1. Знайомство з операційною системою Linux.

1.1. Вступ

Всі матеріали першої і наступної тем будемо розглядати на прикладах, пов'язаних з використанням однієї з різновидів операційної системи UNIX – операційної системи Linux.

Для забезпечення безпеки системи пропонується розпочати знайомство з операційною системою Linux в середовищі віртуального комп'ютера, організованого засобами системи керування віртуальними комп'ютерами VirtualBox (http://www.virtualbox.org).

В якості дистрибутива Linux рекомендується обрати дистрибутив Debian GNU Linux (http://www.debian.org) або користуватись LiveDVD Knoppix (http://www.knoppix.net).

Перший варіант вимагає інсталяції на віртуальний комп'ютер, а при виборі другого — користувач може одразу після завантаження віртуального комп'ютера працювати з системою Linux.

Однак особливістю роботи в LiveDVD Knoppix, як і більшості Live-систем при використанні налаштувань за замовчуванням, є втрата усіх користувацьких змін у системі після перезавантаження комп'ютера, в тому числі і створених під час роботи документів. Тому важливо слідкувати за тим, де будуть зберігатись створені документи. Найкраще їх зберігати на жорсткому диску (у випадку роботи з віртуальним комп'ютером його потрібно створити в середовищі віртуальних комп'ютерів і потім розбити на розділи та відформатувати), в мережній папці або на флеш-носії. У LiveDVD Knoppix є можливість створити спеціальний файл на жорсткому диску, де будуть зберігатись усі зміни, зроблені користувачем під час роботи в системі (встановлені нові програми, налаштування програм, нові документи).

У версії Linux від Knoppix робота в системі за замовчуванням здійснюється під користувачем **knoppix**. Можна створити також свого користувача. Якщо є потреба виконати команду з правами адміністратора системи (**root**), користуються командою **sudo**

```
sudo ls
```

Після виконання команди, вказаної у **sudo**, інші команди будуть виконуватися з правами звичайного користувача.

При інсталяції системи обов'язково слід запам'ятати пароль **суперкористувача** (користувача з іменем **root**), а також інших користувачів, які створювались у системі під час інсталяції.

У тексті практичних занять програмні конструкції, включаючи імена системних викликів, стандартних функцій і команди оболонки операційної системи, виділені іншими шрифтами. У Linux системні виклики і команди оболонки ініціюють складні послідовності дій, які стосуються різних аспектів функціонування операційної системи. Як правило, у рамках однієї теми практичних занять повне пояснення всіх особливостей їхньої поведінки є

неможливим. Тому детальні описи більшості системних викликів, системних функцій і деяких команд оболонки операційної системи при першій зустрічі з ними винесені з основного тексту і виділені двома вертикальними рисками, а в основному тексті розглядаються тільки ті деталі їхнього опису, для розуміння яких вистачає набутих знань. Крім того, винесені описи не є повним описом команд, а адаптованим стосовно даного курсу. Для одержання повного опису звертайтеся до Linux Manual.

Якщо деякий параметр у команди оболонки є необов'язковим, він буде вказуватися у квадратних дужках, наприклад, **[who]**. У випадку, коли можливий вибір тільки одного з декількох можливих варіантів параметрів, варіанти будуть перераховуватися у фігурних дужках і розділятися вертикальною рискою, наприклад, **{+|-|=}**.

1.2. Коротка історія операційної системи Linux, її структура

Ядро операційної системи Linux є монолітною системою. При компіляції ядра Linux можна дозволити динамічне завантаження і вивантаження дуже багатьох компонентів ядра – так званих модулів. У момент завантаження модуля його код завантажується для виконання в привілейованому режимі і зв'язується з іншою частиною ядра. Всередині модуля можуть використовуватися будь-які експортовані ядром функції.

Свого нинішнього виду ця операційна система набула в результаті тривалої еволюції UNIX-подібних операційних систем. Історія розвитку UNIX детально освітлена практично у всій літературі, присвяченій обчислювальній техніці. Досить детально викладено історію у книзі [8] або в оригінальній роботі одного з родоначальників UNIX [18]. Для нас найбільш важливим є існування двох базових ліній еволюції – лінії System V і лінії BSD, оскільки в процесі навчання ми будемо зіштовхуватися з розходженнями в їхній реалізації.

1.3. Системні виклики і бібліотека libc

Основною постійно функціонуючою частиною операційної системи Linux є її ядро. Інші програми (системні або користувацькі) можуть спілкуватися з ядром за допомогою системних викликів, які фактично є прямими точками входу програм у ядро. При виконанні системного виклику програма користувача тимчасово переходить у привілейований режим, одержуючи доступ до даних або пристроїв, які недоступні при роботі в режимі користувача.

Реальні машинні команди, необхідні для активізації **системних викликів**, природно, відрізняються між комп'ютерами, як і спосіб передачі параметрів та результатів між програмою і ядром. Однак з погляду програмування мовою С використання **системних викликів** нічим зовні не відрізняється від використання інших функцій стандартної ANSI бібліотеки мови С, таких, наприклад, як функції роботи з рядками **strlen(),**

strcpy(). Стандартна бібліотека Linux – **libc** – забезпечує С-інтерфейс до кожного системного виклику. Це призводить до того, що системний виклик виглядає як функція мовою С для програміста. Крім того, багато із уже відомих Вам стандартних функцій, наприклад функції для роботи з файлами: **fopen(), fread(), fwrite()** при реалізації в операційній системі Linux будуть використовувати різні системні виклики. Під час вивчення курсу познайомимося з великою кількістю різноманітних системних викликів та їхніх С-інтерфейсів.

