Міністерство освіти і науки України
Вінницький національний технічний університет

В. П. Семеренко, Л. В. Крилик

Операційна система

Linux

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напрямів підготовки “Комп’ютерна інженерія”, “Інформаційна безпека”, “Комп’ютерні науки” та “Електроніка” всіх спеціальностей. Протокол № 4 від 27 жовтня 2005 р.

Вінниця ВНТУ 2006

УДК 681.3.06.(075)
С32

Р е ц е н з е н т и:

С.В. Юхимчук, доктор технічних наук, професор
В.М. Дубовой, доктор технічних наук, професор
О.О.Коваленко, кандидат технічних наук, доцент

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Семеренко В. П., Крилик Л. В.
С32 **Операційна система Linux**.- Навчальний посібник.-Вінниця: ВНТУ, 2006. - 88 с.

Навчальний посібник присвячено вивченню теорії та практичній роботі в операційній системі Linux. Детально вивчається текстовий інтерфейс користувача (команди операційної системи, мова програмування оболонки bash) і графічний інтерфейс користувача (менеджери вікон, менеджери дисплея, робочі столи KDE і GNOME). Розглянуті також основні функції системного адміністратора та поняття про процеси і роботи. Посібник призначений для студентів напрямів підготовки “Комп’ютерна інженерія”, “Інформаційна безпека”, “Комп’ютерні науки” та “Електроніка” для вивчення дисциплін "Системне програмне забезпечення", “Операційні системи”, “Програмування та алгоритмічні мови”, “Персональні комп’ютери”, а також може бути рекомендований студентам інших спеціальностей, пов’язаних з вивченням сучасного програмного забезпечення.

УДК 681.3.06.(075)

ЗМІСТ

Передмова 5
Вступ 7
Тема №1 – команди операційної системи *Linux* 9
1.1 Каталоги в *Linux* 9
1.2 Файли в *Linux* 9
1.3 Розгляд структури каталогів *Linux* 11
1.4 Основні команди *Linux* 12
1.5 Доступ до файлів і каталогів 19
Порядок виконання роботи 21
Тестові запитання для самоперевірки з теми 1 23
Контрольні питання 24
Тема № 2 – Текстовий редактор *vi* (*vim)* 25
2.1 Загальні відомості про текстовий редактор *vi* (*vim*) 25
2.2 Робота в текстовому редакторі *vi* (*vim*) 25
Порядок виконання роботи 30
Тестові запитання для самоперевірки з теми 2 30
Контрольні питання 31
Тема № 3 – Складання сценаріїв 32
3.1 Загальні відомості про текстові оболонки в *Linux* 32
3.2 Змінні в сценаріях для *bash* 33
3.3 Програмування арифметичних виразів 35
3.4 Оператори введення і виведення 35
3.5 Порівняння виразів 36
3.6 Умовні оператори 37
3.7 Оператор-перемикач 37
3.8 Оператор циклу *for* 38
3.9 Оператори циклу *while* і *until* 39
3.10 Функції 40
3.11 Робота з файлами 40
Порядок виконання роботи 41
Тестові запитання для самоперевірки з теми 3 42
Контрольні питання 42
Тема № 4 – Система *X Window* 43
4.1 Загальні відомості про систему *X Window* 43
4.2 Менеджери вікон 44
4.3 Менеджери дисплея 45
Порядок виконання роботи 46
Тестові запитання для самоперевірки з теми 4 47
Контрольні питання 47
Тема № 5 – Робочі столи користувача 48
5.1 Загальні відомості про робочі столи користувача в *Linux* 48
5.2 Робочий стіл *KDE* 49

5.3 Робочий стіл *GNOME* 50
Порядок виконання роботи 51
Тестові запитання для самоперевірки з теми 5 57
Контрольні питання 57
Тема № 6 – Основи адміністрування 58
6.1 Основні задачі системного адміністратора 58
6.2 Стадії завантаження *Linux* 58
6.3 Керування режимами роботи *Linux* 60
6.4 Особливості завантаження системи *X Window* 63
6.5 Конфігураційний файл *XF86Config* 64
6.6 Монтування файлових систем 67
6.7 Додання нових користувачів і нових груп 69
6.8 Ущільнення і архівування файлів 71
Порядок виконання роботи 71
Тестові запитання для самоперевірки з теми 6 72
Тема № 7 – Процеси і роботи в *Linux* 76
7.1 Основні понятті про процеси і роботи 76
7.2 Активні, фонові та відкладені процеси (роботи) 77
7.3 Створення процесу 78
7.4 Пріоритети процесів 80
7.5 Завершення процесів 80
7.6 Корисна інформація про фонові процеси 82
Порядок виконання роботи 83
Тестові запитання для самоперевірки з теми 7 85
Контрольні питання 85
Правильні відповіді на тестові запитання для самоперевірки 86

Рекомендована література 87

Передмова

В останні роки все більшу популярність отримує дистанційне навчання. Сучасні інформаційні технології надають небачені раніше можливості передачі знань студентам та всім бажаючим підвищити свій рівень освіти. Звичайно, дистанційне навчання не замінює собою повністю традиційне очне навчання, і тому різні форми навчання ще тривалий час будуть співіснувати разом.

Широке використання дистанційної освіти стримується як суб'єктивними, так і об'єктивними факторами. До останніх можна віднести труднощі щодо створення відповідного методичного забезпечення.

Жодні інформаційні технології ні в теперішній час, ні в найближчій перспективі не зможуть замінити собою людину-викладача, а значить, не зможуть самостійно згенерувати електронний комп'ютерний підручник. Тільки фахівець із відповідного навчального предмету може написати текст лекції, розробити практичні завдання і підготувати тести для перевірки знань. Далі необхідно підготовлений матеріал "вбудувати" у стандартну Web-структуру глобальної мережі Internet.

Найбільше часу забирає саме підготовка навчальних і контрольних матеріалів. Оскільки ці матеріали представляють собою основу будь-якого способу навчання, тому очевидним є необхідність їх спільної підготовки. Єдині за змістом методичні матеріали можуть мати різну форму в залежності від особливостей різних видів освіти: очної, заочної, дистанційної. Адже кінцева мета будь-якої освіти однакова – дати слухачам ґрунтовні знання у відповідній сфері.

Аналіз стаціонарної і дистанційної освіти показує, що, незважаючи на ряд суттєвих відмінностей, все ж таки є також і багато в них спільних рис, зокрема, в підготовці методично-навчальних матеріалів. Обидві форми навчання передбачають модульний принцип структури навчальних курсів, наявність лекцій, лабораторно-практичних занять, контрольної частини. Незалежно від форми контакту із слухачами кожен викладач намагається зробити свій предмет цікавим, інформаційно насиченим та корисним.

Таким чином, доцільно, якщо можливо, зблизити між собою методичне наповнення для різних форм навчання. Саме з таких позицій і був підготовлений цей навчальний посібник, присвячений вивченню операційної системи *Linux*.

Весь матеріал посібника розбитий на 7 тем, кожна з яких включає як теоретичну частину, так і практичну роботу на комп'ютері. Кількість тем визначається тривалістю навчального періоду в стаціонарній формі навчання. Дистанційна лекція кожної теми у Web-форматі буде мати лише декілька текстових сторінок, що не буде обтяжливим при роботі із комп'ютером. Кожна лабораторна робота виконується під керівництвом викладача в очній формі навчання. Водночас кожен крок лабораторної

роботи детально роз’яснюється, що полегшує роботу при самостійному вивченні предмета. Тестові питання для самоперевірки будуть корисними для всіх категорій студентів. Вони вимагають відповіді лише у формі “Так/Ні”. Одночасно з кожної теми є також перелік контрольних питань, які потребують розгорнутої відповіді.

Для студентів стаціонарної форми навчання навчальний посібник матиме паперову форму, а для дистанційної освіти – є текстовою основою електронного комп’ютерного підручника у Web-форматі.

Вступ

Linux – це операційна система, яка створена на основі загальновідомої системи Unix. Якщо Unix має більше, як 30-літню історію, то датою народження *Linux* є 1991 рік. Саме цього року фінський студент Лінус Торвальдс написав невелику системну програму, що дозволяла лише керувати процесами та основною пам'яттю комп'ютера, і звернувся до всіх програмістів із закликом продовжити його роботу. Зусиллями багатьох ентузіастів зі всього світу вже через декілька місяців була створена закінчена операційна система сімейства *Unix*. Сьогодні *Linux* стоїть в одному ряду з найпотужнішими операційними системами і продовжує далі розвиватись і розширювати свої функціональні можливості. Жодна серйозна фірма програмного профілю не може ігнорувати цю операційну систему і тому більшість програмних пакетів мають свої версії і для *Linux*.

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

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

Варто відзначити дві характерні особливості *Linux*: безкоштовність та відкритість програмного коду.

Більша частина програмного забезпечення для *Linux* розроблена в рамках проекту GNU фонду FSF (Free Software Foundation – вільного програмного забезпечення), тому ця операційна система може вільно розповсюджуватись. На відміну від ліцензій для комерційних продуктів, ліцензія GPL (GNU Generic Program License) для *Linux* захищає авторські права всіх розробників вимагаючи одночасно від них, щоб їх програми і початкові програмні коди були загальнодоступними. Відкритість програмного коду дає також унікальну можливість для самостійного вивчення нових тенденцій в сучасному системному програмуванні. Саме тому *Linux* є найкращою базою для використання в навчальному процесі.

Будь-який програміст може написати свою власну програму або внести зміни в існуючі програми, що входять до складу *Linux*. Звичайно, новостворені програми не завжди проходять жорстке багатомісячне тестування, як це відбувається із новими продуктами відомих фірм. Однак практика підтверджує достатньо високу надійність *Linux*.

Необхідно розрізняти поняття операційної системи (ОС) і дистрибутиву. ОС – це набір системних програм, призначених, по-перше,

для керування ресурсами комп'ютера чи комп'ютерної мережі, по-друге, для полегшення взаємодії користувача з комп'ютером на основі дружнього інтерфейсу. Дистрибутив включає в себе ОС, а також великий набір службових, навчальних, ігрових та інших сервісних програм, зокрема компілятори різних мов програмування, текстові та графічні редактори тощо. Оскільки можна створити різні поєднання ОС із вказаними програмами, тому існує багато різних дистрибутивів. Найбільш відомі із них: *Red Hat, Mandrake, Debian, ASP*, *LFS*.

Посібник призначений для початкового ознайомлення з *Linux* і отримання основних практичних навичок для роботи із цією ОС. В посібнику є 7 тем:

- команди операційної системи *Linux*;
- текстовий редактор *vi* (*vim*);
- складання сценаріїв;
- графічна система *X Window*;
- робочі столи користувача;
- основи адміністрування в *Linux*;
- процеси і роботи.

Кожна лабораторна робота містить короткий теоретичний матеріал, практичні завдання а також контрольні і тестові запитання.

ТЕМА № 1

Команди операційної системи *Linux*

Зміст теми: Знайомство із структурою каталогів операційної системи *Linux*, правами доступу до файлів і каталогів, практичне засвоєння основних команд *Linux* для роботи із файлами і каталогами.

Теоретичні відомості

1.1 Каталоги в *Linux*

Файлова система *Linux* має багато каталогів, які утворюють ієрархічну деревоподібну систему. На початку цього дерева розташований кореневий каталог, який позначається як символ / (похила лінія). Далі розташовуються інші каталоги та файли. Призначення основних каталогів першого рівня наведено в розділі 1.3. На відміну від операційних систем *MS-DOS* і *Windows*, тут майже всі каталоги мають стандартні імена і їх не можна вилучати або перейменовувати.

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

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

/bin

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

/home/user/tom

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

1.2 Файли в *Linux*

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

В *Linux* поняття файла є універсальним. Всі фізичні пристрої (дисплей, принтер, клавіатура та ін.) представляються як спеціальні файли,

які зберігаються в каталозі */dev*. До речі, каталоги в *Linux* розглядаються як третя різновидність файлів, оскільки вони теж займають деяке місце на диску.

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

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

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

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

*.txt - текстовий файл;

*.bin - бінарний, тобто виконуваний, файл;

*.c - програма мовою Сі.

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

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

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

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

1.3 Розгляд структури каталогів *Linux*

Розглянемо призначення основних каталогів першого рівня.

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

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

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

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

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

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

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

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

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

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

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

/var/log – для зберігання системних журналів;

/var/mail – для організації поштових скриньок користувачів;

/var/spool – для організації буферних черг для принтера, пошти і т.п.

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

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

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

1.4 Основні команди *Linux*

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

man <ім'я команди>

Далі розглядаються лише найпоширеніші команди в скороченому форматі.

1.4.1 Інформація про поточний каталог

Для видачі імені поточного каталогу використовується команда

pwd

1.4.2 Перегляд каталогу

Перегляд вмісту каталогу здійснюється командою ***ls***. Формат команди:

ls [–опції] [<Каталог>] [<Файл1> <…>]

При відсутності всіх опцій та параметрів можна отримати список всіх файлів і підкаталогів поточного каталогу у мінімальному форматі. Мінімальний формат передбачає видачу лише імен файлів та підкаталогів. Розширений формат дозволяє розрізнити між собою файли та підкаталоги (після імен останніх ставиться похила лінія "/"). Повний формат передбачає видачу для кожного файла або підкаталога інформації в такій послідовності:

- тип ("*d*" - підкаталог, "-" - файл, "*b*"- блочний пристрій, "*c*" - символьний пристрій);
- права доступу (для користувача-власника, групи користувачів, всіх інших);
- користувач;
- група користувачів;
- обсяг пам'яті в байтах;
- дата створення;
- час створення;
- ім'я.

Основні опції:

l – видача списку файлів та підкаталогів у повному форматі;

F – видача списку файлів та підкаталогів у розширеному форматі ;

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

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

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

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

ls

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

ls –F /home/user/mydir

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

ls –l file1.txt

1.4.3 Створення каталогу

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

mkdir [–опції] [<Каталог1> <…>]

Основні опції:

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

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

mkdir mydir1

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

mkdir –m 755 mydir2

1.4.4 Зміна поточного каталогу

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

cd [<Каталог>]

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

cd /home/user/dir1/mydir

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

cd . .

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

cd

1.4.5 Знищення каталогу

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

rmdir [–опції] [<Каталог1> <…>]

За замовчуванням знищуються лише пусті каталоги та підкаталоги.

Основні опції:

p – знищення всіх підкаталогів того каталога, який ліквідовується.

Приклад 1. Знищити каталог ***mydir:***

rmdir mydir

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

rmdir -p mydir/mydir1 mydir/mydir2

1.4.6 Створення файла

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

cat > <file>

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

cat > file1.txt

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

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

cat >> file1.txt

1.4.7 Копіювання файлів

Для копіювання файлів використовується команда ***cp***. Формат команди:

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

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

Основні опції:

i – видача запиту на підтвердження заміни існуючого файла.

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

cp file1.txt file2.txt

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

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

1.4.8 Переміщення (перейменування) файлів

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

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

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

Основні опції:

i – видача запиту на підтвердження заміни існуючого файла.

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

Приклад 1.

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

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

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

mv [–опції] <Файл1> < Файл2>

де ***<Файл1> -*** старе імя файла;

< Файл2> - нове імя файла.

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

1.4.9 Знищення файлів

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

rm [–опції] <Файл1> … < ФайлN>

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

Основні опції:

i – видача запиту на підтвердження знищення файла.

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

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

rm –i *.txt

1.4.10 Об’єднання файлів

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

cat [–опції] <Файл1> … < ФайлN>

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

Основні опції:

n – здійснити нумерацію рядків об’єднаного файла.

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

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

cat file1.txt file2.txt

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

cat file1.txt file2.txt > common.txt

1.4.11 Сортування файлів

Для сортування вмісту текстових файлів використовується команда ***sort.*** Формат команди:

sort [–опції] <Файл1> … < ФайлN>

За замовчуванням вміст файлів ***<Файл1> … < ФайлN>*** відсортовується за алфавітом і результат видається на стандартний пристрій виведення, тобто на екран дисплея.

Основні опції:

r – відсортувати в оберненому порядку.

o <File> – результат сортування записати у файл ***<File>*** .

r – розглядати бінарні файли як текстові.

Приклад 1. Відсортувати вміст файла ***file1.txt*** в оберненому порядку і результат записати у файл ***result.txt***:

sort –r –o result.txt file1.txt

1.4.12 Пошук відмінностей між файлами

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

diff [–опції] <Файл1> < Файл2>

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

Основні опції:

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

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

i – ігнорувати відмінності в регістрах.

q – повідомляти лише про сам факт відмінностей без подробиць.

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

diff file1.txt file2.txt

1.4.13 Пошук у файлі за зразком

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

grep [–опції] <зразок> [<файл>]

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

Основні опції:

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

a – розглядати бінарні файли як текстові;

r – ігнорувати відмінності в регістрах.

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

grep –r "display" file1.txt

1.4.14 Пошук файлів

Для пошуку файлів використовується команда ***find***. Формат команди:

find [<місце пошуку>] [<вираз>]

Здійснюється пошук файла із каталогів, заданих в ***<місце пошуку>,*** згідно з критерієм, що визначається у ***<вираз>.*** Можна виконувати пошук файлів за різними критеріями:

- за іменем (***–name <ім'я файла>***);
- за датою або часом створення (***-atime n, mtime n***);
- за розміром (***–size n***);
- за типом (***–type t***);
- за іменем користувача-власника (***–user <ім'я власника>***);
- за іменем групи користувачів (***–group <ім'я групи>***).

Можна шукати один конкретний файл або сукупність однотипних файлів, які можна задати одним іменем із символами підстановки. Якщо не вказано ***<місце пошуку>,*** тоді пошук здійснюється у поточному каталозі.

Приклад 1. Здійснити пошук файла ***inittab*** за всіма каталогами:

find / –name inittab

1.4.15 Перегляд файлів

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

cat file1.txt

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

more file1.txt

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

Переглянути посторінково текст файла можна також і за командою

less file1.txt

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

За допомогою команди

head [–опції] <файл>

можна переглянути лише початок файла, а за допомогою команди

tail [–опції] <файл>

можна переглянути лише кінець цього файла.

Основні опції команд ***head*** і ***tail***:

n c – видати на екран n символів;

n l – видати на екран n рядків;

n d – видати на екран n блоків.

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

head –5l file1.txt

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

tail –40c file1.txt

1.4.16 Статистичні дані про файл

Для отримання статистичних даних про розмір файла використовується команда ***wc***. Формат команди:

wc [–опції] <файл>

За замовчуванням видається інформація про кількість рядків, слів та символів (саме в такому порядку) у ***<файл>.*** За допомогою опцій можна отримати цю інформацію вибірково:

l – видати кількість рядків;

w – видати кількість слів;

c – видати кількість символів.

