Операційні системи. Практикум

1. Вступ до курсу практичних занять. Знайомство з операційною системою UNIX.

План заняття

Вступ до курсу практичних занять

Даний курс практичних занять призначений для оволодіння студентами дисципліною "Операційні системи" на прикладі конкретної операційної системи, а саме – операційної системи UNIX. Передбачається, що до початку практичних занять студенти вміють програмувати мовою Сі (з використанням функцій стандартної бібліотеки для роботи з файлами та рядками) і мають уявлення про внутрішню будову ЕОМ.

Для ілюстрації була обрана операційна система UNIX, як найбільш відкрита і проста для розуміння, хоча в можливе і для інших операційних систем, наприклад для Windows NT.

У цілому практичний курс містить у собі 16 занять. Деяким темам виділено по два заняття, і відповідні семінари мають подвійні номери. Природно, розбивка тем на заняття є досить умовною, бажано лише, щоб вони безпосередньо слідували за лекціями, на яких ґрунтуються.

Далі перейдемо до викладу матеріалу семінарсько-практичного курсу.

За своїм змістом матеріал семінарів 1–2 є найбільш критичним стосовно використовуваного виду операційної системи і політики адміністрування. Тому багато питань будуть містити посилання **"довідайтеся у свого системного адміністратора"**. Перш ніж приступати до занять, необхідно забезпечити наявність користувацьких account'ів. **"Довідайтеся у свого системного адміністратора"**, як це зробити.

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

Особливістю роботи в LiveDVD Knoppix, як і в усіх Live-системах при налаштуваннях за замовчуванням, є втрата усіх користувацьких змін у системі після перезавантаження комп'ютера, в тому числі і створених під час роботи документів. Тому важливим є слідкування користувача за тим де будуть зберігатись створені ним документи. Найкраще їх зберігати на жорсткому диску (у випадку роботи з віртуальним комп'ютером його потрібно створити в середовищі віртульних комп'ютерів і потім розбити на розділи та відформатувати), в мережевій папці, або на флеш-носії. У LiveDVD Knoppix є можливість створити спеційльний 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. У випадку потреби 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.

В тексті семінарів програмні конструкції, включаючи імена системних викликів, стандартних функцій і команди оболонки операційної системи, `виділені іншим шрифтом`. В UNIX системні виклики і команди оболонки ініціюють складні послідовності дій, торкаючись різних аспектів функціонування операційної системи. Як правило, у рамках одного семінару повне пояснення всіх нюансів їхньої поведінки є неможливим. Тому детальні описи більшості використовуваних системних викликів, системних функцій і деяких команд оболонки операційної системи при першій зустрічі з ними винесені з основного тексту на сірий фон і обведені рамкою, а в основному тексті розглядаються тільки ті деталі їхнього опису, для розуміння яких вистачає набутих знань.

Якщо який-небудь 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, `{+ | - | =}`.

Коротка історія операційної системи UNIX, її структура

На першій лекції ми розібрали зміст поняття "операційна система", обговорили функції операційних систем і способи їхньої побудови. Всі матеріали першої і наступної лекцій ми будемо ілюструвати практичними прикладами, пов'язаними з використанням однієї з різновидів операційної системи UNIX – операційної системи Linux, хоча постараємося не зв'язувати розповідь саме з її особливостями.

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 [23] або в оригінальній роботі одного з родоначальників UNIX [7]. Для нас найбільш важливим у всій цій історії є існування двох базових ліній еволюції – лінії System V і лінії

BSD, оскільки в процесі навчання ми будемо зіштовхуватися з розходженнями в їхній реалізації.

Системні виклики і бібліотека libc

Основною постійно функціонуючою частиною операційної системи UNIX є її ядро. Інші програми (системні або користувацькі) можуть спілкуватися з ядром за допомогою системних викликів, які по суті 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, природньо, відрізняються від машини до машини, разом зі способом передачі параметрів і результатів між програмою і ядром. Однак з погляду програміста мовою С використання системних викликів нічим зовні не відрізняється від використання інших функцій стандартної ANSI бібліотеки мови С, таких як функції роботи з рядками `strlen()`, `strcpy()` і т.д. Стандартна бібліотека UNIX – `libc` – забезпечує С-інтерфейс до кожного системного виклику. Це приводить до того, що системний виклик виглядає як функція мовою С для програміста. Більше того, багато із уже відомих вам стандартних функцій, наприклад функції для роботи з файлами: `fopen()`, `fread()`, `fwrite()` при реалізації в операційній системі UNIX будуть застосовувати різні системні виклики. Напротязі курсу доведеться познайомитися з більшою кількістю різноманітних системних викликів і 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 використовуть значенням `NULL`. Для точного визначення причини помилки С-інтерфейс надає глобальну змінну `errno`, що описана у файлі `<errno.h>` разом 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` на 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 рядка (`'\n'`).