Більшість системних викликів, що повертають ціле значення, використовують значення **-1** для повідомлення про виникнення помилки, а значення більше або рівне **0** – при нормальному завершенні. Системні виклики, що повертають покажчики, зазвичай для ідентифікації помилкової ситуації використовують значення **NULL**. Для точного визначення причини помилки С-інтерфейс надає глобальну змінну **errno**, яка описана у файлі **<errno.h>** з її можливими значеннями і їхніми короткими визначеннями. Відмітимо, що аналізувати значення змінної **errno** необхідно одразу ж після виникнення помилкової ситуації, тому що системні виклики, які успішно завершилися, не змінюють її значення. Для одержання символьної інформації про помилку на стандартному виводі програми для помилок (за замовчуванням екран термінала) може застосовуватися стандартна UNIX-функція **perror()**.

Функція perror()

Прототип функції

```
#include <stdio.h>

void perror(char *str);
```

Опис функції

Функція **perror()** призначена для виводу повідомлення про помилку, що відповідає значенню системної змінної **errno** в стандартний потік виводу помилок. Функція друкує вміст рядка **str** (якщо параметр **str** не дорівнює **NULL**), двокрапку, пробіл і текст повідомлення, що відповідає помилці, з наступним символом переходу на новий рядок ('**\n**').

1.4. Вхід в операційну систему Linux і зміна пароля

Операційна система Linux є багатокористувацькою операційною системою. Для забезпечення безпечної роботи користувачів і цілісності системи, доступ до неї повинен бути санкціонований. Для кожного користувача, якому дозволений вхід у систему, створюється спеціальне реєстраційне ім'я – **username** *(*або **login)** і зберігається спеціальний пароль – **password**, що відповідає цьому імені. Як правило, при створенні нового користувача початкове значення пароля для нього задає системний адміністратор. Після першого входу в систему користувач повинен змінити

початкове значення пароля за допомогою спеціальної команди. Надалі він може в будь-який момент змінити пароль за своїм бажанням.

Якщо в системі встановлена графічна оболонка поряд зі звичайними алфавітно-цифровими терміналами, найкраще здійснити перший вхід в систему з алфавітно-цифрового термінала або його емулятора. Перемкнутися з графічної оболонки в консольний термінал можна за допомогою комбінації клавіш **Ctrl+Alt+Fx**, де **Fx** – одна із функціональних клавіш **F1, F2,** … Залежно від використовуваного дистрибутиву Linux, налаштувань системи, є 7 алфавітно-цифрових терміналів, які відповідають функціональним клавішам **F1, F2, …, F7**. Дуже часто перший термінал використовується для завантаження системи і з нього завантажується графічна оболонка. Для перемикання між алфавітно-цифровими терміналами достатньо комбінації клавіш **Alt+Fx**. Для повернення до графічної оболонки використовують комбінацію клавіш **Alt+F7** або **Alt+F8**.

Під час входу в систему з алфавітно-цифрового термінала на екрані з'являється напис, що пропонує ввести реєстраційне ім'я. Найчастіше це **"login:"**. Набравши своє реєстраційне ім'я, натискають клавішу **<Enter>**. Система запитає пароль, який відповідає введеному імені, видавши спеціальне запрошення – **"Password:"**. Уважно наберіть пароль, встановлений для Вас системним адміністратором, і натисніть клавішу **<Enter>**. **Пароль, що вводиться, на екрані не відображається, тому набирайте його акуратно!** Якщо все було зроблено правильно, у Вас на екрані з'явиться запрошення до вводу команд операційної системи.

Пароль, встановлений системним адміністратором, необхідно змінити. Найчастіше для цього використовується команда **passwd.** У більшості UNIX-подібних систем потрібно, щоб новий пароль мав не менше шести символів і містив, принаймні, дві не букви і дві не цифри.

Придумайте новий пароль і добре його запам'ятайте, а краще запишіть. Паролі в операційній системі зберігаються в закодованому виді, і якщо Ви його забули, ніхто не зможе допомогти Вам його згадати. Єдине, що може зробити системний адміністратор, так це встановити Вам новий пароль.

Введіть команду для зміни пароля. Зазвичай система просить спочатку набрати старий пароль, потім ввести новий і підтвердити правильність його набору повторним введенням. Після зміни пароля вже ніхто сторонній не зможе ввійти в систему під вашим реєстраційним іменем.

Тепер Ви повноцінний користувач операційної системи Linux.

1.5. Спрощене поняття про пристрій файлової системи в Linux. Повні та відносні імена файлів

В операційній системі Linux існують три базових поняття: **"процес"**, **"файл"** і **"користувач"**. З поняттям "користувач" ми щойно ознайомилися і будемо користуватися ним надалі при вивченні роботи цієї операційної системи. Поняття "процес" характеризує динамічну сторону того, що відбувається в обчислювальній системі. Поняття "файл" характеризує

статичну сторону обчислювальної системи.

З попереднього досвіду роботи з обчислювальною технікою Ви вже маєте уявлення про файл, як про іменований набір даних, що зберігається де-небудь на магнітних дисках або інших носіях. Нам досить такого розуміння, щоб розібратися в тому, як організована робота з файлами в операційній системі Linux. Детальніший розгляд поняття "файл" і організації файлових систем для операційних систем у цілому буде наведено в темі 7, яка присвячена організації файлових систем в Linux.