Приклад. Видати на екран кількість рядків файла ***file1.txt:***

wc –l file1.txt

1.4.17 Переадресація введення-виведення

За замовчуванням як пристрій введення використовується стандартний пристрій введення, тобто клавіатура, а як пристрій виведення використовується стандартний пристрій виведення, тобто екран дисплея. Для більшості команд можна зробити так, щоб команда отримувала дані із

файла, а не з клавіатури, і виводила свої результати на інший дисплей або у файл. Для позначення переадресації введення-виведення в командному рядку використовуються символ **<** або символ **>**. Можна розглядати напрям стрілки як напрям передачі даних.

Наприклад, якщо за командою ***ls*** ми отримаємо список файлів каталога на екрані, то за допомогою команди

ls > dir.txt

цей список поміщається у файл ***dir.txt***.

Якщо необхідно дописати нові дані у існуючий файл, тоді використовується символ переадресації **>>**. Приклад такої переадресації для команди ***cat*** був наведений раніше.

Якщо файл не існує, тоді використання символів переадресації **>** і **>>** викликає створення відповідного файла.

Крім переадресації існує ще один спосіб зміни стандартного виконання введення і виведення – це використання конвеєра, коли вихід від однієї команди стає входом для іншої команди. Конвейер позначається вертикальною лінією. Наприклад, за допомогою команди

ls -l | wc

можна підрахувати кількість файлів і підкаталогів у поточному каталозі.

1.5 Доступ до файлів і каталогів

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

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

Права доступу до каталогів і файлів розподіляються на три типи: читання (*read*), запис (*write*), і виконання (*execute*). Ці типи прав доступу можуть бути надані трьом категоріям користувачів: власникові файла, групі користувачів або всім іншим користувачам.

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

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

ls - l file1.txt

На екрані дисплея отримаємо такий результат

- rw- r- - r- - 1 user1 505 Mar 13 19:05 file1.txt

Перший символ дозволяє розрізнити між собою файли і каталоги: для файлів записується “-“ , а для каталогів “d”. Далі йде рядок прав доступу до цього файла з боку різноманітних категорій користувачів. Потім виводиться кількість (*1*) синонімів, під якими даний файл відомий системі. Третє поле – ім’я власника файла (*user1*) і четверте - група (*505*), до якої належить власник файла. Очевидно, що останнє поле містить ім’я файла (*file1.txt*), а інші поля означають дату та час створення файла.

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

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

Для зміни прав доступу до файлів і каталогів використовується команда

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

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

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

chmod u+x file5.txt

1.6 Рекомендована література з теми 1

[1 с.25-74], [2, с.66-102], [3, с.3-538], [4, с.31-52], [5, с.528-549], [6, с.749-751], [7, с.96-109].

Повний список літератури знаходиться на стор. 87.

Порядок виконання роботи

Таблиця 1.1 – Вивчення основних команд *Linux*

Завдання	*Виконання*
1. В своєму домашньому каталозі створити два нових каталоги*:* *mydir1* та *mydir2*	*Ввести команди* **mkdir mydir1** <Enter> **mkdir mydir2** <Enter>
2. Перейти в каталог *mydir1*	*Ввести команду* **cd mydir1** <Enter>
3. Створити в каталозі *mydir1* файл *file1.txt* і ввести в нього заданий текст із клавіатури	*Ввести команди* **cat > file1.txt** <Enter> *Ввести текст* **1. Turn on the machine.** **2. Start the program.** **3. Request the option.** **4. End the program.** **5. Turn off the machine.** *Натиснути клавіші* <Ctrl><d>
4. Створити дві копії цього файла: *file2.txt* та *file3.txt*	*Ввести команди* **cp file1.txt file2.txt** <Enter> **cp file1.txt file3.txt** <Enter>
5. Дописати у файл *file2.txt* новий рядок: *It is new string1.*	*Ввести команду* **cat >> file2.txt** <Enter> *Ввести текст* **It is new string1.** *Натиснути клавіші* <Ctrl><d>
6. Дописати у файл *file3.txt* нові рядки: *It is new string 2.* *It is new string 3.*	*Ввести команду* **cat >> file3.txt** <Enter> *Ввести текст* **It is new string 2.** **It is new string 3.** *Натиснути клавіші* <Ctrl><d>
7. Перемістити файли *file2.txt* та *file3.txt* в каталог *mydir2*	*Ввести команди* **mv file2.txt /home/user/mydir2** <Enter> **mv file3.txt /home/user/mydir2** <Enter>
8. Видати на екран дисплея вміст файла *file1.txt*	*Ввести команду* **cat file1.txt** <Enter>
9. Перевірити права повноваження на файл *file1.txt*	*Ввести команду* **ls -l file1.txt** <Enter>

Продовження таблиці 1.1

Завдання	*Виконання*
10. Додати повноваження на виконання файла *file1.txt* власнику цього файла та групі користувачів, в яку входить власник файла	*Ввести команду* **chmod u+x, g+x file1.txt** < Enter>
11. Скопіювати *file1.txt* в початковий домашній каталог	*Ввести команду* **cp file1.txt /home/user** <Enter>
12. Перейменувати файл *file1.txt* у файл *file.txt*	*Ввести команду* **mv file1.txt file.txt** <Enter>
13. Вилучити файл *file1.txt* із поточного каталогу	*Ввести команду* **rm file1.txt** <Enter>
14. Перейти в початковий домашній каталог	*Ввести команду* **cd . .** <Enter>
15. Знищити каталог *mydir1*	*Ввести команду* **rmdir mydir1** <Enter>
16. Перейти в каталог *mydir2*	*Ввести команду* **cd mydir2** <Enter>
17. Порівняти між собою файли *file2.txt* та *file3.txt.* і спільні рядки записати у файл *res.txt*	*Ввести команду* **comm file2.txt file3.txt > res.txt** <Enter>
18. З'єднати файли *file2.txt* та *file3.txt* в один файл *file.txt*	*Ввести команду* **cat file2.txt + file3.txt > file.txt** <Enter>
19. Перейти в домашній каталог.	*Ввести команду* **cd . .** <Enter>
20. Видати на екран дисплея список всіх файлів домашнього каталогу в повному форматі та записати цей список в файл *dir.txt*	*Ввести команду* **ls -l /home/user > dir.txt** <Enter>
21. Знайти в одному із системних каталогів файл *Xclients* та скопіювати його в домашній каталог	*Ввести команди* **find / -name Xclients** <Enter> **cp /etc/X11/xinit/Xclients /home/user** <Enter>
22. Переглянути посторінково файл *Xclients* на екрані дисплея	*Ввести команду* **more Xclients** <Enter>
23. Видати на екран дисплея першу сторінку файла *Xclients*	*Ввести команду* **head Xclients** <Enter>
24. Видати на екран дисплея останню сторінку файла *Xclients*	*Ввести команду* **tail Xclients** <Enter>

Продовження таблиці 1.1

Завдання	*Виконання*
25. Видати на екран дисплея перші 5 рядків файла *Xclients*	*Ввести команду* **Head -5l Xclients** <Enter>
26. Виконати сортування рядків файла *Xclients*	*Ввести команду* **sort Xclients** <Enter>
27. Виконати статистичне дослідження файла *Xclients:* - підрахувати кількість рядків, слів та символів; - підрахувати кількість рядків, які починаються із букви *f*;	*Ввести команди* **wc Xclients** <Enter> **grep "f*" Xclients | wc –l** <Enter>
28. Видати на екран дисплея календар 2006 року (перше півріччя)	*Ввести команду* **cal 2006 | more** <Enter>
29. Видати на екран дисплея список всіх команд інтерпретатора *bash*	*Ввести команду* **help** <Enter>
30. Видати список останніх 10 команд	*Ввести команду* **history -10** <Enter>

Примітка. Передбачається, що *login* користувача - *user*.

Тестові запитання для самоперевірки з теми 1

1. В каталозі */mnt* зберігається інформація про конфігурацію системи. (Так / Ні).
2. За допомогою команди
   cat file1.txt
   можна створити файл *file1.txt*. (Так / Ні).
3. Команда *ls* видає вміст поточного каталогу на екран, якщо цей каталог має право на читання. (Так / Ні).
4. В каталозі */bin* зберігаються програми для роботи з файлами, каталогами, а також текстові оболонки, редактори та інші системні і службові програми. (Так / Ні).
5. За допомогою команди
   tail -20 file1.txt
   можна вивести на екран перші 20 рядків файла *file1.txt*. (Так / Ні).
6. Каталог */usr* є домашнім каталогом користувача *user*. (Так / Ні).
7. Каталог */dev* містить спеціальні файли пристроїв. (Так / Ні).
8. За допомогою команди
   mkdir dir1
   можна ліквідувати каталог *dir1*. (Так / Ні).

9. Каталог */root* є власним каталогом суперкористувача. (Так / Ні).
10. За допомогою команди

cat file1.txt + file2.txt

можна об'єднати файли *file1.txt і file2.txt*. (Так / Ні).
11. За допомогою команди

chmod g+w file1.bin

можна надати групі користувачів право на виконання файла *file1.bin*. (Так / Ні).
12. За допомогою команди *pwd* можна вивести на екран список всіх файлів поточного каталогу. (Так / Ні).
13. За допомогою команди

mv /home/user/file1.txt /tmp

рядовий користувач *user* має право переслати файл *file1* із свого домашнього каталогу в каталог, де зберігаються допоміжні файли. (Так / Ні).
14. За допомогою команди

rm /home/user

можна ліквідувати пустий каталог */home/user*. (Так / Ні).
15. За допомогою команди

cd . .

можна перейти із поточного каталога на один каталог вище по дереву каталогів. (Так / Ні).
16. За допомогою команди

head -w file1.txt

можна підрахувати кількість слів у файлі *file1.txt*. (Так / Ні).
17. За допомогою команди

wc -20 file1.txt

можна вивести на екран перші 20 рядків файла *file1.txt*. (Так / Ні).
18. Право доступу на запис у каталог дає можливість вилучення файлів із цього каталогу. (Так / Ні).
19. За допомогою команди

sort file1.txt

можна відсортувати вміст файла *file1.txt*. і результат сортування записати у цей же файл. (Так / Ні).

Контрольні питання

1. Як створити новий каталог?
2. Якою командою переглянути вміст каталогу?
3. Як виконати знищення файла?
4. Яким чином знищити каталог в ОС *Linux*?
5. Які відмінності ви помітили під час знищення файлів і каталогів в ОС *Linux* порівняно з цими ж діями в *MS DOS* та *Windows*?
7. Як здійснити посторінковий перегляд файла?

ТЕМА № 2

Текстовий редактор *vi* (*vim*)

Зміст теми: Отримання практичних навичок роботи при введенні та редагуванні тексту в редакторі ***vi*** (***vim***).

Теоретичні відомості

2.1 Загальні відомості про текстовий редактор *vi* (*vim*)

В епоху панування графічного інтерфейсу може здаватися дивним вивчення роботи текстового редактора, історія створення котрого бере початок ще з часів перших версій операційної системи *Linux*. Проте, насправді нічого дивного в цьому не має. По-перше, редактор ***vi*** є дуже потужною системою щодо створення і редагування текстової інформації. Зрозуміло, в ньому не можна створювати графічну інформацію і він позбавлений багатьох корисних властивостей, до яких звикли користувачі редакторів типу Microsoft Word. Проте сучасні текстові редактори вимагають великих системних ресурсів і обов'язково наявності графічного режиму роботи операційної системи.

Хоча часто бувають ситуації, коли необхідно виконати різноманітні налагодження операційної системи при відсутності графічного режиму. І тут велику допомогу може дати простий і невибагливий з точки зору системних ресурсів текстовий редактор. Багато функцій операційної системи *Linux* реалізовані у вигляді сценаріїв – спеціальних програм, які дуже легко створюються і редагуються в текстовому редакторі.

Останні версії редактора ***vi*** (модернізований ***vi*** – (***vim***)) дозволяють працювати з пристроєм „миша" і оперувати на візуальному рівні з великими блоками тексту. Про це, доречі, говорить і сама назва редактора ***vi*** (visual interpretator – візуальний інтерпретатор).

За довгі роки існування редактор ***vi*** постійно поповнювався новими командами та новими можливостями. Для того, щоб створити новий текстовий файл і провести його найпростіше редагування досить засвоїти лише найпростіші навички роботи з редактором ***vi***, а потім можна вивчити і багато інших додаткових можливостей.

2.2 Робота в текстовому редакторі *vi* (*vim*)

Для початку необхідно знати, що редактор ***vi*** працює в двох основних режимах: командному та введення тексту. У командному режимі натискання клавіш інтерпретуються як команди, що дозволяють зберігати текст, переміщувати курсор в різні частини файла, редагувати фрагмент тексту і виходити з редактора. В режимі введення тексту натискання

клавіш клавіатури сприймається як текст редагованого файла. В процесі роботи можна вільно переходити з одного режиму в інший. Більш детальну інформацію про редактор ***vi*** дають таблиці 2.1-2.6.

Таблиця 2.1 - Початкове знайомство з редактором. Введення тексту

Завдання	*Виконання*
1. Ввійти в редактор	*Ввести команду* **vi** <Enter>
2. Ввійти в режим введення даних	*Натиснути клавішу* <Insert>
3. Ввести заданий текст	
4. Вийти із режиму введення даних	*Натиснути клавішу* <ESC>
5. Перейти в командний режим	*Натиснути клавіші* <Shift> <:>
6. Записати текст в файл *file1.txt*	*Ввести команду* : **w file1.txt** <Enter>
7. Перейти в режим заміни	*Натиснути клавіші* <Insert><Insert>
8. Виконати заміну частини тексту	
9. Вийти із режиму заміни	*Натиснути клавішу* <ESC>
10. Перейти в командний режим	*Натиснути клавіші* <Shift> <:>
11. Вийти із редактора із збереженням введених змін	*Ввести команду* : **wq** <Enter>

Таблиця 2.2 - Вивчення команд переміщення курсора

Завдання	*Виконання*
1. Ввійти в редактор із заданим файлом *XF86Config*.	*Ввести команду* **vi XF86Config** <Enter>
2. Вивчити команди глобального переміщення курсора: а) перехід на один кадр вперед б) перехід на один кадр назад в) перехід до верху екрана г) перехід до низу екрана д) перехід до кінця тексту	*Натиснути клавіші:* а) <Ctrl> <**F**> або <PageUp> **б**) <Ctrl> <**B**> або <PageDown> в) **H** г) **L** д) **G**
3. Вивчити команди локального переміщення курсора: а) до початку речення б) до кінця речення в) до початку розділу г) до кінця розділу д) на рядок з номером n	*Натиснути клавішу:* а) **(** б) **)** в) **{** г) **}** д) n**G**

4. Вийти із редактора без збереження введених змін	*Ввести команду* : **q!** <Enter>

Таблиця 2.3 - Редагування тексту в редакторі

Завдання	*Виконання*
1. Ввійти в редактор із заданим файлом *XF86Config*.	*Ввести команду* **Vi XF86Config** <Enter>
2. Вивчити команди для вилучення фрагмента тексту: а) вилучити слово б) вилучити 3 слова в) вилучити рядок г) вилучити 3 рядки д) вилучити частини рядка від позиції курсора до кінця е) вилучити речення ж) вилучити розділ	*підвести курсор до початку заданого фрагмента тексту і натиснути клавіші:* а) **<d><w>** б) **<d><3><w>** або **<3><d><w>** в) **<d><d>** г) **<d><3><d>** або **<3><d><d>** д) **<d><$>** е) **<d><)>** ж) **<d><}>**
3. Вивчити команди для копіювання фрагмента тексту: а) копіювати слово б) копіювати 3 слова в) копіювати рядок г) копіювати 3 рядки д) копіювати частини рядка від позиції курсора до кінця е) копіювати речення ж) копіювати розділ	*підвести курсор до початку заданого фрагмента тексту і натиснути клавіші:* а) **<y><w>** б) **<y><3><w>** або **<3><y><w>** в) **<y><y>** г) **<y><3><y>** або **<3><y><y>** д) **<y><$>** е) **<y><)>** ж) **<y><}>**
4. Вставити фрагмент тексту із буфера	*підвести курсор в нове місце і натиснути клавішу <**p**> (<**P**>)*
5. Вивчити команди для переміщення фрагмента тексту в буфер: а) перемістити 3 слова в буфер б) перемістити рядок в буфер в) перемістити 3 рядки в буфер г) перемістити розділ.	*підвести курсор до початку заданого фрагмента тексту і натиснути необхідні клавіші:* а) **<d><3><w>** або **<3><d><w>** б) **<d><d>** в) **<d><3><d>** або **<3><d><d>** г) **<d><}>**
6. Вставити фрагмент тексту із буфера	*підвести курсор в нове місце і натиснути клавішу <**p**> (<**P**>)*
7. Вийти із редактора без збереження введених змін.	*Ввести команду* : **q!** <Enter>

Примітка 1. Багато із перерахованих вище команд можуть мати множник для одночасного виконання однакових дій, наприклад 2b, 3w, 4(, 5} і т. д.

Примітка 2. За командою <**p**> об'єкт вставляється справа від позиції курсора, а за командою <**P**> – перед позицією курсора.