Поняття login і password

Операційна система UNIX 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, заводиться 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.

"Довідайтеся у свого системного адміністратора" встановлені реєстраційні імена і паролі.

Вхід у систему і зміна пароля

Прийшов час перший раз увійти в систему. Якщо в системі встановлена графічна оболонка поряд зі звичайними алфавітно-цифровими терміналами, найкраще це зробити з алфавітно-цифрового термінала або його емулятора. На екрані з'являється напис, що пропонує ввести реєстраційне ім'я, як правило, це `"login:"`. Набравши своє реєстраційне ім'я, натисніть клавішу `<Enter>`. Система запитає пароль, який відповідає введеному імені, видавши спеціальне запрошення – зазвичай `"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. **"Довідайтеся у свого системного адміністратора"**, яка команда для цього використовується на вашій обчислювальній системі (найчастіше це команда `passwd` або `yppasswd`). У більшості UNIX-подібних систем потрібно, щоб новий пароль мав не менше шести символів і містив, принаймні, дві не букви і дві не цифри. **"Довідайтеся у свого системного адміністратора"**, які обмеження на новий пароль існують у вашій операційній системі.

Придумайте новий пароль і добре його запам'ятаєте, 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. **"Довідайтеся у свого системного адміністратора"**, що потрібно зробити, якщо ви забули 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 UNIX.

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

В операційній системі UNIX існують три базових поняття: **"процес"**, **"файл"** і **"користувач"**. З поняттям "користувач" щойно ознайомилися і будемо користуватися надалі при вивченні роботи операційної системи UNIX. Поняття "процес" характеризує динамічну сторону того, що відбувається в обчислювальній системі, воно буде детально обговорюватися на лекції 2 і в описі наступних семінарів. Поняття "файл" характеризує статичну сторону обчислювальної системи.

З попереднього досвіду роботи з обчислювальною технікою ви вже маєте деяке поняття про файл, як про іменований набір даних, що зберігається де-небудь на магнітних дисках або стрічках. Для нашого сьогоднішнього обговорення нам досить такого розуміння, щоб розібратися в тому, як організована робота з файлами в операційній системі UNIX. Детальніший розгляд поняття "файл" і організації файлових систем для операційних систем у

цілому буде наведено в лекції 11 і лекції 12, а також на семінарах 11-12, присвячених організації файлових систем в UNIX.

Всі файли, доступні в операційній системі UNIX, 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 якої-небудь директорії. Директорії у свою чергу можуть входити до складу інших директорій. Допускається існування порожніх директорій, у які не входить жоден файл, і жодна інша директорія (див. рис. 1.1). Серед всіх директорій існує тільки одна директорія, що не входить до складу інших директорій – її прийнято називати **коренева директорія (кореневий каталог)**. На початковому рівні пізнання UNIX ми можемо вважати, що у файловій системі UNIX є присутніми, принаймні, два типи файлів: звичайні файли, які можуть містити тексти програм, код для виконання, дані і т.д. – їх прийнято називати **регулярними файлами**, і директорії.

