МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

Б. І. Погребняк, М. В. Булаєнко

ОПЕРАЦІЙНІ СИСТЕМИ

НАВЧАЛЬНИЙ ПОСІБНИК

Харків
ХНУМГ ім. О. М. Бекетова
2018

УДК 004.451(075.8)
П43

Автори:
***Погребняк Борис Іванович*,** кандидат технічних наук, доцент;
***Булаєнко Марина Володимирівна*,** кандидат технічних наук, доцент

Рецензенти:
О. В. Грицунов, доктор фізико-математичних наук, професор, професор кафедри МЕЕПП Харківського національного університету радіоелектроніки;
Н. Д. Сізова, доктор фізико-математичних наук, професор, в. о. завідувача кафедри економічної кібернетики та інформаційних технологій Харківського національного університету будівництва та архітектури

Рекомендовано Вченою Радою ХНУМГ ім. О. М. Бекетова, протокол № 2 від 27 жовтня 2017 р.

Погребняк Б. І.
П43 Операційні системи : навч. посібник / Б. І. Погребняк, М. В. Булаєнко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 104 с.

У навчальному посібнику наведено матеріал щодо засад адміністрування сучасних операційних систем, моніторингу та налаштування їхньої продуктивності. Представлено довідкові матеріали стосовно питань використання сучасних операційних систем на підприємствах різних галузей.

Навчальний посібник призначається для студентів і магістрів ІТ-спеціальностей, слухачів навчальних закладів із наданням другої вищої технічної освіти в галузі інформаційних технологій, аспірантів, працівників підприємств, а також спеціалістів, які бажають поглиблювати свої знання в сфері операційних систем.

УДК 004.451(075.8)

ЗМІСТ

ВСТУП

Навчальний посібник «Операційні системи» присвячено питанням засад адміністрування сучасних операційних систем, моніторингу та налаштуванню їхньої продуктивності.

До складу сучасного комп'ютера входять один або декілька процесорів, оперативна пам'ять, диски, принтер, клавіатура, миша, дисплей, мережні інтерфейси та інші пристрої вводу-виводу. У підсумку формується досить складна система. Якщо кожен програміст, що створює прикладну програму, буде вникати у всі тонкощі роботи цих пристроїв, то він не встигне написати жодного рядка корисного коду. Більше того, керувати всіма цими складниками та їхнім оптимальним використанням дуже непросто. Із огляду на це комп'ютери обладнані спеціальним рівнем програмного забезпечення, що називається операційною системою (далі – ОС). Саме такі програмні системи і є предметом розгляду цього навчального посібника.

Операційні системі є сполучною ланкою між користувачем, прикладними програмами та апаратними засобами комп'ютера. Від їхнього правильного вибору, налаштування та адміністрування в решті-решт залежить ефективність функціонування всього обчислювального комплексу. Особливо гостро ця проблема постає сьогодні, коли на ринку комп'ютер з'являється все більше мобільних операційних систем та вільне програмне забезпечення.

У навчальному посібнику розглянуто питання ефективного використання сучасних операційних систем за різних умов експлуатації. Робота структурована за сьома розділами: основні концепції, еволюція та різновиди операційних систем; архітектура й ресурси операційних систем; логічна та фізична організація файлових систем; реалізація файлових систем; взаємодія з користувачем в операційних системах; засади віртуалізації та установлення операційних систем; засади адміністрування, користувачів та групи. Кожен розділ містить ґрунтовні теоретичні розробки, розгорнуті й докладні приклади розв'язання практичних задач і питання для самостійної перевірки знань.

Посібник має прикладне спрямування й стане в пригоді студентам і аспірантам вищих технічних навчальних закладів. Крім того, посібник можна використати для перепідготовки фахівців у системі післядипломної освіти та під час дистанційної форми навчання. Його можуть також використовувати розробники програмного забезпечення та системні адміністратори для підвищення кваліфікації в галузі операційних систем.

У створенні навчального посібника брали участь багато осіб. Автори висловлюють щиру вдячність завідувачу кафедри прикладної математики та інформаційних технологій Харківського національного університету міського господарства імені О. М. Бекетова Марині Володимирівні Новожиловій, та усім співробітникам кафедри за цінні поради під час роботи над книгою.

1 ГОЛОВНІ КОНЦЕПЦІЇ, ЕВОЛЮЦІЯ І РІЗНОВИДИ ОПЕРАЦІЙНИХ СИСТЕМ

1 Предмет і завдання курсу

Предметом вивчення курсу є сучасні *операційні системи* (далі – ***ОС***).

У першому наближенні ОС можна визначити як комплекс програм, що забезпечують інтерфейс між апаратурою комп'ютера, прикладними програмами й користувачем. Відповідно до цього визначення, усі функції, виконувані ОС, націлені на вирішення двох основних завдань:

- організація ефективної роботи апаратури комп'ютера;
- забезпечення зручного використання ресурсів комп'ютера як прикладними програмами, так і користувачем.

Основною ***метою*** курсу є *вивчення понять, тверджень і особливостей теорії операційних систем, формування сталих поглядів щодо методів розроблення елементів системного програмного забезпечення й адміністрування операційних систем у міському господарстві*. Буде розглянуто два різновиди питань:

- головні принципи побудови ОС, найпоширеніші алгоритми виконання різних функцій ОС, типові структури даних, використовувані для забезпечення роботи ОС;
- практичне втілення цих принципів, алгоритмів, структур у найпоширеніших сучасних ОС.

До завдань курсу не належить вивчення практичних прийомів роботи з ОС. Це краще робити самостійно. Не ставиться й завдання навчити слухачів розробляти нові ОС. Операційні системи не є предметами масового виробництва, і їх розробляє лише невелика частина програмістів. Рівень знань, який передбачено досягти під час вивчення цього предмета, можна порівняти з рівнем знань про склад автомобіля, що властивий водієві-фахівцеві. Він не обов'язково повинен бути автомеханіком, однак в загальних рисах має розуміти принципи роботи автомобіля.

2 Короткий нарис історії ОС

Історії розвитку ОС свідчить про те, що всі істотні просування в сфері архітектури ОС обумовлені впливом двох головних факторів:

- прогрес технології, що призводить до швидкого збільшення характеристик апаратури ЕОМ і до появи принципово нових типів апаратури;
- принципово нові ідеї, що виникають у проектувальників.

Не беручи до уваги давню суперечку матеріалістів з ідеалістами, необхідно визнати, що перший, матеріальний фактор визначав розвиток ОС на 80–90 %. Такі технологічні прориви, як винахід магнітних дисків, мікропроцесорів, створення високоякісних відеомоніторів, потребували радикальних змін у технології роботи з комп'ютером, і внаслідок цього спричинялися створення принципово нових типів ОС або їхніх окремих підсистем. З іншого боку, деякі ідеї в сфері організації обчислювального процесу та інтерфейсу призвели до удосконалювання архітектури комп'ютерів.

Без знання про головні етапи розвитку апаратного та програмного забезпечення важко зрозуміти багато особливостей сучасних ОС.

Додатковий аргумент на користь знання історії полягає в тому, що багато технічних рішень, які, здавалося, втратили свою актуальність разом з конкретними системами, зненацька знову виявляються актуальними на новому витку розвитку. Деякі такі приклади будуть розглянуті в курсі.

Передісторія ОС

Незабаром після того, як наприкінці 40-х років XX століття були створені перші електронні комп'ютери, стала актуальною проблема підвищення ефективності використання їхніх потужностей і, насамперед, центрального процесора.

Типовий комп'ютер першого-другого поколінь являв собою велику кімнату, заставлену шафами й оповиту кабелями. Кожен із головних пристроїв – центральний процесор, оперативна пам'ять, накопичувачі на магнітних стрічках, пристрої введення з перфокарт, принтер – займали «шаф» або «тумбу», які наповнені радіолампами й механічними частинами.

Усе дорого коштувало, споживало достатню кількість електроенергії та регулярно ламалося.

У таких умовах машинний час коштував дуже дорого. Проте, звичайна практика використання ЕОМ не сприймала економіку. Зазвичай, програміст, що розробляє програму, замовляв щодня декілька годин машинного часу та протягом цього часу монопольно використовував машину. Виконавши кожен новий запуск налагоджуваної програми (яку необхідно було щораз уводити або із клавіатури, або, у найкращому разі, з перфокарт), користувач одержував роздруківку (зазвичай у вигляді масиву цифр), аналізував результати, вносив зміни в програму й знову запускав її. Отже, у процесі сеансу налагодження дороге обладнання простоювало 99 % часу, поки програміст осмислював результати та працював із пристроями введення/виведення. Крім того, збій під час введення однієї перфокарти міг призвести до запуску з самого спочатку роботу всієї програми.

Далі сформувалася важлива ідея – використати сам комп’ютер для підвищення ефективності роботи з ним.

Одне з відгалужень цієї ідеї – створення мов і систем програмування – розглядається в окремих курсах. Іншим важливим кроком стало покладання на спеціальну комп’ютерну програму частини тих функцій, які до цього виконував оператор або сам програміст.

Такі програми називалися зазвичай ***моніторами*** (не плутати з монітором як пристроєм виводу, який тоді був дуже рідкісним). Монітор приймав команди, що складаються зазвичай з 1–2 букв назви та 1–3 аргументів, заданих 8-ковими або 16-ковими числами. Типовими командами були, наприклад:

- завантаження даних із перфокарт за зазначеною адресою пам’яті;
- перегляд і коректування (із друкарської машинки) значень у визначеному діапазоні адрес;
- покрокове виконання програми з виведенням результатів кожної команди на друкарську машинку;
- запуск програми із зазначеної адреси із заданням адрес контрольних крапок зупинки.

Незважаючи на вбогість, за сучасними стандартами, подібних засобів, вони водночас значно підвищили продуктивність роботи програмістів. Однак кардинального підвищення завантаження процесора не відбулося.

Часом широкого поширення моніторів у світі були 50-і роки минулого століття (у СРСР – 60-і роки). Нині щось подібне можна зустріти на найбільш примітивних мікропроцесорних контролерах.

Пакетні ОС

Історія ОС починається з появи наприкінці 50-х років XX століття перших систем, що організують роботу за пакетним принципом.

Найважливішою зміною, яка сталася на цьому етапі розвитку, було масове вигнання програмістів із машинних залів, як фактора, що тільки вносить сум'яття в роботу.

Тепер від програміста було потрібно зібрати пакет перфокарт, що містить його програму, дані до неї, а також керувальні перфокарти. Ці карти на спеціально розробленій ***мові керування завданнями*** (JCL, Job Control Language) пояснювали операційній системі, чиє це завдання, що потрібно зробити із програмою (наприклад передати її транслятору з Фортрану), що виконати у разі успішної трансляції (імовірно, запустити на виконання), що – за наявності помилок (наприклад, перейти до іншої програми), звідки взяти вхідні дані (наприклад із такого-то циліндру магнітного диска). Крім того, там могли бути навіть вказівки на те, скільки метрів паперу можна виділити на роздруківку і який максимальний час може зайняти робота програми.

Обійтися без настільки докладних інструкцій було не можна, програміст не був присутній під час запуску завдання та не міг впливати особисто.

Підготовлений пакет передавався разом з іншими подібними пакетами операторові ЕОМ, перед яким стояли два головні завдання: щоб у пристрої введення не переводилися пакети завдань і щоб у принтері не скінчився папір. Коли процесор закінчував обробку завдання і друкування його результатів, він зчитував наступний пакет і приступав до його обробки. Так досягалася головна мета пакетного режиму – позбавлення простоїв процесора через нерозторопність людей.

Незабаром розробники ОС усвідомили, що вичерпано не всі резерви підвищення завантаження процесора. Операції введення і друкування потребували тільки дуже невеликої частки від повної продуктивності процесора. Крім того, у процесі роботи програми траплялися звертання до периферійних пристроїв (наприклад, до магнітних стрічок а, пізніше, – дисків), під час виконання яких процесор знову простоював. Доцільно було знайти спосіб, щоб у ці періоди очікування завантажити процесор іншою роботою. Але для цього необхідно, щоб у пам'яті комп'ютера перебували відразу декілька програм, тоді ОС змогла б перемикати процесор на виконання тієї програми, що у цей момент може виконуватись.

Така організація роботи, коли в пам'яті перебувають декілька програм і система в певні моменти перемикає виконання з однієї програми на іншу, була названа ***мультипрограмуванням***. Ця важлива ідея в різних втіленнях пережила ті пакетні системи, у яких вона вперше була реалізована, і є основою для функціонування практично всіх сучасних ОС.

Однією із найбільш поширених пакетних ОС із мультипрограмуванням е OS/360 – головна операційна систем знаменитого в 60÷70 р. XX століття сімейства ЕОМ IBM 360/370.

ОС із розподілом часу

На рубежі 60÷70 р. XX століття досить поширеним і не занадто дорогим периферійним пристроєм стають монітори (спочатку монохромні, які працюють тільки в текстовому режимі). Процесор і ОЗП залишаються найдорожчими та громіздкими пристроями обчислювальної системи. У цих умовах виникає і швидко поширюється новий тип ОС – ***системи з розподілом часу***.

До однієї ЕОМ підключається декілька десятків робочих місць, обладнаних дисплеєм (монітор та клавіатура) та обчислювальні ресурси, що спільно використовуються. Процесорний час ділиться на кванти тривалістю в декілька десятків мілісекунд і після закінчення кожного кванта процесор можна переключати на обслуговування іншого процесу, іншого дисплея. Оскільки тепер підготовку текстів програм виконують самі програмісти за дисплеями, а робота з редагування тексту потребує дуже малих витрат процесорного часу, процесор встигає обслужити всі робочі місця практично без відчутної затримки. Більша частина часу

процесора виділяється невеликій кількості робочих місць, де в цей момент запущені на виконання програми. Зрозуміло, середня швидкість роботи кожної програми зменшується, принаймні в стільки разів, скільки програм виконується одночасно.

Режим розподілу часу став величезним полегшенням для програмістів, які знову змогли до деякої міри відчути себе «хазяїнами» ЕОМ і одержали можливість запускати програми на трансляцію та налаштування хоч кожні п’ять хвилин. Це дало змогу скоротити строки розробки та налагодження програм.

Для трудомістких обчислювальних завдань, що передбачають розрахунок за раніше налагодженими програмами, режим розподілу часу менш ефективний, ніж пакетний, оскільки часте перемикання процесора між одночасно виконуваними програмами потребує додаткових витрат.

Системи розподілу часу використовуються в режимі діалогу з користувачем, тому замість громіздких, деталізованих операторів JCL у них застосовуються простіші команди, що виконують елементарні дії – запуск програми, видавання на екран файлу або каталогу, копіювання або видалення файлу тощо. Користувачеві не потрібно передбачати заздалегідь всі можливі варіанти виконання команди, набагато простіше побачити результат виконання на екрані та після цього прийняти рішення, яку команду виконувати наступною. Водночас, деякі часто повторювані послідовності команд зручно описати один раз у вигляді «пакетного завдання» і потім використати за необхідності. У цьому плані системи розподілу часу зберігають ті зручні можливості, які надавали пакетні системи.

Спочатку як апаратна основа систем розподілу часу використовувалися «великі» ЕОМ, які пізніше стали називатися «мейнфреймами» (mainframes). Пізніше, із прогресом обчислювальної техніки, це стало можливо навіть міні‑ЕОМ (так називався в ті роки клас комп’ютерів, що займали тільки один-два невеликих шафки). Варто згадати серію міні‑ЕОМ PDP-11, що була най поширенішою в усьому світі протягом півтора десятків років.

Цей період (70-і роки XX соліття у світі, 80-і в СРСР) обумовлений глибоким розвитком теорії та практики створення потужних ОС, що містять розвинені засоби керування процесами та пам’яттю, які реалізують багатокористувацький режим роботи. Серед подібних систем необхідно

відокремити UNIX – єдину систему, що благополучно дожила до нашого часу.

Однозадачні ОС для ПЕОМ

У середині 70-х XX століття років був винайдений мікропроцесор, а до початку 80-х мікропроцесори стали упереджувати за функціональними характеристиках «великі» процесори. Ця ситуація зробила майже марним режим розподілу часу: навіщо ділити один процесор між багатьма завданнями та багатьма користувачами, якщо простіше й дешевше дати окремий мікропроцесор кожному користувачеві? Розподіл часу залишилося доцільним хіба що стосовно суперкомп'ютерів.

Поява і бурхливе поширення персональних комп'ютерів (далі – ПК) призвело до виникнення нового покоління ОС, які виявилися в багато разів простіші своїх попередниць. Непотрібним виявився багатокористувацький захист. Спочатку здалася непотрібною і багатозадачність. Усе це можна було розцінити як очевидний регрес у розвитку ОС.

Най популярнішою ОС для ранніх восьмиразрядних ПК була система CP/M відомої тоді фірми Digital Research, однак із появою на початку 80-х XX століття знаменитої машини IBM PC лідерство була міцно перехоплено системою MS-DOS фірми Microsoft.

Багатозадачні ОС для ПК із графічним інтерфейсом

Швидкий розвиток технології призвів до того, що наприкінці 80-х років XX століття ПК змогли вираховувати значно складніші й більш трудомісткі завдання, ніж раніше. Багато що з досягнень колишніх етапів розвитку ОС виявилося знову затребуваними, але в нових умовах, серед яких потрібно зазначити різке підвищення потужності процесорів і об'єму пам'яті, появу високоякісних графічних моніторів і розвиток мережних технологій.

Стала реальною така річ, як багатозадачна ОС для ПК. Варто зазначити, що спочатку ідея системи, у якій один користувач запускає одночасно декілька додатків, більшості фахівців здавалася безглуздою та викликала глузування: «Чому б не виконати декілька програм по черзі?». Зараз із таким поглядом не можна навіть сперечатися.

На зміну ОС, які виконували текстові команди, що вводять користувачем із клавіатури, прийшли системи, у яких взаємодія з користувачем базується на використанні GUI (Graphical User Interface, графічний інтерфейс користувача).

Значна частина ПК працює в складних локальних обчислювальних мережах. Це призвело до того, що питання захисту даних користувача знову придбали першорядне значення.

3 Класифікація ОС

Існують різні види класифікації ОС за тими або іншими ознаками, що відображають різні істотні характеристики цих систем.

За призначенням.

Системи загального призначення. Ця досить розпливчаста назва має на увазі ОС, призначених для рішення широкого кола завдань, зокрема запуск різних додатків, розроблення та налагодження програм, робота з мережею та мультимедіа.

Системи реального часу. Цей важливий клас систем призначений для роботи в контурі керування об'єктами (такими як літальні апарати, технологічні установки, автомобілі, складна побутова техніка тощо). З подібного визначення випливають жорсткі вимоги до надійності й ефективності таких системи. Необхідно забезпечити точне планування дій системи в часі (керувальні сигнали необхідно видавати в задані моменти часу, а не просто «якомога швидко»). Особливий підклас становлять системи, ***убудовані*** в обладнання. Такі системи роками можуть виконувати фіксований набір програм, не потребуючи втручання людини-оператора на глибшому рівні, ніж натискання кнопки «Вмк».

Іноді відокремлюють також такий клас ОС, як системи з «нежорстким» реальним часом. Це такі системи, які не можуть гарантувати точне дотримання часових співвідношень, але «дуже намагаються», тобто містять засоби для пріоритетного виконання завдань, критичних за часом. Такій системі не можна довірити керування ракетою, але вона цілком упорається з демонстрацією відеофільму. Відокремлення подібних систем в окремий клас має, скоріше, рекламне значення, даючи змогу таким системам, як Windows NT і деяким версіям UNIX, теж називати себе «системами реального часу».

Інші спеціалізовані системи. Це різні ОС, орієнтовані насамперед на ефективне рішення завдань певного класу, з більшим або меншим збитком для інших завдань. Можна відокремити, наприклад, мережні системи (такі як Novell Netware), що забезпечують надійне та високоефективне функціонування локальних мереж.

За особливостями взаємодії з користувачем.

Пакетні ОС, що обробляють заздалегідь підготовлені завдання.

Діалогові ОС, що виконують команди користувача в інтерактивному режимі. «Інтерактивність» полягає у постійній взаємодії системи з користувачем.

ОС із графічним інтерфейсом. Загалом, їх також можна віднести до діалогових систем, однак використання мишою та усього, що з нею пов'язане (меню, кнопки тощо) вносить свою специфіку.

Вбудовані ОС не взаємодіють з користувачем.

За кількістю одночасно виконуваних завдань.

Однозадачні ОС. У таких системах у кожен момент часу може існувати не більше ніж один активний користувальницький процес. Варто помітити, що одночасно з ним можуть працювати системні процеси (наприклад, що виконують запити на введення/виведення).

Багатозадачні ОС. Вони забезпечують паралельне виконання декількох користувальницьких процесів. Реалізація багатозадачності потребує значного ускладнення алгоритмів і структур даних, використовуваних у системі.

За кількістю користувачів.

Однокористувацькі ОС. Їм властивий повний доступ користувача до усіх ресурсів системи. Подібні системи прийнятні переважно для ізольованих комп'ютерів, що не допускають доступу до ресурсів певного комп'ютера по мережі або з терміналів.

Багатокористувацькі ОС. Їхнім важливим компонентом є засоби захисту даних і процесів кожного користувача, що базується на понятті власника ресурсу і на точній вказівці прав доступу, наданих кожному користувачеві системи.

За апаратурною основою.

Однопроцесорні ОС. Вони керують роботою комп'ютера тільки з одним центральним процесором.

Багатопроцесорні ОС. У завдання такої системи входить, крім іншого, ефективний розподіл виконуваних завдань за процесорами та організування погодженої роботи всіх процесорів.

Мережні ОС. Вони включають можливість доступу до інших комп'ютерів локальної мережі, роботи з файловими та іншими серверами.

Розподілені ОС. Їхня відмінність від мережних полягає в тому, що розподілена система, використовуючи ресурси локальної мережі, подає їх користувачеві як єдину систему, не розділену на окремі машини.

4 Критерії оцінки ОС

Під час порівняльного розгляду різних ОС загалом або їхніх окремих підсистемах виникає низка питань – яка з них краще й чому, яка архітектура системи переважніше, який з алгоритмів ефективніший, яка структура даних зручніша тощо.

Дуже рідко можна дати однозначну відповідь на подібні питання, якщо мова йде про практично використовувані системи. Система або її частина, що гірше інших систем за всіма параметрами, просто не мала б права на існування. Насправді можна говорити про типову багатокритеріальну задачу: є декілька важливих критеріїв якості, і система, що випереджає інші за одним критерієм, звичайно уступає за іншим. Порівняльна важливість критеріїв залежить від призначення системи та умов її роботи.

Надійність

Цей критерій взагалі прийнято вважати найважливішим під час оцінки програмного забезпечення, і стосовно ОС він має найбільше значення.

Що розуміється під надійністю ОС?

Насамперед, її живучість, тобто здатність зберігати хоча б мінімальну працездатність в умовах апаратних збоїв і програмних помилок. Висока живучість особливо важлива для ОС комп'ютерів, вбудованих в апаратуру, коли втручання людини ускладнене, а відмова комп'ютерної системи може мати серйозні наслідки.

По-друге, здатність, як мінімум, діагностувати, а як максимум, компенсувати хоча б деякі типи апаратних збоїв. Для цього зазвичай вводиться надмірність зберігання най важливіші дані системи.

По-третє, ОС не повинна містити власних (внутрішніх) помилок. Це вимога рідко буває здійсненою в повному обсязі (програмісти давно зуміли довести своїм замовникам, що в будь-якій великій програмі завжди є помилки, і це звичайна річ), однак потрібно хоча б домогтися, щоб головні, часто використовувані або най відповідальніші частини ОС не мали помилок.

Нарешті, серед надійності системи варто викреслити її здатність протидіяти нерозумним діям користувача. Звичайний користувач повинен мати доступ тільки до тих можливостям системи, які необхідні для його роботи. Якщо користувач, навіть діючи в межах своїх повноважень, намагається зробити щось дуже нетипове (наприклад відформатувати системний диск), то найменше, що повинна зробити ОС, це перепитати користувача, чи впевнений він у правильності своїх дій.

Ефективність

Як відомо, ефективність будь-якої програми визначається двома групами показників, які можна узагальнено назвати «час» та «пам’ять». Під час розроблення системи доводиться приймати багато непростих рішень, обумовлених оптимальним балансом цих показників.

Найважливішим показником часової ефективності є ***продуктивність*** системи, тобто усереднена кількість корисної обчислювальної роботи, виконуваної в одиницю часу. З іншого боку, для діалогових ОС не менш важливо ***час реакції*** системи на дії користувача. Ці показники можуть до деякої міри суперечити один одному. Наприклад, у системах розподілу часу збільшення кванта часу збільшує продуктивність (унаслідок скорочення числа перемикань процесів), але погіршує час реакції.