Таблиця 2.4 - Одночасна робота в редакторі з двома файлами

Завдання	*Виконання*
1. Ввійти в редактор із заданим файлом *XF86Config*	*Ввести команду* **vi XF86Config** <Enter>
2. Виконати операцію копіювання фрагмента тексту із файла *XF86Config* (наприклад, рядка)	*підвести курсор до початку заданого фрагмента тексту;* *і натиснути клавіші:* **<y><y>**
3. Перейти в командний режим	*Натиснути клавіші* <Shift> <:>
4. Зберегти зміни у файлі *XF86Config*	*Ввести команду* **: w**
5. Ввійти в режим редагування файла *File.txt*	*Ввести команду* **: *e* File.txt**
6. Виконати необхідну операцію з редагування (наприклад, вставка рядка із іншого файла)	*підвести курсор в задане місце* *і натиснути клавіші* ***<p>*** *(****<P>****)*
7. Вийти із редактора із збереження введених змін	*Ввести команду* : w**q** <Enter>

Таблиця 2.5 - Вивчення опцій (режимів роботи) редактора

Завдання	*Виконання*
1. Ввійти в редактор із заданим файлом *XF86Config*	*Ввести команду* **vi XF86Config** <Enter>
2. Ввійти в режим введення даних	*Натиснути клавішу* <Insert>
3. Вийти із режиму введення даних	*Натиснути клавішу* <ESC>
4. Перейти в командний режим	*Натиснути клавіші* <Shift> <:>
5. Перевірити роботи таких опцій: а) перегляд списку всіх опцій б) перенумерація рядків в) установлення автозапису (перед викликом нового файла) г) пошук в тексті заданого слова "Monitor' д) встановлення ширини відступу, рівного 5 е) підтримка режиму роботи з пристроєм "миша"	*Ввести команду:* а) : **set all** <Enter> б) : **set nu** <Enter> в) : **set ai** <Enter> г) : **?Monitor** <Enter> д) : **set sw=5** <Enter> е) : **set mouse=a** <Enter>

6. Вийти із редактора без збереження введених змін	*Ввести команду* : **q!** <Enter>

Таблиця 2.6 - Вивчення роботи редактора у візуальному режимі

Завдання	*Виконання*
1. Ввійти в редактор із заданим файлом *XF86Config*	*Ввести команду* **vi XF86Config** <Enter>
2. Перейти в символьний візуальний режим	*Натиснути клавішу* **<v>**
3. Виділити курсором кілька символів тексту	*Використати клавіші клавіатури для переміщення курсора* <↑> , <↓>, <←>, <→>.
4. Вийти із цього режиму	*Натиснути клавішу* <ESC>
5. Перейти в рядковий візуальний режим	*Натиснути клавіші* <Shift> **<v>**
6. Виділити курсором 5 рядків тексту	*Використати клавіші клавіатури для переміщення курсора* <↑> , <↓>
7. Скопіювати виділений фрагмент тексту	*- натиснути клавішу* **<y>**; *- підвести курсор в нове місце;* *- натиснути клавішу* ***<p>(<P>)***
8. Виділити курсором 5 рядків тексту	*Використати клавіші клавіатури для переміщення курсора* <↑> , <↓>
9. Вилучити виділений фрагмент тексту	*- натиснути клавішу* <Del> або **<d>**; *- підвести курсор в нове місце;* *- натиснути клавішу* ***<p> (<P>)***
10. Вийти із цього режиму	*Натиснути клавішу* <ESC>
11. Перейти в блочний візуальний режим	*Натиснути клавіші* <Ctrl> **<v>**
12. Виділити візуальний блок	*Використати клавіші клавіатури для переміщення курсора* <↑> , <↓>, <←> , <→>
13. Виконати операції редагування візуального блоку.	*Аналогічно, як для візуального рядка.*
16. Виділити візуальний блок за допомогою пристрою “миша”	*Використати курсор пристрою “миша”*
17. Виконати операції редагування для візуального блока	*Аналогічно, як для візуального блока при клавіатурному виділенні*

20. Вийти із редактора без збереження введених змін	*Ввести команду* : **q!** <Enter>

2.3. Рекомендована література з теми 2

[1, с.138-221], [2, с.141-148], [7, с.203-209].

Повний список літератури знаходиться на стор. 87.

Порядок виконання роботи

- ознайомитись з теоретичними відомостями про редактор ***vi*** (***vim***);
- набрати в редакторі ***vi*** (***vim***) запропонований викладачем фрагмент тексту;
- виконати збереження створеного файла;
- виконати команди переміщення курсора в тексті, який було набрано;
- виконати редагування створеного тексту в редакторі ***vi*** (***vim***);
- виконати команди копіювання фрагмента тексту в редакторі ***vi*** (***vim***);
- виконати команди переміщення фрагмента тексту в редакторі ***vi*** (***vim***);
- вивчити одночасну роботу в редакторі ***vi*** (***vim***) з двома файлами;
- вивчити роботу редактора ***vi*** (***vim***) у візуальному режимі.

Тестові запитання для самоперевірки з теми 2

1. Редактор ***vi*** (***vim***) може працювати в командному режимі, в режимі вставки і в режимі заміни. (Так / Ні).
2. Перехід в режим вставки можна здійснити при натисненні на клавішу *ESC*. (Так / Ні).
3. Редактор ***vi*** (***vim***) дозволяє створювати графічні зображення. (Так / Ні).
4. Редактор ***vi*** (***vim***) дозволяє переносити фрагменти тексту через буфер від одного файла до іншого. (Так / Ні).
5. Редактор ***vi*** (***vim***) не підтримує пристрій “миша”. (Так / Ні).
6. Для вилучення 5-ти текстових рядків необхідно почергово натиснути клавіші <*d*><*5*><*d*>. (Так / Ні).
7. Для переходу в командний режим необхідно почергово натиснути клавіші <*Shift*><:>. (Так / Ні).
8. Редактор ***vi*** (***vim***) дозволяє візуально виділяти блоки тексту. (Так / Ні).
9. Для перенесення 5-ти слів текстових рядків необхідно почергово натиснути клавіші <*m*><*5*><*d*>. (Так / Ні).

10. Для копіювання 7-ти текстових рядків необхідно почергово натиснути клавіші <*y*><*7*><*y*>. (Так / Ні).
11. Для вставки фрагмента тексту із буфера на самий початок файла необхідно помістити курсор на початок файла і натиснути на клавішу <*p*>. (Так / Ні).

Контрольні питання

1. Як створити текст в редакторі ***vi*** (***vim***) ?
2. Що таке режим зміни в редакторі ***vi*** (***vim***) ?
3. Як записати набраний текст в редакторі ***vi*** (***vim***) ?
4. Як вийти із редактора ***vi*** (***vim***) без збереження введених змін?
5. Що ви знаєте про команди глобального переміщення курсора в редакторі ***vi*** (***vim***) ?
6. Що ви знаєте про команди локального переміщення курсора в редакторі ***vi*** (***vim***) ?
7. Як вилучити фрагмент тексту в редакторі ***vi*** (***vim***) ?
8. Як скопіювати фрагмент тексту в редакторі ***vi*** (***vim***) ?
9. Як перенести фрагмент тексту із одного файла до іншого в редакторі ***vi*** (***vim***) ?
10. Які додаткові можливості надає візуальний режим в редакторі ***vi*** (***vim***) ?

ТЕМА № 3

Складання сценаріїв

Зміст теми: Знайомство із текстовою операційною оболонкою ***bash***, вивчення мови програмування оболонки ***bash*** та практичне складання найпростіших сценаріїв за допомогою редактора ***vi*** (***vim***).

Теоретичні відомості

3.1 Загальні відомості про текстові оболонки в *Linux*

Досить часто в процесі роботи з комп'ютером потрібно часто повторювати одні і ті ж команди *Linux*. Операційна система дозволяє записати необхідну послідовність команд в спеціальний файл, який називається сценарієм оболонки. Далі цей сценарій можна виконувати подібно виконанню звичайної команди Linux, набравши ім'я файла. Такий принцип організації файлів існує в MS DOS – це так звані командні файли.

Створювати сценарії надають спеціальні програми – оболонки, які виступають посередником між користувачем та операційною системою. Існують як текстові, так і графічні оболонки. Найбільш відомі в *Linux* текстові оболонки – ***bash, csh, tcsh, ksh, pdksh***.

Наприклад, необхідно виконати таку послідовність команд:

mkdir dir1
cp file1.txt /home/user/dir1
cd dir1

Для того, щоб оформити вказані команди у вигляді сценарію, необхідно спочатку вказати назву оболонки, в рамках якої вони будуть виконані. Якщо у сценарії використовуються лише команди операційної системи, як у нашому випадку, вибір оболонки не є суттєвим. Тому виберемо найбільш поширену оболонку, яка практично завжди є в більшості дистрибутивів *Linux* – оболонку ***bash***. Таким чином, першим рядком сценарію має бути такий запис:

#!/bin/bash

Цей рядок є по суті ознакою того, що даний файл відноситься до сценаріїв. А далі вже записуються команди, які мають бути виконані.

Після створення файла сценарію (наприклад, під іменем *run*) необхідно перевірити, чи надано право виконати цей сценарій даному користувачеві. Як правило, для того, щоб перетворити будь-який власний файл у виконуваний файл, рядовий користувач повинен надати собі таке право за допомогою команди *chmod:*

chmod u+x run

Далі сценарій можна запустити на виконання із командного рядка:

./run

або

bash run

Для того, щоб зробити сценарій більш універсальним (в нашому випадку – щоб сценарій був придатний для різних імен каталогів та файлів), використовуються позиційні параметри. За допомогою позиційних параметрів операційна система може передавати оболонці конкретні параметри (імена файлів, каталогів, змінних і т.д.) під час виклику сценарію у командному рядку або з іншого сценарію. Такі позиційні параметри мають спеціальні імена. Перший параметр зберігається у змінній з іменем *1*(один) і отримати його значення в сценарії можна за допомогою виразу *$1*. Другий параметр зберігається у змінній з іменем *2*(два) і т. д. Кількість одночасно використовуваних позиційних параметрів обмежена – не більше 9. Однак дозволяється за допомогою оператора

shift n

пересувати кожний позиційний параметр на *n* позицій вліво і відкидати *n* попередніх значень справа.

В нашому прикладі сценарій з використанням двох позиційних параметрів матиме вигляд:

```
#!/bin/bash
mkdir %1
cp %2 /home/user/%1
cd %1
```

Запуск на виконання цього сценарію матиме вигляд:

./run dir1 file1.txt

В оболонці можна також використовувати і власні команди оболонки. Суттєво збільшуються можливості сценаріїв при використанні спеціальних операторів, які утворюють мову програмування оболонки.

Розглянемо основні змінні та оператори мови програмування оболонки **bash.**

3.2 Змінні в сценаріях для *bash*

Мова програмування оболонки підтримує три основних типи змінних:

- змінні середовища;
- вбудовані змінні;
- змінні користувача.

Змінні середовища надаються системою, їх не потрібно визначати, але до них можна звертатись. Значення деяких із них, наприклад *PATH*, можна змінювати в програмі оболонки. Основні змінні середовища наведені в таблиці 3.1.

Вбудовані змінні також надаються системою, але їх не можна змінювати. До цих змінних відносяться, зокрема, такі:

$# - ряд позиційних параметрів, які передаються сценарію оболонки;

$0 - ім'я сценарію;

*$** - рядок зі всіма параметрами, які передаються сценарію оболонки під час його виклику.

Таблиця 3.1 – Змінні середовища

Змінна середовища	Значення
EDITOR	Редактор, який викликається за замовчуванням
HOME	Домашній каталог користувача
LOGNAME	Імя, під яким користувач зареєструвався в системі
MAIL	Шлях до поштової скриньки (файла e_mail) користувача
PATH	Список каталогів, що переглядаються системою при пошуку команди
PS1	Початкове запрошення оболонки ($ - для рядового користувача, # - для суперкористувача)
SHELL	Місцезнаходження оболонки
TERM	Тип термінала користувача

Змінні користувача визначаються самим користувачем під час написання сценарію оболонки. В своєму власному сценарії користувач може довільно використовувати та змінювати їх. На відміну від звичайних універсальних мов програмування, в командних інтерпретаторах не визначається тип змінної (як, наприклад, в мові Сі – int). Система самостійно "здогадується" про тип змінної за тим значенням, яке їй присвоюється. Це легко зробити, оскільки в сценарії можна оперувати тільки цілими числами або текстовими рядками. В такому випадку одна і та ж змінна в один момент часу може слугувати для збереження цілого числа, а через деякий час – для збереження рядка. Однак це не рекомендується робити.

Для присвоєння змінній користувача цілого значення або одного текстового слова досить записати оператор присвоєння в загальноприйнятій формі, наприклад

```
n=5
mas=0
str1=process
```

Характерною особливістю сценаріїв є те, що для доступу до значення змінної, перед її іменем ставиться знак долара ($). Наприклад, для того, щоб змінній *m* присвоїти значення змінної *n* необхідно записати

```
m=$n
```

Якщо текстовий рядок складається з кількох слів, тобто містить пропуски, тоді використовуються лапки, наприклад

str2='long text string'

Для присвоєння текстових рядків використовуються також і подвійні лапки. В цьому випадку забезпечується підстановка значень змінних всередині рядків. Наприклад, якщо записати

str2="long text string"

str3="Value of str2 is $str2"

тоді значенням змінної *str3* буде

Value of str2 is long text string

3.3 Програмування арифметичних виразів

Для програмування арифметичних виразів можна застосувати такі знаки операцій:

+ сума,

- різниця,

***** множення,

/ ділення,

% ділення з остачею.

Арифметичні вирази можна записати двома способами:

а) з використанням оператора ***let***;

б) з використанням оператора ***expr***.

Перший спосіб найбільш простий і зрозумілий, наприклад:

let *y=$a*$b-$c*

Оператор ***expr*** розглядає свої аргументи як арифметичний або логічний вираз. В такому випадку потрібно враховувати деякі додаткові особливості такого запису, наприклад, символи арифметичних операцій відділяти від операндів пропуском, символ операції множення а також і весь вираз брати в лапки:

y='expr $a '' $b '+' $c'*

3.4 Оператори введення і виведення

Для введення змінних з клавіатури використовується оператор ***read***. Наприклад, для введення змінних *var*, *var2*, *var3* в сценарії необхідно записати:

read *var1 var2 var3*

Для виведення повідомлень на екран дисплея використовується оператор ***echo***. Наприклад,

echo *This is message*

Якщо в сценарії необхідно вивести значення змінної, тоді використовуються подвійні лапки, наприклад

echo *"result is $y"*

Користуючись операторами введення та виведення можна написати найпростіший сценарій для обчислення арифметичних виразів:

```
#!/bin/bash
a=3
b=5
echo "Введіть значення змінної x"
read x
let y=($a+$b)*$x
echo "result is $y"
```

3.5 Порівняння виразів

Розглянемо порівняння чисел, рядків а також логічні і файлові операції порівняння.

3.5.1 Порівняння чисел

Для порівняння двох чисел можуть використовуватись такі операції:

a ***–eq*** *b* визначення рівності чисел *a* і *b*;
a ***–ne*** *b* визначення нерівності чисел *a* і *b*;
a ***–gt*** *b* визначення того, чи число *a* більше числа *b* ;
a ***–ge*** *b* визначення того, чи число *a* більше або дорівнює числу *b*;
a ***–lt*** *b* визначення того, чи число *a* менше числа *b*;
a ***–le*** *b* визначення того, чи число *a* менше або дорівнює числу *b*.

3.5.2 Порівняння рядків

Для порівняння двох рядків можуть використовуватись такі операції:

str1 **=** *str2* визначення рівності рядків *str1* і *str2*;
str1 ***!=*** *str2* визначення нерівності рядків *str1* і *str2*;
-n перевірка ненульової довжини рядка;
-z перевірка нульової довжини рядка.

3.5.3 Логічне порівняння

Логічні операції використовуються для порівняння виразів логічних операцій *NOT*, *AND* і *OR*:

! логічна операція *NOT* над логічним виразом;
-a логічна операція *AND* над двома логічними виразами;
-o логічна операція *OR* над двома логічними виразами.

3.5.4 Файлові операції порівняння

Такі операції можуть використовуватись для перевірки файлів:

-d перевірка того, чи є файл каталогом;
-f перевірка того, чи є файл звичайним файлом;
-r перевірка того, чи є право доступу для читання файла;
-w перевірка того, чи є право доступу для запису у файл;

-***x*** перевірка того, чи є право доступу для виконання файла;
-***s*** перевірка того, чи є файл з ненульовою довжиною.

3.6 Умовні оператори

Умовний оператор ***if*** дозволяє в залежності від виконання заданої умови *<виразу>* виконувати *<оператори 1>* або *<оператори 2>*.Формат цього оператора такий:

```
if <вираз>
  then <оператори 1>
else <оператори 2>
 fi
```

Умовні оператори можуть бути вкладеними, наприклад:

```
if <вираз 1>
   then <оператори 1>
else if <вираз 2>
   then <оператори 2>
else <оператори 3 >
 fi
 fi
```

Ключове слово ***fi*** означає закінчення одного умовного оператора, тому їх кількість у вкладеному умовному операторі повинна дорівнювати кількості ключових слів ***if***. Існує також форма скороченого запису вкладеного умовного оператора, коли достатньо лише одного ключового слова ***fi***:

```
if <вираз 1>
  then <оператори 1>
elif <вираз 2>
  then <оператори 2>
else <оператори 3>
 fi
```

3.7 Оператор-перемикач

Існує спеціальний оператор, який зручно використовувати при великій кількості розгалужень. Оформити такий запис дозволяє оператор ***case***, формат якого такий:

```
case var in
     S1) <оператори 1>;;
     S2) <оператори 2>;;
     S3) < оператори 3>;;
     *)  <оператори 4>;;
esac
```

В залежності від того, чи збігається значення змінної *var* із значенням ***S1, S2*** або ***S3***, виконуються відповідно <*оператори 1*>, <*оператори 2*> або <*оператори 3*>. Якщо вказаного збігу немає, тоді виконуються <*оператори 4*>.

3.8 Оператор циклу *for*

Оператор ***for*** має декілька форматів. Найпростіший формат цього оператор циклу, який використовує одновимірний список, має такий вигляд:

```
for  var in list
 do
      <оператори>
 done
```

В даному випадку <*оператори*> виконуються по одному разу для кожного значення змінної *var* із списку *list*. Приклад сценарію для знаходження суми елементів одновимірного масиву:

```
#!/bin/bash
mas=’3 7 12 5 8’
sum=0
for var in $mas
do
    let sum=$sum + $var
done
echo “result  is $sum”
```

Формат циклу ***for*** з використанням масивів дуже схожий на відповідний формат циклу в мові Сі.

Приклад сценарію з використанням циклу ***for*** для знаходження максимального значення серед елементів одновимірного масиву:

```
#!/bin/bash
mas[0]=3
mas[1]=7
mas[2]=12
mas[3]=5
mas[4]=8
max=mas[0]
for((i=0; i<5; i++))
do
   if  [ $m ax  –lt  ${mas[i]} ]
   then let max=${mas[i]}
fi
done
echo “result is $max”
```

3.9 Оператори циклу *while* та *until*

Оператор циклу ***while*** можна використовувати для повторного виконання <*операторів*> до тих пір, поки заданий <*вираз*> буде залишатись істинним:

while <*вираз*>
do
<*оператори*>
done

Можливо, що цикл не буде виконано жодного разу, якщо заданий <*вираз*> виявиться хибним з самого початку.