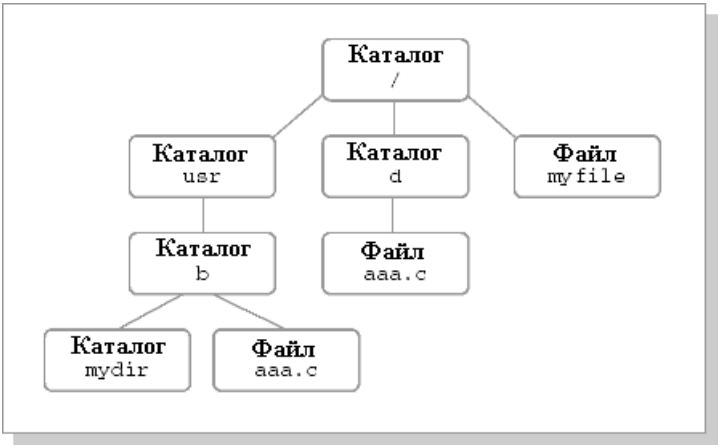

Рис. 1.1. Приклад структури файлової системи

Всі файли, доступні в операційній системі Linux, як і у вже відомих Вам операційних системах, об'єднуються в деревоподібну логічну структуру. Файли можуть об'єднуватися в **каталоги** або **директорії**. Не існує файлів, які не входили б до складу якого-небудь каталогу. Каталоги можуть входити до складу інших каталогів. Допускається існування порожніх каталогів, у які не входить жоден файл і жоден інший каталог (див. рис. 1.1). Серед всіх каталогів існує тільки один каталог, що не входить до складу інших каталогів – його прийнято називати **кореневий каталог (коренева директорія)**. На початковому рівні вивчення Linux ми можемо вважати, що у файловій системі Linux є, принаймні, два типи файлів: **звичайні файли**, які можуть містити тексти програм, код для виконання, дані – їх ще прийнято називати регулярними файлами, і **каталоги**.

Кожен файл (регулярний або каталог) повинен мати ім'я. У різних версіях операційної системи UNIX існують певні обмеження на побудову імені файлу. У стандарті POSIX на інтерфейс системних викликів для операційної системи UNIX міститься лише три явних обмеження:

- Не можна створювати імена більшої довжини, ніж це передбачено операційною системою (для Linux – 255 символів).
- Не можна використовувати символ NUL (не плутайте з покажчиком

NULL!) – він же символ з нульовим кодом, він же ознака кінця рядка в мові С.

- Не можна використовувати символ '/'.

Також небажано застосовувати символи "зірочка" – "**`*`**", "знак запитання" – "**`?`**", "лапки" – "**`"`**", "апостроф" – "**`'`**", "пробіл" – " " і "зворотній слеш" – "**`\`**" (символи записані в стандарті запису символьних констант мови С)**.**

Єдиним винятком є **кореневий каталог**, що **завжди** має ім'я **`"/"`**. Цей каталог є єдиним файлом, що повинен мати **унікальне ім'я у всій файловій системі**. Для всіх інших файлів імена повинні бути унікальними тільки в рамках того каталогу, у який вони безпосередньо входять. Яким же чином відрізнити два файли з іменами **`"aaa.c"`**, що входять у директорії **`"b"`** і **`"d"`** на рис. 1.1, щоб було зрозуміло про який з них йде мова? Тут на допомогу приходить поняття повного імені файлу.

Побудуємо шлях від кореневої вершини дерева файлів до файлу, що цікавить нас, і випишемо всі імена вузлів дерева, що зустрічаються на нашому шляху, наприклад, "**`/ usr b aaa.c".`** У цій послідовності першим буде завжди стояти ім'я кореневого каталогу, а останнім – ім'я файлу, що цікавить нас. Відокремимо імена вузлів один від одного в цьому записі не пробілами, а символами "**`/`**", за винятком імені кореневого каталогу і наступного імені каталогу ("**`/usr/b/aaa.c`**"). Отриманий запис однозначно ідентифікує файл у всій логічній конструкції файлової системи. Такий запис і одержав назву **повного імені файлу**.

1.6. Поняття про поточний каталог. Команда pwd. Відносні імена файлів. Домашній каталог користувача

Повні імена файлів можуть містити в собі досить багато імен каталогів і бути дуже довгими, а тому з ними не завжди зручно працювати. У той же час, існують такі поняття як **поточний** або **робочий каталог** і **відносне ім'я файлу**.

Для кожної програми, яка працює в операційній системі, включаючи командний інтерпретатор (**`shell`**), що обробляє введені команди і виводить запрошення для їхнього введення, один з каталогів у логічній структурі файлової системи призначається поточним або робочим для даної програми. Довідатися, який каталог є поточним для вашого командного інтерпретатора, можна за допомогою команди операційної системи **`pwd`**.

Знаючи поточний каталог, ми можемо побудувати шлях по графу файлів від поточного каталогу до файлу, що цікавить нас. Запишемо послідовність вузлів, які зустрінуться на цьому шляху, у такий спосіб. Вузол, що відповідає поточному каталогу, у запис не включаємо. При русі в напрямку до кореневого каталогу кожний вузол будемо позначати двома символами "крапка" – "**`..`**", а при русі по напрямку від кореневого каталогу будемо записувати ім'я вузла, що зустрівся**.** Розділимо позначення, що відносяться до

різних вузлів у цьому записі, символами "/". Отриманий рядок прийнято називати відносним іменем файлу. **Відносні імена файлів змінюються при зміні робочого каталогу.** Так, у нашому прикладі, якщо робочий каталог – це "**/d**", то для файлу "**/usr/b/aaa.c**" відносним іменем буде **"../usr/b/aaa.c",** а якщо робочий каталог – це **"/usr/b",** то його відносне ім'я **– "aaa.c".**

Для повноти картини ім'я поточного каталогу можна вставляти у відносне ім'я файлу, позначаючи поточний каталог одним символом "крапка" – **"."**. Тоді наші відносні імена будуть виглядати як **"./../usr/b/aaa.c" і "./aaa.c"** відповідно.

Програми, запущені за допомогою командного інтерпретатора, будуть мати як робочий каталог його робочий каталог, якщо всередині цих програм не змінити її розташування за допомогою спеціального системного виклику.