У програмуванні відома аксіома: виграш у часі досягається за допомогою програшу в пам’яті, і навпаки. Це повною мірою стосується і ОС, розроблювачам яких постійно доводиться шукати баланс між витратами часу та пам’яті.

Турбота за ефективність довгий час мала найбільше значення у процесі розроблення програмного забезпечення, і особливо ОС. На жаль, зворотнім боком стрімкого збільшення потужності комп’ютерів стало ослаблення інтересу до ефективності програм. На сьогодні ефективність є найважливішою вимогою хіба що стосовно систем реального часу.

Зручність

Цей критерій найсуб'єктивніший. Можна запропонувати, наприклад, такий підхід: система або її частина зручна, якщо вона дає змогу легко й просто вирішувати ті завдання, які зустрічаються найчастіше, водночас містить засоби для рішення широкого кола менш стандартних завдань (навіть якщо ці засоби не настільки прості). Наприклад: таку часту дію, як копіювання файлу, необхідно виконуватися за допомогою однієї простої команди або легкого руху миші; водночас для зміни розділів диска рекомендується почитати керівництво, оскільки це може знадобитися навіть не щороку.

Розроблювачі кожної ОС мають власні уподобання про зручність, і кожна ОС має своїх прихильників, що вважають саме її ідеалом зручності.

Масштабованість

Термін «масштабованість» (scalability) означає можливість настроювання системи для використання в різних варіантах, залежно від потужності обчислювальної системи, від набору конкретних периферійних пристроїв, від функції, що виконує комп'ютер (сервер, робоча станція або ізольований комп'ютер) від призначення комп'ютера (домашній, офісний, дослідницький тощо).

Гарантією масштабованості є продумана модульна структура системи, що дає змогу в процесі установлення системи збирати та набудовувати потрібну конфігурацію. Можливий і інший підхід, коли під загальною назвою поєднуються різні системи, що забезпечують у раціональних межах програмну сумісність. Прикладом можуть слугувати версії Windows, UNIX, Linux.

У деяких випадках фірми, що виготовляють програмне забезпечення, штучно відключають у дешевших версіях системи ті можливості, які насправді реалізовані, але стають доступні, тільки якщо користувач купує ліцензію на дорожчу версію. Але це вже питання, обумовлено не технічною стороною справи, а маркетинговою політикою.

Здатність до розвитку

Щоб складна програма довго існувала, у неї необхідно закласти можливості для майбутнього розвитку.

Однією з головних умов здатності системи до розвитку є добре продумана модульна структура, у якій чітко визначені функції кожного модуля і його взаємозв'язку з іншими модулями. Створюється можливість удосконалювання окремих модулів з мінімальним ризиком призвести до небажаних наслідків для інших частин системи.

Важливою вимогою до розвитку ОС є ***сумісність версій знизу вгору***, що означає можливість безболісного переходу від старої версії до нової, без втрати раніше напрацьованих прикладних програм і без необхідності різкої зміни всіх навичок користувача. Зворотна сумісність – зверху вниз – зазвичай, не гарантується, оскільки в процесі розвитку система здобуває нові можливості, не реалізовані в старих версіях. Наприклад, програма з Windows 3.1 буде нормально працювати і в Windows 10; а от, навпаки – навряд чи.

Фірми-виробники ОС додають максимум зусиль для забезпечення сумісності знизу вгору, щоб не відлякувати користувачів. Однак фірми намагаються в кожну нову версію закласти яку-небудь нову «цукерку», щоб спонукати користувачів якомога швидше купити її.

Сумісність версій – благо для користувача, однак на практиці вона часто призводить до консервації давно віджилих свій вік особливостей або ж просто невдалих рішень, прийнятих у ранній версії системи. У документації подібні архаїзми позначаються як «застарілі» (obsolete), але повної відмови від них, зазвичай, не відбувається (з огляду на те, що може десь ще працювати прикладна програма, написана двадцять років тому з використанням саме цих можливостей).

Зазвичай, найбільше консервативною стороною будь-якої ОС є не алгоритми, а структури системних даних, тому розробники заздалегідь будують структури «на виріст»: закладають у них резервні поля, використають змінні замість деяких констант, установлюють кількісні обмеження з більшим запасом тощо.

Мобільність

Під ***мобільністю*** (portability) розуміється можливість перенесення програми (у цьому разі ОС) на іншу апаратну платформу, тобто на інший тип процесора й іншу архітектуру комп'ютера. Йдеться про перенесення із помірними трудовитратами, що не потребує повного перероблення системи.

Властивість мобільності не настільки однозначно позитивна, як може здатися. Щоб програма була мобільна, під час її розроблення варто відмовитися від глибокого використання особливостей конкретної архітектури (таких як кількість і функціональні можливості регістрів процесора, нестандартні команди тощо). Мобільну програму необхідно написати мовою досить високого рівня (зазвичай використовується мова С), якому можна реалізувати на комп'ютерах будь-якої архітектури. Платою за мобільність завжди є деяка втрата ефективності, тому немобільні системи розповсюджені досить широко.

З іншого боку існувало безліч ефективних і зручних, але немобільних ОС, які зникли разом із процесорами, для яких вони призначалися. Водночас мобільна система UNIX продовжує процвітати четвертий десяток років, набагато опередивши ті комп'ютери, для яких вона спочатку створювалася. Приблизно 5–10 % початкових текстів UNIX написані мовою асемблера, їх необхідно переписувати заново у разі перенесення на нову архітектуру. Інша частина системи написана на С і практично не потребує змін при переносі.

Деяким компромісом є ***багатоплатформені*** ОС (наприклад Windows NT), спроектовані для використання на декількох апаратних платформах, але вони не гарантували можливості перенесення на нові, не передбачені заздалегідь архітектури.

5 Головні функції та структура ОС

Відповідно до багаторічної традиції, при розгляді основ функціонування ОС прийнято викреслювати чотири головних групи функцій, виконуваних системою.

Керування пристроями. Маються на увазі всі периферійні пристрої, що підключають до комп'ютера, – клавіатура, монітор, принтери, диски тощо.

Керування даними. Під цим давнім терміном зараз розуміється робота з файлами, хоча раніше, коли звертання до даних на магнітних носіях виконувалося шляхом вказування адреси розміщення даних на пристрої, а поняття файлу не існувало.

Керування процесами. Ця частина роботи ОС полягає у запису і завершенні роботи програм, обробці помилок, забезпеченні паралельної роботи декількох програм на одному комп'ютері.

Керування пам'яттю. Оперативна пам'ять комп'ютера – це такий ресурс, якого завжди не вистачає. У цих умовах раціональне планування використання пам'яті є найважливішим чинником ефективної роботи.

Є ще декілька важливих функцій, що виконує ОС, і які важко втиснути в межі традиційної класифікації. До них, насамперед, належать такі.

Організація інтерфейсу з користувачем. Форми інтерфейсу можуть бути різноманітними, залежно від типу та призначення ОС: мова керування пакетами завдань, набір діалогових команд, засоби графічного інтерфейсу.

Захист даних. Як тільки система перестає бути надбанням одного ізольованого від зовнішнього світу користувача, питання захисту даних від несанкціонованого доступу набувають істотну важливість. ОС, що забезпечує роботу в мережі або в системі розподілу часу, повинна відповідати наявним стандартам безпеки.

Ведення статистики. У процесі роботи ОС повинна збиратися, зберігатися та аналізуватися різноманітна інформація: про кількість часу, витраченого різними програмами та користувачами, про інтенсивність використання ресурсів, про спроби некоректних дій користувачів, про збої обладнання тощо. Зібрана інформація зберігається в системних журналах і в облікових записах користувачів.

Для розуміння роботи ОС необхідно вміти викреслити головні частини системи та їх зв'язок, тобто описувати структуру системи. Для різних ОС їхній структурний розподіл може бути досить різним. Найзагальнішими різновидами структуризації можна вважати два. З одного боку, можна вважати, що ОС розподілено на підсистеми, що відповідають переліченим вище групам функцій. Такий розподіл досить обґрунтований, програмні модулі ОС дійсно можна віднести до однієї із цих підсистем. Інший важливий структурний розподіл обумовлюється поняттям ***ядра*** системи.

Ядро, це основна, «найбільше системна» частина операційної системи. Відомо різні визначення ядра. Відповідно до одного з них, ядро –

це ***резидентна*** частина системи, тобто до ядра ставиться той програмний код, що постійно перебуває в пам'яті протягом всієї роботи системи. Інші модулі ОС є ***транзитними***, тобто довантажуються з диска залежно від необхідності на час своєї роботи. До транзитних частин системи відносяться:

- ***утілити*** (utilities) – окремі системні програми, що виконують приватні завдання, такі як форматування та перевірка диска, пошук даних у файлах, моніторинг (відстеження) роботи системи та багато чого іншого;
- ***системні бібліотеки підпрограм***, що дають змогу прикладним програмам використовувати різні спеціальні можливості, підтримувані системою (наприклад, бібліотеки для графічного виведення, для роботи з мультимедіа тощо);
- ***інтерпретатор команд*** – програма, що виконує введення команд користувача, їхній аналіз і виклик інших модулів для виконання команд;
- ***системний загружчик*** – програма, що під час запуску ОС (наприклад під час вмикання живлення) забезпечує завантаження системи з диска, її ініціалізацію і старт;
- ***інші різновиди програм***, залежно від конкретної системи.

Не менш важливим є визначення ядра, полягає у розрізненні режимів роботи комп'ютера. Усі сучасні процесори підтримують, принаймні, два режими: ***привілейований*** режим (він же режим ядра, kernel mode) і ***непривілейований*** (режим завдання, режим користувача, user mode). Програми, що працюють у режимі ядра, мають повний, необмежений доступ до всіх ресурсів комп'ютера: його командам, адресам, портам введення/виводу тощо. У режимі завдання можливості програми обмежені, вона, зокрема, не може виконати деякі спеціальні команди. Апаратне розмежування можливостей є абсолютно необхідною умовою реалізації надійного захисту даних у багатокористувацькій системі. Звідси випливає і визначення ядра як частини ОС, що працює в режимі ядра. Усі інші програми, як системні утиліти, так і програми користувачів, працюють у режимі користувача та повинні звертатися до ядра для виконання багатьох системних дій.

Варто зауважити, що переходи з режиму користувача в режим ядра і назад – це дії, що потребують певного часу, і занадто часте їх виконання може призвести до помітного зниження швидкості роботи програм.

У зв’язку із цим визначення того, які функції повинні підтримуватися ядром, а які краще виконувати в режимі користувача – це непросте і важливе завдання, що повинні вирішити розроблювачі ОС.

Особливу роль у структурі системи грають ***драйвери пристроїв.*** Ці програми, призначені для обслуговування певних периферійних пристроїв, безсумнівно, можна віднести до ядра системи: вони майже завжди є резидентними та працюють у режимі ядра. Але на відміну від самого ядра, що змінюється тільки з появою нової версії ОС, набір використовуваних драйверів досить мобільний і залежить від набору пристроїв, підключених до певного комп’ютера. У деяких системах (наприклад у ранніх версіях UNIX) для підключення нового драйвера було потрібно перекомпілювати все ядро. У більшості сучасних ОС драйвери підключаються до ядра в процесі завантаження системи, а іноді дозволяється навіть завантаження та вивантаження драйверів у процесі роботи системи.

Як програмний інтерфейс системи, тобто набір засобів для звернення прикладних програм до послуг ОС, використовується документований набір ***системних викликів*** або ***функцій API*** (Applied Programming Interface). Між цими двома термінами є деяка різниця. Під системними викликами розуміються функції, реалізовані безпосередньо програмами ядра системи. У процесі їхнього виконання відбувається перехід із режиму користувача в режим ядра, а потім назад. На відміну від цього, API-функції визначаються як функції, описані в документації ОС, незалежно від того, чи виконуються вони ядром або ж системними бібліотеками, що працюють у режимі користувача. У Windows часто декілька різних API-функцій звертаються до одного того самого не документованого системного виклику, але мають різні частини, що обрамляють, які працюють у режимі користувача.

Контрольні питання

1. Що є предметом вивчення в курсі Операційні системи?
2. Що є головною метою курсу Операційні системи?
3. Що входить в завдання курсу Операційні системи?
4. Назвіть фактори, які істотно вплинули на архітектуру сучасних ОС.
5. Як називались програми, які частково замінили функції оператора?
6. Назвіть типові команді програми-монітора.

7. Назвіть час найбільшого поширення програм-моніторів.
8. У чому полягає принцип пакетної обробки даних?
9. Як називалась мова керування завданнями?
10. Що таке мультипрограмний режим роботи ОС?
11. Назвіть найпоширенішу мультипрограмну пакетну ОС, для яких комп'ютерів вона була призначена?
12. Що таке ОС з розподілом часу?
13. Назвіть переваги та недоліки ОС з розподілом часу.
14. Назвіть найпоширеніші ЕОМ, на яких могла виконуватися ОС розподілу часу.
15. Назвіть найпоширеніші ОС для ПЕОМ, на яких комп'ютерах вони працювали?
16. Що таке багатозадачна ОС для ПК із графічним інтерфейсом?
17. Назвіть головні класифікації ОС.
18. Які можуть бути ОС за призначенням?
19. Що таке ОС загального призначення?
20. Що таке ОС реального часу?
21. Які можуть бути ОС за особливостями взаємодії з користувачем?
22. Що таке пакетні ОС?
23. Що таке діалогові ОС?
24. Що таке вбудовані ОС?
25. Які можуть бути ОС по числу одночасно виконуваних завдань?
26. Що таке однозадачна ОС?
27. Що таке багатозадачна ОС?
28. Які можуть бути ОС за кількістю користувачів?
29. Які можуть бути ОС залежно від апаратної основи?
30. Назвіть головні критерії оцінки ОС.

2 АРХІТЕКТУРА І РЕСУРСИ ОПЕРАЦІЙНИХ СИСТЕМ

1 Склад та призначення програмного забезпечення комп'ютера

Широке використання комп'ютерів у різних сферах людської діяльності обумовлюється програмним забезпеченням, що дає змогу використати ту саму машину, побудовану із кремнію та сталі, для рішення величезної кількості найрізноманітніших завдань. На відміну від перших програм, що вводилися в машину за допомогою стомлювального перемикання безлічі тумблерів і цифро-складальних пристроїв, програмне забезпечення, що є істотно важливим для сучасного комп'ютера, записується, зазвичай, на магнітних дисках і починає діяти відразу після ввімкнення машини. Однак команди комп'ютера, хоча вони і задаються тепер набагато зручнішим способом, помітних змін не зазнали. Кожен комп'ютер повинен розкласти завдання на послідовність машинних команд, а потім виконувати їх одну за іншою.

В англійській мові для програмного забезпечення (далі – ПО) обране (а точніше, створене) досить влучне слово *SoftWare* (буквально – «м'який виріб»), що підкреслює рівнозначність програмного й апаратного забезпечення («заліза» – *HardWare* – «твердий виріб»), і водночас свідчить про його гнучкість, здатність модифікуватися, пристосовуватися, розвиватися. Уведення цих термінів було обумовлюється необхідністю провести чітку межу між командами-інструкціями, що керують комп'ютером, і його фізичними компонентами або апаратним забезпеченням, що, власно, і становить комп'ютер. Тому перемикання комп'ютера з побудови малюнка на розроблення стандартного контракту або з розроблення архітектурного проекту на створення карти погоди земної кулі здійснюється шляхом зміни послідовності команд, що керують його роботою, тобто програм.

Перш, ніж обговорювати склад та призначення програмного забезпечення, необхідно ввести ще два терміни: *програміст* і *користувач*. Якщо програма складна, до її створення долучається велика група людей, що виконують різні функції, необхідно вживати загальний термін «програміст» для позначення людини, що проектує, записує та вдосконалює нову програму, або модифікує стару. Людина, для якої пишеться програма та яка, імовірно, буде постачати її вхідними даними та використовувати отримані результати, називається користувачем. І хоча та

сама людина може бути і користувачем, і програмістом, важливо розрізняти ці дві функцій.

Загальний термін програмне забезпечення (далі – ПО, *SoftWare*) застосовується для позначення програм, використовуваних у комп'ютерних системах. Усе ПО можна класифікувати за безліччю найрізноманітними ознаками. Найістотнішими з них є класифікації за такими ознаками:

- *призначенням*, або місцем використання в технологічному процесі обробки інформації;
- *різновидом* (*типом*) *ліцензії*.

Залежно від призначення програми підрозділяються на два великих класи:

1) прикладне програмне забезпечення (далі – ППО);
2) базове програмне забезпечення (далі – БПО).

Прикладне програмне забезпечення складають програми для користувачів і виконуючі інформаційні процеси, які потрібні користувачам. Воно призначено для виконання цільових завдань користувача. Наприклад, у бізнесі найпоширенішою прикладною програмою є 1С:Підприємство, що призначена для виконання різноманітних економічних завдань.

Базове програмне забезпечення – це «накладні витрати» комп'ютерної обробки інформації. Тобто воно безпосередньо не бере участь у виконанні цільових завдань користувача. Але й без нього ці проблеми вирішити не можливо.

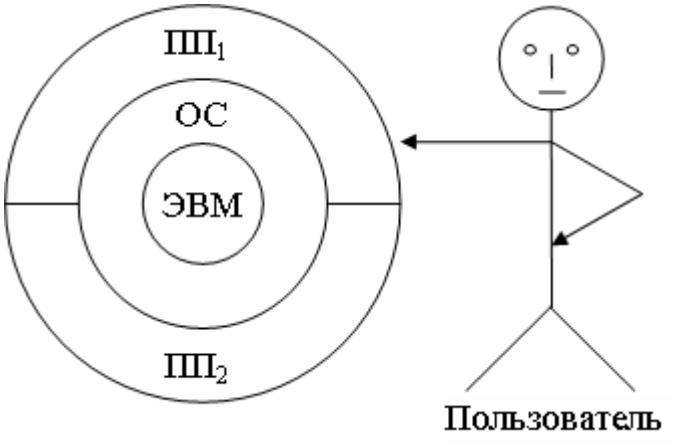
Центральне місце в базовому програмному забезпеченні займає спеціальна сукупність програм, яка називається *операційною системою*, і яка забезпечує взаємодію:

- людини;
- комп'ютера;
- програм.

Така взаємодія виконується відповідно до рисунка 2.1.

Операційна система вказує комп'ютеру, як інтерпретувати команди і дані, як розподіляти апаратні ресурси для виконання завдань і як управляти периферійними пристроями (наприклад, принтером або дисплеєм).

Вона також забезпечує можливість безпосередньої взаємодії людини та комп'ютера, виконуючи такі дії, як зберігання програм і даних на зовнішніх запам'ятовувальних пристроях.



ЕВМ – електронна обчислювальна машина (комп'ютер)
ОС – операційна система
$ПП_1$, **$ПП_2$** – прикладна програма1, прикладна програма2

Рисунок 2.1 – Взаємодія користувача, комп'ютера і програм

Якщо розглядати операційну систему як «режисера» комп'ютерного дійства, то прикладні програми відіграють роль «артистів». Саме за допомогою таких програмам, як текстові процесори, ігри та електронні таблиці, комп'ютер є багатофункціональним.

Поняття «базове програмне забезпечення» включає також такі програми, як транслятори та утиліти. Транслятор – це програма, що приймає як вхідні дані іншу програму, написану на так називаній мові програмування високого рівня, що віддалено нагадує мову людини, і відмінному від машинного коду, з яким має справа процесор, і переводить цю програму в машинну мову, що становить комбінацію нулів та одиниць, які комп'ютер обробляє як послідовності електричних імпульсів. Утиліти виконують рутинні, але часто вкрай необхідні, функції, наприклад, видалення непотрібної інформації з магнітних дисків. Ці скромні «робочі конячки» допомагають вирішувати типові завдання обробки інформації.

Отже, повна *обчислювальна система*, що включає апаратуру (*HardWare*) та програмне забезпечення (S*oftWare*), створює середовище, у якій можуть розроблятися, зберігатися та виконуватися програми. Границі між окремими шарами в цій обчислювальній системі (рис. 2.1) називаються *інтерфейсами*.

Тобто *інтерфейс – це набір правил взаємодії між окремими компонентами системи*. Серед цих правил можна викреслити такі:

- границя між користувачем і прикладною програмою називається *Графічним Інтерфейсом Користувача* (англ. *Graphical User Interface – GUI*);
- границя між прикладною програмою та операційною системою – *Інтерфейсом Прикладного Програмування* (англ. *Application Programs Interface – API*);
- границю між операційною системою і комп'ютером утворює так звана *система (набір, архітектура) команд центрального процесора*.

Одні операційні системи можуть працювати тільки з однією, цілком певною, системою команд, а інші – підтримувати різні системи команд. Наприклад, операційні системи класу Microsoft Windows «розуміють» тільки так звану систему команд *x86* (процесори корпорацій Intel та AMD, на основі яких створюються персональні комп'ютери IBM PC), тоді як операційні системи сімейства Linux можуть працювати з різними системами команд.

У ситуації, коли програмне забезпечення є об'єктом продажу нарівні із предметами побуту, на нього автоматично поширюються вже не тільки закони наукового розроблення, але й властивості матеріальних предметів, якими можна торгувати, обмінюватися, право володіння та користування якими охороняється законом. Так програмне забезпечення потрапило в розряд *інтелектуальної власності*, тобто програми почали розглядатися як об'єкти застосування *авторського права*.

Щоб захистити свої інтереси, виробники програмного забезпечення використають ліцензії – особливий різновид договору між власником авторських прав і користувачем (покупцем) програмного забезпечення. Ліцензія на програмне забезпечення – це правовий інструмент, що визначає порядок поширення й використання програмного забезпечення. Загалом, ліцензія гарантує, що видавець ПО, якому належать виняткові права на програму, не подасть до суду на того, хто нею користується.

Ліцензії на програмне забезпечення, загалом, розділяються на дві великі категорії:

1) невільні (вони ж власницькі, пропрієтарні, напіввільні);
2) ліцензії вільного та відкритого ПО.

Їхні відмінності істотно впливають на права кінцевого користувача стосовно використання ПО.

Головною рисою пропрієтарних ліцензій є те, що видавець ПО в ліцензії дає дозвіл її одержувачеві використати одну або декілька копій програми, але при цьому сам залишається правовласником всіх цих копій. Один із наслідків такого підходу полягає в тому, що практично усі права на ПО залишаються за видавцем, а користувачеві передається тільки дуже обмежений набір строго обкреслених прав. Типовою пропрієтарною ліцензією може слугувати ліцензія на операційну систему Microsoft Windows, що включає великий список заборонених варіантів використання, таких, наприклад, як установлення однієї її копії на декілька комп'ютерів.

На відміну від пропрієтарних, вільні й відкриті ліцензії не залишають права на певну копію програми її видавцеві, а передають найважливіші з них кінцевому користувачеві, який і стає її власником. Прикладом найпоширенішої вільної ліцензії є GPL (General Public License), що дає користувачеві право самому поширювати ПО під цією ліцензією, брати участь у його розробленні або змінювати іншими способами. Проте перелічені права зобов'язують користувача ПО під GPL підкорятися певним правилам. Наприклад, будь-які зміни програми, зроблені користувачем і розповсюджені далі, повинні супроводжуватися вхідним кодом цих змін.

2 Структура комп'ютерної програми

Будь-яка комп'ютерна програма складається з трьох частин (рис. 2.2):

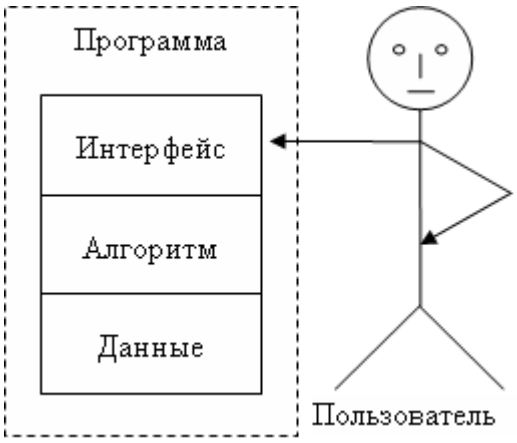


Рисунок 2.2 – Структура комп'ютерної програми

1. Інтерфейс – набір правил взаємодії між користувачем і програмою.

2. Алгоритм, або *бізнес-логіка* – набір правил перетворення інформації із вхідної у вихідну.

3. *Дані* – те, що обробляє програма.

Усі три частини обов'язково присутні в будь-якій комп'ютерній програмі. Однак, «питома вага» кожної з них залежить від призначення програми. Наприклад, у науково-технічних (інженерних) програмах переважають алгоритми, а інтерфейс і дані в них, зазвичай, нескладні. Економічні задачі, навпаки, обробляють великі обсяги інформації (тобто переважають дані), а алгоритми й інтерфейс у таких програмах порівняно прості. Ігрові програми мають добре пророблені інтерфейси, цікаві алгоритми та нескладні дані.