Приклад сценарію з використанням циклу ***for*** для знаходження максимального значення серед елементів двовимірного масиву, який вводиться із клавіатури:

```
#!/bin/bash
for((i=0; i<5; i++))
do
for((j=0; j<5; j++))
do
    read mas[i][j]
done
done
    max=mas[0]
for((i=0; i<5; i++))
do
for((j=0; j<5; j++))
do
if [ $max –lt ${mas[i][j]} ]
   then let max=${mas[i][j]}
fi
done
done
echo “result is $max”
```

Оператор циклу ***until*** можна використовувати для повторного виконання <*операторів*> до тих пір, поки заданий <*вираз*> буде залишатись хибним:

until <*вираз*>
do
<*оператори*>
done

3.10 Функції

Як і в мовах високого рівня, окремі частини сценаріїв можна записувати у вигляді функцій. Формат визначення функції такий:

```
func() {
  <оператори>
}
```

Виклик функції, якій передаються параметри *param1, param2, param3* :

```
func param1 param2 param3
```

Можна також передати параметри у вигляді одного рядка, наприклад, $@. Функція може інтерпретувати параметри за тими же принципами, за якими виконується інтерпретація позиційних параметрів, що передаються сценарію оболонки. Наприклад, для обчислення виразу

можна використати дві функції:

```
#! /bin/ bash
 a = 9
 b = 5
 c = 7
 d = 2
calc1() {
      let y= ($a+$b) /$1
     echo "Result is $y"
      }
 calc2() {
      let y = ($a+$b)*$1
     echo "Result is $y"
          }
echo "input x"
read x
if  [ $x  - eq 5 ]
      then calc1  c
      else calc2  d
fi
```

3.11 Робота з файлами

Використовуючи файлові операції порівняння, можна із заданого списка імен знаходити файли або каталоги, а також визначати їх права доступу. Наприклад:

```
#! /bin/ bash
if  [ -d  name1 ]
        then  echo " name1 is directory"
el  if  [ -f  name2 ]
        then   echo " name2  is file"
else  echo " name1 and name2  is not directory or file"
fi
if  [ -w  name2 ]
        then  echo " file has write permission"
else  echo " file has not write permission"
fi
```

В системних сценаріях *Linux* часто зустрічаються випадки, коли потрібно виконати задану послідовність операцій в залежності від інформації, яка записана у відповідних файлах.Складемо сценарій, в результаті виконання якого на екрані з'являється вікно системної програми годинника або калькулятора, якщо у файлі */home/user/Select.txt* змінній *Program* присвоєно значення відповідно "*XCLOCK*" або "*XCALC*". Звертаємо увагу, що цей сценарій може бути виконано лише в графічній оболонці ***X*** (детальніше графічний режим *Linux* розглядається в наступній лабораторній роботі).

```
#! /bin/ bash
. /home/user/Select.txt
if [ "$Program" = "XCLOCK" ]
        then  exec  xclock &
elif [ "$Program" ="XCALC" ]
        then  exec  xcalc &
fi
```

3.12 Рекомендована література з теми 3

[1, с.266-331], [5, с.451-464], [6, с.324-327].

Повний список літератури знаходиться на стор. 87.

Порядок виконання роботи

1. Скласти сценарій за отриманим завданням.
2. Викликати редактор ***vi*** (***vim***) та набрати текст сценарію.
3. Надати сценарію право на використання (права доступу задаються командою ***chmod***).
4. Виконати сценарій. В разі отримання повідомлення про помилки виправити їх.

Тестові запитання для самоперевірки з теми 3

1. Сценарій (командний файл) може містити лише команди операційної системи. (Так / Ні).
2. Оператор циклу "***while***" закінчується ключовим словом "***next***". (Так / Ні).
3. Умовний оператор "***if"*** завжди закінчується ключовим словом "***fi***". (Так / Ні).
4. ***bash*** – найпоширеніша графічна оболонка в *Linux*. (Так / Ні).
5. За допомогою позиційних параметрів можна передати оболонці імена файлів і каталогів. (Так / Ні).
6. В сценаріях можна використовувати числа з плаваючою комою. (Так / Ні).
7. В сценарії параметр $# не можна змінювати. (Так / Ні).
8. Команда "**echo**" використовується для виведення інформації на екран дисплея. (Так / Ні).
9. Операція "=" використовується для порівняння чисел. (Так / Ні).
10. Оператор циклу "***for***" використовується тільки для одновимірних масивів. (Так / Ні).
11. Оператор "***case***" можна використати, якщо можливі два розгалуження в системі. (Так / Ні).
12. В сценарії існує арифметична операція піднесення до степеня. (Так / Ні).
13. Оператор циклу "***unti*** " можна використовувати для повторного виконання операторів, поки умовний вираз буде залишатись істинним. (Так / Ні).
14. За допомогою файлових операцій порівняння можна визначити права доступу до каталогів. (Так / Ні).

Контрольні питання

1. Яка різниця між компіляторами та інтерпретаторами? До якого типу відноситься мова оболонки ***bash***?
2. Що ви знаєте про змінні в оболонці ***bash***?
3. Які Ви знаєте оператори введення-виведення?
4. Як відбувається порівняння чисел, рядків та файлів в сценаріях?
5. Які умовні оператори та оператори циклів Ви знаєте?

ТЕМА № 4

Система X Window

Зміст теми: Знайомство з графічним інтерфейсом користувача *Linux*, призначення менеджерів вікон, менеджерів дисплея, а також інших складових системи *X Window*.

Теоретичні відомості

4.1 Загальні відомості про систему *X Window*

Система *X Window* версії 11 (далі *X Window* або *X11*) – це бібліотека графічних програм, що використовується для створення графічного інтерфейсу користувача в операційній системі *Linux*.

В основу *X Window* покладена мережна архітектура типу "клієнт-сервер", проте її реалізація відрізняється від загальноприйнятих уявлень. В функції *X*-клієнта, який може знаходитись на будь-якій машині комп'ютерної мережі, входить обробка даних, тобто виконання будь-якої конкретної задачі. *X*-сервер приймає запити від користувача, відсилає їх *X*-клієнту, а потім відображає на дисплей користувача отримані відповіді від *X*-клієнта. Саме тому *X*-сервер повинен працювати на локальному комп'ютері, відображаючи інформацію користувачеві, в той час як *X*-клієнт може знаходитись на будь-якій машині, підключеній до мережі. До одного *X*-клієнта можуть надходити запити від різних користувачів, тобто *X*-клієнт стає спільно використовуваним ресурсом. Така клієнт-серверна архітектура дозволяє взаємодіяти програмам, які працюють під керуванням різних операційних систем і на різних апаратних платформах.

X Window орієнтована не тільки на мережі, в багатьох випадках *X*-сервер і *X*-клієнт знаходяться на одному комп'ютері. Наприклад, якщо запустити на виконання програму *xcalc*, тоді головна програма системи *X Window*, що знаходиться в каталозі */usr/X11R6/bin/X*, виконує роль *X*-сервера, а програма *xcalc* – роль X-клієнта. Є велика кількість різноманітних *X*-клієнтів, найважливішими з яких є такі:

- менеджери дисплея;
- менеджери вікон;
- робочі столи;
- стандартні програми (калькулятор, годинник і т.д.)

В *Linux*, як і в *Unix*, на одному комп'ютері є 7 умовних консолей (консоль – це сукупність "клавіатура + дисплей"). Після завантаження *Linux* користувач попадає в одну з консолей, а потім він може перемикатися між різними консолями і працювати в будь-якій з них. В перших шести консолях реалізований текстовий режим роботи, а в сьомій консолі працює *X*-сервер. Для переходів із текстової консолі в будь-яку

іншу консоль необхідно натиснути комбінацію клавіш <*Alt*> + <*Fz*> (де *z*=1,2,…,7), а для переходу з графічної консолі потрібно натиснути комбінацію клавіш <*Ctrl*> + <*Alt*> + <*Fz*>, де *z*=1,2,…,6.

4.2 Менеджери вікон

Менеджер вікон (інша назва – диспетчер вікон, адміністратор вікон) – це *X*-клієнт, який дає можливість керувати вікнами прикладних і системних програм: змінювати їх розміри, переміщати по екрану, згортати вікна в піктограму і виконувати багато інших функцій. На основі менеджерів вікон реалізовані найбільш складні програмні продукти графічного інтерфейсу *Linux* – робочі столи *GNOME* та *KDE*. Проте, менеджери вікон можуть працювати і самостійно, забезпечуючи мінімальний набір послуг з керування вікнами.

Свої функції менеджери вікон реалізують через меню. В більшості віконних адміністраторів відсутня панель головного меню зверху екрана: замість цього використовується висхідне меню, яке викликається після натиснення кнопки миші на вільному місці екрана. Відкривши меню, необхідно, не відпускаючи кнопку, перемістити курсор на потрібний пункт меню і лише потім відпустити кнопку.

Розглянемо можливості менеджерів вікон на прикладі менеджера *twm* (*tab window manager*). Завдяки малому розміру і зручності в використанні *twm* до цих пір широко використовується. За його допомогою легко вивчати основні функції менеджерів вікон: переміщення вікон програм, зміна розмірів вікон, згортання вікна в піктограму, виконання команд миші і клавіатури, запуск на виконання *X*-клієнтів. Головне меню менеджера *twm* має такий вигляд:

Twm
Iconify
Resize
Move
Raise
Lower

Focus
Unfocus
ShowIconmgr
HideIconmgr

Xterm

Kill
Delete

Restart
Exit

Наприклад, за допомогою пункту меню *Xterm* можна вивести на екран дисплея вікно текстового терміналу, в якому за допомогою команд операційної системи можна далі викликати будь-якого іншого *X*-клієнта, чи виконати ще яку-небудь дію. За допомогою пункту меню *Kill* можна зняти з виконання *X*-клієнта.

Важливою перевагою *twm* є можливість зміни системних функцій і додавання нових функцій у відповідності з потребами користувачів.

До найбільш відомих менеджерів вікон в *Linux* можна також віднести: *fvwm, fvwm2, fvwm95, kwin, Enlightenment, sawfish, mvm*.

Для робочого стола *KDE* основним менеджером вікон є *kwin*. Для робочого стола *GNOME* раніше базовим адміністратором вікон був *Enlightenment* (або просто *E*), а тепер його замінив *sawfish*.

4.3 Менеджери дисплея

Менеджер дисплея вкликається автоматично при завантаженні системи *X Window*. Цей *X*-клієнт відповідає за реєстрацію користувачів в системі. Таким чином, при введенні login та пароля, ми взаємодіємо саме із менеджером дисплея.

Як правило, менеджер дисплея працює лише в складі із більш потужними *X*-клієнтами – робочими столами. Для кожного робочого стола використовується свій менеджер дисплея: *gdm* для *GNOME* і *kdm* для *KDM*. Тип менеджера дисплея, який за замовчуванням запускається під час початкового завантаження, визначається сценарієм оболонки *prefdm*. Цей сценарій звертається до файла *desktop* із каталога */etc/sysconfig*. Знаючи мову програмування оболонки *bash*, корисно проаналізувати такий фрагмент сценарію *prefdm.*

```
preferred=
if [ -f /etc/sysconfig/desktop ]; then
        . /etc/sysconfig/desktop
        if [ "$DISPLAYMANAGER" = GNOME ]; then
                preferred=gdm
        elif [ "$DISPLAYMANAGER" = KDE ]; then
                preferred=kdm
        elif [ "$DISPLAYMANAGER" = XDM ]; then
              preferred=xdm
        fi
fi
```

Неважко зрозуміти із наведеного фрагмента сценарію, що вибір менеджера дисплея здійснюється в залежності від того, яке ключове слово міститься у файлі *desktop*: *GNOME*, *KDE* або *XDM*.

4.4. Рекомендована література з теми 4

[2, с.45-58], [5, с.57-60, 78-85, 92-94], [6, с.129-166], [7, с.259-274].

Повний список літератури знаходиться на стор. 87.

Порядок виконання роботи

1. В першій текстовій консолі викликати систему *X Window*, виконавши команду

X

2. Перейти із графічної консолі в другу текстову консоль, натиснувши клавіші *Ctrl+Alt+F2*, і виконати такі команди:

- ввести login та пароль;
- експортувати змінну середовища *DISPLAY*:

export DISPLAY=:0

- запустити на виконання програму *X*-клієнта:

xclock

3. Перейти в графічну консоль, натиснувши клавіші *Alt+F7*, і пересвідчитись в присутності там *X*-клієнта.

4. Перейти в третю текстову консоль, натиснувши клавіші *Ctrl+Alt+F3*, і виконати такі команди:

- ввести login та пароль;
- перевірити кількість активних консолів, виконавши команду

w

- експортувати змінну середовища DISPLAY:

export DISPLAY=:0

- запустити на виконання програму менеджера вікон:

twm

5. Знову перейти в графічну консоль і виконати такі дії:

- перетягнути мишею зображення годинника в інше місце екрана;
- натиснувши ліву клавішу миші, викликати на екран дисплея меню менеджера вікон ***twm*** і виконати пункт меню *Xterm* для виклику вікна термінала;
- у вікні термінала виконати команди для виклику ще одного *X*-клієнта:

xeyes

6. Перейти в другу текстову консоль, натиснувши клавіші *Ctrl+Alt+F2*, і зняти з виконання першого *X*-клієнта, натиснувши клавіші *Ctrl+C*.

- перевірити кількість активних консолів, виконавши команду

w

7. Перейти в графічну консоль і виконати такі дії:
- натиснувши ліву клавішу миші, викликати на екран дисплея меню менеджера вікон ***twm*** і виконати пункт меню *kill* ;
- активізувати вікно X-клієнта і, натиснувши ліву клавішу миші, зняти з виконання *X*-клієнта;
- закрити менеджер вікон ***twm***, виконавши пункт меню *exit*;
- закрити систему *X Window*, натиснувши одночасно клавіші *Ctrl+Alt+BackSpace*.

8. Знайти в каталозі /*etc* файли ***desktop*** і ***prefdm*** та визначити тип менеджера дисплея, який буде використовуватись на даному комп'ютері в графічному режимі *Linux*.

Тестові запитання для самоперевірки з теми 4

1. За допомогою системи *X Window* реалізується текстовий інтерфейс користувача. (Так / Ні).
2. В системі *X Window X*-сервер виконує запити *X*–клієнтів. (Так / Ні).
3. Віконні та дисплейні менеджери – приклади *X*–клієнтів. (Так / Ні).
4. Програми – ***xcalc*** і ***xclock*** приклади *X*–клієнтів. (Так / Ні).
5. Віконний менеджер виконує функцію ідентифікації користувача в системі. (Так / Ні).
6. ***kdm***, ***xdm***, ***gdm*** – приклади дисплейних менеджерів. (Так / Ні).
7. Дисплейний менеджер виконує функцію керування апаратною частиною дисплея. (Так / Ні).
8. ***Sawhich***, ***Enlightenment***, ***fwm*** – приклади віконних менеджерів. (Так / Ні).
9. Тип дисплейного менеджера визначається вмістом файла ***desktop***. (Так / Ні).
10. Віконний менеджер ***twm*** не може функціонувати окремо від робочих столів ***GNOME*** чи ***KDE***. (Так / Ні).
11. Система *X Window* може бути викликана із текстового режиму командою *startx.* (Так / Ні).
12. Система *X Window* працює на п'ятій консолі. (Так / Ні).

Контрольні питання

1. Що таке клієнт/сервер в ОС *Linux*?
2. Для чого можна застосовувати систему *X Window*?
3. Назвіть відомі Вам X-клієнти.
3. Скільки є консолей в ОС *Linux*? Які з них текстові, а яка графічна?
4. Як здійснюється перехід від одної консолі до іншої?
5. Як перевірити кількість активних консолей?
6. Яке призначення менеджерів вікон та менеджерів дисплея?

ТЕМА № 5

Робочі столи користувача

Зміст теми: Основні функції робочого стола, знайомство із робочими столами *GNOME* і *KDE*, вивчення графічної оболонки *Konqueror,* найпростіші операції настроювання екрана дисплея, настроювання панелі задач робочого стола.

Теоретичні відомості

5.1 Загальні відомості про робочі столи користувача в *Linux*

Робочий стіл – це графічний інтерфейс користувача з операційною системою. Робочий стіл дозволяє:

- переглядати в графічному режимі файлову систему і виконувати всі операції над файлами (копіювання, перейменування, вилучення тощо);
- розміщувати ярлики файлів і каталогів (папок) для швидкого доступу до них;
- розміщувати ярлики змінних дисків для їх монтування та доступу до їх вмісту;
- розміщувати ярлики принтерів для прискорення початку друку;
- зіставляти програми з файлами певного типу для їх автоматичного запуску.

Як правило, робочий стіл містить панель задач, на якій розташовуються кнопки ярликів, меню, програм і аплетів. Аплет – це невеличка вбудована програма для роботи і контролю за станом системи (наприклад: годинник, калькулятор і т.д.). Зазвичай панель задач реалізована у вигляді рядка в нижній частині екрана, хоча в більшості випадків передбачена можливість її переміщення до будь-якої сторони екрана.

Характерною особливістю робочих столів в *Linux,* яка відсутня у *Windows,* є можливість працювати з багатьма віртуальними робочими столами. На кожному із них може розташовуватись свій набір ярликів та відкритих вікон працюючих програм. Кожен віртуальний робочий стіл може мати свою гаму кольорів і рисунків, тобто ці столи функціонують незалежно один від одного.

Для переходу між віртуальними столами існує спеціальний перемикач – пейджер, ярлик якого розташовується на панелі. Можна також переміщувати вікна програм з одного столу на інший. До задач настроювання входить також зміна кількості віртуальних столів.

З точки зору ОС робочий стіл - це набір X-клієнтів та бібліотек для створення графічного інтерфейсу користувача із системою. Серед X-

клієнтів найважливішими є менеджери дисплеїв та менеджери вікон. Кожний робочий стіл має свій конкретний менеджер дисплея та конкретний менеджер вікна, тобто вони орієнтовані тільки на свій робочий стіл.

5.2 Робочий стіл KDE

Найбільш відомим робочим столом є *KDE* (*K Desktop Environment* – робочий стіл *K*). Авторські права на цей робочий стіл належать фірмі Trolltech.

Головна перевага *KDE* – забезпечення єдиного стандарту для всіх його складових частин і програм на основі об'єктно-орієнтованого підходу. В *KDE* всі елементи трактуються як об'єкти, до яких можна отримати доступ і виконати з ними певні дії.

KDE пропонує користувачеві весь спектр можливостей для керування зовнішнім видом і функціональними можливостями системи. Можна коректувати дуже багато речей – загальний фон робочого стола, вигляд кнопок та ярликів, вміст панелі або меню запуску програм та багато іншого. Для операцій настроювання робочого стола є спеціальний Центр Керування (*Control Center*), який дуже нагадує Панель керування у Microsoft Windows.