Для кожного нового користувача в системі заводиться спеціальний каталог, що стає поточним одразу після його входу в систему. Цей каталог одержав назву **домашнього каталогу користувача**.

Домашні каталоги користувачів знаходяться у каталозі **/home**. Наприклад, для користувача **stud** домашнім буде каталог **/home/stud**. Для адміністратора (**root**) домашнім каталогом є **/root**. Синонімом домашнього каталога є символ **"~"**.

1.7. Одержання допомоги про команди у ОС Linux

Під час вивчення операційної системи Linux Вам часто буде потрібна інформація про те, для чого призначена команда або системний виклик, які в них параметри та опції, для чого призначені деякі системні файли, який їхній формат. Більша частина інформації в UNIX Manual доступна в інтерактивному режимі за допомогою утиліти **man**.

Користуватися утилітою **man** досить просто – наберіть команду

man ім'я

де **ім'я** – це ім'я команди, що цікавить вас, утиліти, системного виклику, бібліотечної функції або файлу. Спробуйте з її допомогою подивитися інформацію про команду **pwd**.

Щоб пролистати сторінку отриманого опису, якщо він не помістився на екрані повністю, потрібно натиснути клавішу **<пробіл>**. Для прокручування одного рядка скористайтеся клавішею **<Enter>**. Повернутися на сторінку назад дозволить одночасне натискання клавіш **<Ctrl>** і **<b>**. Вийти з режиму перегляду інформації можна за допомогою клавіші **<q>**.

Іноді імена команд інтерпретатора і системних викликів або які-небудь ще імена збігаються. Тоді, щоб знайти інформацію, необхідно задати утиліті **man** категорію, до якої належить ця інформація (номер розділу). Розподіл інформації на категорії може трохи відрізнятися у різних версіях UNIX.

Якщо Ви знаєте розділ, до якого відноситься інформація, то утиліту **man** можна викликати в Linux з додатковим параметром

```
man номер_розділа ім'я
```

В інших операційних системах цей виклик може виглядати інакше. Для одержання точної інформації про розбивку на розділи, форму вказання номера розділу і додаткових можливостей утиліти **man** наберіть команду

```
man man
```

Для одержання допомоги можна використовувати також команду **info**

```
info ls
```

Щоб отримати коротшу інформацію про команду

```
ls –info
```

Одержати підказку про команду можна за допомогою команди **whatis**

```
whatis ls
```

1.8. Перегляд вмісту каталогу, зміна поточного каталогу

Для зміни поточного каталогу командного інтерпретатора можна скористатися командою **cd (change directory)**. Для цього необхідно набрати команду у вигляді

```
cd ім'я_директорії
```

де **ім'я_директорії** – повне або відносне ім'я каталогу, який потрібно зробити поточним. Команда **cd** без параметрів зробить поточним каталогом домашній каталог.

Переглянути вміст поточного або будь-якого іншого каталогу можна, скориставшись командою **ls** (від **list**). Якщо ввести її без параметрів, ця команда роздрукує Вам список файлів, що перебувають у поточному каталозі. Якщо ж як параметр задати повне або відносне ім'я каталогу

```
ls ім'я_директорії
```

то вона роздрукує список файлів у зазначеній директорії. Слід відзначити, що в отриманий список не ввійдуть файли, імена яких починаються із символу "крапка" – **"."**. Такі файли зазвичай створюються різними системними програмами для своїх цілей (наприклад, для налаштування). Подивитися повний список файлів можна, додатково вказавши команді **ls** опцію **-a**, тобто набравши її у вигляді

```
ls -a ім'я_директорії
```

У команди **ls** існує також багато інших опцій, деякі з яких ми ще розглянемо на практичних заняттях. Для одержання повної інформації про команду **ls** скористайтеся утилітою **man**.

1.9. Команда cat і створення файлу. Перенапрямлення вводу і виводу

Ми вміємо переміщуватися логічною структурою файлової системи і переглядати її вміст. Хотілося б уміти ще й переглядати вміст файлів, і створювати їх. Для перегляду вмісту невеликого текстового файлу на екрані можна скористатися командою **cat**. Якщо набрати її у вигляді

```
cat ім'я_файлу
```

то на екран виведеться увесь його вміст.

Не намагайтеся переглядати на екрані вміст директорій – однаково не вийде! Не намагайтеся переглядати вміст невідомих файлів, особливо якщо Ви не знаєте, текстовий він чи бінарний. Вивід на екран бінарного файлу може привести до непередбачуваної поведінки вашого термінала.

Якщо навіть Ваш файл і текстовий, але великий, то Ви побачите тільки його останню сторінку. Великий текстовий файл зручніше переглядати за допомогою утиліти **more**.

Якщо як параметри для команди **cat** задамо не одне ім'я, а імена декількох файлів

```
cat файл1 файл2 ... файлN
```

то система видасть на екран їхній вміст у зазначеному порядку. Вивід команди **cat** можна перенаправляти з екрана термінала в який-небудь файл, скориставшись символом перенаправлення вихідного потоку даних – знаком "більше" – "**>**". Команда

```
cat файл1 файл2 ... файл > файл_результату
```

об‘єднає вміст всіх файлів, чиї імена розміщені перед знаком ">", у файл_результату – їхню конкатинацію (від слова concatenate і походить її назва). Прийом перенаправлення вихідних даних зі стандартного потоку виводу (екрана) у файл є стандартним для всіх команд, виконуваних командним інтерпретатором. Ви можете одержати файл, що містить список всіх файлів поточного каталогу, якщо виконаєте команду **ls -a** з перенаправленням вихідних даних

```
ls -a > новий_файл
```

Якщо імена вхідних файлів для команди **cat** не задані, то вона буде використовувати в якості вхідних даних інформацію, що вводиться із клавіатури, доти, поки Ви не наберете ознаку закінчення вводу – комбінацію клавіш **<CTRL> + <d>**.