3 Операційна система Linux

Linux належить до серії операційних систем UNIX. Свій початок серія веде від 1969 року, коли співробітники Bell Labs (одного з дослідницьких підрозділів американської компанії AT&T) Кен Томпсон і Денніс Рітчі створили найперший варіант операційної системи, що надалі одержала назву UNIX.

4 Коротка історія

До початку 80-х років XX-го століття операційна система UNIX стала найпоширенішою ОС у світі, особливо в академічному середовищі та Інтернеті. Для цієї операційної системи існувала безліч програм, що вільно поширювалися в науковому співтоваристві. Але особливість цієї ситуації полягала в тому, що ОС UNIX тоді була невільним програмним продуктом.

Початок рішення цієї проблеми було покладено Річардом Столменом, що 27 вересня 1983 року в групах новин net.unix-wizards та net.usoft опублікував оголошення про початок розроблення операційної системи *GNU*. Метою проекту GNU було створення «цілісної UNIX-сумісної програмної системи», що повністю складається з вільного програмного забезпечення. Як писав пізніше сам Річард Столмен: «Абревіатура GNU розшифровується як «GNU – це не UNIX» (GNU's – Not UNIX). Головна ідея, закладена в процес розроблення системи GNU – це її повне відокремлення від UNIX. UNIX завжди була й залишається

невільним ПО…»... Тобто те, що належить проекту GNU, не є частиною UNIX (оскільки на той час навіть саме слово UNIX уже було зареєстрованою товарною маркою, тобто перестало бути вільним).

У межах проекту GNU була створена більшість компонентів, необхідних для функціонування вільної операційної системи, і вільно розповсюджуваних в Інтернеті. Крім текстового редактора *Emacs*, Р. Столмен створив компілятор *gcc* (GNU C Compiler) і відладчик *gdb*. Будучи видатним програмістом, Річард Столмен поодинці зумів створити ефективний і надійний компілятор, що перевершує за своїми якостями продукти комерційних постачальників, створювані цілими колективами програмістів. Як зазначає Р. Столмен: «ДО 1990 року система GNU була практично закінчена; не вистачало тільки одного з базових компонентів – ядра». Тобто для перетворення GNU у повноцінну операційну систему не вистачало тільки одного ядра.

Розроблення ядра велося (воно називалося Hurd), але з якихось причин затримувалася. Тому поява ядра, розробленого фінським студентом Лінусом Торвальдсом (Linus Torvalds) було дуже своєчасним. Воно «упало на підготовлений ґрунт» і призвело до появи нової операційної системи з повністю відкритим вхідним кодом, що у згодом одержала назву Linux. Линус Торвальдс виявився зі своєю розробкою в потрібнім місці в потрібний час.

Безпосереднім прообразом (прототипом) для Linux була операційна система MINIX – один із варіантів UNIX-подібної операційної системи. MINIX була розроблена в 1987 році Ендрю Таненбаумом (Andrew S. Tanenbaum), професором Університету Врie (Vrije University), Амстердам, Нідерланди. Свою операційну систему Таненбаумом розробив як навчальний посібник, на прикладі якого він показував студентам внутрішній устрій реальної операційної системи.

Зрозуміло що операційна система MINIX не відрізнялася великою досконалістю, але вона різнилася однією дуже важливою ознакою – її вхідні коди були відкриті. MINIX можна було придбати разом із книгами Таненбаума або окремо, і її можна було встановити на персональний комп’ютер. Таненбаум зумів залучити найвідоміших фахівців комп’ютерної техніки до обговорення напрямів створення операційних систем. Студенти комп’ютерних факультетів світу вивчали твори Таненбаума, вчитуючись у коди, намагаючись зрозуміти, як працює

система, яка курує їхнім комп'ютером. І одним із таких студентів був Лінус Торвальдс.

Офіційним днем народження операційної системи Linux вважається 25 серпня 1991 року, коли Лінус Торвальдс надіслав перше повідомлення про своє розроблення в групу новин, присвячену ОС MINIX comp.os.minix. А 17 вересня 1991 року він виклав в Інтернеті вхідний код програми (версії 0.01) для загальнодоступного завантаження. Каталог на FTP-сервері, де була викладена система, називався pub/OS/Linux. Згодом ця назва поступово закріпилася за новою операційною системою.

Якщо бути зовсім точним, то слово «Linux» означає тільки ядро. Тому, коли мова йде про операційну систему, правильніше було б говорити про «операційну систему, що базується на ядрі Linux». Річард Столлмен, засновник руху вільного ПЗ, наполягає на тому, що операційну систему варто називати не Linux, а GNU/Linux. Але назва Linux історично вже закріпилося за цією ОС, тому далі теж будемо називати її просто Linux (не забуваючи про заслуги Р. Столмена та інших дослідників).

Сучасний Linux

Сучасний Linux – це багатокористувацька, багатозадачна операційна система. Це означає, що одночасно на одному комп'ютері може працювати багато користувачів, кожний з яких одночасно може запускати на виконання багато різних програм. На сучасному персональному комп'ютері одночасно може працювати тільки одна людина. Однак багатокористувацька модель має багато переваг. Наприклад, один користувач може запустити тривалий процес (скачування фільму з Інтернету) і віддати комп'ютер іншому, або той самий користувач може зареєструватися в системі під різними іменами, і з різними повноваженнями, для виконання різних завдань тощо.

У будь-якій операційній системі загального призначення, і Linux зокрема, можна викреслити чотири головні складники:

1) ядро;
2) файлову систему;
3) інтерфейс;
4) утиліти.

Ядро

Ядро (англ. kernel) – це головна, визначальна, частина ОС, що керує роботою апаратної частини комп'ютера й виконанням програм. Вона розподіляє ресурси комп'ютера між програмами, що виконуються одночасно, а також забезпечує їхнє безконфліктне виконання. Звичайний користувач із ядром безпосередньо не працює. Він взаємодіє з такими компонентами ОС як інтерфейс, файлова система й утиліти. З ядром працюють переважно програмісти й системні адміністратори. Ядро ОС Linux, на відміну від інших складових системи, протягом усього часу розробляється й удосконалюється централізовано під загальним керівництвом Лінуса Торвальдса. На сьогодні тільки близько двох відсотків ядра написано самим Торвальдсом, але за ним як і раніше залишається рішення про внесення змін в офіційну гілку.

Файлова система

Файлова система являє собою спосіб (метод) зберігання й організації доступу до даних на зовнішніх запам'ятовувальних пристроях (далі – ЗЗП) комп'ютера (таких, наприклад, як магнітні й оптичні диски, флеш-накопичувачі тощо). Будь-яка файлова система складається з такого:

1) набору правил зберігання інформації на ЗЗП;
2) набору програм, які ці правила реалізують.

Кожен такий набір правил має своє унікальне ім'я – *назва файлової системи*. Докладніше файлові системи розглядаються у відповідних розділах.

Інтерфейс

Інтерфейс слугує для організації взаємодії користувача з комп'ютером. Керування роботою ОС Linux здійснюється за допомогою так званого *термінала*. Під терміналом (консоллю) розуміється пристрій, призначений для обміну інформацією між користувачем і комп'ютером. У мінімальній конфігурації термінал складається з монітора й клавіатури, миша – не обов'язкова. В ОС Linux існує два типи терміналів (інтерфейсів):

1) текстовий (інтерфейс командного рядка – англ. Command Line Interface – *CLI*);

2) *графічний* (англ. Graphical User Interface – *GUI*).

У більшості сучасних дистрибутивів Linux параметри настроєні так, що за замовчуванням першим завантажується графічний інтерфейс (графічний термінал).

Докладніше питання взаємодії користувача з операційною системою розглянуті у відповідному розділі.

Утиліти

Утиліти – це просто окремі програми, які виконують деякі службові функції, наприклад – копіювання або видалення файлів. Більшість утиліт в ОС Linux розташовані в каталогах `/bin` та `/usr/bin`. Повний список каталогів, де командна оболонка шукає утиліти, називається «*шлях пошуку*». Він зберігається в змінній оточення `PATH`. Подивитися її поточне значення можна по команді «`echo $PATH`». Результатом роботи команди буде послідовність шляхів, розділених символом двокрапки («`:`»). А команда «`set`» без параметрів видає поточні значення всіх змінних оточення.

Дистрибутиви

Загальна назва Linux не потребує яку-небудь єдину «офіційну» комплектацію. Вона поширюється зазвичай безкоштовно у вигляді різних готових «дистрибутивів», що мають свій набір прикладних програм і вже настроєних під конкретні потреби користувача. І водночас, як один із наслідків вільного поширення ПО для Linux є те, що велика кількість різноманітних фірм і компаній, а також просто незалежних груп розроблювачів стали випускати так звані «дистрибутиви Linux». Дистрибутив – це набір програмного забезпечення, що включає чотири головні складники ОС, тобто ядро, файлову систему, інтерфейс та утиліти, а також деяку сукупність прикладних програм. Зазвичай всі програми, що включають у дистрибутив Linux, поширюються на умовах GPL. Однак розроблювач дистрибутива повинен принаймні укомплектувати його програмою інсталяції, що буде встановлювати Linux на комп'ютер, на якому ніякої ОС ще немає. Крім того, необхідно забезпечити врегулювання взаємозалежностей та протиріч між різними *пакетами програм*, а також їхніми версіями. На сьогодні у світі існує вже більше сотні різних дистрибутивів Linux, і увесь час з'являються нові.

Наприклад, такі: Ubuntu Linux, Linux Mint, Debian Linux, Red Hat Linux, Fedora Linux, ASPLinux, Gentoo Linux, Arch Linux, ALT Linux, openSUSE Linux, PCLinuxOS тощо.

5 Поліморфізм

Поліморфізм (*polymorphism*) – це властивість комп'ютерних програм, що дає змогу використати ті самі прийоми й методи для рішення декількох схожих, але технічно різних завдань. У загальному вигляді ідея поліморфізму може бути коротко сформульована в такий спосіб – «один інтерфейс – та безліч реалізацій».

Принцип поліморфізму не є чим-небудь зовсім новим, або «революційним винаходом» програмістів, – він давно й широко використовується в різних галузях науки і техніки. Наприклад, послідовність розташування педалей зчеплення, гальма й газу однаковий у всіх автомобілях, у не залежно від типу їхнього двигуна (карбюраторний, або дизельний), сфері застосування (легкового або вантажний), класу, марки, місця виробництва тощо. Це дає змогу водіям зовсім легко й просто «пересідати» з одного автомобіля на інший. З іншого боку, не важко уявити собі «комерційний успіх» автомобіля, у якого конструктори «оптимізували» і, відповідно, змінили (без вагомих на те підстав) порядок розташування цих педалей. Єдине, що зробили програмісти – дали влучну назву одному із найістотніших ознак свого товару.

У програмуванні поліморфізм досить широко використовується під час створення безлічі програм найрізноманітнішого призначення. Після освоєння деяких прийомів роботи з одним додатком більшість із них можна використати й у роботі з іншими додатками. Зрозуміло, що в кожному додатку і його стані можуть бути задіяні зовсім різні механізми. Наприклад, переміщення або копіювання виділеного фрагмента в межах одного документа реалізуються за допомогою елементарних функцій переміщення та копіювання певного додатка. Ті самі дії, але між різними документами одного додатка, можуть задіяти зовсім інші механізми (наприклад, технологію DDE – Dynamic Data Exchange в Microsoft Windows), а між різними додатками – треті (наприклад, технологію OLE – Object Linking and Embedding в Microsoft Windows). Однак із погляду користувача, усі ці «тонкосщі реалізації» ніякого значення не мають – для

нього істотним є тільки те, що щоб однакові дії в різних додатках призводили б до однакових результатів.

Дотримування принципів поліморфізму, як і будь-яких стандартів, має як свої сильні, так і слабкі сторони. У науці будь-які стандарти є своєрідним «гальмом» для досягнення нових результатів. У промисловості у процесі виробництва товарів, наприклад, походження стандартам є необхідністю, оскільки надає багато додаткових переваг. Зокрема, у масовому виробництві використання уніфікованих технологій завжди є дешевшим. Користувачі також завжди віддають перевагу тим товарам, які їм давно й добре відомі, і зовсім неохоче «освоюють» нові вироби.

А оскільки, програми також є товаром, то, зрозуміло, що більшість розробників послідовно й цілеспрямовано дотримуються ідеї поліморфізму. Для користувачів ця особливість означає тільки одне – що після освоєння інтерфейсу та прийомів роботи з однією із програм, їх можна використати й під час роботи з іншими додатками. Тому, цю методику варто освоїти якомога раніше та якнайкраще, оскільки нею прийде користуватися постійно й повсюдно.

Контрольні питання

1. Поясніть значення слова «software».
2. Поясніть значення слова «hardware».
3. Поясніть значення слова «програміст».
4. Поясніть значення слова «користувач».
5. Назвіть найістотніші ознаки класифікації програм.
6. Що таке класифікація програм за призначенням.
7. Що таке класифікація програм за типом (різновидом) ліцензії.
8. Що таке прикладне програмне забезпечення?
9. Що таке базове програмне забезпечення?
10. Назвіть програми, що входять до складу базового програмного забезпечення.
11. Які функції виконує операційна система?
12. Що таке утиліти?
13. Які функції виконують програми-транслятори?
14. Які компоненти входять до складу повної обчислювальної системи?
15. Що таке інтерфейс у комп’ютерній системі?

16. Як називається інтерфейс між користувачем і прикладною програмою?

17. Як називається інтерфейс між прикладною програмою та операційною системою?

18. Як називається інтерфейс між операційною системою та комп'ютером?

19. З якою системою команд можуть працювати операційні системи Microsoft Windows, а з якою – Linux?

20. Що таке ліцензія на програмні засоби?

21. Назвіть головні категорії ліцензій на програмні засоби.

22. Назвіть найпоширенішу вільну ліцензію на ПЗ.

23. Які складники містить будь-яка комп'ютерна програма?

24. До якої серії операційних систем належить Linux?

25. Як називались проекти та операційна система, які ввійшли до складу Linux?

26. Хто вніс найбільший вклад у створення ОС Linux?

27. Назвіть головні складники ОС загального призначення.

28. Що таке термінал у ОС Linux?

29. Які типи терміналів існують в ОС Linux?

30. Що таке дистрибутив ОС Linux? Назвіть найвідоміші з них.

3 ФІЗИЧНА ТА ЛОГІЧНА ОРГАНІЗАЦІЯ ФАЙЛОВИХ СИСТЕМ

1 Поняття файлової системи

Однією з головних функцій операційної системи є організація зберігання інформації на *зовнішніх запам'ятовувальних пристроях* (далі – *ЗЗП*) комп'ютера. Основу зберігання інформації на ЗЗП (магнітних та оптичних дисках, flash-дисках тощо.) становить файлова система. Вона являє собою спосіб (метод) зберігання й організації доступу до даних на ЗЗП. Під файловою системою розуміють:

1) набір правил зберігання інформації на ЗЗП;
2) набір програм, які ці правила реалізують.

Кожен такий набір правил має своє унікальне ім'я – *назву файлової системи*.

2 Фізична організація зберігання інформації на магнітних дисках

З погляду на те, що основна маса інформації, оброблюваної на комп'ютері, зберігається на магнітних дисках, і більшість програм у тому або іншому ступені взаємодіють із ними, далі трохи докладніше зупинимося на способах організації фізичного зберігання й обміну інформацією на магнітних дисках.

Запис інформації на магнітні диски виконується за допомогою *головок запису-зчитування* (*write-read heads*) по концентричних колах, які називаються *доріжками*, або *треками* (*tracks*). Кожна доріжка має свій унікальний номер, починаючи з 0 від краю диска. Доріжки розділені на дрібніші одиниці зберігання інформації, які називаються *секторами* (*sectors*). Розмір сектора фіксований, і для більшості дисків становить 512 байт. Сектор є занадто малою одиницею зберігання й обміну інформацією, особливо для великих дисків. Отже, як компроміс між доріжкою й сектором був прийнятий *кластер* (*cluster*) – декілька секторів, що розташовуються підряд. Обмін інформацією між оперативною пам'яттю комп'ютера й диском виконується блоками даних, рівним кластеру. Дисковий простір під файли так само виділяється в кластерах. Останній виділений файлу кластер може бути заповнений тільки частково. Розмір кластера – величина, фіксована для кожного диска, і може бути встановлена в процесі підготовлення його

до експлуатації, що називається *форматування* (*formatting*). Вибір розміру кластера – це завжди компроміс між швидкістю запису-зчитування та ефективністю використання дискового простору. З одного боку він може бути рівним сектору, і дисковий простір буде використано максимально ефективно, а швидкість обміну – мінімальною. З іншого боку його розмір обмежений розміром доріжки; швидкість обміну максимальна, а ефективність використання – мінімальна. *Поверхні* (*sides*) магнітних дисків так само пронумеровані. Нумерація починається з 0 для самої верхньої поверхні, найвищого (або єдиного) диска. Кожному номеру поверхні однозначно відповідає також номер головки запису-зчитування. Конструкція приводів магнітних дисків така, що головки записи-зчитування не можуть позиціюватися на необхідні доріжки кожна індивідуально – вони переміщаються всі разом. Кожне таке їхнє положення над певною доріжкою називається *циліндром* (*cylinder*). Якщо подумки з'єднати за вертикаллю всі доріжки з однаковими номерами на всіх поверхнях магнітних дисків, то вийде циліндр. Апаратні й програмні засоби комп'ютера зазвичай працюють, використовуючи циліндри. Коли дані записуються на диск по циліндрах, до них можна звертатися без необхідності переміщення головок запису-зчитування. Оскільки переміщення головок повільне, порівняно зі швидкістю обертання пластин і перемиканнями між окремими головками, то використання циліндрів значно зменшують час доступу до даних.

У деяких випадках один *фізичний диск* може бути розділений на декілька *логічних дисків*. Це робиться у разі, коли розмір фізичного магнітного диска більше, ніж може підтримувати певна операційна система. Фізичний диск розділяється на логічні так само у разі, якщо на одному комп'ютері використається декілька операційних систем, які підтримують несумісні файлові системи.

Як було зазначено вище, дисковий простір під файл виділяється в блоках фіксованого розміру – кластерах. Кластери, займані файлом, не завжди виділяються підряд – зазвичай файли бувають фрагментованими. Інформація про імена файлів, займаних ними кластерах тощо зберігається в спеціально викресленому для цього місці – *змісті тому*.

3 Логічна організація файлових систем

Головними об'єктами логічної організації зберігання інформації на ЗЗП (файлової системи) є такі:

- *файл*;
- *папка* (*каталог*);
- *диск*.

Файлом називається пойменована сукупність однотипних даних на ЗЗП комп'ютера. Він має безліч різних параметрів. Головні з них такі:

- *ім'я*;
- *тип*;
- *розмір*.

Ім'я файлу вказується користувачем під час його створення. Необхідно дотримуватися декількох правил:

- ім'я файлу повинне бути унікальним у межах папки (каталогу);
- *довжина* імені файлу не повинна перевищувати 255 символів;
- ім'я файлу не повинне збігатися з одним із зарезервованих імен, таких як, **COM**, **LPT** тощо.
- в імені файлу не повинні зустрічатися деякі символи, такі, наприклад, як «*****», «**/**», «**?**» тощо.

На тип файлу вказує *розширення імені файлу*. Розширення імені файлу – це декілька символів (зазвичай три) після останньої праворуч крапки в імені файлу. Тип файлу вказує спосіб кодування інформації в ньому, а також програму, у якій він був створений. В операційній системі Microsoft Windows існує декілька зарезервованих типів файлів. Так, розширення **.TXT** мають текстові файли, створені у програмі Блокнот, **.DOC** – у Microsoft Word, а **.XLS** – у Microsoft Excel. Розширення імені файлу є необов'язковою характеристикою файлу. У разі його відсутності неможливо визначити програму, призначену для роботи з ним.

Спеціальним типом файлів є *ярлики*. Вони призначені для забезпечення швидкого доступу до інших об'єктів файлової системи. Тобто ярлик – це файл, що містить «адресу» іншого об'єкта файлової системи й має розширення **.LNK**.

Папки (*каталоги*) – це контейнерні об'єкти, які можуть містити файли й інші папки. Вони призначені для угруповання файлів у вигляді

деревоподібної структури. Імена папок будуються за тими самими правилами, що й файлів, тільки без розширення.

Диски можуть бут двох різновидів:

- *фізичні*;
- *логічні*.

Фізичні диски – це реальні ЗЗП комп'ютера, а логічні – це частини фізичних. Тобто, фізичний диск можно розбити на декілька логічних дисків, кожний з яких може мати свою файлову систему. В операційних системах сімейства Microsoft Windows диски йменуються початковими буквами латинського алфавіту, за яким обов'язково потрібно ставити двокрапку «**:**», наприклад, «**C:**».

Для роботи з об'єктами файлової системи (дисками, папками й файлами) призначені спеціальні програми, які називаються *файловими менеджерами*.

Контрольні питання

1. Що таке файлова система?
2. Що таке назва файлової системи?
3. Що таке доріжка (трек) магнітного диска?
4. Як нумеруються доріжки на магнітних дисках?
5. Що таке сектор на магнітному дискові?
6. Який розмір має сектор для більшості дисків?
7. Що таке кластер на магнітному диску?
8. В якому діапазоні лежить розмір кластера магнітного диска?
9. Коли встановлюється розмір кластера магнітного диска?
10. На що впливає розмір кластера магнітного диска?
11. Що таке циліндр магнітного диска?
12. Що таке логічний диск?
13. У яких випадках потрібні логічні диски?
14. Що таке зміст тому?
15. Назвіть об'єкти, що входять до складу файлової системи.
16. Що таке файл?
17. Якими параметрами характеризується файл?
18. Яких правил необхідно дотримуватися під час побудови імен файлів?

19. Що таке розширення імені файлу?

20. Де вказується розширення імені файлу?

21. Наведіть приклади розширення імен файлів для операційної системи Microsoft Windows.

22. Чи є обов'язковою характеристикою файлу розширення його імені?

23. Що таке ярлики у файлових системах?

24. Навіщо потрібні ярлики у файлових системах?

25. Яке розширення імені файлу мають ярлики?

26. Що таке каталоги (папки) у файлових системах?

27. Навіщо потрібні каталоги (папки) у файлових системах?

28. За якими правилами будуються імена каталогів (папок)?

29. Як називаються диски в операційних системах сімейства Microsoft Windows?

30. Як називаються програми, призначені для роботи з об'єктами файлових систем?

4 РЕАЛІЗАЦІЯ ФАЙЛОВИХ СИСТЕМ

1 Файлові системи Microsoft Windows

Операційні системи сімейства Microsoft Windows підтримують такі файлові системи:

1. **NTFS** – основна файлова система Microsoft Windows. Вона відрізняється підвищеною надійністю та розмежуванням прав доступу до інформації.

2. **FAT** – головна система попередніх версій операційної системи Microsoft Windows. Залишена для забезпечення сумісності з ними.

3. **CDFS** – файлова система оптичних дисків.

Для роботи з об'єктами файлової системи (дисками, папками й файлами) в операційній системі Microsoft Windows призначена спеціальна програма – *файловий менеджер – Проводник.*

2 Файлова система Linux

«Рідними» для операційних систем сімейства Linux є файлові системи *Ext3fs/Ext4fs* (Extended File System ver. 3/4 – «розширена файлова система версій 3 й 4»). Однак, Linux підтримує роботу й інші файлові системи, таких, наприклад, як FAT, NTFS, CDFS тощо, що забезпечує сумісність (прямий обмін інформацією) з іншими операційними системами. Програми, які реалізують роботу файлових систем, входять до складу так званої підсистеми вводу-виводу операційної системи. Докладнішу інформацію щодо файлових системах ОС Linux можна одержати по команді «`man fs`».

Головними об'єктами файлової системи Linux є такі:

1) *файли*;
2) *каталоги*.

Важливо пам'ятати, що у файловій системі Linux немає ні дисків, ні папок, ні документів – є тільки каталоги та файли.

Файл – це пойменована сукупність даних на ЗЗП. В інших операційних системах файли ще іноді називають документами. Кожен файл має безліч різних характеристик. Головною серед них, з погляду користувача, є ім'я файлу. Ім'я файлу зазвичай у якийсь спосіб відображає його вміст. У Linux, на відміну від інших операційних систем, не існує якого-небудь формату щодо імен файлів (як, наприклад, «8.3» у MS-DOS).

