступні три символи – права групи користувачів, і останні три символи – права всіх інших користувачів. В даному випадку власник user1 файла file1.txt може читати (r - read) свій файл і записувати (w - write) в нього нові дані, а всі інші категорії користувачів можуть лише читати цей файл. Жодний із користувачів не має прав на виконання (x - executive) файла file1.txt, оскільки в третій, шостій та девятій позиціях є знак заборони “-“.

За замовчуванням файлам надається захист – *rw- r-- r--*, який дозволяє іншим користувачам читати файли, але ніяким чином їх не змінювати. Каталогам за замовчуванням дається право доступу *d rwx r-x r-x*, що дозволяє іншим користувачам заходити з правами екскурсантів у каталог власника. Проте багато користувачів хоче тримати інших користувачів подалі від своїх файлів. Встановивши права доступу файла, *d rw- --- ---* ви нікому не покажете цей файл, і не дозволите записати в нього. Також добре захищає файли від всіх захист відповідного каталогу *d rwx --- ---*.

Для зміни прав доступу до файлів і каталогів використовується команда

chmod {a,u,g,o} {+, - ,} {r, w, x} <filename>

Спочатку необхідно вказати категорію користувача (a – всі користувачі; u - користувач-власник; g – група користувачів; o – всі інші). Далі потрібно вказати, чи додається відповідне право доступу (+), чи забирається право (-). І на завершення вказується конкретне право доступу: r - read, w - write, x - execute.

Приклад 1. Надати власнику файла file5.txt право на виконання свого файла:

chmod u+x file5.txt