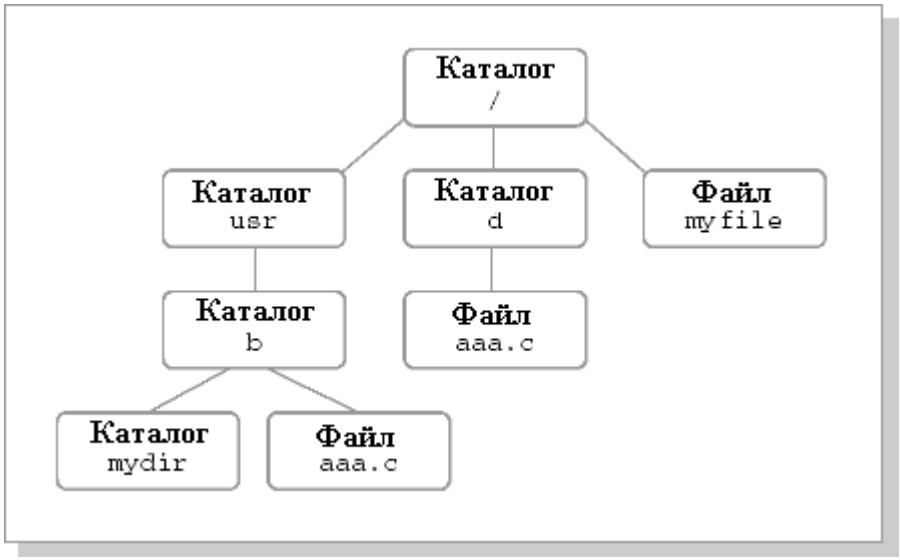

Рис. 1.1. Приклад структури файлової системи

Кожен файл (регулярний або директорія) 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 C.
- Не можна використовувати символ '/'.

Від себе додамо, що також небажано застосовувати символи "зірочка" – "*", "знак запитання" – "?", "лапки" – "\"", "апостроф" – "\'", "пробіл" – " " і "зворотній слеш" – "\\" (символи записані в стандарті запису символьних констант мови C).

Єдиним винятком є коренева директорія, що **завжди** має ім'я "/". Ця ж директорія за цілком зрозумілими причинами є єдиним файлом, що повинен мати унікальне ім'я у всій файловій системі. Для всіх інших файлів імена повинні бути унікальними тільки в рамках тієї директорії, у яку 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**.

Давайте подумки побудуємо шлях від кореневої вершини дерева файлів до файлу, що цікавить нас, і випишемо всі імена файлів (тобто вузлів дерева), що зустрічаються на нашому шляху, наприклад, "/ usr b aaa.c". У цій послідовності першим буде завжди стояти ім'я кореневої директорії, а останнім – ім'я файлу, що цікавить нас. Відокремимо імена вузлів

один від одного в цьому записі не пробілами, а символами "/", за винятком імені кореневої директорії і наступного імені директорії ("/usr/b/aaa.c"). Отриманий запис однозначно ідентифікує файл у всій логічній конструкції файлової системи. Такий запис і одержав назву повного імені файлу.

Поняття про поточну директорію. Команда pwd. Відносні імена файлів

Повні імена файлів можуть містити в собі досить багато імен директорій і бути дуже довгими, а тому з ними не завжди зручно працювати. У той же час, існують такі поняття як поточна або робоча директорія 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, одна з директорій у логічній структурі файлової системи призначається поточною або робочою для даної програми. Довідатися, яка директорія є поточною для вашого командного інтерпретатора, можна за допомогою команди операційної системи `pwd`.

Команда pwd

Синтаксис команди

`pwd`

Опис команди

Команда `pwd` виводить повне ім'я поточної директорії для запущеного командного інтерпретатора.

Знаючи поточну директорію, ми можемо побудувати шлях по графу файлів від поточної директорії до файлу, що цікавить нас. Запишемо послідовність вузлів, які зустрінуться на цьому шляху, у такий спосіб. Вузол, що відповідає поточній директорії, 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 директорія "/d" , то для файлу "/usr/b/aaa.c" відносним іменем буде "../usr/b/aaa.c", а якщо робочий каталог – це директорія "/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 одиночним 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 робочою директорією його робочу директорію, якщо всередині цих програм не змінити її розташування за допомогою спеціального системного виклику.

Домашня директорія користувача і її визначення

Для кожного нового користувача в системі заводиться спеціальна директорія, що стає поточною одразу після його входу в систему. Ця директорія одержала назву домашньої директорії користувача. Скористайтеся командою `pwd` для визначення своєї домашньої директорії.