У ній немає ніяких приписів щодо розширення імені файлу: в імені файлу може бути будь-яка кількість крапок (зокрема й ні однієї), а після останньої крапки може бути будь-яка кількість символів. Навпаки, якщо ім'я файлу починається із символу крапка «.» – він стає «невидимим» для деяких команд. Ім'я файлу в Linux може бути завдовжки до 255 символів і містити будь-які символи, за винятком двох символів: з кодом 0 і похилої риски (слеш) «/». До того ж, *Linux завжди розрізняє прописні й малі літери, як в іменах файлів, так і в іменах каталогів*.

Е декілька символів, допустимих в іменах файлів і каталогів, які, при цьому, потрібно використати з обережністю. Це – так звані *спецсимволи*: «~», «`», «!», «@», «#», «$», «%», «&», «*», «(», «)», «[», «]», «{», «}», «\», «|», «;», «:», «'», «″», «<», «>», а також символи пробілу й табуляції. Якщо ж все-таки використається один або декілька таких символів, то все ім'я необхідно укласти в подвійні лапки «″».

Однак, з погляду ОС Linux ім'я файлу аж ніяк не є його головною характеристикою. У файловій системі Linux кожному файлу однозначно відповідає так званий «*індексний дескриптор*» (*inode*). Комп'ютеру зручніше працювати із числами, а людина краще сприймає символічні імена. Із цього виходить, що в Linux кожен файл може мати кілька *символічних імен* (*посилань*). Посилання бувають двох типів:

1) *жорсткі* (*hard links*);

2) *символічні* (*symbolic links*).

Жорсткі (тверді) посилання – це різні імена того самого вмісту. Кількість твердих посилань на ту саму область даних (файл) не обмежена, тобто у файлі може бути багато різних імен. Перше тверде посилання (перше ім'я) формується після створення файлу. Наступні тверді посилання створюються за допомогою команди «ln» (від англ. «link» – «з'єднувати, зв'язувати»), що має два параметри: перший параметр – це ім'я файлу, на який потрібно створити посилання, а другий – ім'я нового посилання. Після створення твердого посилання неможливо розрізнити, де вхідне ім'я, а де посилання. Коли видаляється файл, що має декілька різних імен – твердих посилань, то фактично видаляється тільки одне посилання – те, яке зазначене в команді видалення файлу. Для того, щоб видалити файл повністю (тобто звільнити займану їм область, і не мати більше можливості звертатися до цих даних) необхідно видалити всі його імена – усі тверді посилання. Номер індексного дескриптора будь-якого

файлу можна довідатися за допомогою команди «`ls`» із ключем «`-i`». *Тверді посилання створюються тільки в межах одного носія (однієї файлової системи), їх також не можна створювати на каталоги*.

Символьне (символічне) посилання – це просто файл, у якому записане ім'я іншого файлу. Вони називаються так тому, що містять символи – шлях до файлу або каталогу. Символьні посилання, як і тверді, надають можливість звертатися до того самого файлу по різних іменах. Крім того, символьні посилання можуть указувати на файли й каталоги в інших файлових системах (тобто на інших носіях, або навіть на інших комп'ютерах), чого не дозволяють тверді посилання. Символьні посилання в Linux аналогічні до ярликів у Microsoft Windows. Якщо вхідний файл або каталог вилучити, символьне посилання не видаляється, але стає марним – воно не буде «працювати». Наприклад, якщо спробувати вивести вміст такого «битого посилання» за допомогою команд «`cat`» або «`ls`», буде видане повідомлення про помилку. Створюються символьні посилання за допомогою тієї самої команди, що й тверді – «`ln`», тільки із ключем «`-s`» (від англ. «symbolic» – «символічний»).

Доти, доки в системі кількість файлів не занадто велика, не існує й проблеми їхнього пошуку. Якщо кількість файлів збільшується, наприклад, до декількох тисяч, то гостро встає питання їх упорядкування та систематизації. Для рішення цього завдання й були створені (придумані) *каталоги*.

Поняття *каталогу* (*directory*) дає змогу систематизувати всі об'єкти, розміщені на носії даних (наприклад, на диску). У більшості сучасних файлових систем використається деревоподібна структура організації каталогів, тобто каталоги можуть містити інші каталоги. Як зауважує Огастес Де Морган: «У більших каталогів є маленькі каталоги…». Каталог, на який є посилання у цьому каталозі, називається підкаталогом, вкладеним або дочірнім каталогом. Каталог, що містить поточний каталог, називається батьківським каталогом.

У будь-якій файловій системі Linux (і UNIX загалом) завжди є тільки один кореневий каталог («корінь», «root»), що позначається символом «`/`». Користувач Linux завжди працює з єдиним деревом каталогів, навіть якщо різні дані розташовані на різних носіях: декількох твердих або мережних дисках, знімних дисках, CD-ROM тощо. Для того, щоб підключати й відключати файлові системи на різних пристроях в одне загальне дерево,

використаються процедури *монтування* й *розмонтування* (які може виконувати тільки користувач із правами адміністратора – «`root`»). Після того, як файлові системи на різних носіях підключені до загального дерева дані, що на них записані доступні так, ніби всі вони становлять єдину файлову систему: користувач може навіть не знати, на якому пристрої які файли зберігаються.

Організація каталогів файлової системи у вигляді дерева не допускає появи циклів: тобто каталог не може містити в собі каталог, у якому втримується сам. У наслідок цього обмеження шлях до будь-якого каталогу або файлу завжди буде однозначним і кінцевим.

Імена каталогів у Linux будуються за тими самими правилами, що й імена файлів. І, взагалі, каталоги в принципі нічим, крім своєї внутрішньої структури, не відрізняються від «звичайних» файлів, наприклад, текстових. Тобто кожен каталог у Linux – це окремий файл особливого типу («`d`» від англ. «directory»), що відрізняється від звичайного файлу з даними тим, що в ньому можуть утримуватися тільки посилання на інші файли й каталоги.

У кожен момент часу користувач працює тільки з одним, так званим, «*поточним*» каталогом. Поточний каталог – це каталог, у якому запущені програми користувача за замовчуванням (тобто якщо явно не зазначений інший каталог) читають і записують дані. У Linux є спеціальна команда «`pwd`» (абревіатура від англ. «print working directory» – «надрукувати (вивести) робочий каталог»), що повертає ім’я (повний шлях) поточного каталогу.

Шляхом (англ. «path, pathname»; «стежкою» до файлу) називається список каталогів, яким необхідно «пройти в дереві каталогів» (маршрут, траєкторія), щоб потрапити в цільовий каталог. Каталоги в шляху від інших відокремлюються тим самим символом «`/`», що слугує для позначення кореневого каталогу. Тому символом «`/`» не можна використати в іменах файлів і каталогів. Розрізняють повний шлях, що починається в кореневому каталозі (і починається символом «`/`»), і відносний шлях, що починається в поточному каталозі (і, відповідно, не починається символом «`/`»).

Крім поточного каталогу, в ОС Linux для кожного користувача визначений ще його «*домашній каталог*» (аналог папки «*Мои документи*» Microsoft Windows) – каталог, призначений для зберігання власних даних

користувача. У цьому каталозі користувач має всі права: може створювати та видаляти файли й каталоги, міняти права доступу до них тощо. У каталоговій структурі Linux домашні каталоги користувачів зазвичай розміщаються в каталозі `/home` і мають імена, що збігаються з іменами користувачів. Наприклад, `/home/student21` – повний шлях у домашній каталог користувача `student21`. Кожен користувач може звертатися до свого домашнього каталогу за допомогою символу «тильда» – «~» (у системі не існує каталогу з ім'ям «~» – це просто «синтаксичний цукор»). Тому рядок «`~/temp`» означає каталог `temp`, що перебуває в домашньому каталозі. Коли користувач входить у систему, що цей каталогом стає його домашній каталог.

Для зміни поточного каталогу (переміщення, «подорожі» по файловій системі) слугує команда «`cd`» (від англ. «change directory» – «змінити каталог»). Команда «`cd`» має один параметр – ім'я каталогу (повний або відносний шлях), у який потрібно переміститися, тобто зробити поточним. Команда «`cd`» без параметрів еквівалента команді «`cd ~`» і робить поточним каталогом домашній каталог користувача.

Команда «`ls`» (від англ. «list» – «список») без параметрів служить для виводу на екран вмісту поточного каталогу – імен файлів і вкладених підкаталогів. Якщо потрібно переглянути вміст не поточного, а якогось іншого, потрібно зрозуміло вказати команді «`ls`» повний або відносний шлях до цього каталогу. Для одержання додаткової інформації про файли й каталоги використаються додаткові параметри – *ключі* (*опції*, *прапорці*). Наприклад, команда «`ls`» із ключем «`-l`» (від англ. «long» – «довгий») виводить розширену інформацію про вміст каталогу. Крім імен файлів і каталогів, у цьому разі відображається інформація про їхніх власників, кількість твердих посилань, правах доступу, розмірі й даті останньої модифікації. Якщо задати ключ «`-i`» (від англ. «index»), то будуть відображені номери індексних дескрипторів. Ключ «`-F`» до кожного імені додає суфікс: символ «/» – для каталогів, символ «*» – для програм (виконавчих, двійкових файлів). Для відображення схованих файлів (імена яких починаються на символ крапка «`.`») слугує ключ «`-a`» (від англ. «all» – «всі»). Ключ «`-d`» (directory) слугує для того, щоб команда «`ls`» виводила інформацію не про вміст каталогу, а про сам цей каталог (як про файл, що містить список посилань на файли та вкладені каталоги).

Порядок ключів у команді довільний; їх так само можна перелічити як окремо, так і поєднувати, наприклад, «`ls -l -i`», «`ls -i -l`», «`ls -li`», або «`ls -il`».

На структуру каталогів UNIX-подібних операційних систем існує досить строгий стандарт – так званий Filesystem Hierarchy Standard (FHS). Він регламентує не тільки розташування основних каталогів, але і їхні підкаталоги, а іноді навіть приводить список файлів, які повинні бути присутнім у певних каталогах. Цей стандарт послідовно дотримується у всіх Linux-системах. Корінний каталог більшості ОС Linux має приблизно такий зміст:

- `/bin` (від англ. «binaries» – «двійкові, що виконуються») – у цьому каталозі перебувають виконувані файлі, найбільш необхідних утиліт (команд) Linux, таких, наприклад, як «`pwd`», «`ls`», «`mv`» тощо;
- `/boot` («завантаження системи») – у цьому каталозі перебувають файли, необхідні для завантаження системи, зокрема саме ядро Linux;
- `/dev` (від англ. «devices» – «пристрій») – у цьому каталозі перебувають есі наявні в системі драйвери пристроїв – вони використаються для доступу до пристроїв і ресурсів системи, таким, наприклад, як диски, модеми, пам’ять тощо;
- `/etc` (від англ. «et cetera» – «різне, і так далі») – містить різні службові файли, зокрема, – системні конфігураційні файли;
- `/home` («додому») – тут розташовані домашні каталоги користувачів;
- `/lib` (від англ. «libraries» – «бібліотеки») – у цьому каталозі перебувають образи поділюваних бібліотек (shared library images) – файли, що містять код, що можуть використати багато програм;
- `/lost+found` («втратити й знайти») – цей каталог використається під час відновлення файлів системи командою `fsck`;
- `/mnt` (від англ. «mount» – «монтувати») – каталог для монтування – тимчасового підключення файлових систем, наприклад, на знімних носіях (CD-ROM тощо);
- `/proc` – це «віртуальна файлова система», у якій файли зберігаються в пам’яті, а не на диску; вони обумовлені різними процесами, що відбуваються в системі, і дають змогу одержати інформацію про те, що роблять програми й процеси в зазначений час;

• /root – домашній каталог адміністратора системи – користувача «root»;

• /sbin (від англ. «system binaries») – на додаток до утиліт каталогу /bin тут перебувають програми, необхідні для завантаження, резервного копіювання та відновлення системи, повноваження, на виконання яких, є тільки в системного адміністратора;

• /tmp (від англ. «temporaries» – «тимчасові файли») – цей каталог призначений для тимчасових файлів: у таких файлах програми зберігають проміжні дані, необхідні їм тільки на час роботи; після завершення роботи програми тимчасові файли втрачають зміст і, зазвичай, видаляються;

• /usr – це «держава в державі». Тут можна знайти такі самі підкаталоги bin, etc, lib, sbin, як й у кореневому каталозі. Однак у кореневий каталог надходять тільки утиліти, необхідні для завантаження і відновлення системи в аварійній ситуації, всі інші програми й дані розташовуються в підкаталогах /usr;

• /var (від англ. «variable» – «змінні») – тут розміщаються ті дані, які створюються в процесі роботи різними програмами та призначені для передачі іншим програмам і системам (черги друку, електронної пошти тощо) або для системного адміністратора (системні журнали, що містять протоколи роботи системи). На відміну від каталогу /tmp сюди надходять ті дані, які можуть знадобитися після того як програма, що їх створила, завершила свою роботу.

Докладніше з головними каталогами Linux можна познайомитися по тексту стандарту FHS, повністю якому можна знайти в Інтернеті за адресою http://www.pathname.com/fhs/.

Файлова система ОС Linux підтримує наступні типи файлів, які позначаються першим символом у полі прав доступу, і приймають такі значення:

• «-» – звичайний файл – може містити текстові дані, музику, фільм, програму тощо;

• «d» – каталог – файл, що може містити тільки список посилань на інші вкладені файли й каталоги;

• «b» – файл блокового пристрою – пристрій, запис і читання інформації, в якому виконується блоками, наприклад, жорсткий диск;

• «c» – файл символьного пристрою – пристрій, запис і читання інформації в якому виконується посимвольно, наприклад, термінал;

• «s» – доменне гніздо (від англ. «socket») – це з'єднання між процесами, що дає змогу їм взаємодіяти, не підпадаючи під вплив інших процесів. Вони є ключовим поняттям TCP/IP і, відповідно, на них цілком будується Інтернет. Серед стандартних засобів, що використають гнізда – система *X Window*, система друку і система *syslog*;

• «p» – іменований канал (від англ. «pipe» – «труба») – або буфер FIFO (First In – First Out) слугує переважно для того, щоб організувати обмін даними між різними додатками (процесами). Усе, що один процес поміщає в канал, інший може звідти прочитати. Іменовані канали створюються за допомогою команди «mkfifo»;

• «l» – символічне посилання (від англ. «link») – файл, що може містити тільки символічний покажчик (посилання) на інший файл.

Визначити тип файлу можна і на підставі його вмісту. Багато форматів файлів передбачають вказівку на початку файлу, як варто інтерпретувати подальшу інформацію: як програму, вихідні дані для текстового редактора, сторінку HTML, звуковий файл, зображення або щось інше. Для цього в Linux є команда «file», що читає перші декілька сотень байт файлу, шукає в них ключові послідовності символів і на цій підставі робить припущення про тип файлу.

У Linux для відділення шляху до файлу від його імені слугують команди «dirname» та «basename», відповідно. Наприклад, команда «basename /etc/fstab» видасть ім'я файлу «fstab», а команда «dirname /etc/fstab» – шлях до нього – «/etc». Однак команди «dirname» та «basename» ніяк не аналізують тип файлу – вони просто розділяють останній компонент повного імені та все інше. Наприклад, команди «dirname /etc/foomatic» й «basename /etc/foomatic» видадуть «/etc» й «foomatic», відповідно, – вони просто відокремили останній компонент «foomatic». Хоча «foomatic» – це каталог («ls -ld /etc/foomatic»).

Оскільки Linux – система багатокористувацька та багатозадачна, питання розмежування прав доступу до файлів і каталогів є одним з істотних моментів функціонування операційної системи (необхідно

захистити файли кожного користувача від «дурного впливу» файлів інших користувачів).

В основі механізму розмежування доступу лежать імена користувачів та імена груп користувачів. У Linux кожен користувач має унікальне ім'я, під яким він входить у систему. Крім того, у системі створюється деяка кількість груп користувачів, до того ж кожен користувач може бути включений в одну або декілька груп. Члени різних груп можуть мати різні права доступу до файлів і каталогів. Наприклад, група адміністраторів має більше прав, ніж група програмістів. Створює та видаляє групи користувачів суперкористувач («root»), він же може змінювати склад тієї або іншої групи.

Наприклад, команда «ls –l /bin/ls» відображає інформацію про саму себе – про бінарний (виконавчому, програмному) файлі «/bin/ls», що запускається за командою «ls»

```
-rwxr-xr-x 1 root root 92136 Сен 20 2009 /bin/ls.
```

Пояснення до команди, з права на ліво: «/bin/ls» – повне ім'я файлу, «Сен 20 2009» – дата останньої модифікації файлу, «92136» – розмір файлу (у байтах), «root» – ім'я групи, «root» – ім'я власника файлу, «1» – кількість твердих посилань на файл, «rwxr-xr-x» – права доступу до файлу, «-» – тип файлу (звичайний файл).

Права доступу (друге поле ліворуч) підрозділяються на три типи: *читання* (*read*), *запис* (*write*) і *виконання* (*execute*). Ці типи прав доступу можуть бути надані трьом класам користувачів: власникові файлу, групі, у яку входить власник, і всім іншим користувачам.

Дозвіл на читання дає змогу користувачеві читати вміст файлів, а у разі каталогів – переглядати перелік імен файлів у каталозі (наприклад, за командою «ls»). Дозвіл на запис дає змогу користувачеві писати у файл і змінювати його. Для каталогів це надає право створювати та видаляти в ньому вкладені файли й каталоги. Нарешті, дозвіл на виконання дає змогу користувачеві виконувати файли (як бінарні програми, так і командні файли). Дозвіл на виконання стосовно каталогів означає: по-перше, зробити цей каталог поточним (команда «cd»), а по-друге, звертатися за

доступом до файлів, що містяться в ньому. Тоді для файлу /bin/ls: власник («root») має всі права (rwx – читати, змінювати й виконувати), а група й всі інші користувачі – тільки читати й виконувати, але не змінювати (r-x).

Важливо відмітити, що права доступу, які має файл, залежать також від прав доступу до каталогу, у якому цей файл знаходится. Наприклад, навіть якщо файл має «-rwxrwxrwx», інші користувачі не зможуть до нього добратися, якщо в них не буде прав на читання й виконання каталогу, у якому перебуває цей файл. Наприклад, якщо користувач захоче обмежити доступ до всіх своїх файлів, він може просто змінити права доступу свого домашнього каталогу на «drwx------». Отже, ніхто не буде мати доступ до його каталогу (крім «root»), а, отже, стороннім будуть недоступні й всі файли – отже, можна не піклуватися про індивідуальний захист своїх файлів. Інакше кажучи, щоб мати доступ до файлу, необхідно мати доступ до всіх каталогів, що лежать на шляху від кореня до цього файлу, а також дозвіл на доступ власно до цього файлу. У разі звертання до файлу або каталогу, доступу до якого в користувача нема, йому буде видане повідомлення типу «Permission denied» – «Відмовлено в доступі».

У символічному посиланні не використаються права доступу – вони завжди відображаються як «lrwxrwxrwx». Замість цього, права доступу до файлу, отриманому символічним посиланням, визначаються правами доступу до файлу, на який посилаються.

Для установлення (зміни) прав доступу до файлу служить команда «chmod» (від англ. «change mode» – «змінити режим (доступу)»), що має такий синтаксис:

```
chmod {a|u|g|o}{+|-}{r|w|x} <ім'я файлу>.
```

Пояснення до синтаксису:

1) застосувати до такого:

- «a» (від англ. «all» – «всі») – користувачам всіх категорій (власникові, групі та всім іншим);
- «u» (від англ. «user» – «користувач») – власникові файлу;
- «g» (від англ. «group» – «група») – групі;
- «o» (від англ. «other» – «інші») – усім іншим користувачам.

2) права:

- «+» – додати;
- «−» – позбавити.

3) на таке:

- «r» (від англ. «read») – читання;
- «w» (від англ. «write») – запис
- «x» (від англ. «execute») – виконання (використання).

Наприклад команда

```
chmod a+x file_name
```

надає всім користувачам системи право на виконання файлу `file_name`, команда

```
chmod go-rw file_name
```

позбавляє права на читання та запис для всіх, крім власника файлу, а команда

```
chmod ugo+rwx file_name
```

дає всім право на читання запис і виконання. Якщо опустити вказівку на те, кому надається це право, то йдеться про те, що мова йде взагалі про всіх користувачів, тобто замість

```
chmod a+x file_name
```

можна записати просто

```
chmod +x file_name.
```

Виконувати зміну права доступу до файлу або каталогу за допомогою команди «chmod» може тільки власник або суперкористувач («root»). Для того щоб мати можливість змінювати права групи, власник повинен додатково бути членом тієї групи, якій він хоче дати права на цей

файл. Змінювати власника й групу, можливо за допомогою команд «chown» та «chgrp», відповідно, виконувати які може тільки суперкористувач.

Для створення каталогів в ОС Linux слугує команда «mkdir» (від англ. «make directory» – «створити каталог»), що як єдиний обов'язковий параметр приймає ім'я (шлях) створюваного каталогу. Наприклад, команда «mkdir my_dir» створює в поточному каталозі підкаталог з ім'ям my_dir. Якщо тепер виконати команду «ls -a my_dir», то вона видасть список із двох імен «.» та «..» – у знову створеному каталозі автоматично створюються два записи: «.» – посилання на цей каталог та «..» – посилання на батьківський каталог. Хоча створювати тверді посилання на каталог не можна, команда «ls -ld my_dir», крім усього іншого, покаже, що тип файлу «d» – каталог, що має два тверді посилання: власне ім'я – «my_dir» та ім'я «.». Якщо у каталозі створити підкаталог, кількість твердих посилань на цей каталог збільшиться на одне за допомогою імені «..» у створеному підкаталозі.

Для видалення непотрібних об'єктів файлової системи – файлів і каталогів – слугують команди «rm» (від англ. «remove» – «видаляти») та «rmdir» (від англ. «remove directory» – «видалити каталог»), відповідно. Команда «rm» призначена саме для видалення твердих посилань, а не самих файлів. У Linux, щоб повністю видалити файл, потрібно послідовно видалити всі тверді посилання на нього. Усі тверді посилання на файл (його імена) рівноправні – серед них нема «головного», зі зникненням якого зникне файл. Поки є хоч одне посилання, файл (його вміст) продовжує існувати. Для того щоб скористатися командами «rm» та «rmdir», необхідно мати право на запис у каталог, у якому розташовані файли або каталоги, що видаляються. При цьому повноваження на зміни самих файлів не обов'язкові. Для видалення каталогу необхідно на початку видалити в ньому всі файли та вкладені підкаталоги, а тільки потім видалити порожній каталог за допомогою команди «rmdir». Однак можна видалити й непустий каталог, якщо скористатися командою «rm» із ключем «-r» (від англ. «recursive» – «рекурсивно»). Команда «rm -r <ім'я каталогу>» – дуже зручний спосіб втратити відразу всі файли: вона рекурсивно обходить весь каталог, видаляючи все, що попадеться: файли, підкаталоги, символьні посилання. При цьому завжди необхідно

пам’ятати, що в Linux не передбачено процедури відновлення вилучених файлів і каталогів (вилучені файли й каталоги відразу попадають у піч, а не в сміттєвий кошик, і відновити їх із попелу вже не вдасться). Більшість дистрибутивів Linux настроєні так, що видалення файлу або каталогу відбувається «мовчки» – ніякого попереджувального повідомлення при цьому не видається. Для видавання додаткового запиту на підтвердження операції видалення слугує ключ «`-i`», зневажати яким не варто.

У Linux для копіювання файлів і каталогів призначена утиліта «`cp`» (від англ. «copy» – «копіювати»), що потребує присутності двох обов’язкових параметрів: перший – файл або каталог, що копіюють (звідки), другий – файл або каталог призначення (куди). Потрібно зважити на те, що під час копіювання файлу або каталогу поверх уже наявного не виводиться ніякого попередження! (Говорячи про копіювання, доречно згадати широко відоме висловлення, приписуване Вільяму Оккаму: «Не слід множити сутності більш необхідного».) Втім, можна використати команду «`cp`» із ключем «`-i`», тоді перед записом поверх наявного файлу буде запитуватися підтвердження.

Якщо необхідно не скопіювати, а перемістити файл із одного каталогу в інший – для цього існує команда «`mv`» (від англ. «move» – «переміщати»). Синтаксис й опції в неї такі ж, як і команди «`cp`». Вона спочатку копіює файл або каталог, а тільки потім видаляє вхідний. Команда «`mv`» може використатися не тільки для переміщення, але й для перейменування файлів і каталогів – для цього необхідно просто задати як аргументи старе й нове ім’я. Під час використання команди «`mv`», також як і під час використання команди «`cp`», рекомендується використати опцію «`-i`» для того, щоб одержати попередження, коли файл буде перезаписуватися.