Завдяки використанню об'єктної графічної бібліотеки *Qt*, дотримується єдиний стиль при створенні кнопок, меню, перемикачів та інших атрибутів вікон. Наприклад, у всіх програмах *KDE* в правій частині меню розташована опція виклику довідки. Вся довідкова система витримана в єдиному форматі та стилі. У всіх програмах, кнопки, що виконують однакові функції, позначаються однаковими піктограмами. На перший погляд, це може здатись дрібницями. Але така однотипність сприяє швидкому засвоєнню нових програм, адже не завжди є можливість детального опису призначення кожного елементу.

Важливою перевагою *KDE* є повна підтримка всіма його програмами національних стандартів. Засоби багатомовної підтримки вбудовані безпосередньо в *KDE*, разом із документацією і файлами довідки.

На сьогоднішній день існує велика кількість програмних пакетів, спеціально створених для *KDE*. Бібліотеки *KDE* і *Qt*, які використовують мову C++, стають основою для розробки нових сучасних програм.

Найважливіше є те, що наявність однієї фірми-розробника означає високу ступінь відповідальності за свій програмний продукт, тобто високу надійність роботи *KDE*.

KDE – це великий проект, до складу якого в стандартному дистрибутиві входить більше сотні програм. Варто запам'ятати лише декілька із них, які є немов би "візитною карткою" *KDE*: менеджер дисплея *kdm,* менеджер вікна *kwin*, файловий менеджер *Konqueror*.

Необхідно відзначити, що в кожній новій версії *Linux* розширюються функції *Konqueror,* поступово перетворюючи його із простого файлового менеджера в потужну графічну оболонку.

5.3 Робочий стіл GNOME

Робочий стіл *GNOME* був розроблений дещо пізніше, як альтернатива *KDE*. Основною причиною появи *GNOME* було побоювання, що фірма Trolltech має право заборонити вільне розповсюдження свого *KDE*. І хоча цього поки що не сталось, все ж був створений міжнародний проект для розробки графічного інтерфейсу, який повністю підпадає під дію загальної ліцензії GPL (тобто для вільного розповсюдження). Про це свідчить і сама назва нового робочого столу: *GNOME* - GNU Network Obect Model Environment (GNU – GNU's Not Unix). Цей комплекс програм став по суті результатом співпраці багатьох програмістів-ентузіастів та фірм зі всього світу. З 2000 року координує всі роботи організація GNOME Foundation.

В *GNOME* основний наголос було зроблено на гнучкість, швидке оновлення компонентів, неповторність. Хоча цей робочий стіл більш динамічний, його компоненти, створені різними авторами, не завжди детально протестовані на предмет спільного користування. З однієї сторони, це полегшує налагодження *GNOME*, а з іншої – ускладнює сумісність різноманітних компонентів системи.

GNOME менш інтегрований порівняно з *KDE*, більше орієнтований на стандарт Open Source.

Якщо стиль *KDE* витримано в традиціях Microsoft Windows, то *GNOME* пропонує інтерфейс користувача, який основано на використанні менеджера вікон *Enlightenment*, (тепер його замінив *sawfish*). Для *GNOME* базовим менеджером дисплея є *gdm*, а графічною оболонкою, яка включає файловий менеджер - *Nautilus.*

З розвитком *KDE* і *GNOME* стало ясно, що вони будуть між собою конкурувати. Проте, і розробники *KDE*, і розробники *GNOME* прямують до того, щоб всі вказані розбіжності були непомітними для користувача і перехід від одного робочого столу до іншого не викликав великих проблем.

Як і *KDE*, робочий стіл *GNOME* пропонує такі основні функції:

- наявність головного меню для доступу до всіх програм та настроювання робочого столу;
- наявність панелі задач із кнопками для швидкого виклику необхідних програм або виконання налагоджувальних операцій;
- наявність піктограм і ярликів файлів та пристроїв;
- підтримка технології *drag-and-drop* для копіювання, створення посилань, переміщення або вилучення файлів і пристроїв;

- наявність пейджера для перемикання між різними віртуальними робочими столами.

Якщо на комп'ютері було інстальовано обидва розглянутих робочих столи, тоді для переходу від одного робочого стола до іншого необхідно спочатку ввести команду

switchdesk ,

потім вказати ім'я нового робочого столу і виконати перезавантаження *X Window*. Під час нового завантаження операційної системи завжди завантажується той робочий стіл, назва якого вказана в файлі "*/etc/sysconfig/desktop*". Змінити вміст цього файла може тільки адміністратор (детальніше про це в наступній лабораторній роботі).

5.4. Рекомендована література з теми 5

[2, с.250-255], [5, с.79-92], [4, с.75-129], [6, с.59-70, 95-122], [7, с.31-52], [8, с.60-97, 132-157].

Повний список літератури знаходиться на стор. 87.

Порядок виконання роботи

А. Найпростіші операції із компонентами робочого стола

1. Відкрити стартове меню, натиснувши лівою клавішою миші на кнопці стартового меню (K-меню) в лівому нижньому кутку екрана.
2. Переглянути меню і відповідні висхідні підменю.
3. Знайти серед наявних програм програми "*Научный калькулятор*" ("*Scientific Calculator*"} і "*Терминал*"("*Terminal*") та запустити їх на виконання.
4. Перейти у вікно програми "*Терминал*" і виконати такі операції: переміщення вікна по екрану, зміна розмірів вікна.
5. Вивчити контекстне меню програми "*Терминал*", яке викликається натисненням правої кнопки миші у заголовку вікна програми.
6. Перемістити у вікно програми "*Научный калькулятор*", натиснувши комбінацію клавіш "Alt-Tab".
7. Перевірити роботу програми "*Научный Калькулятор*" для кількох математичних операцій.
8. Перемістити у вікно програми "*Терминал*" і у вікні програми виконати команду для запуску програми "*xeyes*".
9. Перевірити різні варіанти взаємного розташування відкритих вікон програм і значків на робочому столі KDE:
   а) - викликати контекстне меню робочого стола KDE, натиснувши праву кнопку миші на вільній області стола;

- вибрати в контекстному меню пункт ”*Упорядочить значки*”: за іменами, за типом, за розміром.

б) - викликати контекстне меню робочого стола KDE, натиснувши праву кнопку миші на вільній області стола;
- вибрати в контекстному меню пункт ”*Окна* ”;
- далі вибрати пункт ”*расположить каскадом*”

10. Створити на робочому столі KDE нову папку:
- викликати контекстне меню робочого стола KDE, натиснувши праву кнопку миші на вільній області стола;
- вибрати в контекстному меню пункти “*Создать/Папку*”;
- ввести ім’я нового каталога.

11. Створити на робочому столі KDE новий файл:
- викликати контекстне меню робочого стола KDE, натиснувши праву кнопку миші на вільній області стола;
- вибрати в контекстному меню пункти “*Создать Файл /Текстовый файл*”;
- ввійти в текстовий редактор, двічі натиснувши на піктограмі створеного файла;
- ввести текст і зберегти його у файлі командою “*Файл/ Сохранить*” текстового редактора.

12. Створити на робочому столі KDE посилання на програму:
- викликати контекстне меню робочого стола KDE, натиснувши праву кнопку миші на вільній області стола;
- вибрати в контекстному меню команди “*Создать/Ссылка на приложение*”;
- в діалоговому вікні “*Свойства*” у вкладці “*Общие*” двічі натиснути клавішою миші на стандартному рисунку піктограми посилання на програму і вибрати із списку інший варіант рисунка піктограми;
- в діалоговому вікні “*Свойства*” у вкладці “*Права доступа*” установити потрібні права доступу до посилання на програму;
- в діалоговому вікні “*Свойства*” у вкладці “*Приложение*” в командному рядку вказати ім’я програми, яка буде виконуватись при активізації даного посилання (наприклад, “*xclock*”).

13. Запустити на виконання програму або виконати команду операційної системи різними способами:

а) натиснути лівою клавішою миші на піктограмі посилання на дану програму;

б) - натиснути одночасно дві клавіші <Alt> ,<F2>;
- в командному рядку ввести ім’я необхідної програми або команди;

в) - відкрити стартове меню, натиснувши лівою клавішою миші на кнопці стартового меню в лівому нижньому кутку екрана;
- вибрати пункти меню “*Выполнить программу*”;

- в командному рядку текстового терміналу ввести ім’я необхідної програми або команди.

14. Вилучити файл із робочого стола KDE за допомогою контекстного меню або клавіші “*Delete*” .
15. Відкрити вікно програми “*Корзина*” (“*Trash*”) і вивчити її властивості.
16. Перевірити основні операції програми “*Корзина*”: відновлення вилучених файлів та її очищення.
17. Закрити екран дисплея:
    - виконати команду в стартовому меню “*Закрити екран*” (“*Lock-Screen*”);
18. Відкрити екран дисплея:
    - ввести свій пароль (Примітка. При закритому екрані можна перейти на іншу консоль (консоль *n*), натиснувши клавіші <Alt>,<F*n*>).

Б. Вивчення графічної оболонки Konqueror (*версії 3.03*)

1. Запустити на виконання графічну оболонку *Konqueror*, натиснувши лівою клавішою миші на ярлику “*Домашній каталог*” (“*Home*”).
2. Вибрати пункт меню ”*Вид*” (“*View*”) головного меню і переглянути вміст домашнього каталога в різних масштабах та в різних виглядах:
   - у вигляді піктограм (“*Icon View*”);
   - у вигляді кількох колонок (“*MultiColumn View*”);
   - у вигляді дерева (“*Tree View*”);
   - у вигляді детального списку (“*Detail View*”);
   - у вигляді тексту (“*Text View*”).
3. Виконати упорядкування та сортування значків у вікні графічної оболонки за алфавітом, за типом, за розміром (пункти меню ”*Вид/*” (“*View/*”)).
4. Змінити фон робочої частини вікна графічної оболонки (пункти меню ”*Вид/Цвет фона*” (“*View/Background Color*”))
5. Перейти в пункт “*Настройка*” (“*Setting*”) головного меню і виконати операції настроювання:
   - показати/забрати головне меню (“*Show MenuBar*”);
   - показати/забрати панель інструментів (“*Show ToolBar*”);
   - показати/забрати додаткову панель інструментів (“*Show Extra-ToolBar*”);
6. Перейти в пункт “*Окно*” (“*Window*”) головного меню і виконати такі операції настроювання:
   - розбити вікно графічної оболонки на дві частини по вертикалі (команда “*Split View Left/Right*”);

- розбити вікно графічної оболонки на дві частини по горизонталі (команда "*Split View Top/Botton*");
- перейти в повноекранний режим роботи (команда "*Full-Screen Mode*");
- повернутись в попередній одновіконний режим (натиснувши на панелі інструментів на крайню справа кнопку "*Exit Full-Screen Mode*").

7. Перейти в різні каталоги файлової системи такими способами:
   а) за допомогою пункту "*Перейти*" ("*Go*") головного меню і команд:
   - "*Вверх*"("↑") : на рівень вище по ієрархічному дереву,
   - "*Вниз*"("←") : на попередній каталог ,
   - "*Вперед*"("→") : на наступний каталог;

   б) використанням відповідних кнопок панелі інструментів;
   в) із заданням імені каталогу в рядку адрес панелі інструментів;
   г) з використанням швидкого переходу в домашній каталог за допомогою команди "*Home URL*" пункту "*Перейти*" ("*Go*") головного меню.
8. Запустити на виконання програму, наприклад "*xcalc &*" різними способами:
   - через командний рядок (команда "*Tools/Run Command* ");
   - через виклик емулятора текстового терміналу внизу графічної оболонки (команда "*Window/Show Terminal Emulator*");
   - через виклик текстового термінала командою "*Tools/Open Terminal*");.
9. Перейти в пункт "*Инструменты*" ("*Tools*") головного меню і виконати пошук заданого файла, наприклад *Xsession (команда* "*Поиск файлов*" ("*Find files*")).
10. Скопіювати із знайденого каталогу файл *Xsession* в домашній каталог через буфер обміну (за допомогою буферу в пункті "*Правка*" ("*Edit*") головного меню графічної оболонки).
11. Перейти в домашній каталог і перейменувати файл *Xsession* в файл *Myfile.txt* (за допомогою команди "*Rename*" в пункті "*Правка*" ("*Edit*") головного меню графічної оболонки).
12. Створити посилання (ярлик) для файла *Myfile.txt* (за допомогою команди "*Create New/Link to Application*" в пункті "*Правка*" ("*Edit*") головного меню графічної оболонки).
13. За допомогою посилання відкрити файл *Myfile.txt* в найпростішому текстовому редакторі.
14. Внести зміни у вміст файла і зберегти його під новим ім'ям *Myfile2.txt*.
15. Вийти із текстового редактора.

16. Створити нову папку *Mydir1* (за допомогою команди "*Create New/Directory*" в пункті "*Редакторование*" ("*Edit*") головного меню графічної оболонки).
17. Перемістити файл *Myfile2.txt* в папку *Mydir1* за допомогою курсора миші.
18. Вилучити файл *Myfile.txt:*

    а) перемістити файл в корзину (за допомогою команди "*Move To Trash*" в пункті "*Правка*" ("*Edit*") головного меню графічної оболонки);

    б) вилучити файл із файлової системи з одночасним фізичним збереженням його на вінчестері (за допомогою команди "*Delete*" в пункті "*Правка*" ("*Edit*") головного меню графічної оболонки);

    в) фізично знищити файл на вінчестері (за допомогою команди "*Shred*" в пункті "*Правка*" ("*Edit*") головного меню графічної оболонки).
19. Закрити вікно графічної оболонки *Konqueror.*

В. Найпростіші операції настроювання екрана дисплея

1. В стартовому меню вибрати пункт "*Центр управления*" ("*Control Center*") і вибрати розділ "*Внешний вид и интерфейс*" ("*Look & Feel*").
2. Настроїти зовнішній вигляд та фон екрана дисплея:
   - вибрати підрозділ "*Внешний вид и интерфейс/Фон*" і в діалоговому вікні "*Фон*" ("*Background*") у вкладці "*Фон*" натиснути лівою клавішою миші на рядку "*Выбор цвета 1*" і вибрати бажану гаму кольорів;
   - натиснути лівою клавішою миші на рядку "*Выбор цвета 2*" і вибрати бажану гаму кольорів;
3. Змінити шпалери робочого стола:
   - в діалоговому вікні "*Фон*" вибрати вкладку "*Обои*" ("*Wallpaper*") в командному рядку "*Рисунок*" ввести ім'я графічного файла із бажаним рисунком шпалер (Примітка. Варіанти рисунків шпалер зберігаються у форматах *jpeg* та інших в таких каталогах: */usr/share/pixmaps/backgrounds/files, /usr/share/pixmaps/backgrounds/space, /usr/share/pixmaps/backgrounds/fPropaganda, /usr/share/wallpapers)*.
4. Настроїти екранну заставку дисплея:
   - вибрати підрозділ "*Внешний вид и интерфейс/Хранитель экрана*" і в діалоговому вікні "*Хранитель экрана*" ("*Screen saver*") встановити величину затримки зображення та сам рисунок зображення на екрані дисплея.
5. Настроїти параметри піктограм:

- вибрати підрозділ “*Внешний вид и интерфейс/Пиктограммы*” в діалоговому вікні “*Пиктограммы*” (“*Icons*”) і встановити бажані типи піктограм для каталогів, файлів та посилань.

6. Настроїти параметри віртуальних робочих столів, вибравши підрозділ “*Внешний вид и интерфейс/Рабочий стол*”, і в діалоговому вікні “*Рабочий стол*” (“*Desktop*”) встановити:
   - параметри елементів віртуальних робочих столів (вкладка “*Desktop*”);
   - кількість віртуальних робочих столів (вкладка “*Number of Desktop*”);
7. Настроїти параметри панелі задач, вибравши підрозділ “*Внешний вид и интерфейс/Панель*” і в діалоговому вікні “*Панель*” (“*Panel*”) виконати такі операції:
   - задати розмір, стиль і позицію панелі задач на робочому столі (вкладка “*Position*”);
   - задати параметри приховування панелі задач (вкладка “*Hiding*”);
   - задати фонові рисунки кнопок програм на панелі задач (вкладка “*Look & Feel* ”);
   - сконфігурувати стартове меню (вкладка “*Menus*”);

Г. Основні операції настроювання панелі задач робочого стола

1. Переглянути структуру панелі задач, розташованої у вигляді рядка в нижній частині екрана дисплея.
2. Запустити на виконання програму, кнопка якої знаходяться на панелі задач.
3. За допомогою пейджера викликати наступний віртуальний робочий стіл і запустити на виконання іншу програму (перехід між віртуальними робочими столами можна здійснити або курсором миші або одночасним натисненням клавіші клавіатури <Alt> та однієї із функціональних клавіш (<F1>…<F8>).
4. Додати на панель задач нові кнопки виклику програм:
   - знайти необхідну програму за допомогою стартового меню;
   - встановити на піктограму програми курсор миші, натиснути ліву клавішу миші і, не відпускаючи її, перетягнути піктограму на панель задач.
5. За допомогою контекстного меню панелі задач, яке викликається натисненням правої кнопки миші у вільній від кнопок області панелі задач, виконати такі операції:
   - додати на панель задач нові кнопки виклику програм (наприклад, програми “*Термінал*”);
   - змінити кількість кнопок пейджера.
   - вилучити із панелі задач кнопку виклику програм;

Примітка. Для інших версій робочого стола *KDE* та графічної оболонки *Konqueror* деякі команди можуть відрізнятись назвою або додатковими можливостями, у структуру меню можуть бути внесені зміни.

Тестові запитання для самоперевірки з теми 5

1. Команда *switchdesk* служить для переходу між робочими столами *KDE* і *GNOME.* (Так / Ні).
2. *Nautilus* – графічна оболонка робочого стола *KDE.* (Так / Ні).
3. *sawfish* – віконний менеджер робочого стола *GNOME.* (Так / Ні).
4. Програма *xclock* – приклад аплету. (Так / Ні).
5. Кількість віртуальних столів не повинна перевищувати 4-х. (Так / Ні).
6. Робочий стіл *GNOME* використовує об'єктну бібліотеку *Qt*. (Так / Ні).
7. *Konqueror* – файловий менеджер робочого стола *GNOME.* (Так / Ні).
8. Пейджер – це перемикач між окремими робочими столами. (Так / Ні).
9. Робочий стіл *KDE* підпадає під дію загальної ліцензії *GPL.* (Так / Ні).
10. На робочому столі дозволяється розміщувати ярлики принтерів для прискорення початку друку. (Так / Ні).

Контрольні питання

1. Які функції виконують робочі столи?
2. Які Ви помітили відмінності у графічному інтерфейсі користувача в операційних системах *Windows* та *Linux*?
3. Які особливості має робочий стіл *KDE*?
4. Які особливості має робочий стіл *GNOME*?
5. Як здійснити перехід від одного робочого столу до іншого?
6. Як здійснити настроювання параметрів робочих столів *KDE* та *GNOME*?