Таким чином, команда

```
cat > новий_файл
```

дозволяє створити новий текстовий файл із іменем **новий_файл** і вмістимим, яке користувач введе з клавіатури. У команди **cat** існує багато різних опцій. Подивитися її повний опис можна в UNIX Manual.

Відзначимо, що поряд з перенаправленням вихідних даних існує спосіб перенаправляти вхідні дані. Якщо під час виконання деякої команди потрібно ввести дані із клавіатури, можна помістити їх заздалегідь у файл, а потім перенаправляти стандартний ввід цієї команди за допомогою знака "менше" – "**<**" і імені файлу із вхідними даними. Інші варіанти перенаправлення потоків даних можна подивитися в UNIX Manual для командного інтерпретатора.

1.10. Найпростіші команди для роботи з файлами – cp, rm, mkdir, mv

Для нормальної роботи з файлами необхідно не тільки вміти створювати файли, переглядати їхній вміст і переміщуватися логічним деревом файлової системи. Потрібно вміти також створювати власні підкаталоги, копіювати і вилучати файли, перейменовувати їх. Це мінімальний набір операцій, не володіючи яким не можна почувати себе впевнено при роботі з комп'ютером.

Для створення нового підкаталогу використовується команда **mkdir** (скорочення від make directory). У найпростішому вигляді команда виглядає так:

mkdir ім'я_директорії

де **ім'я_директорії** – повне або відносне ім'я створюваного каталогу. У команди **mkdir** є набір опцій, опис яких можна переглянути за допомогою утиліти **man**.

Команда cp

Синтаксис команди

```
cp файл_джерело файл_призначення
cp файл1 файл2 ... файлN дир_призначення
cp -r дир_джерело дир_призначення
cp -r дир1 дир2 ... дирN дир_призначення
```

Опис команди

Команда **cp** у формі

cp файл_джерело файл_призначення

призначена для копіювання одного файлу з іменем **файл_джерело** у файл із іменем **файл_призначення**.

Команда **cp** у формі

cp файл1 файл2 ... файлN дир_призначення

призначена для копіювання файлу або файлів з іменами **файл1**, **файл2**, ... **файлN** у вже існуючу директорію з іменем **дир_призначення** під своїми іменами. Замість імен файлів, що копіюються, можуть використовуватися їхні шаблони.

Команда **cp** у формі

cp -r дир_джерело дир_призначення

призначена для рекурсивного копіювання однієї директорії з іменем **дир_джерело** в нову директорію з іменем **дир_призначення**. Якщо директорія **дир_призначення** вже існує, то ми одержуємо команду **cp** у наступній формі

cp -r дир1 дир2 ... дирN дир_призначення

Така команда призначена для рекурсивного копіювання директорії або директорій з іменами **дир1**, **дир2**, ... **дирN** у вже існуючу директорію з іменем **дир_призначення** під своїми власними іменами. Замість імен директорій, які копіюються, можуть використовуватися їхні шаблони.

Для копіювання файлів може використовуватися команда **cp** (скорочення від **copy**). Команда **cp** може копіювати не тільки окремий файл, але й набір файлів, і навіть директорію разом з усіма піддиректоріями (рекурсивне копіювання). Для задання набору файлів можуть використовуватися шаблони імен файлів. Так само шаблон імені може бути використаний і в командах перейменування файлів і їхнього вилучення, які ми розглянемо нижче.

Шаблони імен файлів

Шаблони імен файлів можуть застосовуватися як параметр для задання набору імен файлів у багатьох командах операційної системи. При використанні шаблона проглядається вся множина імен файлів, що перебувають у файловій системі, і ті імена, які задовольняють шаблону, включаються в набір. У загальному випадку шаблони можуть задаватися з використанням наступних метасимволів:

***** – відповідає всім ланцюжкам літер, включаючи порожній;

? – відповідає всім одиночним літерам;

[...] – відповідає будь-якій літері, що міститься в дужках. Пара літер, розділених знаком мінус, задає діапазон літер.

Так, наприклад, шаблону ***.c** задовольняють всі файли поточної директорії, чиї імена закінчуються на **.c.** Шаблону **[a-d]*** задовольняють всі файли поточної директорії, чиї імена починаються з букв **a, b, c, d**. Існує одне обмеження на використання метасимвола ***** на початку імені файлу, наприклад, у випадку шаблона ***c**. Для таких шаблонів імена файлів, що починаються із символу крапка, вважаються такими, що не задовольняють шаблону.

Для вилучення файлів або директорій застосовується команда **rm** (скорочення від remove). Якщо Ви хочете вилучити один або декілька регулярних файлів, то найпростіший вигляд команди **rm** буде:

rm файл1 файл2 ... файлN

де **файл1**, **файл2**, ..., **файлN** – повні або відносні імена регулярних файлів, які потрібно вилучити. Замість імен файлів можуть використовуватися їхні шаблони. Якщо Ви хочете вилучити одну або декілька директорій разом з їхнім вмістом (рекурсивне вилучення), то до команди додається опція **-r**:

rm -r дир1 дир2 ... дирN

де **дир1**, **дир2**, ... **дирN** – повні або відносні імена директорій, які потрібно вилучити. Замість безпосередньо імен директорій також можуть використовуватися їхні шаблони. У команди **rm** є ще набір корисних опцій, які описані в UNIX Manual. Насправді процес вилучення файлів не такий простий, яким здається на перший погляд. Детальніше він буде розглянутий в темі 7, коли розглядатимемо операції над файлами в операційній системі Linux.