Багатоцільова команда «`cat`» (від англ. «concatenate» – «об’єднати»), слугує для такого:

- конкатенації (об’єднання, злиття) декількох файлів в один. Наприклад, команда

```
cat file1 file2 file3 > new_file
```

поєднує три файли в один, тобто `file1` + `file2` + `file3` = `new_file`.

- копіювання файлів. Наприклад, команда

```
cat file1 > file2
```

копіює `file1` в `file2`.

- створення текстових файлів. Наприклад, команда

```
cat > new_file
```

створює файл `new_file` на основі введення зі стандартного пристрою уведення – клавіатури. Для завершення введення (і команди «`cat`») слугують сполучення клавіш **`Ctrl+D`** та **`Ctrl+C`**. Сполучення клавіш **`Ctrl+C`** посилає програмі, що зчитує із клавіатури, сигнал аварійного припинення роботи, а **`Ctrl+D`** повідомляє їй, що закінчено уведення чергового рядка, і можна продовжувати роботу далі; якщо ж рядок порожній – то закінчити роботу програми. Можна вважати, що **`Ctrl+C`** – це скорочення від англійського «Cancel» («Скасування»), а **`Ctrl+D`** – від «Done» («Зроблено»);

- відображення вмісту файлу. Наприклад, команда

```
cat file
```

виводить вміст зазначеного файлу `file` на системний пристрій виводу – на дисплей. Однак, команду «`cat`» зручно використовувати для виводу тільки дуже невеликих по обсягу файлів – вміст великого файлу миттєво проскакує по екрану, і користувач бачить тільки останні рядки.

Для перегляду більших за обсягом файлів зручніше користуватися командами «`more`» («більше») і «`less`» («менш»). Команда «`more`» виводить вміст файлу посторінково. Для того, щоб побачити наступну сторінку, необхідно натиснути клавішу пробіл, а попередню – «**`B`**». Натискання клавіші **`Enter`** призводить до зсуву на один рядок. Для завершення роботи команди (не пролистуючи весь файл до кінця) необхідно нажати клавішу «**`Q`**». Команда «`less`» містить всі функції команди «`more`» і деякі додаткові.

Керування роботою команди «less» просто – за допомогою таких клавіш:

- **Q** (від англ. Quit) – завершити;
- ↓ або **Enter** – рядок вниз;
- ↑ – рядок нагору;
- **PgDn** або **Space** (**Пробіл**) – сторінка (екран) вниз;
- **PgUp** або **B** – сторінка (екран) нагору;
- **g** – початок тексту;
- **G** – кінець тексту;
- **?** або **h** (від англ. «help» – «допомога») – вичерпна інформація роботи з командою.

До речі, для відображення сторінок інтерактивного керівництва команда «man» за замовчуванням використовує команду «less».

Для пошуку файлів і каталогів в ОС Linux використається команда «find» («пошук»). Вона може шукати файли за іменем, розміром, датою створення або модифікації та деякими іншими критеріям, а також виконувати різні операції над знайденими об'єктами.

Загальний синтаксис команди «find» має такий вигляд:

```
find [<список каталогів>] <критерій пошуку>.
```

Параметр <список каталогів> визначає сферу пошуку. Якщо задати як початковий каталог кореневий каталог «/», то пошук може затягтися дуже надовго, оскільки буде проглядатися вся структура каталогів, включаючи змонтовані файлові системи (зокрема й мережні, якщо такі є). Якщо не вказати жодного шляху, пошук буде вестися тільки в поточному каталозі та його підкаталогах.

Початком <критерію пошуку> вважається перший аргумент, що починається на один із символів: «-», «(», «)», «,», «!». Усі аргументи, що передують <критерію пошуку>, трактуються як імена каталогів, у яких необхідно робити пошук.

Зазвичай пошук виконується за іменем файлу, наприклад, команда

```
find /usr/share/doc /usr/doc /usr/local/doc -name Mousepad
```

у зазначених каталогах шукає розташування документації по текстовому редакторові *Mousepad*, команда

```
find /usr/share/doc /usr/doc /usr/local/doc -name xfce*
```

шукає документацію по графічному інтерфейсі *xfce*, а команда

```
find -name file_name?
```

у поточному каталозі шукає файли з іменами `file_name1`, `file_name2`, `file_namec` тощо.

Для пошуку файлів і каталогів, імена яких відомі не зовсім точно, найчастіше використаються такі символи підстановки (шаблона, заміни):

- «`*`» – означає будь-яка кількість будь-яких символів;
- «`?`» – один будь-який символ.

Наприклад, за шаблоновом «`*.doc`» будуть знайдені всі файли з будь-яким іменем та розширенням «`.doc`», по шаблонові «`file_name.*`» – файли з ім'ям «`file_name`» і будь-яким розширенням. Символи підстановки використаються не тільки в команді «`find`», але й у багатьох інших командах Linux для масової обробки файлів однією командою. Детальнішу інформацію по команді «`find`» можна одержати на інтерактивних `man-сторінках`.

У разі редагування та збереження файлів досить часто виникає завдання порівняння різних їхніх версій (копій), а також автоматичного внесення змін у старішу версію. Для порівняння файлів в ОС Linux є команда «`diff`» («різниця»), що порівнює два файли й видає інформацію про наявні розходження. Якщо ці розходження зберегти в окремому файлі, то потім за допомогою команди «`patch`» («шматок», «златка») їх можна внести в старішу версію файлу.

Наприклад, команда

```
diff file1 file2 > file.diff
```

записує у файл `file.diff` відмінності між файлами `file1` й `file2`, а команда

```
patch file1 file.diff
```

застосовує до файлу `file1` «златки» – файл `file.diff`. У наслідок цього вміст файлу `file1` стає таким самим, як і `file2`. Така процедура використається, наприклад, якщо над одним файлом працює декілька людей одночасно в різних місцях і розсилання тільки змін дає змогу кожному авторові самостійно вирішувати, застосовувати їх чи ні.

Для роботи з об'єктами файлової системи в ОС Linux найчастіше використаються такі команди:

- `pwd` – відобразити повний шлях у поточний каталог;
- `ls [<ключі>] [<ім'я файлу або каталогу>]` – відобразити вміст каталогу або параметри файлу;
- `file [<ключі>] [<ім'я файлу>]` – визначити тип файлу;
- `dirname <повне ім'я файлу>` – вивести ім'я каталогу із заданого повного імені;
- `basename <повне ім'я файлу>` – вивести ім'я файлу із заданого повного імені;
- `cd [<ім'я каталогу (шлях)>]` – змінити поточний каталог;
- `mkdir [<ключі>] <ім'я каталогу>` – створити каталог;
- `rm [<ключі>] <ім'я файлу або каталогу>` – видалити файл або каталог;
- `rmdir [<ключі>] <ім'я каталогу>` – видалити непустий каталог;
- `cp [<ключі>] <ім'я вихідного файлу або каталогу (звідки)> <ім'я результуючого файлу або каталогу (куди)>` – копіювати файли й каталоги;

• mv [<ключі>] <ім'я вихідного файлу або каталогу (звідки)> <ім'я результуючого файлу або каталогу (куди)> – перемістити або перейменувати файли й каталоги;

• cat [<ключі>] <імена файлів> – об'єднати файли або копіювати, у тому числі, на стандартний вивід – монітор;

• more [<ключі>] <ім'я файлу> – посторінково відобразити вміст файлу на стандартному пристрої виводу – моніторі;

• less [<ключі>] <ім'я файлу> – посторінково відобразити вміст файлу на стандартному пристрої виводу – моніторі;

• ln <існуюче ім'я файлу> <нове ім'я> – створити тверде посилання;

• ln –s <існуюче ім'я файлу або каталогу> <нове ім'я> – створити символічне посилання;

• find [<список каталогів>] <критерій пошуку> – знайти файл або каталог;

• diff [<ключі>] <ім'я файлу1> <ім'я файлу2> – знайти розходження між двома файлами;

• patch [<ключі>] [<ім'я вихідного файлу> [<латки>]] – застосувати «латки» до вихідного файлу;

• chown <нове ім'я власника> <ім'я файлу> – змінити власника файлу (може виконувати тільки суперкористувач);

• chgrp <нове ім'я групи> <ім'я файлу> – змінити групу (може виконувати тільки суперкористувач або власник файлу, але він повинен входити в групу, який повинен дати права на файл).

Контрольні питання

1. Що таке файлова система?
2. Що таке назва файлової системи?
3. Які файлові системи є головними для ОС Linux?
4. Назвіть файлові системи, з якими може працювати ОС Linux.
5. Як називається та частина операційної системи, що забезпечує роботу файлових систем?
6. З якими об'єктами працює файлова система Linux?
7. Що таке файл?

8. Назвіть головну характеристику файлу з погляду користувача.

9. За якими правилами будується ім’я файлу?

10. Що таке індексний дескриптор?

11. Що таке тверде посилання? Що таке символічне посилання? Що в них загального та чим вони відрізняються?

12. Що таке каталог? Навіщо він потрібний?

13. Як підключити й відключити файлову систему на іншому носії в загальне дерево каталогів ОС Linux?

14. За якими правилами будується ім’я каталогу?

15. Що таке поточний каталог? Що таке домашній каталог?

16. Як, і де, можна подивитися ім’я поточного каталогу?

17. Як змінити поточний каталог?

18. Як подивитися вміст каталогу? Яку інформацію при цьому можна побачити?

19. Що таке FHS? Назвіть головні (під)каталоги кореневого каталогу ОС Linux.

20. Назвіть типи файлів, які підтримує Linux. Як можна подивитися тип файлу?

21. Як організоване розмежування доступу до файлів і каталогів в Linux? Як подивитися права доступу до файлу?

22. Як змінити права доступу до файлу? Хто може їх змінювати?

23. Як створити каталог? Що означають імена: «`.`» і «`..`»?

24. Як видалити файл, каталог?

25. Як скопіювати файл, каталог?

26. Як перемістити, перейменувати файл, каталог?

27. Для чого слугує (що може виконувати) команда «`cat`»?

28. Як переглянути файл, що не міститься на екрані монітора?

29. Як знайти файл або папку? Що таке символи шаблона, як вони використаються?

30. Як зрівняти два файли? Як автоматично відкоригувати старішу версію файлу?

5 ВЗАЄМОДІЯ З КОРИСТУВАЧЕМ В ОПЕРАЦІЙНИХ СИСТЕМАХ

1 Взаємодія з користувачем

Керування роботою UNIX-подібної операційної системи (Linux) здійснюється за допомогою *термінала*. Під терміналом (*консоллю*) розуміється пристрій, призначений для обміну інформацією між користувачем і комп'ютером. У мінімальній конфігурації термінал складається з монітора й клавіатури, миша – не обов'язкова. Відомо два типи терміналів (*інтерфейсів*):

1) *текстовий* (*інтерфейс командного рядка – Command Line Interface – CLI*);

2) *графічний* (*Graphical User Interface – GUI*).

У більшості сучасних дистрибутивів Linux параметри настроєні так, що за замовчуванням завантажується графічний інтерфейс (графічний термінал).

ОС Linux дає змогу організувати одночасну роботу до дванадцяти віртуальних терміналів (віртуальних консолей) – за кількістю функціональних клавіш на клавіатурі. Зазвичай, є шість текстових віртуальних терміналів і не менш одного графічного. Перемикання із графічного інтерфейсу (режиму, термінала) на перший текстовий віртуальний термінал виконується за допомогою сполучення клавіш **Ctrl+Alt+F1**, на другий – **Ctrl+Alt+F2**, а на шостий – **Ctrl+Alt+F6**. Перемикання віртуальних терміналів у текстовому режимі виконується за допомогою сполучень клавіш **Alt+F$_n$**, де **n** – номер віртуального термінала. Сполучення клавіш **Alt+F7** здійснює перемикання з текстового режиму в графічний – сьомий віртуальний термінал. Однак для перемикання віртуальних терміналів зручніше запам'ятати сполучення клавіш **Ctrl+Alt+F$_n$** – воно спрацьовує в обох режимах: і в текстовому, і в графічному.

Для того щоб почати роботу с ОС Linux, користувач повинен зареєструватися в системі. Процедура реєстрації складається із двох кроків:

1) введення імені;

2) введення паролю.

Їх можна одержати у викладача або системного адміністратора. Під час введенні імені та пароля необхідно звертати увагу на прописні й малі літери – Linux «чутлива» до регістра символів. Пароль під час введення не відображається – щоб його ніхто не зміг підглянути. Якщо під час введення імені або пароля була допущена помилка, система видасть приблизно таке повідомлення: «`Login incorrect`», і процедуру реєстрації прийдеться повторити.

Після успішної реєстрації (початку сеансу; входу, «як у кімнату»; «логіювання» і тощо) на екрані монітора з'являється запрошення для введення команди (як правило, символ «`$`»). У текстовому режимі введення кожної команди (порції інформації) завершується шляхом натискання клавіші **`Enter`**. Команда в ОС Linux складається з імені, за яким можуть бути її параметри – якщо такі є. Параметри команди від її імені та одне від одного відокремлюються, принаймні, одним пробілом.

Кожен користувач операційної системи Linux рано або пізно зіштовхується з питанням: «Як виконати ту, або іншу дію?» При цьому у новачків такі питання виникають частіше, у досвідчених користувачів – рідше. Розроблювачі Linux супроводили її великою та потужною довідковою системою, одержати інформацію в якій можна декількома різними способами. Наприклад, за допомогою наступних команд:

- `help` (допомога);
- `man` (від. англ. «manual» – «керівництво»);
- `info` (гіпертекстове керівництво).

Правило, відповідно до якого рішення будь-якого завдання потрібно починати з вивчення документації, по-англійському називається «Read That Fine Manual» (RTFM) – Читайте це найкраще керівництво. Однак довідкова система Linux – це зовсім не підручник, це – дійсно довідник. У ньому утримується інформація, достатня для освоєння описуваного об'єкта, але ніяких навчальних прийомів, ніяких визначень, повторень і виокремлення головного в ньому звичайно немає. Тим більше не допускається усікання керівництва з метою представити невеликі по обсягу, але найбільш важливої інформації як це прийнято в підручниках. Все це має на меті не навчити, а розкрити зміст, пояснити. Найбільш раціональний спосіб одержання довідкової інформації з об'єкта, що

цікавить в даний момент, – відкрити довідку на іншому віртуальному терміналі.

Традиційно документація по Linux у переважній більшості випадків написана на простій англійській мові (навіть у русифікованих дистрибутивах). Якщо англійський – не рідна мова для автора документа, вона буде тільки простіше. Так що вибір, як завжди, за Вами – учити англійську, чи ...

Головний спосіб одержання підказки у всіх UNIX-системах – команда `man`. Довідкова інформація в ній організована у вигляді так званих сторінок керівництва (`manpages`). Кожна сторінка керівництва (для стислості – просто «керівництво») присвячена якому-небудь одному об'єкту системи.

Сторінка керівництва займає, зазвичай, більше одного екрана. Керування роботою команди `man` просто – за допомогою таких клавіш:

- **`Q`** (від англ. «quit» – «вихід з») – завершити;
- ↓ або **`Enter`** – рядок униз;
- ↑ – рядок нагору;
- **`PageDn`** або **`Space`** (**`Пробіл`**) – сторінка (екран) униз;
- **`PageUp`** або **`B`** – сторінка (екран) нагору;
- **`g`** – початок тексту;
- **`G`** – кінець тексту;
- **`?`** або **`h`** (від англ. «help» – «допомога») – вичерпна інформація що до роботи з командою.

Сторінка керівництва складається з так званих «полів» – стандартних розділів, із різних боків описують об'єкт. У полі `NAME` (`ІМ'Я`, `НАЗВА`) знаходиться короткий опис об'єкта – таке, щоб його призначення було зрозуміло з першого погляду. У полі `SYNOPSIS` (`ОГЛЯД`, `СИНТАКСИС`) дається формалізований опис способів використання об'єкта. У квадратних дужках у цьому полі наведені необов'язкові параметри команди, які можна їй передати, а можна й опустити. Текст у полі `DESCRIPTION` (`ОПИС`) – це розгорнутий опис об'єкта, достатній для того, щоб ним скористатися. У полі `SEE ALSO` (`ДИВИСЯ ТАКОЖ`) знаходиться список інших джерел за тією самою темою.

Усі сторінки керівництва ОС Linux згруповані у вісім розділів, які за замовчуванням переглядаються в такому порядку:

- (1) – Уведення в користувальницькі команди;
- (8) – Уведення в адміністративні й привілейовані команди;
- (2) – Уведення в системні виклики;
- (3) – Уведення в бібліотечні функції;
- (4) – Уведення в спеціальні файли (пристрої);
- (5) – Уведення у формати файлів;
- (6) – Уведення до ігор;
- (7) – Уведення про угоди й про різний.

Команда `man` переглядає всі розділи в зазначеному порядку і вказує на перше знайдене керівництво із заданим ім'ям. Щоб подивитися посібник для об'єкта з певного розділу, необхідно як перший параметр команди `man` указати номер цього розділу, наприклад, `man 1 man`. Якщо як перший параметр `man` використати ключ «`-a`», то будуть послідовно видані усі керівництва із заданим ім'ям. Наприклад, за командою «`man -a intro`» будуть видані введення (introductions) за кожною з восьми розділів довідкової системи. Усередині сторінок керівництва прийнято безпосередньо після імені об'єкта ставити в круглих дужках номер розділу, з якого виданий посібник із цього об'єкта. Наприклад, `man(1)` – довідкова інформація з команди `man` видана з першого розділу.

Команда info є деякою альтернативою команди man. Документ info – це дійсний гіпертекст, у якому безліч невеликих сторінок об'єднані в деревоподібну структуру. У кожному розділі документа `info` завжди є зміст, з якого можна перейти відразу до потрібного підрозділу, звідки завжди можна повернутися назад. Крім того, `info`-документ можна читати і як безперервний текст – тому в кожному підрозділі є посилання на попередній і наступний підрозділи. Команда `info` використовує весь екран: на більшій його частині вона показує текст документа, а перший та два останніх рядки відведені для орієнтації в його структурі.

Одна або кілька сторінок, які можна перегортати клавішею **`Пробіл`** (**`Space`**) або **`PageUp`**/**`PageDn`** – це вузол (node). Вузол містить звичайний текст і меню (menu) – список посилань на інші вузли, що лежать у дереві на більш низькому рівні. Посилання всередині документа мають

вигляд «`* ім'я_вузла::`», переміщати по них курсор можна клавішею **`Tab`**, а переходити до перегляду обраного вузла – клавішею **`Enter`**. Повернутися до попереднього переглянутого вузла можна клавішею «**`l`**» (від англ. «last» – «востаннє, далі»). Вийти із програми `info` можна, натиснувши клавішу «**`q`**». Докладнішу довідку про керування програмою `info` можна в будь-який момент одержати з `info` – нажавши клавішу «**`?`**» або «**`h`**». Вузли, що становлять документ `info`, можна переглядати та підряд, один за іншим за допомогою клавіш «**`n`**» (від англ. «next» – «потім, потім, далі») і «**`p`**» (від англ. «previous» – «попередній»), однак у цьому зазвичай немає потреби.

У верхньому рядку екрана `info` показує ім'я поточного вузла, ім'я наступного вузла й ім'я батьківського (або верхнього) вузла, у якому перебуває посилання на поточний. Ім'я поточного вузла «`Top`» та ім'я верхнього вузла «`dir`» означають, що проглядається кореневий вузол документа, вище якого – тільки каталог зі списком всіх `info`-дерев. У нижній частині екрана розташований рядок з інформацією про поточний вузол, а за ним – рядок для введення довгих команд (наприклад, для пошуку тексту за допомогою команди «`/`»).

Якщо деякий об'єкт системи не має документації ні у форматі `man`, ні у форматі `info`, необхідно звернутися до каталогів `/usr/share/doc/<ім'я_об'єкта>`. У них утримується обширна документація по системі Linux загалом, і окремим аспектам її застосування. Більша частина цієї документації являє собою звичайні текстові ASCII-файли, які не мають, на жаль, ні стандартного формату, ні більше того – посилань на посібники з інших об'єктів системи. Для їхнього перегляду можна скористатися командами `more` та `less`.

2 Інтерфейс командного рядка

Спілкування користувача з UNIX-подібною операційною системою (Linux) у текстовому режимі відбувається через так звану «*командну оболонку*» («*оболонку командного рядка*», «*оболонку*» або, просто, «*shell*»). У термінах інших операційних систем цей компонент називається ще «*інтерпретатором команд*» або «*командним процесором*». Але в Linux (а також інших UNIX-подібних ОС) назва «*оболонка*» дуже точно відбиває сутність проблеми – користувач працює не з ОС безпосередньо,

а з ОС через спеціальну програму-оболонку – shell. Тобто усе, що вводить користувач на віртуальному терміналі, спочатку обробляється оболонкою. Далі, вона викликає на виконання необхідні програми, а результати їхньої роботи повертає користувачеві на віртуальний термінал.

Ім'я програмного файлу однієї з перших оболонок операційної системи UNIX було «`sh`» (скорочення від shell). Потім були розроблені трохи її поліпшених варіантів. Зокрема, *Bourne shell* – розширена версія `sh`, написана Стівом Борномом (Steve Bourne). У межах проекту GNU Браеном Фоксом (Brain Fox – головний розроблювач) і Четом Ремі (Chet Ramey) була розроблена оболонка `bash` (від англ. «Bourne again shell» – «знову оболонка Борна» або «заново породжена shell»).

На сьогоднішній день розмаїтість назв командних оболонок у світі UNIX може спричинити збентеження у будь-кого. Але до цього потрібно ставитися як до результату еволюційного розвитку. Насправді існує тільки дві головні групи (два головних типи) оболонок:

- *Bourne shell*;
- *C shell*.

Найновіша з командних оболонок, сумісна з оболонкою Борна – це *оболонка Z* (`zsh`). Оболонки групи C shell (`csh` і `tcsh`) використовують синтаксис, що нагадує синтаксис, мови програмування Сі.

Вибір типу оболонки – це майже як вибір релігії. Хтось віддає перевагу синтаксису shell Борна, а хтось –структурірованішому синтаксису C shell. Однак, із погляду рядового користувача, розходження між всіма цими оболонками практично відсутні. Такі команди, наприклад, як «`cp`» і «`ls`», працюють однаково у всіх типах оболонок. Розходження починають проявлятися тільки під час написання командних файлів або використання деяких нових властивостей оболонок. Однак, якщо є така можливість, спробуйте попрацювати з декількома оболонками, перш ніж зупинитися на одній з них. Це буде корисно у разі роботи з системою, де вибір оболонок буде обмежений.

Найрозповсюдженішою оболонкою, на сьогодні, у більшості дистрибутивів Linux є `bash`. А терміном shell позначають будь-яку командну оболонку, у будь-який UNIX-подібної ОС.

Дізнатися яка оболонка запущена, можна за допомогою команди «`echo $SHELL`». Команда «`echo`» відображає значення змінної оточення

SHELL, у якій записане повне ім'я файлу командної оболонки. Змінні оточення служать для завдання параметрів роботи операційної системи. В ОС Linux таких змінних оточення досить багато. Їхні імена зазвичай схожі на прописні букви. А подивитися значення такий змінної можна за допомогою команди «echo» (важливо не пропускати знак долара «$» перед ім'ям змінної). Команда «set» без параметрів відображає поточні значення всіх змінних оточення shell.

Керування роботою Linux у текстовому режимі здійснюється за допомогою команд. Тобто користувач уводить команду, ОС її виконує, повертає результат і далі – чекає введення наступної команди (найчастіше, видаючи символ запрошення «$»). Уведення кожної команди завершується шляхом натискання клавіші **Enter**. Формально, синтаксис команди Linux виглядає в такий спосіб:

```
<ім'я команди> [<ключі>] [<параметри>].
```

Тобто, кожна команда обов'язково має *ім'я*, наприклад, «ls» – вивести вміст каталогу. Вона, також, може мати (а може і не мати) *параметри*, які уточнюють спосіб її виконання. Наприклад, у команді «ls», ім'я каталогу, вміст якого необхідно відобразити. Команда може містити ще й *ключі* (*опції*, *прапори*), які точніше визначають спосіб її виконання. Наприклад, ключ «-l» команди ls викликає виведення інформації про зміст каталогу в розширеному (довгому) форматі. Ключі в команді обов'язково починаються символом «тире» («-»). Однобуквені ключі можна поєднувати. Компоненти команди (ім'я, ключі та параметри) одне від одного відокремлюються принаймні одним пробілом.

В ОС Linux є безліч різноманітних команд. Можна також додавати нові команди або змінювати вже наявні. Це в корені відрізняє UNIX-подібні ОС від більшості інших операційних систем, які містять строго обмежений набір команд.

Всі команди, які «розуміє» Linux, поділяються на дві групи:

- внутрішні;
- зовнішні.