ТЕМА № 6

Основи адміністрування

Зміст теми: Вивчення основних операцій адміністрування в *Linux*: керування режимами завантаження ОС, редагування конфігураційних файлів, монтування файлових систем, додавання нових користувачів і груп, архівування та ущільнення файлів.

Теоретичні відомості

6.1 Основні задачі системного адміністратора

До основних задач системного адміністратора (суперкористувача) в *Linux* можна віднести:

- інсталяцію (установку) ОС;
- керування процесом завантаження ОС;
- установку режимів роботи ОС;
- редагування конфігураційних файлів;
- монтування і демонтування файлових систем;
- введення та вилучення користувачів ОС;
- оновлення програмного забезпечення;
- конфігурування ядра ОС;
- забезпечення надійного функціонування ОС;
- конфігурування комп'ютерної мережі.

Деякі із перерахованих задач далі будуть розглянуті більш детально. В цій лабораторній роботі будуть також розглянуті питання ущільнення та архівування файлів. Хоча для виконання цих задач на потрібні повноваження суперкористувача, однак бажано практично їх опанувати, враховуючи їх значимість в повсякденній роботі з ОС.

6.2 Стадії завантаження *Linux*

Після включення живлення комп'ютерів з архітектурою Intel відразу розпочинається самостійний процес перевірки працездатності основних пристроїв, відомий під назвою *POST* (*power-on self test* - самотестування при включенні живлення). При відсутності проблем із апаратурою далі вступає в роботу базова система введення-виведення або скорочено *BIOS* (*basic input/output system*). Ця програма зчитує параметри настроювання системи із спеціальної системної памяті *CMOS*. Системний *BIOS* і системна *CMOS*-память реалізовані апаратно у вигляді спеціальних мікросхем.

BIOS визначає, з якого пристрою (вінчестера, дискети, CD-ROM) буде здійснюватись завантаження ОС та перевіряє готовність цього

пристрою. Далі *BIOC* передає керування спеціальній програмі (початковому завантажувачу), яка власне і завантажує в оперативну память основні системні файли ОС. Для завантаження сучасних операційних систем сімейства *Linux* в ролі початкового завантажувача найчастіше виступає *LILO* (*Linux Loader*).

LILO насправді представляє собою сукупність кількох програм та файлів, основними з яких є такі:

- інсталятор *map*-файла (програма із файла */sbin/lilo*, в якому вказано місцезнаходження завантажувальних файлів *Linux*);

LILO може знаходитись в різних місцях:

- в головному завантажувальному запису (*MBR*) на вінчестері;
- в суперблоці кореневого розділу *Linux* на вінчестері;
- в завантажувальному секторі дискети.

Вибір розташування програми завантаження залежить від кількості ОС на вінчестері. Якщо на вінчестері інстальовано одночасно *Windows* і *Linux,* тоді зручно вибрати перший варіант. В цьому випадку можна вибирати ОС перед її завантаженням. Варто не забувати, що будь-яка переустановка *Windows* може знищити *LILO* в *MBR*.

Для керування процесом завантажування ОС корисно розібратись із конфігураційним файлом */etc/lilo.conf*, до якого звертається LILO. Типовий приклад файла */etc/lilo.conf :*

```
prompt
timeout=300
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
lba32

other=/dev/hda1
optional
label=windows

image=/boot/vmlinuz-2.4.18-14
label=linux
initrd=/boot/initrd-2.4.18-14.img
read-only
append="root=LABEL=/"

other=/dev/hda1
optional
label=WINDOWS
```

Параметри із конфігураційного файла */etc/lilo.conf* відповідають трьом розділам: загальному розділу, розділу образів *Linux* та розділу образів інших інстальованих ОС. Пояснення основних параметрів цього файла наведені в табл. 6.1.

Таблиця 6.1 - Основні параметри файла */etc/lilo.conf*

Команда	Опис команди
promt	Виводить на екран меню завантаження
timeout = 300	Часова затримка перед вибором ОС за замовчуванням (в мілісекундах)
default = linux	Назва ОС, яка завантажується за замовчуванням
Boot=/dev/hda	Розташування програми завантаження *LILO* (*/dev/hda – MBR*, або суперблок)
install=/boot/boot.b	Інсталяція вказаного файла (*/boot/boot.b*) як нового розташування програми завантажування *LILO*
label=windows *label=linux*	Задання мітки, як імені для вибору однієї із кількох інстальованих ОС
image=/boot/vmlinuz-2.4.18-14	Ім’я ядра *Linux*

Змінити деякі параметри процесу завантаження ОС можна не тільки безпосереднім редактування файла */etc/lilo.conf*, але також і використанням команди *lilo*. Наприклад, для того, щоб зробити *Windows* системою, яка завантажується за замовчуванням, достатньо в режимі суперкористувача набрати команду

lilo -D Windows .

При цьому ім’я ОС в даній команді повинно збігатись з іменем цієї ОС, яке було вказано в файлі */etc/lilo.conf.*

6.3 Керування режимами роботи *Linux*

Як тільки ядро системи завантажиться в оперативну память система *Linux* вже функціонує. Але ядро не взаємодіє безпосередньо із користувачем. Воно запускає на виконання програму *init* (правильніше сказати, створюється головний процес *init*; детальніше про процеси дивись останню лабораторну роботу). Ця програма відповідає за останні стадії завантаження ОС і визначає конкретний режим роботи *Linux*. З цією метою програма *init* звертається до спеціального файла */etc/inittab*. Типовий приклад цього файла має такий вигляд:

 # Default runlevel. The runlevels used by RHS are:
 #   0 - halt (Do NOT set initdefault to this)
 #   1 - Single user mode
 #   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
 #   3 - Full multiuser mode
 #   4 - unused
 #   5 - X11
 #   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:

 # System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

 # Things to run in every runlevel.
ud::once:/sbin/update

 # Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

 # When our UPS tells us power has failed, assume we have a few minutes
# of power left. Schedule a shutdown for 2 minutes from now.
 # This does, of course, assume you have powerd installed and your
 # UPS connected and working correctly.

pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

 # If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

 # Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4

5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

 # Run xdm in runlevel 5
 # xdm is now a separate service
x:5:respawn:/etc/X11/prefdm –nodaemon

Файл *inittab* розпочинається із пояснення можливих режимів роботи *Linux:*

- режим 0 – зупинка;
- режим 1 – для одного користувача;
- режим 2 – текстовий, для багатьох користувачів, без мережі;
- режим 3 – текстовий, для багатьох користувачів, з використанням мережі;
- режим 4 – не використовується;
- режим 5 – графічний режим X11;
- режим 6 – перезавантаження.

Режим роботи *Linux*, який буде використано за замовчуванням, тобто без втручання адміністратора, вказується в рядку *id:3:initdefault:* (в даному випадку режим 3). На практиці найчастіше використовуються лише режими 3 та 5, рідше режими 1 та 2, а режими 0 та 6 взагалі забороняється використовувати за замовчуванням.

Нагадаємо, що рядки, які відмічені знаком #, означають коментарій. Всі інші рядки файла *inittab* зумовлюють виконання відповідних дій, про які далі і йтиме мова.

Після рядка, який визначає режим роботи за замовчуванням, знаходяться рядки *s0-s2*, що запускають на виконання сценарії системної ініціалізації.

Для кожного вибраного режиму роботи системи виконується відповідний сценарій. Всі сценарії ініціалізації різних режимів роботи розташовані в каталозі */etc/rc.d*. Цей каталог, в свою чергу, містить декілька підкаталогів, по одному для кожного із режимів роботи (*rc0.d, ... , rc6.d*), підкаталог *init.d* та декілька інших підкаталогів. Імена всіх файлів, які розташовані в підкаталогах */etc/rc.d/rc*.d*, (замість символу * використовується число) розпочинаються буквами *s* або *k*, що позначають запуск (*start*) або завершення (*kill*) процесів. Наступне число в імені вказує на відносний порядок сценарію, далі – ім’я самого файла - це ім’я пов’язаного із ним головного сценарію (який, в свою чергу, знаходиться в підкаталозі */etc/rc.d/init.d*).

Рядки, які відмічені ідентифікаторами *pf* і *pr*, зацікавлять вас тільки в тому випадку, якщо робота джерела безперебійного живлення, встановленого на вашому комп’ютері, підтримуються спеціальною програмою *powerd.* Ця програма видає на екран дисплея відповідні попередження в разі аварійної ситуації із живленням комп’ютера.

Після рядка

 # Run gettys in standard runlevels

відбувається запуск програм *getty* у віртуальних консолях (віртуальних терміналах) для різних режимів роботи. Віртуальна консоль дозволяє отримати доступ до екранного вмісту відповідної консолі. Вони недоступні для читання рядовими користувачами, тому інші користувачі не можуть підглядати за вашим сеансом.

Нарешті, в самому кінці файла *inittab* відбувається запуск сценарію *prefdm* для графічного режиму роботи системи *Linux*.

6.4 Особливості завантаження системи *X Window*

Якщо у файлі *inittab* за замовчуванням вибрано режим роботи 5, тоді процес завантаження ОС продовжується. Від файла *inittab* "естафета" в подальшій роботі передається сценарію *prefdm* із каталогу */etc/X11*.

Існує файл *desktop*, в якому записано лише одне із трьох ключових слів: *GNOME, KDE* або *Anotherlevel*. В залежності від того, яке там записане слово, сценарій *prefdm* викликає менеджер дисплея : *gdm, kdm* або *xdm,* відповідно.

Основна функція менеджера дисплея - реєстрація користувача в системі і запуск відповідного сценарію *Xsession*, який визначає наступний сеанс роботи в середовищі *X Window*. Незважаючи на те, що кожному менеджеру дисплея відповідає свій сценарій *Xsession,* всі сценарії виконують такі дії:

- перенаправлення виведення повідомлень в спеціальний файл *{Home}/.xsession-errors*;
- завантаження файлів ресурсів користувачів із файла *{Home}/.Xresources*;
- запуск сценаріїв користувачів (при їх наявності);
- запуск сценарію *Xclients*.

Основне призначення сценарію *Xclients* із каталога */etc/X11/init* – запуск одного із робочих столів (як правило, *GNOME* або *KDE*) або менеджера вікон.

Завершення завантаження робочого стола або менеджера вікон свідчить про завершення завантаження ОС в цілому і готовність її до роботи в графічному режимі.

Якщо у файлі *inittab* за замовчуванням вибрано один із текстових режимів роботи, тоді процес *init* запускає одну із текстових оболонок (як правило, це оболонка *bash*) і процес завантаження ОС на цьому завершується. Для переходу *Linux* в графічний режим роботи досить виконати команду *startx*. Ця команда звертається до сценарію *xinitrc* із каталога */etc/X11/init*, який виконує по суті ту ж послідовність дій, що і сценарії *Xsession* та *Xclients.*

6.5 Конфігураційний файл *XF86Config*

Файл *XF86Config* - це основний конфігураційний файл системи *X Window*. Редагуючи цей файл, можна вирішити проблеми із настроюванням дисплея, клавіатури, пристрою “миша” а також і шрифтів. Розглянемо призначення основних розділів цього файла.

Розділ *Module*

Цей розділ служить для завантаження засобів підтримки сервера та візуалізації. Тут вказуються імена модулів, які завантажуються із каталога */usr/X11R6/lib/modules:*

```
Section "Module"
        Load  "dbe"
        Load  "extmod"
        Load  "fbdevhw"
        Load  "dri"
        Load  "glx"
        Load  "record"
        Load  "freetype"
       Load  "type1"
EndSection
```

Рядки

```
Load  "freetype"
Load  "type1"
```

свідчать, що система *X Window* підтримує векторні шрифти *TrueType* і *Type 1*.

Розділ *File*

Цей розділ вказує X-серверу місцезнаходження бази даних, яка містить палітри кольорів, розташування системних шрифтів та модулів:

```
Section "Files"
        RgbPath    "/usr/X11R6/lib/X11/rgb"
        FontPath   "unix/:7100"
        FontPath   "/usr/X11R6/lib/X11/fonts/local"
        FontPath   "/usr/X11R6/lib/X11/fonts/misc"
        FontPath   "/usr/X11R6/lib/X11/fonts/Type1"
        FontPath   "/usr/X11R6/lib/X11/fonts/TrueType"
EndSection
```

Рядок

RgbPath "/usr/X11R6/lib/X11/rgb"

вказує на місцезнаходження бази даних, яка містить палітри *RGB*-кольорів. Інші рядки вказують на каталоги із відповідними шрифтами.

Розділ *InputDevice*

Цей розділ показує, які типи клавіатури та "миші' використовуються:

```
Section "InputDevice"
        Identifier  "Keyboard0"
        Driver      "keyboard"
        Option        "XkbRules"  "xfree86"
        Option        "XkbModel" "pc102"
        Option        "XkbLayout"        "ru"
EndSection

Section "InputDevice"
        Identifier  "Mouse0"
        Driver      "mouse"
        Option      "Protocol" "PS/2"
        Option      "Device" "/dev/psaux"
        Option      "ZAxisMapping" "4 5"
        Option      "Emulate3Buttons" "yes"
EndSection

Section "InputDevice"
        Identifier     "Mouse1"
        Driver              "mouse"
        Option              "Device"            "/dev/input/mice"
        Option              "Protocol"          "IMPS/2"
        Option              "Emulate3Buttons"       "no"
        Option              "ZAxisMapping"          "4 5"
EndSection
```

Якщо в комп'ютері має використовуватись тип клавіатури розкладкою US (США), тоді необхідно записати:

Option "XkbModel" "us"

А якщо використовується розкладка клавіатури іншої країни, тоді має бути:

Option "XkbModel" "pc102"

Про підтримку російської розкладки клавіатури свідчить рядок:

Option "XkbLayout" "ru"

В даному випадку використовується два типи "миші":

- із протоколом PS/2 і можливістю емуляції трикнопкової "миші";

- із протоколом IMPS/2 і без емуляції трикнопкової “миші”.

Розділ *Monitor*

Цей розділ містить такі дані: ідентифікатор монітора, назву фірми-виробника, найменування моделі та діапазони частот горизонтальної та вертикальної розгортки:

```
Section "Monitor"
        Identifier   "Monitor0"
        VendorName   "Monitor Vendor"
        ModelName    "Monitor Model"
        HorizSync   30-61
        VertRefresh 50-120
        Option "dpms"
EndSection
```

Розділ *Device*

Цей розділ містить відомості про чіпсет відеокарти, драйвер для її підтримки, параметри настроювання частоти тактового генератора та інші важливі опції:

```
Section "Device"
        Identifier   "S3 Savage4"
        Driver       "savage"
        VendorName   "S3 Savage4"
        BoardName    "S3 Savage4"
EndSection
```

Розділ *Screen*

В цьому розділі описуються ідентифікатор екрана, назва відеокарти, тип монітора, параметри передачі кольорів, роздільна здатність екрана:

```
Section "Screen"
        Identifier   "Screen0"
        Device       "S3 Savage4"
        Monitor      "Monitor0"
        DefaultDepth        16

        Subsection "Display"
        Depth       16
        Modes       "800x600" "640x480"
        EndSubsection
EndSection
```

В даному випадку надаються режим глибини кольорів *High Color* (*DefaultDept 16*) за замовчуванням та два варіанти роздільної здатності екрана (800×600 і 640×480).

6.6 Монтування файлових систем

Linux підтримує багато різних файлових систем. Зрозуміло, що в першу чергу підтримується власна файлова система *ext2* (або її покращений варіант - *ext3*). Інші файлові системи необхідно спочатку підключити (змонтувати). Точніше кажучи, підключаються пристрої, на яких містяться відповідні файлові системи: вінчестери, гнучкі диски, компакт-диски. З однієї сторони, такий підхід, звичайно, дещо незручний, особливо це стосується дискет. Однак, ми маємо можливість в межах однієї операційної системи легко переходити від одної файлової системи до іншої. До речі, у *Windows* такої можливості немає.

Власна файлова система *ext2* (*ext3*) монтується до кореневого каталога / ще під час інсталяції *Linux*. Інші файлові системи із необхідністю монтуються до каталога */mount* за допомогою команди *mount*:

mount [-t type] [-o options] device mount-point

де *device* – пристрій, на якому знаходиться підключувана файлова система;

mount-point – точка монтування;

type – тип файлової системи (необов'язково);

options – додаткові опції файлової системи (необов'язково).

Якщо файлова система знаходиться на дискеті, тоді іменем пристрою буде *fd0* або *fd1*. Якщо ж підключається компакт-диск, то іменем прстрою найчастіше може бути *cdrom* або *cd0*.

Дещо складніше правильно записати ім'я пристрою при монтуванні вінчестерів. Як відомо, на вінчестері є розділи. У *Windows* такі розділи називаються логічними дисками і їм присвоюються букви латинського алфавіту. В *Linux* такого поділу на диски немає. Кожний розділ – це самостійний пристрій, що має своє стандартизоване ім'я. Це ім'я формується таким чином: перші дві букви – це *hd*, що, очевидно, означає *hard disk*. Наступною йде одна із чотирьох букв латинського алфавіту (*a,b,c,d*). Буква "*a*" присвоюється вінчестеру, який підключений як *Primary Master*, "*b*" – *Primary Slave*, "*c*" – *Secondary Master*, "*d*" – *Secondary Slave*. Таким чином, повна назва вінчестера, який підключений як *Primary Master* буде *hda*. Далі нумеруються розділи на вінчестері. *Primary*-розділів на жорсткому диску може бути всього чотири. Їм відповідають чотири можливих записи в *Master Boot Record* (*MBR*), а також номери з 1-го по 4-ий. Тобто, ці розділи маркуються в *Linux* як пристрої *hda1-hda4*. А якщо дисків буде більше, то вони будуть розташовані в *extended*–розділі і нумеруватись за порядком їх розташування, починаючи із *hda5*.

Точка монтування – це підкаталог каталога */mount*, до якого монтується пристрій із файловою системою. Цей підкаталог має бути створено ще до початку монтування; його назва може бути довільною.

В команді *mount* можна вказати конкретний тип підключуваної файлової системи:

- *dos* – файлова система *FAT16* (операційної системи *MSDOS*);
- *vfat* – файлова система *VFAT* (операційної системи *Windows 95*);
- *ntfs* – файлова система *NTFS* (операційної системи *Windows NT/2000/XP*)

Можливі значення додаткових опцій команди *mount* наведені в Табл. 6.1.