Команда mv

Синтаксис команди

```
mv ім'я_джерела ім'я_призначення
mv ім'я1 ім'я2 ... ім'яN дир_призначення
```

Опис команди

Команда **mv** у формі

```
mv ім'я_джерела ім'я_призначення
```

призначена для перейменування або переміщення одного файлу (неважливо, регулярного або директорії) з іменем **ім'я_джерела** у файл із іменем **ім'я_призначення**. При цьому перед виконанням команди файлу з іменем **ім'я_призначення** існувати не повинно.

Команда **mv** у формі

```
mv ім'я1 ім'я2 ... ім'яN дир_призначення
```

призначена для переміщення файлу або файлів (неважливо, регулярних файлів або директорій) з іменами **ім'я1**, **ім'я2**, ... **ім'яN** у вже існуючу директорію з іменем **дир_призначення** під власними іменами. Замість імен переміщуваних файлів можуть використовуватися їхні шаблони.

Командою вилучення файлів і директорій варто користуватися з обережністю. Вилучену інформацію відновити неможливо. Якщо Ви системний адміністратор і Ваша поточна директорія – це коренева директорія, не виконуйте команду `rm -r *`!

Для перейменування файлу або його переміщення в інший каталог застосовується команда **mv** (скорочення від move). Для задання імен переміщуваних файлів у ній теж можна використовувати їхні шаблони.

1.11. Історія редагування файлів. Система Midnight Commander

Коли з'явилися перші варіанти операційної системи UNIX, пристрої введення та відображення інформації істотно відрізнялися від існуючих сьогодні. На клавіатурах були присутні тільки алфавітно-цифрові клавіші (не було навіть клавіш курсорів), а дисплеї не передбачали екранного редагування. Тому перший редактор операційної системи UNIX – редактор **ed** – вимагав від користувача строгої вказівки того, що і як буде редагуватися за допомогою спеціальних команд.

Редактор **ed** був пострічковим редактором. Згодом з'явився екранний редактор – **vi**, однак і він вимагав строгої вказівки того, що і як у поточній позиції на екрані ми повинні зробити, або яким чином змінити поточну позицію, за допомогою спеціальних команд, що відповідають алфавітно-цифровим клавішам. Ці редактори можуть здатися нам зараз анахронізмами, але вони дотепер входять до складу всіх варіантів UNIX і іноді (наприклад, при роботі з віддаленим комп'ютером через повільний канал зв'язку) є єдиним

засобом, що дозволяє віддалено редагувати файл.

Напевно, Ви вже переконалися в тому, що робота в Linux винятково на рівні командного інтерпретатора та вбудованих редакторів далека від уже звичних для нас зручностей. Але не все так погано. Існують різноманітні пакети, що полегшують завдання користувача в Linux. До таких пакетів варто віднести **Midnight Commander** – аналог програм Norton Commander для DOS і FAR для Windows 9x і NT – зі своїм вбудованим редактором, що запускається командою **mc**. Інформацію про них можна знайти в UNIX Manual. Більшими можливостями володіють багатофункціональні текстові редактори, наприклад, emacs.

1.12. Користувач і група. Команди chown і chgrp. Права доступу до файлу

Для входу в операційну систему Linux кожний користувач повинен бути зареєстрований у ній під певним іменем. Обчислювальні системи не вміють оперувати іменами, тому кожному імені користувача в системі відповідає деяке числове значення – його ідентифікатор – **UID** (**user identificator**).

Всі користувачі в системі діляться на групи. Наприклад, студенти однієї навчальної групи можуть становити окрему групу користувачів. Групи користувачів також одержують свої імена і відповідні ідентифікаційні номери – **GID** (group identificator). В одних версіях UNIX кожний користувач може входити тільки в одну групу, в інших – у кілька груп.

Команда chown

Синтаксис команди

chown owner файл1 файл2 ... файлN

Опис команди

Команда **chown** призначена для зміни власника (хазяїна) файлів. Нового власника файлу можуть призначити тільки попередній власник файлу або системний адміністратор.

Параметр **owner** задає нового власника файлу в символьному виді, як його **username**, або в числовому виді, як його **UID**.

Параметри **файл1**, **файл2**, ... **файлN** – це імена файлів, для яких здійснюється зміна власника. Замість імен можуть використовуватися їхні шаблони.

Для кожного файлу, створеного у файловій системі, запам'ятовуються імена його власника і групи власників. Відмітимо, що група власників не обов'язково повинна бути групою, у яку входить власник. Спрощено можна вважати, що в операційній системі Linux при створенні файлу його власником стає користувач, що створив файл, а його групою власників – група, до якої

цей користувач належить. Згодом власник файлу або системний адміністратор можуть передати його у власність іншому користувачеві або змінити його групу власників за допомогою команд **chown** і **chgrp**, опис яких можна знайти в UNIX Manual.

Команда chgrp

Синтаксис команди

chgrp group файл1 файл2 ... файлN

Опис команди

Команда **chgrp** призначена для зміни групи власників (хазяїв) файлів. Нову групу власників файлу можуть призначити тільки власник файлу або системний адміністратор.

Параметр **group** задає нову групу власників файлу в символьному виді, як ім'я групи, або в числовому виді, як її **GID**.

Параметри **файл1**, **файл2**, ... **файлN** – це імена файлів, для яких здійснюється зміна групи власників. Замість імен можуть використовуватися їхні шаблони.

Отже, для кожного файлу виділяється три категорії користувачів:

- користувач, що є власником файлу;
- користувачі, що відносяться до групи власників файлу;
- всі інші користувачі.