Внутрішні команди убудовані («вшиті», «зашиті», «запрограмовані») безпосередньо в оболонку. Це, наприклад, такі команди як «cd», «echo»

або «exit». Зовнішні команди – *утиліти* – це просто файли. Наприклад, команда «ls» – виконуваний файл, розташований у каталозі /bin. Тому замість команди «ls» можна ввести повне ім'я файлу, що виконуватиметься – «/bin/ls».

Список вбудованих команд shell можна подивитися за командою «help». Докладну інформацію з певної вбудованої команди видає та сама команда «help» із вказівкою як параметр імені вбудованої команди, наприклад: «help cd».

Тип будь-якої команди shell можна визначити за допомогою команди «type». Убудовані команди називаються «builtin», а для зовнішніх команд виводиться повне ім'я виконуваного файлу. Наприклад команди «type man» і «type echo» видадуть, відповідно, таке:

```
man is /usr/bin/man
echo is a shell builtin.
```

Деякі – найпотрібніші – команди можуть бути і убудованими, і у вигляді окремого файлу. Наприклад «echo». Працює убудована команда так само, як і зовнішня, але оскільки часу на її виконання йде істотно менше, командний інтерпретатор вибере саме її, якщо буде така можливість. Ключ «-a» змушує «type» вивести всі можливі варіанти інтерпретації команди, а ключ «-t» – вивести тип команди замість повного імені. Наприклад, команди «type -a echo» і «type -at echo» видадуть, відповідно, таке:

```
echo is a shell builtin
echo is /bin/echo
```

та

```
builtin
file.
```

Саме набір зовнішніх команд можна розширювати та змінювати. Унаслідок цього Linux є дуже потужною й гнучкою системою. Щоб додати нову команду, системний адміністратор може просто встановити її в стандартний каталог, де розміщаються програми. Список каталогів, де

оболонка відшукує команди, називається «*шлях пошуку*». Він зберігається в змінній оточення `PATH`. Подивитися її поточне значення можна за командою «`echo $PATH`». Результатом роботи команди буде послідовність шляхів, розділених символом двокрапки («`:`»), наприклад:

```
/home/student21/bin:/bin:/usr/bin:/usr/local/bin:/
usr/X11R6/bin:/usr/games.
```

Перший шлях – `/home/student21/bin`, другий – `/bin`, тощо. Отже, якщо існують дві команди з однаковими іменами, одна з яких перебуває в каталозі `/home/student21/bin`, а інша в каталозі `/bin`, то виконається та, яка перебуває в каталозі `/home/student21/bin`.

Більшість сучасних командних оболонок (за винятком найранніших `sh` та `csh`) допускають редагування вводимої інформації (командного рядка). Можливості редактора командного рядка специфічні для різних командних оболонок. Однак найбільше необхідні команди редагування підтримуються у всіх різновидах shell у подібний спосіб. Наприклад:

- **`Backspace`** – видалити символ ліворуч від курсору;
- **`Delete`** – видалити символ у позиції від курсору;
- **`Ctrl+U`** – видалити підстрочу від курсору до початку рядка;
- **`Ctrl+K`** – видалити підстрочу від курсору до кінця рядка;
- < «стрілка вліво» – перемістити курсор на одну позицію вліво;
- > «стрілка вправо» – перемістити курсор на одну позицію вправо;
- **`Home`** або **`Ctrl+A`** – перемістити курсор у початок командного рядка;
- **`End`** або **`Ctrl+E`** – перемістити курсор у кінець командного рядка;
- **`Alt+B`** – перемістити курсор на одне слово вліво;
- **`Alt+F`** – перемістити курсор на одне слово вправо;
- **`Ctrl+W`** – видалити слово ліворуч від курсору;
- **`Alt+T`** – поміняти місцями символ у позиції курсору й ліворуч від нього.

Натискання клавіш **`Ctrl+C`** спричиняє негайне припинення виконання команди, а **`Ctrl+Z`** – тимчасово зупинити. Комбінація **`Ctrl+S`**

зупиняє вивід на екран, поки він не буде продовжений за допомогою **Ctrl+Q**.

Змінити стандартні призначення комбінацій клавіш можна командою «stty», використовуючи такий синтаксис:

```
stty <функція> <сполучення_клавіш>.
```

Наприклад, команда «stty erase ^H» призначає видалення символу ліворуч від курсору сполученню клавіш **Ctrl+H** – те, що зазвичай виконує клавіша **Backspace**. Діакритичний знак «кришечка» («^») перед клавішею «**H**» позначає клавішу **Ctrl** – це загальноприйнята нотація у всіх UNIX-системах.

Список поточних установок термінала можна одержати, ввівши команду «stty -a». Але розібратися в результатах її роботи не просто: «stty» – складна команда, що має численні застосування, і для деяких з них потрібні великі пізнання. Докладнішу інформацію з команди «stty» можна одержати, виконавши команду «man stty».

Якщо за якою небудь причиною настроювання текстового віртуального термінала зіб'ються (таке може відбутися, наприклад, при спробі переглянути вміст двійкового (виконуваного) файлу за допомогою команди «cat») і він перебуває в деякому незрозумілому стані – не реагує на **Enter**, не показує введення, не видаляє символи, виводить текст «сходами» тощо, то «лікувати» його рекомендується за допомогою таких команд:

- «stty sane» – спеціальна форма «stty», що скидає настроювання термінала в придатний для роботи стан;
- «setterm -reset» – поверне настроювання термінала у звичайний стан;
- «tput reset», «reset» або «tset» – ініціалізації термінала.

Усі сучасні командні оболонки, у більшому або меншому ступені, уміють добудовувати (англ. «completion») імена файлів і команд. Для цього необхідно набрати трохи перших символів імені й натиснути клавішу **Tab**. Наприклад, необхідно ввести такі символи «cd /usr/inc» та, не натискаючи клавішу **Enter**, натисніть клавішу **Tab**.

Оболонка сама додасть «lude/», доповнивши рядок до повного імені каталогу «/usr/include/». Тепер можна натиснути **Enter**, і команда буде виконана.

Критерієм визначення імені файлу є мінімальна повнота. Необхідно ввести символи в кількості, достатній, щоб за ними можна було відрізнити необхідне ім'я файлу або каталогу від усіх інших у цьому каталозі. Оболонка зможе відшукати це ім'я та доповнити його – аж до символу слеш, якщо знайдене ім'я є ім'ям каталогу. Функція автодоповнення може використовуватися не тільки з іменами файлів і каталогів, але й із командами – якщо це слово стоїть на початку командного рядка.

Виконуючи добудовування, shell може вивести не весь рядок, а тільки ту його частину, до якої немає сумнівів. Якщо подальше добудовування може піти декількома шляхами, то однократне натискання **Tab** ні до чого не призведе, а повторне – до виводу над командним рядком списку всіх можливих варіантів. У цьому разі необхідно підказати командній оболонці продовження: дописати декілька символів, що визначають, по якому шляху піде добудовування, і знову натиснути **Tab**. Якщо двічі натиснути клавішу **Tab** у порожньому командному рядку, то буде видано приблизно таке повідомлення: «Display all 2148 possibilities? (y or n)» – «Показати всі 2148 можливості? (y або n)». Аналогічним чином можна намагатися «угадувати» закінчення команд, якщо замість клавіші **Tab** декілька разів натискати клавішу **Esc**.

Усі команди, які вводив користувач, зберігаються в так називаній «*історії команд*». Найпростіше «витягти» їх відтіля можна за допомогою стрілок керування курсором:

- «стрілка нагору» (↑) або **Ctrl+P** (від англ. «previous» – «попередній») «прокручують» історію команд від останньої до першого;
- «стрілка вниз» (↓) або **Ctrl+N** (від англ. «next» – «потім») – у зворотному напрямі.

Відповідна команда відображається в командному рядку як тільки що набрана. Її можна відредагувати або виконати безпосередньо, натиснувши клавішу **Enter**. Максимальна кількість команд, що може зберігатися в історії, містить змінна оточення HISTSIZE.

Якщо необхідно знайти в історії якусь певну команду, простіше не «перегортати» список історії стрілками, а пошукати в ній за допомогою сполучення клавіш **Ctrl+R** (від англ. «reverse search» – «зворотний пошук»). Виводиться підказка спеціального виду – «(reverse-i-search)`': ». Натискаючи символи для пошуку, знаходимо першу команду, у якій є присутнім введена підстрока пошуку. Далі, за допомогою **Ctrl+R**, відшукуються всі інші команди із заданою підстрокой.

Щоб історія команд могла зберігатися між сеансами роботи користувача, shell записує її у файл, ім'я якого записане в змінній оточення HISTFILE (звичайно, це .bash_history, що перебуває в домашньому каталозі користувача). Робиться це в момент завершення сеансу роботи користувача – накопичена за час роботи історія дописується в кінець цього файлу. Під час наступного запуску shell цей файл зчитує цілком. Історія зберігається не вічно – кількість команд, що запам'ятовують, обмежене – звичайно, близько 10000 команд. Точне значення цієї величини зберігається в змінній оточення HISTFILESIZE.

Для перегляду всієї історії команд слугує команда «history». Вона виводить інформацію у дві колонки: номер команди й саму команду. «Погортати» декілька останніх сторінок виведення команди «history» можна за допомогою сполучень клавіш **Shift+PgUp** й **Shift+PgDown**.

Переважна більшість команд Linux, за замовчуванням, дані вводять із клавіатури, а результат відображають на екрані монітора. Клавіатура є «*стандартним пристроєм введення*» (stdin), а екран віртуального термінала – «*стандартним пристроєм виведення*» (stdout). Однак, оскільки Linux – система гнучка, ці настроювання за замовчуванням можна динамічно змінювати у процесі виконання команд.

Наприклад, shell дає змогу вільно перенаправляти виведення будь-якої команди з термінала у файл. Для цього необхідно після команди додати знак «більше чим» («>»), а слідом за ним указати ім'я файлу, куди буде послане виведення команди. Наприклад, щоб перехопити й зберегти результати роботи команди «ls», можна ввести:

```
ls /usr/bin > ~/dir.txt
```

Перелік файлів, що перебувають у каталозі `/usr/bin`, буде записаний у файл `dir.txt` у домашньому каталозі користувача. Якщо файл `dir.txt` уже існує, то операція перенаправлення виведення («>») затре його старий вміст. Перезапис наявного файлу є частою помилкою користувачів. Тому, щоб продовжити записувати (дописувати) дані в уже наявний файл, в shell є інша операція перенаправлення виведення, а саме – два знаки «більше чим» («>>»). Наприклад, команда

```
date >> dir.txt
```

додасть у кінець уже існуючого файлу `dir.txt` дані зі стандартного виведення чергової команди `date` – поточну дату – буде отриманий саме той результат, що був потрібний. Прийом перенаправлення виведення виявляється дуже зручним, коли потрібно неодноразово виконати ту саму програму та зберігати результати її роботи, наприклад, для аналізу помилок.

Однак нерідко виникають ситуації, коли необхідно, щоб одна програма обробляла дані, виведені іншою програмою. Користуючись перенаправленням введення-виведення, можна зберегти вивід однієї програми у файлі, а потім направити цей файл на введення іншій програмі. Але те саме можна зробити й ефективніше – перенаправляти виведення можна не тільки у файл, але й безпосередньо на вхід іншій програмі. У цьому разі замість двох команд буде потрібна тільки одна – програми передають один одному дані «із рук у руки». У Linux такий спосіб передавання даних називається «*програмним каналом*» або «*конвеєром*». В англомовних, оригінальних, текстах терміну «*конвеєр*» відповідають терміни «*pipe*» – «труба» або, більш точніше, «pipeline» – «трубопровід».

У shell для перенаправлення виведення однієї програми на вхід іншій програмі слугує символ «`|`». Найпростіший і найпоширеніший випадок, коли потрібно використати конвеєр, виникає, якщо вивід програми не вміщається на екрані монітора й дуже швидко «пролітає» перед очами, от же користувач не встигає його прочитати. У цьому разі можна направити вивід у програму перегляду, наприклад, «`less`», що дасть змогу не кваплячись, собі на втіху, «полистати» весь текст, повернутися до початку тощо:

```
ls -l /usr/bin | less
```

Можна послідовно обробити дані декількома різними програмами, перенаправляючи вивід однієї на вхід наступній програмі й організувати у такий спосіб як завгодно довгий конвеєр для одержання необхідного результату. Тобто програмні канали використаються для того, щоб скомбінувати декілька маленьких програм, кожна з яких виконує тільки певні перетворення над своїм вхідним потоком, для створення узагальненої команди, результатом якої буде якесь складніше перетворення.

Необхідно зважити на те, що перенаправлення та конвеєри, це засоби, надавані оболонкою shell, це синтаксис shell, і символи «>», «>>» і «|» ніяк не стосуються до команд.

Крім перенаправлення виводу команди у файл, або на вхід іншої команди, у shell є ще один механізм передвання інформації – вивід будь-якої команди може бути поміщений у командний рядок. Для цього така команда повинна бути вкладена в «одинарні зворотні лапки» – «`» (які знаходяться на одній клавіші із символом «тильда» – «~»). Наприклад, команда «`echo Файлів: `ls -l /usr/bin | wc -l`` відображає слово «`Файлів: `», за яким буде число, на одиницю більше кількості файлів у каталозі `/usr/bin` (рядок «разом»). Тобто вивід команди, записаної в одинарні зворотні лапки – «`ls -l /usr/bin | wc -l`» – був поміщений у командний рядок – доповнив параметр, і був виведений – результат роботи команди «`echo`».

Однак вводити такі довгі команди, та ще й не помиляючись, дуже важко. А якщо робити це доводиться багаторазово – те не тільки важко, але й утомливо. Для цього в shell із довгих, які часто повторюються, «старих» команд створюють «нові», коротші, команди – їх просто записують у спеціальні файли – *сценарії*.

Наявне велике сімейство програм, мета роботи яких полягає в тому, щоб сприймати вхідний потік даних, робити над ним деякі перетворення та видавати результат на стандартний вивід (звідки його можна перенаправляти кудись ще за бажанням користувача). У Linux такі програми називаються *фільтрами*: дані проходять через них, до того ж щось «застряє» у фільтрі й не з'являється на виході, щось змінюється, а щось проходить крізь фільтр незмінним. Фільтри в Linux зазвичай читають дані зі стандартного входу, а результат виводять на стандартний вивід.

З поміж найчастіше використовуваних програм-фільтрів викреслюють, наприклад, такі команди, як: «cat», «ls», «cp», «more», «less», «wc», «cmp», «diff», «sort» тощо. Найпростішим фільтром, безумовно, є команда «echo». Вона призначена для виведення на стандартний пристрій виведення рядка символів, що заданий їй як аргумент. Після цього вона видає сигнал переведення рядка та завершується. Наприклад, команда «echo Здраствуй, мир!» як результат своєї роботи поверне рядок символів «Здраствуй, мир!».

Наступним прикладом програми-фільтра є команда «sort», що сортує вхідні дані й посилає результат на стандартний вихід. Крім цієї очевидної дії, виконуваної за замовчуванням, є безліч інших способів сортування, для кожного з яких «sort» має відповідний ключ. Наприклад, ключ «–f» викликає сортування не чутливим до регістра символів. Зазвичай сортування виконується за символами, але іноді потрібно сортувати числа. Ключ «–n» забезпечує сортування за числовим значенням, а ключ «–r» змінює порядок сортування на протилежний. Отже, команди сортують уміст каталогу /usr/bin:

```
ls /usr/bin | sort –f          – за алфавітом;
ls –s /usr/bin | sort –n       – за збільшенням розміру;
ls –s /usr/bin | sort –nr      – за зменшенням розміру.
```

За замовчуванням «sort» порівнює рядки цілком, але їй можна вказати, по якому полю повинне виконуватися сортування. Для цього призначений ключ «-k». Запис «-kn» означає, що сортування буде виконуватися по полю номер n. Наприклад, команда

```
ls -la /usr/bin | sort -k5nr
```

виконує сортування вмісту каталогу /usr/bin за зменшенням розміру файлу.

Ще одна, часто використовувана програма-фільтр – це команда «wc» (від англ. «word count» – «лічильник слів»), що підраховує кількість рядків, слів і байт. Для підрахунку тільки рядків необхідно вказати ключ «-l»,

слів – «`-w`», а байт – «`-c`». Наприклад, команда «wc –w Фрукти.txt» видасть число – кількість слів, що містить файл Фрукти.txt.

Особливим фільтром є команда «`tee`» («мішень»), що «роздвоює» вхідний потік: із одного боку – направляючи його на стандартний вивід, а з іншого боку – у файл, ім'я якого задано їй як параметр.

Дані, що проходять через фільтр, являють собою текст: у стандартних потоках вводу-виводу всі дані передаються у вигляді символів, рядок за рядком, як і у терміналі. Тому за допомогою конвеєра можуть бути зістиковані будь-які дві програми, що підтримують стандартні потоки вводу-виводу. Це, у якомусь ступені, нагадує стандартний дитячий конструктор, де всі деталі можуть зістиковуватися між собою.

У будь-якому дистрибутиві Linux обов'язково є присутнім набір стандартних утиліт, призначених для роботи з файловою системою та обробки тексту. Кожна із цих утиліт призначена для виконання якоїсь однієї операції над файлами або текстом: виведення списку файлів у каталозі, копіювання, сортування рядків тощо. Кожна утиліта може виконувати свою функцію трохи по-різному, залежно від переданих їй ключів і параметрів. Усі утиліти працюють зі стандартними потоками вводу-виводу, тому добре пристосовані для побудови конвеєрів: послідовно виконуючи прості операції над потоком даних, можна вирішувати досить нетривіальні завдання.

Принцип комбінування елементарних операцій для виконання складних завдань успадкований Linux-ом від операційної системи UNIX (як, втім, і багато інших принципів). Переважна більшість утиліт UNIX, не втратили свого значення і в Linux. Усі вони орієнтовані на роботу з даними в текстовій формі, багато хто є фільтрами, усі не мають графічного інтерфейсу та викликаються з командного рядка. Цей пакет утиліт називаються *coreutils – головні утиліти*.

Командна оболонка shell може використатися не тільки для виконання окремих команд, але також і для автоматичного виконання деякої серії таких команд за допомогою «*сценарію*». Тобто, якщо якусь послідовність команд доводиться виконувати досить часто, те її можна перетворити в «нову» команду, що має власне ім'я для того, щоб її можна було запускати надалі як звичайну команду. Зібрані в одному файлі такі серії команд в англомовній літературі останнім часом переважно

називаються *script – сценарій*, хоча те, що під цим розуміється, у багатьох книгах на російській традиційно називається як «*програма на shell*», «*командний файл оболонки*» або, просто, «*командний файл*».

Сценарій – це звичайний текстовий файл, призначений для обробки якою-небудь утилітою. Зазвичай така утиліта – це інтерпретатор деякої мови програмування, а вміст такого файлу – програма на цій мові. Якщо системі спеціально не «натякнути», то як інтерпретатор вона запускає стандартний shell – `/bin/sh`.

Наприклад, якщо створити текстовий файл `Hello.script` і записати в нього рядок «`echo Здраствуй, Мир!`», а потім запустити його на виконання по команді «`sh Hello.script`» – буде видане повідомлення «`Здраствуй, Мир!`». Для того, щоб щораз не доводилося вказувати програму-інтерпретатор (`sh` або `bash`) і передавати їй сценарій у вигляді параметра, а просто запускати скрипт на виконання по імені файлу (наприклад, `Hello.script`), не залежно від того, у якому каталозі працює користувач, необхідно:

1) за допомогою команди «`chmod`» установити параметр доступу «виконуваний», оскільки за замовчуванням текстові файли створюються доступним на запис і читання, але не на виконання (наприклад, «`chmod +x Hello.script`»);

2) помістити його в один з каталогів пошуку команд оболонки shell, які перераховуються в змінної оточення `PATH`. Якщо скрипт призначений не для загального застосування, а для особистого, то його краще помістити в особистий каталог програм (команд), що, зрозуміло, перебуває в домашньому каталозі користувача й називається `bin`. Такий каталог найкраще записати першим у списку змінної оточення `PATH`, оскільки, якщо ім'я скрипта збіжиться з ім'ям однієї з утиліт в інших каталогах пошуку, то виконуватися буде саме він, а не навпаки – команда з іншого каталогу.

«Правильний» скрипт командної оболонки shell у першому рядку повинен містити ім'я програми-інтерпретатора, що буде його виконувати. Для цього першими двома байтами сценарію повинні бути символи «`#!`». Тоді весь його перший рядок, починаючи із третього байта, система сприймає як команду обробки сценарію. Виконання такого сценарію приведе до запуску зазначеної після «`#!`» команди, останнім параметром

якої буде ім’я самого файлу сценарію. Однак, якщо між символами «#» та «!» виявиться пробіл, тоді ця директива не спрацює, оскільки буде сприйнятий як звичайний коментар. Тоді зміст «правильного» сценарію `Hello.script` повинен бути таким:

```
#!/bin/sh
# Сценарій shell, що виводить повідомлення
echo Здраствуй, Мир!
```

У ньому перший рядок «`#!/bin/sh`» свідчить про те, що цей файл – сценарій, і повідомляє shell, як його виконати. У цьому разі необхідно передати сценарій на виконання команді `/bin/sh`, тобто самій програмі shell. Чому це важливо? У більшості систем UNIX `/bin/sh/` – це shell Баурновського типу, наприклад `bash`. Гарантується, що сценарій буде виконуватися саме під shell Баурновського типу, а не, скажімо, під C shell. Тобто цей сценарій буде виконуватися під shell Баурна, навіть у тому разі, коли поточним інтерпретатором команди `tcsh` (або який-небудь інший C shell). Усі ці командні оболонки одна від одної відрізняються тільки синтаксисом мови, на якій пишуться сценарії.

Другий рядок – це коментар. Коментарі в скриптах починаються символом «ґрати» («#») і тривають до кінця рядка. Тобто усе, що перебуває в такому рядку після символу «ґрати» і до кінця рядка, shell буде вважати коментарем та ігнорувати – цей текст не розглядається як команда, а використаються тільки для пояснень.

Інші рядки сценарію – звичайні команди в тому вигляді, у якому вони вводяться безпосередньо на виконання. Інтерпретатор читає кожен рядок сценарію та виконує цей рядок, начебто він введений у відповідь на запрошення.

Однак є можливість написати сценарій не тільки для shell, а взагалі, для будь-якої утиліти, у тому числі написаної самостійно. Наприклад, виконання такого сценарію:

```
#!/bin/sort
Яблука
Груші
Сливи
```

призведе до запуску на виконання зазначеної після «#!» команди /bin/sort і передавання їй як останній параметр самого сценарію. Тобто запуститься утиліта «sort», а сценарій (файл із невідсортованими рядками) передасться їй як параметр. Зрозуміло, оформляти файли, які необхідно відсортувати, у вигляді sort-сценаріїв нераціонально. Однак, цей приклад продемонстрував, що сценарій можна написати для всього, чого завгодно. Для цього необхідно тільки замінити перший рядок посиланням на будь-яку програму, що буде читати файл, та виконувати відповідні команди. Наприклад, скрипти мовою Perl починаються з рядка вигляду: «#!/usr/bin/perl».

Найповнішу інформацію стосовно роботи з командною оболонкою bash можна отримати за командою «man bash».

3 Графічний інтерфейс користувача

Графічний інтерфейс користувача (GUI) являє собою різновид користувальницького інтерфейсу, у якому реалізована метафора *оточення робочого столу* (англ. desktop environment). Робочий стіл у сучасних операційних системах – це екран, що бачить користувач відразу ж після завантаження операційної системи. У сучасній комп'ютерній термінології він є частиною графічного середовища користувача, що, зі свого боку, призначена полегшити взаємодію з комп'ютером, створивши інтуїтивно зрозумілий інтерфейс, яким можна керувати як за допомогою клавіатури, так і миші. «Інтуїтивно зрозумілий інтерфейс» досягається тим, що людина на екрані бачить графічні зображення (кнопки, папки, документи, вікна тощо), якими він може управляти подібно до того, як він робив би це з реальними об'єктами.

Графічний інтерфейс користувача в ОС Linux будується на підставі стандарту *X Window System* (зверніть увагу на те, що Window, а не Windows – в однині) або просто «*X*» (у просторіччі – «*ікси*»). Найперший варіант цього стандарту був розроблений у 1987 році в Массачусетському технологічному інституті. Починаючи із другої версії, стандарт підтримується консорціумом X, який був створений у 1988 р. з метою уніфікації графічного інтерфейсу для ОС UNUX. З 1997 р. Консорціум X перетворений у X Open Group. На сьогодні діє версія 11 випуск 6 стандарту на графічну підсистему для UNIX-подібних ОС, що коротко позначається як *X11R6*.