Таблиця 6.1 - Додаткові опції команди *mount*

Опція	Опис опції
1. -f	1. Імітація підключення файлової системи
2. -v	2. Детальний звіт про процес монтування
3. -r	3. Монтування із доступом тільки для читання
4. -w	4. Монтування із доступом для читання і запису
5. -a	5. Монтуються всі файлові системи, які перераховані у файлі */etc/fstab*

Монтування є досить відповідальною системною операцією, тому вона дозволена лише суперкористовачу.

Наприклад, якщо необхідно до каталогу */mnt/windows* підключити розділ *C* операційної системи *Windows 95*, тоді необхідно виконати команду *mount* в такому форматі:

mount -t vfat /dev/hda5 /mnt/windows .

А якщо необхідно до каталогу */mnt/floppy* підключити дискету, тоді можна виконати команду *mount* таким чином:

mount /dev/fd0 /mnt/floppy .

Інформація про всі змонтовані файлові системи зберігається в системному файлі */etc/fstab*.

Перед початком монтування нової файлової системи рекомендується переглянути раніше підключені файлові системи за допомогою команди *df*. Ця команда дає багато корисної додаткової інформації, зокрема обсяг вільного і зайнятого дискового простору.

Якщо потреба у деякій файловій системі відпала, тоді її можна відключити (розмонтувати) за допомогою команди *umount*, яка має два варіанти:

umount device,

або

umount mount-point.

Наприклад, демонтувати дискету можна таким чином:

umount /dev/fd0 ,

або

umount /mnt/floppy .

Монтувати файлові системи можна не тільки в текстовому режимі роботи ОС за допомогою команди *mount.*

В графічному режимі дуже зручно виконувати цю операцію, використовуючи панель керування (*Control Panel*) в робочому столі *KDE*, або спеціальну програму адміністрування *linuxconf* в робочому столі *GNOME*.

6.7 Додання нових користувачів і нових груп

Основним файлом при налагоджені і конфігуруванні облікових записів користувачів є файл */etc/passwd*. Це звичайний текстовий файл в форматі ASCII, який містить інформацію про кожного користувача в окремому рядку в таких семи полях, розділених двокрапкою:

поле 1 – ім′я користувача;

поле 2 – пароль користувача (символ X означає, що пароль у зашифрованому вигляді зберігається у спеціальному файлі */etc/shadow*);

поле 3 – ідентифікатор користувача (*User ID - UID*);

поле 4 – ідентифікатор групи (*Group ID - GID*);

поле 5 – коментар;

поле 6 – робочий каталог користувача;

поле 7 – використана оболонка за замовчуванням;

Приклад 1.

user1:X:501:500:OpenLinux User: /home/user:/bin/bash

Облікові записи нових корстувачів, крім паролів, можуть бути подані шляхом безпосереднього редагування файла */etc/passwd*, проте дещо зручніше використати спеціальну утиліту *useradd*.

Загальний формат цієї утиліти містить багато різноманітних опцій, але на практиці достатньо використовувати тільки основні з них:

useradd [-g group] [-d home_dir] [-s shell] [-p password] name,

де

group – ідентифікатор групи;

home_dir – робочий каталог користувача;

shell – ім′я текстової оболонки, яка використовується за замовчуванням;

password – пароль користувача;

name – ідентифікатор користувача.

Приклад 2.

Для того, щоб в файлі */etc/passwd* з′явився новий запис про користувача user, який збігається з наведеним записом в прикладі 1, необхідно використати утиліту *useradd* з такими опціями:

useradd -g 500 -d /home/user1 -s /bin/bash -p lab331 user1

Вилучити обліковий запис про користувача можна за допомогою утиліти *userdel*. Ця утиліта також має декілька опцій. В мінімальному варіанті для видалення облікового запису про користувача *user1* досить записати в командному рядку:

userdel user1

Якщо потрібно вилучити і робочий каталог цього користувача , тоді треба вказати опцію *r*:

userdel -r /home/user1

Для створення нового облікового запису групи користувачів, використовується утиліта *groupadd*, яка має такий формат:

groupadd -g group name

де *group* – ідентифікатор групи;

name – ідентифікатор користувача;

Приклад 3.

Для створення нової групи *502* і включення в неї користувача *user2* необхідно записати в командному рядку:

groupadd -g 502 user2

Один користувач може входити до складу кількох груп. Рекомендується створювати спочатку нову групу, а лише потім додавати до неї нових користувачів командою *useradd*.

Приклад 4.

Добавимо в нову групу 502 ще одного користувача *user3*:

useradd -d /home/user3 -p lab331 -g 502 user3

Облікові записи про групи зберігаються в конфігураційному файлі */etc/group*. Кожний рядок цього файла містить інформацію про одну групу у вигляді таких чотирьох полів, які розділено двокрапкою:

поле 1 – ім′я групи;

поле 2 – пароль користувача (зазвичай містить символ X або порожнє місце);

поле 3 – ідентифікатор групи (*GID*);

поле 4 – список користувачів, що належать даній групі.

Приклад 5.

Після виконання попередніх команд з прикладів 3 і 4 останній рядок файла */etc/group* матиме вигляд:

users: :502: user2, user3

Для виведення користувача із будь-якої групи потрібно в файлі */etc/group* вказати його ім′я із списку користувачів даної групи.

Варто зазначити, що робочі столи *KDE* і *GNOME* мають власні можливості щодо додання нових користувачів та груп. В графічному режимі ці операції виконуються швидше і не вимагають знання опцій розглянутих раніше команд.

6.8 Ущільнення і архівування файлів

Ущільнення файлів – це операція над файлами для отримання копій файлів, які займають менший розмір в порівнянні із початковими файлами.

Всі методи ущільнення даних можна розділити на два класи:

- з втратою інформації;
- без втрати інформації.

Ущільнення даних з втратою інформації (5-20%) застосовується для аудіо- та відеофайлів. Їх можна ущільнити в 10-15 разів (музика) або в 20-30 разів (фото- та відеоматеріали). Прикладом алгоритмів такого класу можуть служити алгоритми *JPEG*, *MPEG*.

Ущільнення даних без втрати інформації використовується для текстових та програмних файлів. Зрозуміло, що ступінь ущільнення таких файлів набагато менша (1,5-3 рази), однак є гарантія повного збереження початкової інформації. Такі методи базуються на вилученні природної надлишковості даних. До найбільш відомих алгоритмів такого класу відносяться:

- алгоритм Хафмана (символи, які найчастіше зустрічаються, мають більш короткий код);
- методи *RLE* (вилучення фрагментів, які повторюються);
- алгоритм *Zempel-Ziv* (дуже складний, зате більш ефективний).

В *Linux* за останнім алгоритмом працює програма *gzip*.

Архівування файлів – це операція над файлами з метою отримання резервних копій файлів для довготривалого зберігання. Найпростіший метод резервного копіювання в *Linux* – це копіювання на резервний носій даних необхідних файлів чи каталогів за допомогою утиліти *tar*. Звичайно, файли можна спочатку ущільнити, а потім створити їх резервні копії.

6.9 Рекомендована література з теми 6

[2, с.125-128, 237-250, 262-275], [5, с.61-76, 338-346, 408-433], [7, с.370-378].

Повний список літератури знаходиться на стор. 87.

Порядок виконання роботи

1. Здійснити завантаження ОС *Linux* в текстовому режимі.
2. Здійснити завантаження ОС *Linux* в графічному режимі.

3. Відкрити в текстовому редакторі та проаналізувати лістинги файлів *inittab, Xsession, Xclients.*
4. Провести аналіз апаратних характеристик комп'ютера на основі інформації із файла *XF86Config*.
5. Здійснити операції з монтування та розмонтування файлових систем (Табл. 6.2).
6. Здійснити операції з архівування та ущільнення файлів (Табл.6.3).
7. Здійснити операції з додання нових користувачів та груп.

Тестові запитання для самоперевірки з теми 6

1. Монтування і демонтування файлових систем здійснюється в режимі суперкористувача. (Так / Ні).
2. Архівування і ущільнення файлів у *Linux* – це тотожні операції. (Так / Ні).
3. На одному комп'ютері можна одночасно інсталювати *Windows* та *Linux*. (Так / Ні).
4. Користувач може входити до складу тільки однієї групи користувачів. (Так / Ні).
5. Файл *lilo.conf* дозволяється редактувати. (Так / Ні).
6. Режими роботи (текстовий/графічний) *Linux* задається в файлі *XF86Config*. (Так / Ні).
7. В сценарії із файла *prefdm* викликається один із файлових менеджерів для поточного сеансу роботи. (Так / Ні).
8. Утиліти *adduser* та *useradd* виконують однакові функції. (Так / Ні).
9. При архівуванні файлів в *Linux* зменшується їх розмір. (Так / Ні).
10. Роздільна здатність дисплея в *Linux* визначається в файлі *XF86Config*. (Так / Ні).
11. Структура вінчестера у *Linux* аналогічна структурі вінчестера у *Windows.* (Так / Ні).
12. Програма *LILO* використовується для перевірки працездатності апаратних засобів комп'ютера. (Так / Ні).
13. За допомогою програми *tar* можна зменшити розмір файла. (Так / Ні).
14. Під час завантаження системи *X Window* сценарій *Xclients* викликає сценарій *Xsession*. (Так / Ні).
15. В *Linux* дозволяється одночасно змонтувати кілька файлових систем. (Так / Ні).
16. За командою *startx* викликається текстова оболонка *bash*. (Так / Ні).
17. За допомогою програми *gzip* можна зменшити розмір файла. (Так / Ні).
18. Пароль користувачів зберігається у файлі *shadow*. (Так / Ні).
19. В *Linux* дозволяється змонтувати оптичні диски. (Так / Ні).

Таблиця 6.2 - Адміністрування файлових систем

Завдання	*Виконання*
1. Переглянути список змонтованих файлових систем та вільного дискового простору в блоках	*Ввести команду* **df** <Enter>
2. Переглянути список змонтованих файлових систем та вільного дискового простору в мега- та гігабайтах	*Ввести команду* **df -h** <Enter>
3. Визначити розмір в кілобайтах заданого файла (наприклад, *XF86Config*)	*Ввести команду* **du –h XF86Config** <Enter>
4. Визначити тип заданого файла (наприклад, *XF86Config*)	*Ввести команду* **file XF86Config** <Enter>
5. Змонтувати до каталогу *windows* файлову систему операційної системи *Windows*	*Ввести команду* **mount /dev/hda5 /mnt/windos** <Enter>
6. Перейти в папку */mnt/windows* (в кореневий каталог диска C)	*Ввести команду* **cd /mnt/windows** <Enter>
7. Переглянути список файлів в кореневому каталозі диска C	*Ввести команду* **ls** <Enter>
8. Скопіювати файл *autoexec.bat* з диску C: в свій домашній каталог *Linux*	*Ввести команду* **cp /mnt/windows/autoexec.bat /home/user** <Enter>
9. Змонтувати до каталогу *floppy* файлову систему дискети	*Ввести команду* **mount /dev/fd0 /mnt/floppy** <Enter>
10. Скопіювати із дискети заданий файл (наприклад, *file.txt*) в свій домашній каталог *Linux*	*Ввести команду* **cp /mnt/floppy/file.txt /home/user** <Enter>
11. Скопіювати всі текстові файли із свого домашнього каталогу *Linux* на дискету	*Ввести команду* **cp /home/user/*.txt /mnt/floppy** <Enter>

Примітка 1. Монтування файлових систем виконується лише в режимі суперкористувача.

Примітка 2. В каталозі **/mnt** мають бути заздалегідь підготовлені каталоги **windows** та **floppy,** інакше їх необхідно створити.

Таблиця 6.3 - Ущільнення і архівування файлів в текстовому режимі

Завдання	*Виконання*
1. Із початкового файла *file1.txt* створити ущільнений файл *file1.txt.gz*	*Ввести команду* **gzip file1.txt** <Enter>
2. Видати довідку про ступінь ущільнення файла *file1.txt.gz*	*Ввести команду* **gzip -l file1.txt.gz** <Enter>
3. Відновити початковий файл *file1.txt*	*Ввести команду* **gunzip file1.txt.gz** <Enter>
4. Створити новий файл *fff.gz* і занести в нього ущільнену копію файла *file2.txt* із збереженням початкового файла	*Ввести команду* **gzip –c file2.txt > fff.gz** <Enter>
5. Дописати у файл *fff.gz* ущільнену копію файла *file3.txt* із збереженням файла *file3.txt*	*Ввести команду* **gzip –c file3.txt >> fff.gz** <Enter>
6. Із файла *fff.gz* створити новий файл *file.txt*, який містить початкові дані всіх раніше ущільнених файлів	*Ввести команду* **gunzip –c fff.gz > file.txt** <Enter>
7. Створити архів *archiv.tar* із файлів *file4.txt* та *file5.txt*	*Ввести команду* **tar cf archiv.tar file4.txt file5.txt** <Enter>
8. Ущільнити архів *archiv.tar*	*Ввести команду* **gzip archiv.tar** <Enter>
9. Відновити архів *archiv.tar*	*Ввести команду* **gunzip archiv.tar.gz** <Enter>
10. Створити архів *archiv2.tar* із всіх файлів каталога */home/user/dir1*	*Ввести команду* **tar –cvf archiv2.tar /home/user/dir1** <Enter>
11. Створити архів *archiv3.tar* із файлів *file6.txt* та *file7.txt* одночасно з їх ущільненням	*Ввести команду* **tar czf archiv3.tar.gz file6.txt file7.txt** <Enter>
12. Відновити початкові файли із ущільненого архіву *archiv3.tar* і створити новий архів *aaa.tar*	*Ввести команду* **gunzip -c archiv3.tar.gz aaa.tar** <Enter>

Контрольні питання

1. Які основні етапи завантаження ОС *Linux*?
2. Яке призначення файлів *inittab, Xsession, Xclients*?
3. Як змінити режим роботи ОС *Linux* (текстовий або графічний) за замовчуванням?
4. В якому файлі відбувається конфігурування системи *X Window*?
5. Які файлові системи можна підключити (змонтувати) в ОС *Linux*?
6. Як відбувається монтування та розмонтування файлових систем?
7. Як додати нових користувачів і нові групи?
   В чому полягає різниця між архівуванням та ущільненням файлів в ОС *Linux*?

ТЕМА №7

Процеси і роботи в *Linux*

Зміст теми: Керування процесами і роботами в *Linux*: створення, зупинка, відновлення, завершення. Пріоритети процесів. Активні, фонові та відкладені процеси. Демони.

Теоретичні відомості

7.1 Основні поняття про процеси і роботи

Процесом у *Linux* і *Unix* називається програма або команда, що виконується.

Linux – багатозадачна ОС і в ній одночасно може виконуватись кілька процесів. Для того, щоб їх розрізняти, кожному з них присвоюється персональний ідентифікатор (PID – Process IDentificator). Для виведення на екран списку всіх процесів, що існують в даний момент в системі, використовується команда *ps.* В результаті виконання цієї команди на екран дисплея може бути, наприклад, видана така інформація:

PID	*TTY*	*TIME*	*CMD*
701	*pts/0*	*1:16:00*	*bash*
2403	*pts/0*	*0:18:00*	*bs*

TTY – термінал з якого був запущений процес.
TIME – час, протягом якого він виконується.
CMD – назва програми, яка буде працювати у запущеному процесі.

Найперший процес, запущений системою, називається – *init*. Його PID=1. Він є головним батьківським процесом всіх інших процесів. Кожен новий процес повинний обов'язково мати свій батьківський процес. Наприклад, процес текстової оболонки *bash*, який був породжений безпосередньо від процесу *init*, у свою чергу, буде батьківським процесом для всіх інших процесів, створюваних користувачем під час сеансу. Особливістю *Unix* та *Linux* є те, що для кожного нового процесу створюється дублікат батьківського процесу. Такий механізм процесів називається клонуванням**.**

Кожний новостворений процес отримує три уже відкритих файли:

- *stdin* – для вхідних даних,
- *stdout* – для вихідних даних,
- *stderr* – для повідомлень про помилки.

Процеси можуть функціонувати в двох режимах: системному і користувача. Робота в системному режимі означає виконання процесом системних викликів. Він найбільш важливий, тому що виконується обробка переривань, викликаних зовнішніми сигналами і системними викликами, а також керуванням доступом до диска, розподіл додаткової динамічної

пам'яті й інших ресурсів системи. Процес функціонує в режимі користувача, коли виконується програма користувача.

Окрім поняття “процес” використовується також і поняття “робота”. Хоча їх практичний смисл досить близький, проте є деякі відмінності. По-перше, якщо процес характеризується ідентифікатором PID, то робота – номером, що позначається символом %. По-друге, для керування роботами існують спеціальні команди, про які детальніше буде сказано далі. По-третє, до робіт відноситься лише частина наявних в системі процесів, зокрема процеси користувачів.

Варто також зазначити, керування роботами здійснюється власними засобами текстових оболонок *Linux*. Наприклад, оболонка *bash* має команду *jobs* для керування роботами. Для перегляду наявних в системі робіт вказана команда має такий формат:

jobs –<опції>,

де *<опції>* вказують на ідентифікатори типу процесів (робіт).

Отримати інформацію про відповідність між номером роботи і ідентифікатором процесу можна за допомогою команди

jobs –l

7.2 Активні, фонові та відкладені процеси (роботи)

Процеси (роботи) бувають активними (привілейованими), фоновими та відкладеними. В кожний момент часу може бути лише один активний процес. Активним є такий процес, з яким безпосередньо взаємодіє користувач, тобто тільки цей процес отримує інформацію з клавіатури і посилає результати на ваш екран (як кажуть, виконується на передньому плані). З іншого боку, фонові процеси не одержують інформації з термінала, у загальному випадку вони спокійно виконуються, не вимагаючи потреби в спілкуванні з користувачем. Деякі фонові процеси виконуються протягом великого проміжку часу і не здійснюють нічого зовні цікавого. Компіляція програм або ущільнення файлів - приклади таких процесів. Немає потреби чекати, коли ці процеси закінчаться. Їх можна просто запустити у фоні. Поки вони там виконуються, ви можете займатися іншими програмами. Але потрібно знати основні особливості фонової обробки:

- фоновий процес не допускає введення з клавіатури;
- будь-яке виведення від фонового процесу на екран руйнує все, що ви в цей момент ввели з клавіатури;
- при запуску великої кількості фонових процесів можна перевантажити систему.

Процеси можуть бути також відкладені. Відкладений процес - це процес, що у даний момент не виконується і тимчасово зупинений. Після того, як ви призупинили виконання процесу, надалі ви можете його продовжити як на передньому плані, так і в фоні. Поновлення

призупиненого процесу не змінить його стану - при поновленні він почнеться з того місця, на якому сталась зупинка.