Для кожної із цих категорій власник файлу може визначити різні права доступу до файлу. Розрізняють три види прав доступу: право на читання файлу – **r** (від слова **read**), право на модифікацію файлу – **w** (від слова **write**) і право на виконання файлу – **x** (від слова **execute**). Для регулярних файлів значення цих прав збігається із зазначеним вище. Для директорій він трохи інший. Право читання для каталогів дозволяє читати імена файлів, що містяться у цьому каталозі (і тільки імена). Оскільки "виконувати" директорію безглуздо, право доступу на виконання для директорій змінює значення: наявність цього права дозволяє одержати додаткову інформацію про файли, що входять у каталог (їхній розмір, їхній власник, дата створення). Без цього права Ви не зможете ні читати вміст файлів, що містяться у директорії, ні модифікувати їх, ні виконувати. Право на виконання також потрібне для директорії, щоб зробити її поточною, а також для всіх директорій на шляху до неї. Право запису для директорії дозволяє змінювати її вміст: створювати і вилучати в ній файли, перейменовувати їх. Відзначимо, що для вилучення файлу досить мати права запису і виконання для директорії, у яку входить даний файл, незалежно від прав доступу до самого файлу.

1.13. Команда ls з опціями -al. Використання команд

chmod і umask

Одержати детальну інформацію про файли в деякій директорії, включаючи імена власника, групи власників і права доступу, можна за допомогою вже відомої нам команди **ls** з опціями **-al**. У виводі цієї команди третій стовпчик зліва містить імена користувачів власників файлів, а четвертий стовпчик зліва – імена груп власників файлу. Перший лівий стовпчик містить типи файлів і права доступу до них. Тип файлу визначає перший символ у наборі символів. Якщо це символ '**d**', то тип файлу – директорія, якщо там міститься символ '**-**', то це – регулярний файл. Наступні три символи визначають права доступу для власника файлу, наступні три – для користувачів, що входять у групу власників файлу, і останні три – для всіх інших користувачів. Наявність символу (**r**, **w** або **x**), що відповідає праву, для деякої категорії користувачів означає, що дана категорія користувачів має це право.

Команда chmod

Синтаксис команди

```
chmod [who] { + | - | = } [perm] файл1 файл2 ... файлN
chmod [perm8] файл1 файл2 ... файлN
```

Опис команди

Команда **chmod** призначена для зміни прав доступу до одного або декількох файлів. Права доступу до файлу можуть змінювати тільки власник (хазяїн) файлу або системний адміністратор.

Параметр **who** визначає, для яких категорій користувачів встановлюються права доступу. Він може бути одним символом або кількома символами:

a – встановлення прав доступу для всіх категорій користувачів. Якщо параметр **who** не заданий, то за замовчуванням застосовується **a**. При визначенні прав доступу із цим значенням задані права встановлюються з урахуванням значення маски створення файлів;

u – встановлення прав доступу для власника файлу;

g – встановлення прав доступу для користувачів, що входять у групу власників файлу;

o – встановлення прав доступу для всіх інших користувачів.

Операція, що виконується над правами доступу для заданої категорії користувачів, визначається одним з наступних символів:

+ – додавання прав доступу;

- – скасування прав доступу;

= – заміна прав доступу, тобто скасування всіх існуючих і додавання перерахованих.

Якщо параметр **perm** не визначений, то всі існуючі права доступу відміняються.

Параметр **perm** визначає права доступу, які будуть додані, скасовані або встановлені замість існуючих відповідною командою. Він є комбінацію наступних символів або одним із них:

r – право на читання;

w – право на модифікацію;

x – право на виконання.

Параметри **файл1**, **файл2**, ..., **файлN** – це імена файлів, для яких здійснюється зміна прав доступу. Замість імен можуть використовуватися їхні шаблони.

Команда **chmod**, записана в другому форматі, дозволяє встановити всі права для власника, групи власника та інших користувачів. При цьому параметр **perm8** задає права у вісімковому вигляді. Наприклад, команда

```
chmod 762 1.txt
```

встановлює такі права на файл **1.txt**: **111110010** або **rwxrw—w-.**

Власник файлу може змінювати права доступу до нього, користуючись командою **chmod**.

Маска створення файлів поточного процесу

Маска створення файлів поточного процесу (**umask**) використовується системними викликами **open()** і **mknod()** при встановленні початкових прав доступу для створюваних файлів або **FIFO**. Молодші 9 біт маски створення файлів відповідають правам доступу користувача, що створює файл, групи, до якої він належить, і всіх інших користувачів так, як записано нижче із застосуванням вісімкових значень:

0400 – право читання для користувача, що створив файл;

0200 – право запису для користувача, що створив файл;

0100 – право на виконання для користувача, що створив файл;

0040 – право читання для групи користувача, що створив файл;

0020 – право запису для групи користувача, що створив файл;

0010 – право на виконання для групи користувача, що створив файл;

0004 – право читання для всіх інших користувачів;

0002 – право запису для всіх інших користувачів;

0001 – право на виконання для всіх інших користувачів.

Встановлення значення якого-небудь біта в 1 забороняє ініціалізацію відповідного права доступу для створюваного файлу. Значення маски створення файлів може змінюватися за допомогою системного виклику **umask()** або команди **umask**. Маска створення файлів успадковується процесом-нащадком при породженні нового процесу системним викликом **fork()** і входить до складу незмінної частини системного контексту процесу при системному виклику **exec()**. У результаті цього успадкування зміни маски за допомогою команди **umask** вплине на атрибути доступу

файлів, що створюються, для всіх процесів, породжених далі командною оболонкою.

Змінити поточне значення маски для програми-оболонки або подивитися його можна за допомогою команди **umask**.

Команда umask

Синтаксис команди

```
umask [value]
```

Опис команди

Команда **umask** призначена для зміни маски створення файлів командної оболонки або перегляду її поточного значення. При відсутності параметра команда видає значення встановленої маски створення файлів у вісімковому вигляді. Для встановлення нового значення воно задається як параметр **value** у вісімковому виді.