X Window System – це «багатошарова» система, на «верху» якої перебуває *графічне оточення робочого столу* (англ. *desktop environment*), що, власно, і є сполучною ланкою між кінцевим користувачем та операційною системою. Якщо в найпоширенішій операційній системі Microsoft Windows усього одна графічна оболонка, тобто користувач може міняти графічну тему, настроювання деяких графічних елементів (наприклад змінити іконку папки), але, як би він не намагався, сам графічний інтерфейс залишиться тим самим. В ОС Linux ситуація кардинально протилежна – таких графічних оболонок багато. Користувач має можливість установити їх всі, а входячи в систему вибрати ту, яка в цей момент йому більше підходить.

В ОС Linux найпоширеніші графічні середовища це такі:

- *KDE* (K Desktop Environment або українською, просто, – оточення робочого столу);
- *GNOME* (GNU Network Object Model Environment або українсько–мережне середовище об'єктної моделі GNU).

Вибираючи графічне середовище, користувач вибирає набір програм, з якими він будете працювати. Середовище KDE використовує для роботи бібліотеку Qt (The Q Toolkit), а GNOME – інструментарій GTK+ (The GIMP Toolkit). Із цього витікає, що якщо користувач вибере KDE, те будуть установлені програми, що працюють на бібліотеці Qt, якщо ж GNOME, то відповідно встановлюються програми, що базуються на GTK+. Наприклад, як файловий менеджер у разі вибору KDE буде встановлений *Dolphin*, а у разі вибору GNOME – *Nautilus*. Але запустити прикладну програму в невластивою їй середовищі не вийде.

KDE – це графічне середовище користувача, за своїм виглядом і логічним устроєм, максимально наближені до GUI Microsoft Windows. Але це тільки зовні. Головною властивістю KDE є гнучкість у настроюванні системи, що дає змогу конфігурувати додаток без редагування текстових файлів. Недоліки цього графічного середовища є наслідком його переваг – далеко не всім користувачам потрібна така гнучкість і, відповідно, наявність такої великої кількості опцій для настроювання. Для деяких із них важливішими є саме ті функціональні можливості, яким вони віддають перевагу.

Перші версії KDE були створені з використанням невільних інструментів – бібліотека Qt на той момент була пропрієтарним продуктом. Це стало причиною створення двох проектів:

- Harmony.
- GNOME.

Маючи однакові цілі (створення вільного середовища вільними засобами), два проекти вибрали зовсім різні шляхи реалізації задуманого. Проект Harmony ставив своїм завданням переписати бібліотеку Qt, випустивши її під вільною ліцензією, проект GNOME – повністю відмовився від використання Qt. Остання стабільна версія бібліотеки Qt 4 доступна під ліцензією GNU GPL для платформ UNIX, Mac OS, iOS, Android і Windows, що дозволяє KDE мати повну офіційну підтримку на всіх перерахованих платформах.

Розроблення GNOME почалося в серпні 1997 року, як спроба створити повністю вільне робоче середовище для операційної системи Linux. Вона була заснована на інструментарії GTK+, створеному раніше для графічного редактора GIMP і розповсюджуваному на умовах GNU GPL. У GNOME головний пріоритет одержали міркування практичності, простоти та зручності використання середовища, зокрема для недосвідчених користувачів. Ключовим моментом у GNOME є ідея про те, що кожне функціональне навантаження і кожна опція настроювання в програмі мають свою ціну: найчастіше краще вибрати один, оптимальний варіант поведінки програми, чим реалізовувати безліч варіантів і змушувати користувача вибирати один із них.

Xfce є графічною оболонкою, побудованою на основі інструментального пакету GTK+, використовуваного в GNOME, але набагато легшою та призначеною для тих, кому потрібний простий стіл, який ефективно працює, який легко використовувати та розширювати. Перший варіант Xfce був створений Олівером Форданом у 1998 році. Назва Xfce спочатку означало *XForms Common Environment*, але відтоді Xfce була переписана декілька разів і більше не використає цю технологію. (XForms – це технологія Web-форм, що базується на архітектурі *Model-View-Controller*, де дані представляються у вигляді XML.) Назва залишилася, але записується вже не як XFce, а як Xfce і ніяк не розшифровується.

З моменту появи й до сьогодні Xfce позиціюється як GUI, що «леша» ніж GNOME та KDE. Легша в тому розумінні, що їй для роботи потрібно менше оперативної пам'яті та слабший процесор. Тобто за задумом розроблювачів вона повинна була своєю швидкодією задовольнити користувачів, у яких слабкі комп'ютери. Останнім часом Лінус Торвальдс теж став використовувати Xfce, хоча й уважає, що це «великий крок назад у порівнянні з GNOME 2, але це величезний крок уперед порівняно з GNOME 3».

LXDE (Lightweight X11 Desktop Environment) – це наймолодша GUI. Її перший реліз був випущений у 2006 році тайванським програмістом Хун Жень Йі. Вона прагне запропонувати користувачеві швидкий і легкий робочий стіл, невимогливий до ресурсів комп'ютера і заснований на взаємно незалежних компонентах. LXDE часто позиціюється як графічне середовище, що швидше та «легше» ніж Xfce. Фактично в «легкості» різниці між ними практично немає, крім того за функціональністю вона значно уступає Xfce.

Unity є оболонкою робочого стола, яка розроблювальна компанією Canonical для операційної системи Ubuntu Linux, перший випуск якої був 3 червня 2010 року. Середовище робочого стола Unity є відгалуженням (англ. fork – відгалуження) від кодової бази GNOME 3. Вона дає змогу ефективніше використовувати маленькі екрани нетбуків з використанням, наприклад, вертикальної панелі для перемикання між запущеними програмами. Усі стандартні додатки Unity, як і раніше, беруться з GNOME.

MATE – це середовище робочого стола, що є відгалуженням від кодової бази більш непідтримуваного середовища GNOME 2. Назва MATE походить від іспанської назви (ісп. mate) зіновиду падуба, рослини, з листів якого виготовляють однойменний напій. Одна із цілей розроблення MATE – зберегти працездатність робочого стола на старому обладнанні, не орієнтуючись винятково на сучасні конфігурації. Уперше про початок розроблення MATE було оголошено 18 червня 2011 року на форумі Arch Linux користувачем Perberos, який і став засновником проекту. Пізніше до розроблення MATE приєдналася безліч розроблювачів і добровільних помічників.

Cinnamon (від англ. cinnamon – кориця) – це середовище робочого стола, що є відгалуженням від кодової бази GNOME. Головний напрям розроблення – надання користувачеві більш звичного середовища

в стилі GNOME 2, зручного користувачам настільних ПК і ноутбуків, без недоліків Unity. Спочатку Cinnamon розроблявся командою програмістів Linux Mint. Проект уперше був представлений публіці 2 січня 2012 року.

Контрольні питання

1. Що таке shell? Яке її призначення? Назвіть синоніми.
2. Назвіть головні групи Shell. Чим вони відрізняються?
3. Як визначити назву поточної shell?
4. Приведіть структуру команди ОС Linux.
5. На які групи поділятюься команди ОС Linux?
6. Як визначити тип команди?
7. Що робить команда `echo`? До якого типу вона належить?
8. Як подивитися каталоги пошуку утиліт ОС Linux?
9. Як відновити настроювання термінала, що збилися, у вихідний стан?
10. Як в ОС Linux виконується «добудовування» команд?
11. Які є варіанти «добудовування» команд, якщо з першої спроби не вийшло?
12. Що таке «історія команд»? Навіщо вона потрібна? Як нею користуватися?
13. Як подивитися максимальну кількість команд, що може зберігатися в «історії»?
14. Як відшукати потрібну команду в «історії», не пролистуючи її всю, якщо відомо тільки декілька її символів?
15. Як подивитися ім’я файлу «історії», у якому зберігаються команди між сеансами роботи? Де він зазвичай міститься?
16. Як подивитися максимальну кількість команд, що може зберігатися у файлі «історії»?
17. Що таке стандартні пристрої введення та виведення в ОС Linux? Як вони називаються?
18. Як перенаправляти виведення команди ОС Linux у файл?
19. Як дописати виведення команди ОС Linux у вже існуючий файл?
20. Як перенаправляти вивід однієї команди ОС Linux на вхід іншої? Приведіть приклад.

21. Для чого слугує символ «;» у командному рядку?

22. Як помістити вивід команди ОС Linux у командний рядок?

23. Що таке програми-фільтри? Навіщо вони потрібні та як вони використаються?

24. Приведіть приклади програм-фільтрів. Що вони роблять?

25. Як одночасно записати виведення команди у файл і вивести на термінал?

26. Що таке сценарії та навіщо вони потрібні? Назвіть синоніми.

27. Що необхідно зробити, щоб сценарій можна було запускати на виконання, увівши тільки його ім’я?

28. Назвіть типову ознаку «правильного» сценарію.

29. Як написати сценарій, що буде виконувати потрібна програма-утиліта?

30. Як одержати вичерпну інформацію з роботи shell?

6 ЗАСАДИ ВІРТУАЛІЗАЦІЇ ТА ВСТАНОВЛЕННЯ ОПЕРАЦІЙНИХ СИСТЕМ

1 Засади віртуалізації

Технологія віртуалізації призначена для підтримання можливості одночасного запуску на одному комп'ютері декількох (зокрема різних) операційних систем (далі – ОС). Реалізація цієї можливості здійснюється за допомогою спеціальних програм – віртуальних машин (далі – ВМ). Віртуальна машина – це повністю ізольований програмний контейнер, здатний виконувати операційну систему і додатки як фізичний комп'ютер. Операційна система, що виконується на віртуальній машині, називаються *гостьовою операційною системою* (англ. *guest OS*), а платформа, що підтримує віртуальну машину, – *хостом* (англ. *host*). *Операційна система хоста* (англ. *host OS*) – це операційна система фізичного комп'ютера, у якій виконується ВМ. Для успішного виконання віртуальної машини на хост-компьютері він завжди «ділиться» своїми **реальними ресурсами**, такими як пам'ять (оперативна, дискова та відео-) та процесор (виділяючи час роботи та кількість ядер) для ВМ.

Системи віртуальних машин використовуються:

1) як полігон для тестування нових програм та операційних систем – збої та віруси в них ні як не відбиваються на працездатності хоста;

2) як оболонка для серфінгу по Інтернету – у ролі гостьової ОС запускається Linux зі своїм браузером; при цьому ймовірність проникнення шкідливих програм на хост-компьютер зводиться до мінімуму;

3) для підвищення надійності, розподілу навантаження та масштабування серверів корпоративних інформаційних систем;

4) для тренування навичок і підвищення професійного рівня системних адміністраторів;

5) інших аналогічних цілей.

2 Система віртуалізації Oracle VM VirtualBox

Далі як система віртуалізації буде розглядатися Oracle VM VirtualBox (VirtualBox), а як гостьові ОС – Linux Mint та Android-x86, оскільки вони:

1) вільні (безкоштовні);
2) мають російський інтерфейс;
3) кроссплатформенні, тобто, можуть виконуватися під керуванням операційної системи хоста як Windows, так й Linux.

Система VirtualBox поширюється на підставі ліцензії *GNU GPL 2*, тобто *вона вільна*. Дистрибутив VirtualBox (для відповідної операційної системи хоста) можна скачати за адресою: https://www.virtualbox.org/wiki/Downloads/. Як гостьові ОС VirtualBox підтримує роботу практично всіх сучасних операційних систем, зокрема Microsoft Windows, Linux, FreeBSD, Mac OS X, FreeDOS тощо.

Перевагою VirtualBox є простий та інтуїтивно зрозумілий користувацький інтерфейс. Усі головні функції винесені у вигляді кнопок на панелі інструментів. Створення та настроювання віртуальних машин виконується за допомогою покрокового майстра. VirtualBox підтримує роботу з мережами, тому гостьова віртуальна ОС зможе легко вийти в Інтернет. Дуже корисною є функція «знімків» операційної системи. Віртуальна машина записує на диск «крапки відновлення», до яких у будь-який момент можна «відкотити» гостьову систему у разі виникнення помилок або збоїв. VirtualBox підтримує роботу віртуальних машин, створених в інших системах віртуалізації. Наприклад, формат файлу віртуальної машини *.VHD* (вільна Virtual PC від Microsoft) або *.VMDK* (проприетарна Workstation від VMware).

Інтерфейс системи, що виконується, VirtualBox з єдиною віртуальною машиною Linux Mint зображений на рисунку 6.1.

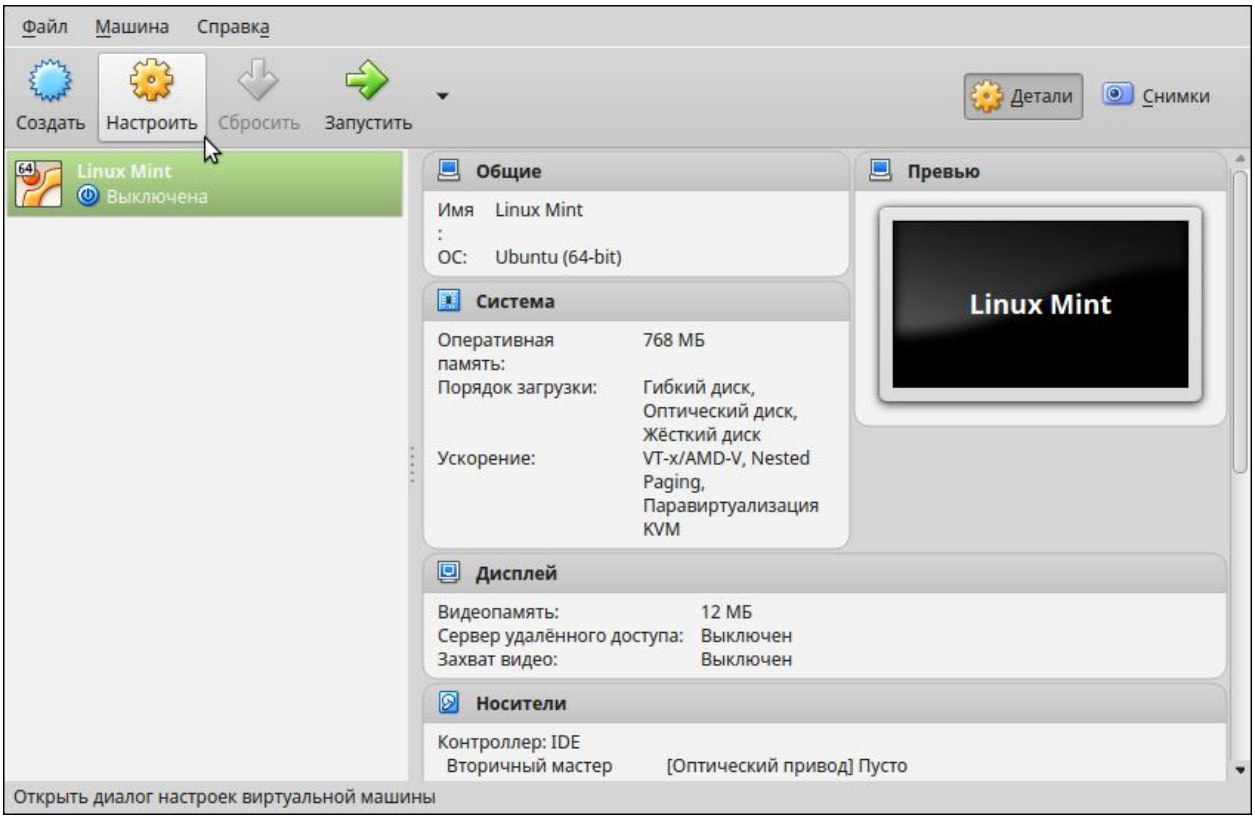


Рисунок 6.1 – Інтерфейс системи VirtualBox

У лівій панелі розташовується список всіх установлених віртуальних машин. Для кожної з них створюється свій віртуальний диск, на який установлюється своя операційна система. При першому запуску програми VirtualBox список віртуальних машин порожній.

Праворуч від списку віртуальних машин відображається інформація про виділену в лівому списку ВМ: найменування, обсяг оперативної пам'яті, порядок завантаження носіїв тощо. Параметри виділеної віртуальної машини можна змінити, клацнувши по кнопці **Настроить** на панелі вгорі. Натискання кнопки **Создать** викликає майстер створення нової віртуальної машини. Кнопка **Запустить** слугує для запуску виділеної віртуальної машина, а кнопка **Сбросить** – для її вимикання.

Головне меню VirtualBox містить три пункти: **Файл**, **Машина** та **Справка**. Пункт меню **Файл** слугує для роботи з віртуальними носіями, експорту/імпорту конфігурацією віртуальних машин, а так само настроювання конфігурації поточної віртуальної машини. Під час вибору пункту меню **Выход** закриється вікно VirtualBox, однак запущені віртуальні машини продовжать виконуватися. Пункт основного меню **Машина** призначений для роботи з віртуальними машинами: створювати,

додавати, змінювати, копіювати, видаляти, запускати, скидати, припиняти роботу. Пункт меню **Справка** дає змогу одержати вичерпну інформацію про програму VirtualBox, а так само обновити її до останньої версії.

3 Установлення гостьової операційної системи

Установлення гостьової операційної системи на віртуальний комп'ютер потребує виконання таких кроків:

1) визначення місця розташування дистрибутива гостьової ОС (привід CD/DVD, каталог файлової системи, мережний ресурс тощо);

2) створення віртуального комп'ютера;

3) настроювання віртуального комп'ютера;

4) установлення гостьової операційної системи на віртуальний комп'ютер.

Контрольні питання

1. Що таке віртуалізація?
2. Що таке віртуальна машина?
3. Що таке гостьова операційна система?
4. Дайте визначення операційній системі хоста.
5. Звідки беруться обчислювальні ресурси для гостьового комп'ютера?
6. Для чого використаються системи віртуальних машин?
7. На яких операційних системах хоста може виконуватися VirtualBox?
8. Які гостьові ОС підтримує VirtualBox?
9. Де можна взяти дистрибутив VirtualBox?
10. Назвіть головні переваги системи VirtualBox?
11. Чи підтримує VirtualBox віртуальні машини, створені в інших системах віртуалізації?
12. Перелічіть формати віртуальних жорстких дисків, які може створювати і підтримувати VirtualBox.
13. Перелічіть головні елементи вікна VirtualBox.
14. Як почати створення нової віртуальної машини?
15. Як змінити настроювання наявної ВМ?
16. Які є способи запуску на виконання (завантаження) віртуальної машини.

17. Перелічіть кроки для установлення гостьової ОС на віртуальний комп’ютер

18. Поясніть призначення віртуального жорсткого диска

19. Від чого залежить мінімальний обсяг віртуального жорсткого диска?

20. Назвіть формат рідного (нативного) жорсткого диска VirtualBox.

21. Як указати місце розташування дистрибутива встановлюваної гостьової ОС?

22. Як погодити параметри мережних настроювань операційної системи хоста та гостьової ОС?

23. Для чого необхідно погоджувати параметри мережних настроювань операційної системи хоста й гостьової ОС?

24. Що необхідно вказати в першому вікні діалогу майстра установлення гостьової ОС Linux Mint?

25. Для чого слугує прапорець «Установить сторонне программное обеспечение для …» у другому вікні діалогу майстра установлення гостьової ОС Linux Mint?

26. Куди установлюється гостьова операційна система Linux Mint?

27. Які параметри вказуються в останньому вікні діалогу майстра установлення гостьової ОС Linux Mint?

28. Як переконатися, що процес установлення гостьової ОС Linux Mint не «завис»

29. Як (або чим) завершується процес установлення гостьової ОС Linux Mint?

30. Що може бути причиною не правильної мови інтерфейсу гостьової ОС Linux Mint?

7 ЗАСАДИ АДМІНІСТРУВАННЯ, КОРИСТУВАЧІ ТА ГРУПИ

1 Механізм розмежування прав доступу

Операційна система Linux є багатокористувацькою багатозадачною ОС. Це означає, що в системі одночасно можуть працювати декілька користувачів, і кожен з них може запустити на виконання декілька додатків. Питання безконфліктного й легітимного розподілу ресурсів комп'ютера (час центрального процесора, оперативна та зовнішня пам'ять тощо) між програмами, що виконуються одночасно, є одним з істотних моментів надійного й безпечного функціонування операційної системи.

В основі механізму розмежування доступу до ресурсів комп'ютера лежать імена користувачів та імена груп користувачів. Кожному користувачеві ОС Linux ставиться у відповідність свій обліковий запис (англ. **account**), що однозначно характеризується ім'ям користувача (**username**) і його ідентифікатором (**uid**). Тобто як ім'я користувача, так і його ідентифікатор повинні бути унікальними в межах усієї системи.

2 Категорії користувачів і групи

В операційній системі Linux існують три категорії користувачів:

1. Єдиний користувач із ім'ям (**username**) **root** – суперкористувач (англ. **root** – корінь; читається як «рут»). Він має максимальні повноваження в системі. Система повністю підвладна цьому користувачеві. Будь-яка команда буде беззастережно виконана. Тому працювати під ім'ям **root** потрібно з обережністю.
2. Звичайні (реальні) користувачі.
3. Системні (фіктивні) користувачі. Вони, зазвичай, мають такі імена, як **daemon**, **bin**, **sys**, **mail**, **news** тощо. Під цими іменами не можна реєструватися в системі, але від їхнього імені можуть запускатися різні сервіси (програми). Властивістю облікових записів із такими іменами є поле **shell** у файлі **/etc/passwd** – воно буде визначено як **/sbin/nologin** або **/bin/false**.

Крім окремих користувачів, у системі реєструється деяке число груп користувачів. До того ж кожен користувач належить одній основній групі

(первинній, англ. **native group**) і одній або декільком додатковим групам (англ. **additional groups**).

3 Реалізація управління користувачами і групами

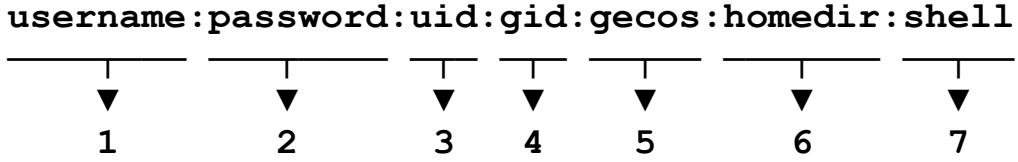
Практична реалізація механізму управління користувачами та групами в ОС Linux здійснюється за допомогою такого:

1) файлів;
2) каталогів;
3) команд.

Система обліку користувачів ОС Linux опирається на такі конфігураційні файли:

- **/etc/passwd** – облікова інформація про користувачів;
- **/etc/shadow** – тіньова інформація про користувачів;
- **/etc/group** – інформація про групи користувачів;
- **/etc/gshadow** – тіньова інформація про групи користувачів;
- **/etc/default/useradd** – властивості, призначувані за замовчуванням новим обліковим записам;
- **/etc/login.defs** – настроювання безпеки підсистеми тіньових паролів;
- каталог **/etc/skel** – шаблон («кістяк») домашнього каталогу і стандартного профілю користувача за замовчуванням. Коли для нового користувача створюється домашній каталог, у нього копіюються файли з **/etc/skel**.

Файл **/etc/passwd** містить по одному рядку (запису) для кожного користувача системи. У кожному з них зазначена низка атрибутів облікового запису, таких як реєстраційне ім’я користувача, його поточне ім’я, домашній каталог, стартовий інтерпретатор тощо. Для поділу атрибутів облікового запису слугує символ двокрапки (**:**). Кожен запис файлу **/etc/passwd** має такий формат:



Пояснення до наданого формату:

1. username – реєстраційне ім’я користувача (англ. **login**), під яким він входить у систему. Його довжина перебуває в діапазоні

1–32 символів. Воно може містити тільки латинські букви (**a**–**z**, **A**–**Z**), цифри (**0**–**9**), символи «**.**» і «**-**».

2. password – пароль. Зазвичай використовуються тіньові паролі, і замість пароля у файлі **/etc/passwd** записано символ «*****», а сам пароль у зашифрованому вигляді зберігається у файлі **/etc/shadow**. Значення цього поля встановлюється утилітою **passwd**.

3. uid – ідентифікатор користувача (англ. **User Identifier**). Унікальне ціле число в діапазоні **0**–**65534**. Це поле система використовує для завдань, пов'язаних із правами доступу до процесів і файлів (використовувати числа простіше і зручніше, ніж рядки символів), а ім'я користувача зручніше для людей. Тобто кожен обліковий запис ідентифікується як полем – **uid**, так й ім'ям користувача – **username**. **uid = 0** завжди присвоюється користувачеві **root**. За замовчуванням діапазон значень поля **uid** для реальних користувачів визначений у файлі **/etc/login.defs** як значення параметрів **UID_MIN** й **UID_MAX**, а для фіктивних користувачів – **SYS_UID_MIN** і **SYS_UID_MAX**, відповідно.