Майте на увазі, що призупинення процесу - це не переривання процесу. Коли ви перериваєте процес, який виконується, натискаючи клавіші переривання (зазвичай, це <*CTRL*><*C*>), то процес знищується назавжди. А якщо процес знищено, то немає іншого способу відновити його, як знову запустити спочатку, використовуючи колишню команду. Зауважимо також, що деякі програми можуть перехоплювати переривання, тоді натискання клавіш <*CTRL*><*C*> не приведе до негайного припинення процесу. Це дозволить програмі виконати необхідні операції акуратного завершення. Деякі програми взагалі не дозволять вам їх перервати.

7.3 Створення процесу

Процес породжується за допомогою системного виклику *fork()*. При цьому виклику відбувається перевірка на наявність вільної пам'яті, доступної для розміщення нового процесу. Якщо необхідна пам'ять доступна, то створюється процес-нащадок поточного процесу, що представляє собою точну копію процесу, що викликається При цьому в таблиці процесів для нового процесу будується відповідна структура. Нова структура створюється також у таблиці користувача. При цьому всі змінні ініціалізуються нулями. Цьому процесу присвоюється новий унікальний ідентифікатор, а ідентифікатор батьківського процесу запам'ятовується в блоці керування процесом.

Процеси, що виконують різні програми, утворюються завдяки застосуванню наявних у стандартній бібліотеці *Linux* функцій “сімейства *exec*”: *execl, execlp, execle, execv, execve, execvp*. Ці функції відрізняються форматом виклику, але в остаточному підсумку виконують одну задачу: заміщають всередині поточного процесу код, що виконується, на код, що міститься в зазначеному файлі. Цей файл може бути не тільки двійковим файлом *Linux*, що виконується (аналог файла **.exe* в системі *Windows*), але і сценарієм командного інтерпретатора, і двійковим файлом іншого формату (наприклад, класом *java*, що виконується файлом *DOS*).

Таким чином, якщо операція запуску програми у *DOS* і *Windows* виконується як єдине ціле, то у *Linux* (і в *Unix* взагалі) розділена на два етапи: спочатку здійснюється запуск, а потім визначається, яка програма буде працювати. Чи є в цьому сенс і чи не занадто великі додаткові витрати? Адже створення копії процесу передбачає копіювання дуже значного обсягу інформації.

Сенс у даному підході є суттєвий. Дуже часто програма повинна зробити деякі дії ще до того, як почнеться власне її виконання. Наприклад, створити непоіменований канал для спілкування з іншими процесами. Реалізується це дуже просто: спочатку “відмежовуються” процеси, потім

виконується системний виклик *pipe()* для створення каналу - і тільки після цього запускається програма із функції *exec()*.

Стосовно додаткових витрат, то вони найчастіше виявляються надто малими: при створенні копії процесу його індивідуальні дані фізично нікуди не копіюються. Замість цього використовується техніка, відома за назвою *copy-on-write* (копіювання при записі): сторінки даних обох процесів особливим чином позначаються, і тільки тоді, коли один процес намагається змінити вміст будь-якої своєї сторінки, він дублюється. Щоб створити новий активний процес рядовому користувачеві зовсім не обов'язково знати про всі ці системні виклики. Йому достатньо лише ініціювати виконання команди або програми, набравши в командному рядку відповідне ім'я файла, або натиснувши на відповідну піктограму на робочому столі. Приклад запуску годинника:

xclock

Для запуску процесу у фоновому режимі необхідно після імені команди чи імені файла ввести символ & і лише потім натиснути на клавішу <Enter>. Приклад запуску годинника в фоновому режимі:

xclock &

Щоб призупинити активний процес необхідно одночасно натиснути клавіші <Ctrl><Z>. Поки процес зупинено на нього не витрачається час процесора. Але ви завжди можете відновити процес, причому як на передньому плані, так і в фоні. Для поновлення процесу в активному режимі використовується команда *fg* (*"foreground"* - передній план), а в фоновому режимі – команда *bg* ("*background*" - задній план).

Ще одне зауваження. Команди *fg* і *bg* звичайно переводять на передній план чи у фоновий режим процеси, що були зупинені останніми (що визначається символом "+" після номера роботи при використанні команди *jobs*).

Якщо ви виконуєте багато процесів (робіт) одночасно, ви можете перевести на передній план чи, навпаки, у фоновий режим конкретну роботу із заданням її номера як аргумента команд *fg* або *bg*:

fg %2 (переведення на передній план роботи номер 2),

bg %3 (переведення у фон роботи номер 3).

Для цих команд не можна використовувати ідентифікатори процесів. Крім того, використання тільки номерів робіт, наприклад

%2

еквівалентно команді

fg %2

Тип процесу (роботи) можна легко визначити за допомогою команди *jobs*. Наприклад, якщо після виконання цієї команди ми отримали на екрані дисплея інформацію

```
[1]-  Running    xclock
[2]-  Running    xeyes  &
[3]+  Stopped    xcalc
```

то це означає, що перший процес (робота) в даний момент є активним, другий процес (робота) – фоновим, а третій процес (робота) – відкладений.

Можна також отримати список процесів (робіт) лише одного типу, вказавши відповідну опцію в команді *jobs*:

jobs –r - видача лише активних процесів (робіт),

jobs –s - видача лише відкладених процесів (робіт).

Пам'ятайте, що керування роботами, це властивість оболонки операційної системи. Команди *fg*, *bg* і *jobs* - внутрішні команди оболонки *bash*. Якщо з якоїсь причини ви використовуєте текстову оболонку, що не підтримує керування роботами, там ви не знайдете цих команд. На додаток до цього, є деякі аспекти керування роботами, що розрізняються в оболонках *bash* і *tcsh*. Деякі оболонки не здатні керувати роботами, хоча більшість оболонок *Linux* мають таку можливість.

7.4 Пріоритети процесів

Зазвичай при запуску всі процеси користувачів мають однаковий пріоритет, рівний 10, що дає змогу операційній системі рівномірно розподіляти між ними процесорний час. Пріоритет можна змінити за допомогою команди

nice –число процес,

де *число* – величина, на яку зменшується початкове значення пріоритету для *процесу*.

Пріоритет активного процесу з ідентифікатором PID можна змінити командою

renice –число PID .

Команда *nice* не завжди знижує пріоритет. Для його підвищення необхідно вказати від'ємне *число*.

Варто відзначити, що лише суперкористувач має таке право.

7.5 Завершення процесів

Для примусового завершення активного і фонових процесів використовуються різні способи. Як вже раніше відмічалось, активний процес можна ліквідувати, натиснувши клавіші <CTRL><C> або клавішу DEL.

Для завершення фонового процесу використовується команда *kill*, яка має кілька форматів:

kill PID

kill -signal PID

kill %n

Ця команда може брати як аргумент номер роботи, або ідентифікатор процесу. Наприклад, для завершення процесу із ідентифікатором *PID=237* необхідно виконати команду

kill 237,

а для завершення роботи із номером 20 необхідно виконати команду

kill %20

Для перевірки ліквідації вказаного процесу, можна виконати команду

ps

в результаті чого на екрані дисплея отримаємо відповідь:

237 Terminated

А якщо виконати команду

jobs

тоді теж одержимо аналогічне підтвердження:

[20]+ Terminated

Ключ "*-signal*" змушує команду *kill* виконати ряд додаткових послуг, тобто послати процесу певний сигнал. Може бути послано понад 20 сигналів, кожний з яких має свій номер.

При виході користувача із системи, *Linux* посилає всім його процесам сигнал 1, що змушує всі процеси завершити роботу. За замовчуванням усім процесам посилається сигнал 15.

Якщо ввести команду *kill 0*, то можна ліквідувати всі фонові процеси.

Якщо який-небудь процес "завис", тоді потрібно перейти до іншої консолі, і з її допомогою ввести команду *kill* для ліквідації "завислого" процесу.

Гарантовано можна знищити процес за допомогою сигналу 9, наприклад:

kill -9 125

Звичайний користувач має право припиняти тільки процеси, запущені з його термінала. Для завершення процесу використовується системний виклик *exit()*, при якому звільняються усі використовувані ресурси, зокрема такі, як пам'ять і структури таблиць ядра. Крім того, завершуються і процеси-нащадки, породжені даним процесом.

Потім з пам'яті вилучаються сегменти коду і даних, а сам процес переходить у стан "зомбі" (у полі *Stat* такі процеси позначаються буквою *"Z"*). "Зомбі" не займає процесорного часу, але рядок у таблиці процесів залишається, і відповідні структури ядра не звільняються.

Якщо батьківський процес з якоїсь причини завершиться раніше дочірнього, останній стає "сиротою" *(orphaned process).* "Осиротілий" "зомбі" на короткий час стає нащадком *init*, після чого вже остаточно "помирає".

Також, процес може впасти в "сон", що не вдається перервати (у полі *Stat* це позначається буквою "*D*"). Процес, що знаходиться в такому стані, не реагує на системні запити і може бути знищений тільки перезавантаженням системи.

7.6 Корисна інформація про фонові процеси

Коли текстова оболонка типу *bash* завершує роботу, вона посилає в усі породжені нею процеси сигнал “відбій”. Якщо процес виконується у фоновому режимі, цей сигнал часто знищує його, що в більшості випадків небажано. Якщо Ви маєте намір запустити у фоновому режимі програму, яка повинна буде працювати і навіть після завершення батьківського процесу (звичайні фонові процеси при цьому завершуються), її потрібно запускати як глибокий фоновий процес командою *nohup*. Така команда має формат:

nohup команда &

Подібний запуск змушує зазначену аргументом команду ігнорувати сигнал відбою. Команда *nohup* має і побічний ефект: до значення *nice* додається п’ять. Уся вихідна інформація, яка генерує процес, якщо стандартний файл виведення і стандартний файл помилок не перепризначені, розміщуються у файлі *nohup.out*.

Демон (від англійського *demon* чи *daemon*) - це фоновий процес, що виконує системну задачу, непомітно для користувача і доповнює операційну систему будь-яким спеціальним сервісом. Ця програма не викликається користувачем у явній формі, а спокійно очікує в пам'яті певної події. У повній відповідності з пануючим у UNIX принципом модульності демони є програмами, а не частинами ядра. Багато демонів запускаються під час початкового завантаження і продовжують працювати увесь час, поки система включена. Інші демони запускаються при необхідності і працюють стільки, скільки передбачено їх функціями.

Демони можна знайти за допомогою команди

ps -ax

До основних демонів можна віднести *init, inetd* та *cron.*

Демон *init* - це перший процес, що запускається після початкового завантаження системи, і є предком майже всіх процесів, його *PID* завжди дорівнює 1),

Демон *inetd* керує іншими демонами. Раніше всі демони запускалися під час початкового завантаження операційної системи, і працювали безупинно (точніше, блокувалися при чеканні роботи). Згодом у систему вводилися все нові і нові демони. Їх стало стільки багато, що почали з’являтися проблеми з продуктивністю роботи системи. У відповідь фахівці BSD розробили *inetd* - демон відповідальний за запуск інших демонів по необхідності. Він запускає демони-клієнти, коли для них є робота, а після виконання задачі дозволяє їм тихо завершитись. Для того щоб працювати під керуванням *inetd*, клієнти повинні дотримуватись особливих правил: якщо конфігурація операційної системи із самого початку не передбачала використання *inetd*, то для його введення в систему необхідно модифікувати багато інших програм. Багато демонів можуть використовуватися або традиційним способом (тобто вони

запускаються однократно і працюють до вимикання системи), або під контролем *inetd*.

Демон *cron* відповідає за виконання команд за графіком. Він обробляє файли з розкладом задач (*cron*-файли), створені як користувачами, так і адміністратором. Демон *cron* часто використовують в адміністративних задачах, таких, як керування обліковими файлами і файлами реєстрації, щоденне чищення файлової системи.

7.7 Рекомендована література з теми 7

[1, с.20-21, 44-46], [2, с.110-123], [5, с.421-422], [7, с.396-400].

Повний список літератури знаходиться на стор. 87.

Порядок виконання роботи

Таблиця 7.1- Вивчення активних, фонових і призупинених процесів

Завдання	Виконання
1. Переглянути список існуючих процесів	*Ввести команду* ***ps*** <Enter>
2. Запустити процес (калькулятор *xcalc*) у фоновому режимі	*Ввести команду* ***xcalc &*** <Enter>
3. Переглянути список існуючих процесів	*Ввести команду* ***ps*** <Enter>
4. Переглянути список наявних робіт	*Ввести команду* ***jobs -l*** <Enter>
5. Запустити процес (годинник *xclock*) в активному режимі	*Ввести команду* ***xclock*** <Enter>
6. Завершити процес (годинник *xclock*) в активному режимі	Натиснути клавіші <***CTRL***><***C***> або клавішу ***DEL***
7. Переглянути список наявних робіт	*Ввести команду* ***jobs -l*** <Enter>
8. Зменшити пріоритет фонового процесу	*Ввести команду* ***renice -5 xcalc*** <Enter>
9. Завершити фоновий процес (калькулятор *xcalc*)	*Ввести команду* ***kill <PID>*** <Enter> (PID – ідентифікатор процесу *xcalc*)

Продовження таблиці 7.1.

Завдання	Виконання
10. Переглянути список наявних робіт	*Ввести команду* ***jobs -l*** <Enter>
11. Запустити процес (команду *yes*) в активному режимі	*Ввести команду* ***yes*** <Enter>
12. Призупинити виконання активного процесу	натиснути клавіші ***<CTRL><Z>***
13. Переглянути список наявних робіт	*Ввести команду* ***jobs -l*** <Enter>
14. Перевести призупинений процес в активний режим	*Ввести команду* ***fg*** <Enter>
15. Призупинити виконання активного процесу	натиснути клавіші ***<CTRL><Z>***
16. Перевести призупинений процес у фоновий режим	*Ввести команду* ***bg*** <Enter>
17. Переглянути список наявних робіт	*Ввести команду* ***jobs -l*** <Enter>
18. Завершити роботу (команду *yes*)	*Ввести команду* ***kill <%1>*** <Enter>

Таблиця 7.2 – Вивчення глибокого фонового процесу

Завдання	*Виконання*
1. Переглянути список існуючих процесів	*Ввести команду* ***ps*** <Enter>
2. Запустити глибокий фоновий процес (сортування файла *classif5*)	*Ввести команду* ***nohup sort classif5 &*** <Enter>
3. До закінчення роботи фонового процесу вийти із консолі	*Ввести команду* ***logout*** <Enter>
4. Повернутись в цю ж консоль	*Ввести login та пароль* <Enter>
5. Переглянути список наявних робіт	*Ввести команду* ***jobs -l*** <Enter>
6. Після закінчення роботи фонового процесу перевірити результати його роботи	*Ввести команду* ***more nohup.out*** <Enter>

Примітка. Завдання із табл. 7.1 виконуються в робочому столі *KDE* або *GNOME*, а завдання із табл. 7.2 - в текстовій консолі.

Тестові запитання для самоперевірки з теми 7

1. Кожна робота характеризується ідентифікатором. (Так / Ні).
2. *init* – це найголовніший процес в системі *Linux*. (Так / Ні).
3. Одночасне натиснення клавіш <*CTRL*> <*C*> приводить до призупинення активного процесу. (Так / Ні).
4. Фоновий процес допускає введення даних їх клавіатури. (Так / Ні).
5. Для отримання інформації про процеси використовується команда *ps*, а про роботи - команда *jobs*. (Так / Ні).
6. Для створення активного процесу необхідно після імені команди чи імені файла ввести символ &. (Так / Ні).
7. Процес (робота) переводиться із активного режиму в фоновий командою *fg*. (Так / Ні).
8. Керування роботами – це властивість оболонок операційної системи. (Так / Ні).
9. Команда *nice* може тільки зменшити пріоритет процесу. (Так / Ні).
10. Команда *kill* може ліквідувати активний процес. (Так / Ні).
11. Демон не є активним процесом. (Так / Ні).
12. Командою *nohup* створюється звичайний фоновий процес. (Так / Ні).

Контрольні питання

1. В чому полягає різниця між процесом і роботою в ОС *Linux* ?
2. Як створити новий процес (роботу) ОС *Linux*?
3. В яких станах може знаходитись процес (робота) ? Як визначити цей стан?
4. Як призупинити активний процес, а потім його відновити ?
7. Які існують способи завершення процесів (робіт) ?
8. Які особливості глибокого фонового процесу ?
9. Які Ви знаєте процеси-демони ?

Правильні відповіді
на тестові запитання для самоперевірки

з теми 1:
“Так” – 3,4,7,9,10,13,15,18.

з теми 2:
“Так” – 1,4,6,7,8,10.

з теми 3:
“Так” – 3,5,7,8,11,14.

з теми 4:
“Так” – 3,4,6,8,9,11.

з теми 5:
“Так” – 1,3,4,8,10.

з теми 6:
“Так” – 1,3,5,8,10,15,17,18,19.

з теми 7:
“Так” – 2,5,8,11.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Топхем Д.,Чыонг Х.В. Юникс и Ксеникс. - М.: Мир, 1988. - 392 с.

2. Скловская С. Red Hat Linux 6.0 – Учебник. К.: “ДиаСофтЮП”, 1999. - 416 с.

3. Скловская С. Команды Linux: Справочник. - К.:“ДиаСофт”, 2001.- 688 с.

4. Глушаков С.В., Сурядный А.С. Linux для дома и офиса: Учебный курс. - Харьков: Фолио, 2002. – 389 с.

5. Болл Б., Питтс Д. Red Hat Linux 7: Энциклопедия пользователя: Пер. с англ.- К.: Изд. “ДиаСофт”, 2001. - 592 с.

6. Бендел Д., Нейпир Р. Использование Linux: Специальное издание: Пер. с англ. 6-е изд. - М.: Вильямс, 2002. - 784 с.

7. Паркер Т. Linux 5.2: Энциклопедия пользователя: Пер. с англ. - К.: ”ДиаСофт”, 1999. - 688 с.

8. Левин М. Операционная система Linux: Пер. с англ. - М.: Оверлей, 2001. - 416 с.

Навчальне видання

Василь Петрович Семеренко,

Людмила Вікторівна Крилик

Операційна система Linux

Навчальний посібник

Оригінал-макет підготовлено авторами

Редактор В.О. Дружиніна

Коректор З.В. Поліщук

Навчально-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК №746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку
Формат 29,7х42 ¼
Друк різографічний
Тираж прим.
Зам. №

Гарнітура Times New Roman
Папір офсетний
Ум. друк. арк.

Віддруковано в комп'ютерному інформаційно-видавничому центрі Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК №746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