Якщо Ви хочете змінити його для **Midnight Commander**, необхідно вийти з **mc**, виконати команду **umask** і запустити **mc** знову. **Маска створення файлів не зберігається між сеансами роботи в системі.** При новому вході в систему значення маски знову буде встановлене за замовчуванням.

1.14. Системні виклики getuid і getgid

Довідатися ідентифікатор користувача, що запустив програму на виконання, – **UID** і ідентифікатор групи, до якої він відноситься, – **GID** можна за допомогою системних викликів **getuid()** і **getgid()**, застосувавши їх всередині цієї програми.

Системні виклики getuid() і getgid()

Прототипи системних викликів

```
#include <sys/types.h>
#include <unistd.h>
uid_t getuid(void);
gid_t getgid(void);
```

Опис системних викликів

Системний виклик **getuid** повертає ідентифікатор користувача для поточного процесу.

Системний виклик **getgid** повертає ідентифікатор групи користувача для поточного процесу.

Типи даних **uid_t** і **gid_t** є синонімами для одного із цілочисельних типів мови С.

1.15. Компіляція програм мовою С в Linux і запуск їх на

виконання

Для компіляції програм в Linux ми будемо застосовувати компілятор **gcc**.

Для того, щоб він нормально працював, необхідно, щоб вихідні файли, що містять текст програми, мали імена, що закінчуються на **.c.**

У найпростішому випадку відкомпілювати програму можна, запускаючи компілятор командою

```
gcc ім'я_вихідного_файлу
```

Якщо програма була написана без помилок, то компілятор створить виконуваний файл з іменем **a.out.** Змінити ім'я створюваного виконуваного файлу можна, задавши його за допомогою опції **-o**:

```
gcc ім'я_вихідного_файлу -o ім'я_виконуваного_файлу
```

Компілятор **gcc** має кілька сотень можливих опцій. Одержати інформацію про них Ви можете в UNIX Manual.

Запустити програму на виконання можна, набравши ім'я виконуваного файлу і натиснувши клавішу **<Enter>**.

Завдання для самостійної роботи

Самостійно опрацюйте матеріали теми 1 до початку практичного заняття, а також такі питання

1. Структура обчислювальної системи.
2. Операційна система як:
    - віртуальний комп'ютер,
    - менеджер ресурсів,
    - захисник користувачів і програм,
    - ядро, що постійно функціонує в пам'яті.
3. Коротка історія еволюції обчислювальних систем.
4. Основні поняття, концепції ОС (системні виклики, переривання, виняткові ситуації, файли, процеси, нитки).
5. Архітектурні особливості ОС (монолітне ядро, багаторівневі системи (layered systems), віртуальні комп'ютери, мікроядерна архітектура, змішані системи).
6. Класифікація ОС (реалізація багатозадачності, підтримка багатокористувацького режиму, багатопроцесорна обробка, системи реального часу).

Завдання для лабораторної роботи

1. Завантажте операційну систему Linux. Увійдіть у систему під своїм реєстраційним іменем, змініть пароль користувача. Які обмеження на новий пароль існують у вашій операційній системі?
2. Завантажте термінал у випадку входу в графічне середовище.
3. Ознайомтеся із структурою файлової системи Linux (файл і каталог,

дозволені символи в іменах файлів, регулярний файл, кореневий каталог, повне та відносне ім'я файлу).

4. Поясніть поняття поточний каталог та домашні каталоги користувачів. Для чого призначена команда **pwd**? Скористайтеся командою **pwd** для визначення своєї домашньої директорії.
5. Одержіть допомогу про відомі Вам команди Linux.
6. Користуючись командами **cd**, **ls** і **pwd**, перемістіться структурою файлової системи і перегляньте її вміст. Можливо, зайти в деякі директорії або переглянути їхній вміст Вам не вдасться. Це пов'язано з роботою механізму захисту файлів і директорій, про який ми поговоримо пізніше. Не забудьте повернутися у свою домашню директорію (**cd ~**).
7. Виконайте команду **ls -al** для своєї домашньої директорії і проаналізуйте її вивід.
8. Переконайтеся, що Ви перебуваєте у своїй домашній директорії, і створіть за допомогою команди **cat** новий текстовий файл **info.txt**. Перегляньте його вміст.
9. Створіть в домашньому каталозі новий каталог **data1**. Скопіюйте в нього створений текстовий файл **info.txt**. Створіть його копію **info2.txt**. Перейменуйте копію на **info_new.txt**.
10. Перегляньте атрибути файлу **info_new.txt**. Зверніть увагу на права доступу до файлу. Як відрізнити у виводі команди **ls** звичайний файл від каталогу, файлу пристрою? Що означають імена "**.**" та "**..**"?
11. Створіть копію файлу **info_new.txt info_user.txt**. Дозвольте читання та запис до **info_user.txt** лише власнику файла, іншим користувачам — забороніть.
12. Створіть нового користувача в системі. Увійдіть під новим іменем в систему. Спробуйте прочитати вміст файла **info_user.txt**. Одержаний результат поясніть.
13. Створіть новий файл і подивіться на права доступу до нього, встановлені системою при його створенні. Чим керується операційна система при призначенні цих прав?
14. Запустіть програму **mc**. Спробуйте переміщатися по директоріях, створювати і редагувати файли. Створіть копію каталогу **data1 data2**. Виконайте дії 9, 10 засобами Midnight Commander.
15. Напишіть, відкомпілюйте і запустіть програму, яка б друкувала ідентифікатор користувача, що запустив програму, і ідентифікатор його групи.
16. Виконайте написану програму з правами адміністратора та двох інших користувачів. Поясніть одержані результати.