4. gid – числовий ідентифікатор первинної групи користувача – групи користувача за замовчуванням (англ. **Group Identifier**). Крім первинної групи, користувач може входити (або не входити) до складу різних груп, але в первинну групу (англ. **native group**) він входить завжди. Користувачі, що входять в одну групу, можуть працювати із загальними ресурсами системи, такими як файли та каталоги. Первинна група суперкористувача називається **root**. Їй привласнений **gid**, що дорівнює **0**. Діапазон значень поля **gid** за замовчуванням для реальних користувачів визначений у файлі **/etc/login.defs** як значення параметрів **GID_MIN** і **GID_MAX**, а для фіктивних користувачів, відповідно – **SYS_GID_MIN** і **SYS_GID_MAX**.

5. gecos – різні відомості про користувача. Такі, наприклад, як повне ім'я, адреса, номер телефону тощо. Це поле системою не контролюється. Назва поля **gecos** – історична: вона з'явилося в 1970-і роки, і походить від General Electric Comprehensive Operating System. До операційних систем сімейства UNIX вона ніяк не стосується, крім того, що спочатку було додано до **/etc/passwd** для забезпечення сумісності з деякими її сервісами.

6. homedir – домашній каталог користувача. Під час входу користувача в систему оболонка вважає поточним робочим каталогом домашній каталог користувача. Зазвичай домашні каталоги користувачів перебувають у каталозі **/home** і називаються за імем власника (наприклад, **/home/student1** для користувача з **username** рівним **student1**). Це зручна угода, що дає змогу уникнути плутанини при пошуку домашнього каталогу конкретного користувача, але технічно домашній каталог можна розташувати в будь-якому місці. Варто завжди дотримуватися такої структури домашніх каталогів, і без особливих причин змінювати цю організацію не рекомендується. Історично склалося так, що домашній каталог користувача **root** перебуває не в каталозі **/home**, а в корені файлової системи – **/root**.

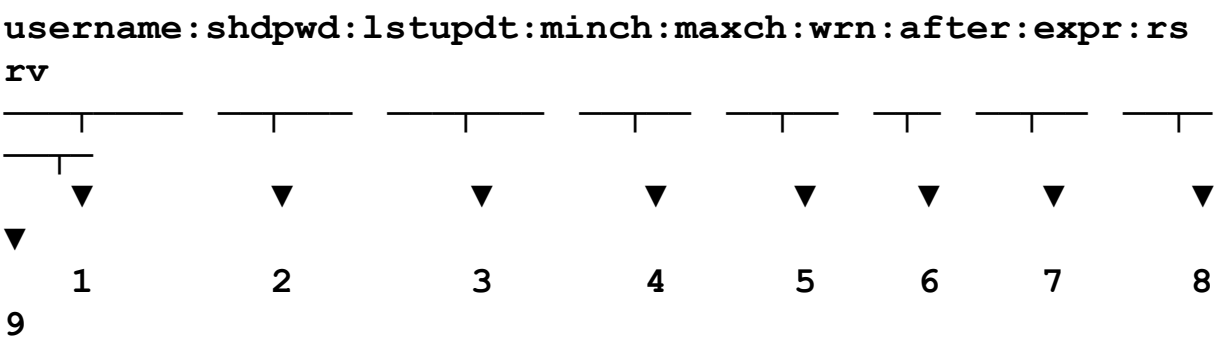
7. shell – командний інтерпретатор, що запускається за замовчуванням під час входу користувача в систему (так званий стартовий командний інтерпретатор, англ. **login shell**). Як правило, це повний шлях до оболонки, наприклад **/bin/bash**, **/bin/sh** або **/bin/tcsh**. Це значення користувач може змінити в будь-який момент за допомогою команди **chsh** (**change shell**). Звичайний користувач може змінити стартовий інтерпретатор тільки на один із перелічених у файлі **/etc/shells**, **root** – на будь-яку програму. Для системних (фіктивних) користувачів воно буде визначено як **/sbin/nologin** або **/bin/false**.

Не всі поля в обліковому запису користувача є обов'язковими. Обов'язковими є тільки поля **username**, **uid**, **gid** та **homedir**. Тому у більшій частині облікових записів заповнені всі поля, і фіктивні (системні) облікові записи використають, зазвичай, тільки деякі з них.

Можливість вільного перегляду файлу **/etc/passwd** для всіх користувачів системи створює певну загрозу безпеці. Тому з метою подолання такої потенційної небезпеки були створені *тіньові паролі* (англ. **shadow passwords**). Під час використання тіньових паролів поле **password** у файлі **/etc/passwd** містить тільки символи «**x**» або «*****» (чого ніколи не може бути в зашифрованому варіанті пароля). Для зберігання самих паролів використається другий файл із ім'ям **/etc/shadow**, що є своєрідним «продовженням» файлу **/etc/passwd**. Зв'язок між ними здійснюється по полю **username**. Файл **/etc/shadow**

переглядати та вносити зміни може тільки користувач **root** – в інших користувачів немає до нього доступу.

Як і **/etc/passwd**, файл **/etc/shadow** містить по одному рядку (запису) для кожного користувача. Кожен рядок, зі свого боку, складається з дев'яти полів, розділених символом двокрапки (**:**) і має такий формат:



Пояснення до наданого формату:

1. username – реєстраційне ім'я користувача. Збігається з **username** у файлі **/etc/passwd**.

2. shdpwd – зашифрований пароль. Щоб дізнатися, за допомогою якого алгоритму зашифрований пароль, потрібно подивитися на шифр. Якщо він досить короткий і не починається із символу «**$**», то застосовано алгоритм **DES** – сам слабкий і ненадійний; зазвичай використається в старих дистрибутивах Linux. Якщо шифр починається із символів ..., то він такий:

- **$1** – **MD5**;
- **$2** – **Blowfish**;
- **$2a** – **eksblowfish**;
- **$5** – **SHA-256**;
- **$6** – **SHA-512**.

3. lstupdt – остання зміна пароля. Кількість днів між 01.01.1970 (початком *ери UNIX*) і днем останньої зміни пароля.

4. minch – мінімальна кількість днів, що повинні пройти від однієї зміни пароля до іншої. Значення **0** у цьому полі показує, що пароль можна змінити у будь-який час.

5. maxch – максимальна кількість днів, поки пароль уважається дійсним. Після закінчення цієї кількості днів система попросить увести новий пароль, оскільки старий втратить силу. Значення **99999** у цьому

полі свідчіть про те, що користувач може не міняти пароль фактично ніколи.

6. wrn – кількість днів, за яку система почне попереджати користувача про необхідність зміни пароля (**7** для повного тижня).

7. after – кількість днів після закінчення дії пароля, коли ще користувач може входити в систему. При спробі входу в систему першою буде запускатися програма зміни пароля. Після певного строку обліковий запис буде заблокована.

8. expr – кількість днів, починаючи з 1.01.1970, після яких пароль буде заблокований.

9. rsrv – зарезервовано для майбутнього використання. Зараз не використається, і повинно бути порожнім.

В **/etc/shadow** обов'язково присутні тільки перші три поля в кожному записі, інші можуть бути відсутні – всі або частина з них. Параметри в ньому настроюються за допомогою утиліт **useradd**, **passwd** і скрипта **adduser**.

Щоб вибірково володіти ресурсами разом з іншими користувачами, кожен користувач приєднується до однієї або декількох групам користувачів. Кожна із цих груп ідентифікується деяким числом, називаним ідентифікатором групи користувачів (англ. **Group ID**). Тобто кожен користувач належить принаймні до однієї групи, зазначеної в полі **gid** файлу **/etc/passwd**. Однак користувач може бути членом і декількох інших груп. *Кожен файл операційної системи Linux асоційований тільки з однією групою користувачів*. Для кожної групи користувачів у файлі **/etc/group** утримується по одному рядку, що дуже схоже на **/etc/passwd**. Формат цього файлу такий:

```
groupname:password:gid:members
    ▼         ▼       ▼       ▼
    1         2       3       4
```

Пояснення до наданого формату:

1. groupname – рядок символів, що ідентифікує групу.

2. password – пароль. Оскільки пароль групи має вроджену проблему з безпекою – він відомий більше ніж одній людині (усім членам групи), тому на сьогодні він практично не використовується. Але разом із командою **newgrp** дає змогу користувачам, що не є членами цієї групи,

прийняти **gid** цієї групи за умови знання пароля. Зазвичай в ньому знаходиться символ «**x**».

3. gid – числовий ідентифікатор групи – **Group ID**. Це саме число перебуває в полі **gid** файлу **/etc/passwd** для вказівки групи користувача за замовчуванням.

4. members – список імен користувачів, розділених комами (без пробілів між ними), що вказує користувачів, які є членами цієї групи, але мають інші значення **gid** у файлі **/etc/passwd**.

Для визначення, у яких групах числиться поточний користувач, слугує команда **groups**. Якщо передати їй як параметр список користувачів, вона виведе списки груп, до яких належить кожен із них.

Файл **/etc/gshadow** містить тіньову інформацію про групи. Він є своєрідним «продовженням» файлу **/etc/group** і пов'язаний з ним по полю **groupname**. Файл **/etc/gshadow** складається із окремих рядків (записів) з полями, розділених символом двокрапки (**:**) і має такий формат:

```
groupname:password:admins:members
    ▼         ▼        ▼        ▼
    1         2        3        4
```

Пояснення до наданого формату:

1. groupname – ім'я групи.

2. password – зашифрований пароль для групи. Він може бути відсутнім. Використовується у разі зміни групи командою **newgrp**. Якщо поле містить неприпустиме значення, наприклад «**!**» або «*****», те скористатися ним буде не можливо.

3. admins – список адміністраторів групи, перелічених через кому, які мають право змінювати пароль за допомогою команди **gpasswd**.

4. members – список імен користувачів, розділених комами (без пробілів між ними), що вказує користувачів, які є членами цієї групи та мають інші значення **gid** у файлі **/etc/passwd**.

Файл **/etc/default/useradd** містить деякі стандартні значення властивостей, призначувані за замовчуванням новим обліковим записам. Він являє собою звичайний текстовий файл; кожен рядок описує один параметр настроювання. Рядок складаються з назви параметра і його значення, які поєднуються за допомогою символу дорівнює «**=**».

Наприклад, **SHELL=/bin/sh** – указує стартовий інтерпретатор **/bin/sh** для нового облікового запису (значення поля **shell** у файлі **/etc/passwd**). Порожні рядки і коментарі ігноруються. Коментарі починаються зі знака «ґрати» «**#**», що повинен бути першим непробільним символом у рядку. Файл **/etc/default/useradd** можна редагувати як вручну, так і скористатися командою **useradd** із ключем **-D** і параметрами.

Файл **/etc/login.defs** містить настроювання підсистеми тіньових паролів (англ. **shadow password suite**). Цей файл є обов'язковим. Його відсутність не заблокує роботу системи, але, імовірно, призведе до виконання деструктивних операцій.

Він являє собою звичайний текстовий файл, кожен рядок якого описує один параметр настроювання. Рядки складаються з назви параметра і його значення, які розділяються пробільним символом. Порожні рядки й коментарі ігноруються. Коментарі починаються зі знака «ґрати» «**#**», що повинен бути першим непробільним символом у рядку.

Значення параметрів у файлі **/etc/login.defs** можуть бути трьох типів:

1) рядок символів;
2) логічні значення;
3) числа (звичайні й довгі, максимальні значення яких залежать від архітектури комп'ютера).

Головними є такі параметри настроювання:

- **UID_MIN** і **UID_MAX** – діапазон ідентифікаторів користувачів **uid**, використовуваний у програмах **useradd**, **adduser** і **newusers** для створення звичайних користувачів;
- **GID_MIN** і **GID_MAX** – діапазон ідентифікаторів груп користувачів, використовуваний у програмах **useradd**, **groupadd**, **adduser** і **newusers** для створення звичайних груп користувачів;
- **SYS_UID_MIN**, **SYS_UID_MAX**, **SYS_GID_MIN** і **SYS_GID_MAX** – діапазони ідентифікаторів системних користувачів і груп, відповідно;
- **PASS_MAX_DAYS**, **PASS_MIN_DAYS**, **PASS_WARN_AGE** – керування «віком» пароля: максимальна кількість днів яке пароль може використовуватися, мінімальна кількість днів між можливістю зміни

пароля і за яку кількість днів до закінчення терміну дії пароля попереджати користувача, відповідно;

- **PASS_MIN_LEN** – мінімальна довжина пароля;
- **ENCRYPT_METHOD** – указує алгоритм шифрування за замовчуванням для паролів (якщо він не зазначений у командному рядку). Можливі такі значення: **DES**, **MD5**, **SHA256** або **SHA512**;
- **CREATE_HOME** – визначає, чи потрібно створювати за замовчуванням домашній каталог для нових користувачів. Ця змінна не впливає на системних користувачів. Вона може бути перевизначена ключем **-m** із командного рядка утиліти **useradd**;
- **USERGROUPS_ENAB** – опція дає змогу програмі **useradd** за замовчуванням створювати групу з ім'ям користувача, а **userdel** видаляти користувальницьку групу, якщо в ній немає більше членів;
- **ENV_PATH** та **ENV_SUPATH** – значення за замовчуванням змінної оточення **PATH** під час входу в систему для звичайного користувача й **root**, відповідно. Воно являє собою список шляхів, розділених двокрапкою (наприклад, **/bin:/usr/bin**) і може передувати **PATH=**;
- **LOGIN_RETRIES** – максимальна кількість спроб входу в систему;
- **LOGIN_TIMEOUT** – максимальний час у секундах, відведений на вхід.

Після створення нового облікового запису за допомогою команди **useradd** з параметром **-m** у домашній каталог користувача копіюються файли з каталогу **/etc/skel**. Він включає деякі (зазвичай сховані) файли, які містять стандартні параметри профілю користувача і значення параметрів додатків (такі як **.emacs**, **.bashrc** тощо). Отже каталог **/etc/skel/** виступає в ролі шаблона («кістяка») домашнього каталогу і стандартного профілю користувача. Системний адміністратор сам вирішує, які файли потрібно помістити в каталог **/etc/skel** (або в будь-який інший каталог шаблонів, зазначений у параметрі **SKEL=** файлу **/etc/default/useradd**).

4 Засоби керування користувачами і групами

В операційній системі Linux для керування користувачами і групами зазвичай використаються такі команди:

- **users** – вивести список імен користувачів, що зараз працюють у системі;
- **who** (укр. **хто** в системі?) – вивести інформацію про всіх користувачів, що зараз працюють у системі (ім'я користувача, робочий термінал, час входу в систему);
- **id [<username>]** – вивести справжні та діючі **uid-и** та **gid-и** про поточного користувача [або про користувача **<username>** – якщо він зазначений];
- **groups [<usernames>]** – вивести інформацію про групи, до яких належить поточний користувач [або користувачі **<usernames>** – якщо вони зазначені];
- **useradd <username>** – створити обліковий запис для користувача **<username>**. Із ключем **-D** – вивести (або змінити) параметри за замовчуванням для знову створюваних облікових записів;
- **usermod <username>** – змінити обліковий запис користувача **<username>** згідно переданим у командному рядку параметрам;
- **userdel <username>** – видалити обліковий запис користувача **<username>**. Із ключем **-r** буде вилучений і домашній каталог, а із ключем **-f** (від англ. **force** – змусити) буде вилучений обліковий запис, навіть якщо користувач у цей момент працює в системі;
- **passwd [<username>]** – змінити пароль користувача. Звичайний користувач може змінити тільки свій пароль, а **root** – будь-якого **<username>**;
- **groupadd <groupname>** – створити нову групу з ім'ям **<groupname>**;
- **groupmod <groupname>** – змінити параметри групи з ім'ям **<groupname>** згідно переданим у командному рядку параметрам;
- **groupdel <groupname>** – видалити групу з ім'ям **<groupname>**;

- **gpasswd <groupname>** – змінити параметри групи з ім'ям **<groupname>** у файлах **/etc/group** й **/etc/gshadow** згідно з переданим у командному рядку параметрам;
- **pwck** – перевірка та відновлення цілісності файлів **/etc/passwd** і **/etc/shadow**. Із ключем **-r** виконується перевірка в режимі «тільки читання», чого зазвичай цілком достатньо, щоб спланувати подальші дії;
- **grpck** – перевірка та відновлення цілісності файлів **/etc/group** і **/etc/gshadow**. Із ключем **-r** виконується перевірка в режимі «тільки читання», чого зазвичай цілком достатньо, щоб спланувати подальші дії;
- **newusers [<filename>]** – обновити, або створити нові, облікові записи користувачів у пакетному режимі. Це не цілком формальний спосіб, але ним зручно користуватися у разі, якщо потрібно додати багато користувачів відразу, наприклад, під час створення облікових записів для навчальної групи. Команда читає файл **<filename>** (або стандартне введення – у разі відсутності) і використовує цю інформацію для відновлення існуючих або створення нових облікових записів. Кожен рядок інформації аналогічний до формату запису стандартного файлу **/etc/passwd**, за винятком того, що поле пароля містить вхідний пароль, заданий відкритим текстом. Отже, необхідно також подбати і про захист файлу **<filename>**.

Для створення, видалення та зміни облікових записів в операційній системі Linux достатніми повноваженнями володіє тільки користувач ***root***. Створення облікового запису користувача виконується в декілька етапів:

1) додавання запису в **/etc/passwd**;
2) створення домашнього каталогу користувача;
3) копіювання в нього файлів з каталогу **/etc/skel** з настроюваннями за замовчуванням;
4) установлення пароля для знову створеного користувача.

Однак цей набір дій не завжди виконуються вручну за допомогою окремих утиліт, таких як **useradd** і **passwd**. У більшості сучасних дистрибутивів Linux для зручнішого додавання-видалення користувачів і груп поставляється набір спеціальних скриптів, таких як **adduser**,

deluser, **addgroup** та **delgroup**. Вони в інтерактивному режимі задають низку питань (здебільшого з відповідями за замовчуванням) і на підставі цієї інформації автоматично виконують усі необхідні дії по адмініструванню облікових записів.

У деяких дистрибутивах Linux (зокрема в Linux Mint) існують так само і графічні засоби адміністрування користувачів і груп. Зазвичай вони розташовані в розділі головного меню **Администрирование ⇨ Пользователи и группы**. З їхньою допомогою можна швидко вирішити визначені завдання адміністрування, але вони не мають тієї гнучкість, як низькорівневі **useradd**, **adduser**, **groupadd** тощо.

Оскільки всі файли керування користувачами і групами (**/etc/passwd**, **/etc/shadow**, **/etc/group**, **/etc/gshadow**) в операційній системі Linux є звичайними текстовими файлами, то вносити в них зміни можна і за допомогою будь-якого текстового редактора. Така необхідність виникає вкрай рідко, наприклад, у разі краху системи, коли інші засоби не працюють. Однак цією можливістю варто користуватися дуже обережно, і в крайньому разі – дуже велика ймовірність внесення деструктивних змін.

Найбезпечнішим способом «ручного» редагування цих настроювальних файлів – це використання утиліт **vipw** та **vigr**. Вони дають змогу дотримувати хоч якоїсь атомарності внесених змін. Тобто, заблокують необхідні файли доти, доки за допомогою текстового редактора будуть виконуватися необхідні зміни. За допомогою утиліт **vipw** і **vigr** можна редагувати файли **/etc/passwd** й **/etc/group**, відповідно. Якщо вказати ключ **-s**, то будуть редагуватися тіньові версії цих файлів, тобто **/etc/shadow** і **/etc/gshadow**. У більшості дистрибутивів Linux під час вибору редактора ці утиліти спочатку переглядають змінну оточення **$VISUAL**, а потім – **$EDITOR**, і якщо нічого не знайдено, запускають стандартний текстовий редактор **vi**. У Linux Mint текстовий редактор за замовчуванням указується як значення змінної **SELECTED_EDITOR=** у конфігураційному файлі **.selected-editor**. Необхідне значення цієї змінної можна встановити за допомогою будь-якого текстового редактора. Найшвидший і найнадійніший спосіб – утиліта **select-editor**. Вона пропонує вибір із

трьох найбільш популярних текстових редакторів: **ed**, **nano** та **vi**. За замовчуванням буде використовуватися **nano**.

Контрольні питання

1. Що лежить в основі механізму розмежування доступу до ресурсів комп'ютера в операційній системі Linux?
2. Що таке обліковий запис користувача?
3. Які категорії користувачів існують в операційній системі Linux?
4. Що таке група користувачів? Навіщо вони потрібні?
5. Як здійснюється реалізація механізму керування користувачами і групами в ОС Linux?
6. Перелічить конфігураційні файли та каталоги, на які опирається система обліку користувачів ОС Linux.
7. Що записано у файлі /etc/passwd?
8. Що записано у файлі /etc/shadow?
9. Що записано у файлі /etc/group?
10. Що записано у файлі /etc/gshadow?
11. Що записано у файлі /etc/default/useradd?
12. Що записано у файлі /etc/login.defs?
13. Що записано в каталозі /etc/skel?
14. Наведіть структуру файлу /etc/passwd.
15. Наведіть структуру файлу /etc/shadow.
16. Наведіть структуру файлу /etc/group.
17. Наведіть структуру файлу /etc/gshadow.
18. Наведіть структуру файлу /etc/default/useradd.
19. Наведіть структуру файлу /etc/login.defs.
20. Наведіть зміст каталог /etc/skel.
21. Як визначити, за допомогою якого алгоритму зашифрований пароль у файлі тіньових паролів /etc/shadow?
22. Назвіть початком ери UNIX – дату початку відліку «віку» паролів в ОС Linux.
23. Назвіть головні параметри настроювання підсистеми тіньових паролів у файлі /etc/login.defs.

24. Хто може створювати, видаляти та змінювати облікові записи користувачів в ОС Linux?

25. З яких етапів складається створення нового облікового запису користувача?

26. За допомогою яких інструментів в ОС Linux можна адмініструвати користувачів і групи?

27. Який текстовий редактор за замовчуванням використають утилітах vipw та vigr?

28. Як в операційній системі Linux можна встановити пароль користувача?

29. Як в операційній системі Linux можна подивитися пароль користувача?

30. Хто в операційній системі Linux може змінити пароль іншого користувача?

СПИСОК ДЖЕРЕЛ

1. Шеховцов В. А. Операційні системи / В. А. Шеховцов. – Київ : Видавнича група BHV, 2005. – 576 с.: іл.

2. Цикритзис Д. Операционные системы / Д. Цикритзис, Ф. Бернстайн. – М. : Мир, 1977. – 336 с.

3. Таненбаум Э. Современные операционные системы / Э. Таненбаум. – 3-е изд. – СПб. : Питер, 2010. – 1120 с: ил.

4. Бах М. Дж. Архитектура операционной системы UNIX / М. Дж. Бах. – М. : Мир, 1995. – 387 с.

5. Вахалия Ю. UNIX изнутри / Ю. Вахалия. – СПб. : Питер, 2003. – 844 с: ил.

6. Руссинович М. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP, Windows 2000. Мастер-класс. / Пер. с англ. / М. Руссинович, Д. Соломон. – СПб. : Питер; М. : Издательско-торговый дом «Русская редакция», 2008. – 992 с.: ил.

7. Гордеев А. В. Операционные системы : учебник для вузов. 2-е изд. / А. В. Гордеев. – СПб. : Питер, 2009. – 416 с.

8. Олифер В. Г. Сетевые операционные системы / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер. 2002. – 544 с.: ил.

9. Введение в операционные системы. Курс лекций. [Електронний ресурс – http://cs.mipt.ru/docs/courses/osstud/os.html]

10. Операционные системы [Електронний ресурс – http://www.avinout.com/]

11. Национальный Открытый Университет «ИНТУИТ» [Електронний ресурс – http://www.intuit.ru]

Навчальне видання

ПОГРЕБНЯК Борис Іванович,
БУЛАЄНКО Марина Володимирівна

ОПЕРАЦІЙНІ СИСТЕМИ

НАВЧАЛЬНИЙ ПОСІБНИК

Відповідальний за випуск *О. Б. Костенко*

Редактор *В. І. Шалда*

Комп’ютерне верстання *І. В. Волосожарова*

Дизайн обкладинки *Т. А. Лазуренко*

Підп. до друку 04.12.2017. Формат 60 × 84/16.
Друк на ризографі. Ум. друк. арк. 4,6.
Тираж 50 пр. Зам. №

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова,
вул. Маршала Бажанова, 17, Харків, 61002.
Електронна адреса: rectorat@kname.edu.ua
Свідоцтво суб’єкта видавничої справи:
ДК № 5328 від 11.04.2017.