Команда man - універсальний довідник

Під час вивчення операційної системи UNIX вам часто буде потрібна інформація про те, що робить та або інша команда або системний виклик, які в них параметри та опції, для

чого призначені деякі системні файли, який їхній формат і т.д. Ми постаралися, у міру можливості, включити описи більшості використовуваних у курсі команд і системних викликів у наш текст. Однак іноді для одержання більш повної інформації ми відсилаємо читачів до UNIX Manual – посібника з операційної системи UNIX. На щастя, 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, варто нажати 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, яка цікавить вас, необхідно задати утиліті `man` категорію, до якої належить ця інформація (номер розділу). Розподіл інформації на категорії може злегка відрізнятися у різних версіях UNIX. В Linux, наприклад, прийнято наступний поділ:

1. Виконувані файли, або команди інтерпретатора.
2. Системні виклики.
3. Бібліотечні функції.
4. Спеціальні файли (зазвичай файли пристроїв) – що це таке, ви довідаєтеся на семінарах 13-14.
5. Формат системних файлів і прийняті умови.
6. Ігри (зазвичай відсутні).
7. Макропакети і утиліти – такі як man.
8. Команди системного адміністратора.
9. Підпрограми ядра (нестандартний розділ).

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 одержати коротшу інформацію про команду

```
ls --info
```

Одержати підказку про команду можна за допомогою команди whatis

```
whatis ls
```

Команди cd – для зміни поточної директорії і ls – для перегляду вмісту директорії

Для зміни поточної директорії 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 директорії, яку вам потрібно зробити поточною. Команда `cd` без параметрів зробить поточною директорією вашу домашню директорію.

Переглянути вміст поточної або будь-якої іншої директорії можна, скориставшись командою `ls` (від list). Якщо ввести її без параметрів, ця команда роздрукує вам список файлів, що перебувають у поточній директорії. Якщо ж як параметр задати повне або відносне ім'я директорії:

```
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
```

або

```
ls -a ім'я_директорії
```

У команди `ls` існує також багато інших опцій, частина з яких ми ще розглянемо на семінарах. Для одержання повної інформації про команду `ls` скористайтеся утилітою `man`.

Переміщення структурою файлової системи

Користуючись командами `cd`, `ls` і `pwd`, переміщайтесь 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 наприкінці подорожі повернутися у свою домашню директорію.

Команда cat і створення файлу. Перенапрямлення вводу і виводу

Ми вміємо переміщатися 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.

Увага! 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` (опис її використання ви знайдете в UNIX Manual). Команда `cat` буде нам цікава з іншого погляду.

Якщо ми 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 перенапрямлення 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 перенапрямлення 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 поточної директорії, якщо виконаєте команду `ls -a` з перенапрямленням 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> і <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 безліч різних опцій. Подивитися її повний опис можна в UNIX Manual.

Відзначимо, що поряд з перенапрямленням 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 перенапрямлення потоків даних можна подивитися в UNIX Manual для командного інтерпретатора.

Створення файлу за допомогою команди cat

Переконайтеся, що ви перебуваєте у своїй домашній директорії, і створіть за допомогою команди `cat` новий текстовий файл. Перегляньте його вміст.

Найпростіші команди роботи з файлами – cp, rm, mkdir, mv

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

Для створення нової піддиректорії 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 створюваної директорії. 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

Даний опис не є повним описом команди cp, а коротким вступом до її використання. Для одержання повного опису команди зверніться до UNIX Manual.

Команда cp у формі

```
cp файл_джерело файл_призначення
```

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

Команда cp у формі

```
cp файл1 файл2 ... файлN дир_призначення
```

служить 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
```

служить 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 сору). Команда cp уміє 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 видалення, 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 сукупність 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 видалення файлів або директорій застосовується команда rm (скорочення від remove). Якщо ви хочете видалити 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 видалити. Замість імен файлів можуть використовуватися їхні шаблони. Якщо ви хочете видалити одну або декілька директорій разом з їхнім вмістом (рекурсивне видалення), то до команди додається опція -r:

```
rm -r дир1 дир2 ... дирN
```

де дир1, дир2, ... дирN – повні або відносні імена директорій, які потрібно видалити. Замість безпосередньо імен директорій також можуть використовуватися їхні шаблони. У команди rm є ще набір корисних опцій, які описані в UNIX Manual. Насправді процес видалення файлів не такий простий, яким здається на перший погляд. Детальніше він буде розглянутий на семінарах 11-12, коли ми будемо обговорювати операції над файлами в операційній системі UNIX.

Команда mv

Синтаксис команди

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

Опис команди

Даний опис не є повним описом команди mv, а є коротким вступом до її використання. Для одержання повного опису команди звертайтеся до UNIX Manual.

Команда mv у формі

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

служить 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
```

служить 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 видалення файлів і директорій варто користуватися з обережністю. Вилучену інформацію відновити неможливо. Якщо ви системний адміністратор і ваша поточна директорія **– це коренева директорія, будь ласка, 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.

Історія редагування файлів – ed, vi

Отримані знання вже дозволяють нам досить вільно оперувати файлами. Але що нам робити, якщо буде потрібно змінити вміст файлу, відредагувати його?

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. Так, наприклад, для заміни першого сполучення символів "ra" на "ru" в одинадцятому рядку файлу, що редагується, треба було б ввести команду

```
11 s/ra/ru
```

Редактор ed[1] був пострічковим редактором. Згодом з'явився екранний редактор – vi[2], 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 віддаленою машиною через повільний канал зв'язку) є єдиним засобом, що дозволяє віддалено редагувати файл.

Система Midnight Commander – mc. Вбудований в mc редактор і редактор joe

Напевно, ви вже переконалися в тому, що робота в UNIX 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 UNIX. 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, і екранний редактор joe. Інформацію про них можна знайти в UNIX Manual. Більшими можливостями володіють багатофункціональні текстові редактори, наприклад, emacs[3].

Увійдіть в mc і спробуйте переміщатися по директоріях, створювати і редагувати файли.

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

Як уже говорилося, для входу в операційну систему UNIX 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 інші – у кілька груп.

Команда chown

Синтаксис команди

```
chown owner файл1 файл2 ... файлN
```

Опис команди

Команда `chown` призначена для зміни власника (хазяїна) файлів. Даний опис не є повним описом команди, а адаптованим стосовно даного курсу. Для одержання повного опису звертайтеся до UNIX Manual. 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

1 Опис редактора ed можна знайти, наприклад, в [11]. В електронному виді опис є в документі http://cs.mipt.ru/docs/comp/rus/os/unix/user_guide/unixuser/gl6_1.htm.

2 Опис редактора vi теж можна знайти в [11]. В електронному виді опис є в документі http://cs.mipt.ru/docs/comp/rus/os/unix/user_guide/unixuser/gl7_1.htm.

3 В електронному виді опис редактора emacs див. у документі http://cs.mipt.ru/docs/comp/rus/os/unix/user_guide/emacs/index.html.

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. Даний опис не є повним описом команди, а адаптованим стосовно даного курсу. Для одержання повного опису звертайтеся до UNIX Manual. 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.

Як ми бачимо, 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 (як, втім, і невиконуваний регулярний файл), право доступу на виконання для директорій змінює значення: наявність цього права дозволяє одержати додаткову інформацію про файли, що входять у каталог (їхній розмір, хто їхній власник, дата створення і т.д.). Без цього права ви не зможете ні читати вміст файлів, що лежать 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 видаляти в ній файли, перейменовувати їх. Відзначимо, що для видалення файлу досить мати права запису і виконання для директорії, у яку входить даний файл, незалежно від прав доступу до самого файлу.

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 ліворуч містить імена користувачів власників файлів, а четвертий стовпчик ліворуч – 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 '-', те 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.

Виконайте команду `ls -al` для своєї домашньої директорії і проаналізуйте її вивід.

Команда chmod

Синтаксис команди

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

Опис команди

Команда `chmod` призначена для зміни прав доступу до одного або декількох файлів. Даний опис не є повним описом команди, а адаптованим до даного курсу. Для одержання повного опису звертайтеся до UNIX Manual. 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.

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

Створіть новий файл і подивіться на права доступу до нього, встановлені системою при його створенні. Чим керується операційна система при призначенні цих прав? Вона використовує для цього маску створення файлів для програми, що створює файл. Початково для програми-оболонки вона має деяке значення за замовчуванням.

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 рівним 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 зміна маски за допомогою команди umask вплине на атрибути доступу до знову створюваних файлів 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 встановлено за замовчуванням.

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.

Компіляція програм мовою C в UNIX і запуск їх на виконання

Тепер ми готові до того, щоб написати першу програму в нашому курсі. Залишилося тільки навчитися компілювати програми мовою C і запускати 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.

"Довідайтеся у свого системного адміністратора", як називається компілятор мови C для вашої операційної системи і які опції він має. Завичай у всіх версіях UNIX є компілятор з ім'ям `cc`, що підтримує опцію `-o`.

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

Написання, компіляція і запуск програми з використанням системних викликів getuid() і getgid()

Напишіть, відкомпілюйте і запустіть програму, яка друкувала б ідентифікатор користувача, що запустив програму, і ідентифікатор його групи.