# Робота №2. Система розмежування доступу в UNIX і

# Linux, права доступу до файлів і керування ними

## Мета

### Оволодіння практичними навичками керування

**правами доступу до файлів і їхній аналіз в ОС UNIX та Linux**

## Завдання для самостійної підготовки

1. Вивчити (крім довідкового матеріалу, наведеного далі, рекомендується [2, пп. 1.2.3, 1.2.4]:
   * поняття “право доступу” і “метод доступу”;
   * атрибути доступу до файлів в UNIX;  перегляд інформації про права доступу;  зміна прав доступу.
2. Детально ознайомитись з довідкової системи man з такими командами UNIX (Linux): **ls –l, chmod, chown, umask[[1]](#footnote-1), setfacl, getfacl.**
3. Відповідно до завдання підготувати послідовність команд для його виконання.

## Довідковий матеріал

Складовою кожної захищеної системи є система розмежування доступу. В системі UNIX контролюється доступ до файлів. Кожний файл має свого власника, а також відноситься до визначеної групи. Вся детальна інформація про файл, що включає тип файлу, ідентифікатори власника файлу та його групи, розмір файлу, час останньої модифікації файлу, інформацію щодо прав доступу до файлу, а також про його розміщення (номери блоків), міститься не в каталогах, як це зроблено в багатьох інших файлових системах, в тому числі FAT (MS-DOS, Windows), а в системній таблиці так званих індексних дескрипторів (i-node). Безпосереднє звернення до цієї таблиці заборонене. Каталоги містять лише ім’я файлу та індекс – посилання на відповідний запис в таблиці індексних дескрипторів. Саме завдяки такій організації кожний файл у файловій системі UNIX може мати кілька абсолютно рівноправних імен (жорстких посилань), що в загальному випадку містяться в різних каталогах (це було розглянуто в Роботі №1). Інформацію з таблиці індексних дескрипторів можна переглянути командою ls –l. При цьому для кожного файлу інформація видається у вигляді одного рядка. Перший символ – тип файлу: **‘–’** – звичайний файл (текстовий або бінарний), **d** –каталог (*directory*), **l** – символічне посилання (*link*), **c** – символьний пристрій (*character device*), **b** – блочний пристрій (*block device*). Наступні дев’ять символів описують права доступу до файлу (див. далі). Далі надається інформація про кількість жорстких посилань на файл, власника файлу, групу, розмір файлу, дату останньої модифікації і останнє поле – ім’я файлу.

UNIX реалізує дискреційну[[2]](#footnote-2) модель розмежування доступу, в якій для кожного файлу визначається, які права мають всі користувачі на доступ до файлу. Для цього з кожним файлом асоціюється спеціальна інформація, що містить ідентифікатор власника файлу, ідентифікатор групи файлу і права доступу до файлу. Права доступу поділяються на 3 частині: права власника, права групи і права всіх інших. У кожному класі користувачів виділено по 3 біти, що відповідають правам читання, запису й виконання (r, w, x), відповідно. Для каталогів право виконання трактується як право доступу до таблиці індексних дескрипторів на читання і запис, не маючи цього права неможливо зробити поточним цей каталог чи будь-який з його підкаталогів, неможливо ознайомитись і змінити права доступу до об’єктів цього каталогу, можна тільки переглядати його вміст, якщо є право читання. Навіть маючи право запису, без права виконання не можна змінити вміст каталогу. Навпаки, якщо є право на виконання, але не встановлено право на читання для каталогу, то неможливо переглянути вміст каталогу, але можна заходити в його підкаталоги чи звертатись до файлів, що містяться в ньому, якщо знати їхні імена.

Є ще три біти, що впливають на доступ до файлу. Перший з них називається SUID (Set-User-ID-on-execution bit), і якщо він установлений для файлу, що виконується, то цей файл для будь-якого користувача виконується не з його правами, а з правами власника цього файлу — зазвичай, власником є root. Другий біт — SGID (Set-Group-ID-on-execution bit), якщо він встановлений, то ця програма виконується для будь-якого користувача із правами членів групи цього файлу. Для каталогів SGID визначає, що для усіх файлів, створюваних у цьому каталозі, ідентифікатор групи буде встановлений такий же, як у каталогу, а не такий, що визначає первинну групу користувача (ці правила залежать від версії UNIX). Останній біт називається Sticky, він використовується тільки для каталогів і визначає, що користувачі, які мають право на записування в каталог, не мають права видаляти чи перейменовувати файли інших користувачів у цьому каталозі. Це необхідно для каталогів загального використання, наприклад /tmp.

Права доступу до файлів задаються при створенні файлу і в подальшому можуть бути змінені командою **chmod <***нові права***> <***файл(и)***>**

**<**нові права**>** задаються двома способами. Перший – символьний. Використовуються такі позначення: **u** – власник файлу (user), **g** – група (group), **o** – всі інші (others), **a** – всі категорії користувачів одночасно (all). Після категорії користувачів задається дія: ‘**+**’ – додати права до існуючих, ‘**-**’ – відібрати права (якщо існують), ‘**=**’ – встановити права замість існуючих. Далі позначаються права: **r** – зчитувати (Read), **w** – записувати (Write), **x** – виконувати (eXecute). Можна формувати список зміни прав, розділяючи окремі категорії користувачів комами (без пробілів). Наприклад, команда **chmod u+rw,g=rx,o-w my\_file**

встановить для користувача права на зчитування і записування (право на виконання для користувача буде залежати від того, чи було воно встановлено раніше), для групи встановить права на зчитування і виконання (незалежно від того, які права були раніше), а для всіх інших гарантовано заборонить записування (права на зчитування і виконання будуть встановлені в залежності від того, чи були вони встановлені раніше).

Другий – числовий спосіб задавання прав доступу – зручніше використовувати, коли треба встановлювати нові права незалежно від попередньо встановлених. При цьому використовується представлення у восьмеричній системі числення, яке легко зрозуміти з наведеної таблиці:

|  |  |  |
| --- | --- | --- |
| **Восьмеричне представленн я** | **Двійкове представленн я** | **Права доступ у** |
| 0 | 000 | **--**  **-** |
| 1 | 001 | **-x** |
| 2 | 010 | **w-** |
| 3 | 011 | **wx** |
| 4 | 100 | **r-**  **-** |
| 5 | 101 | **r-**  **x** |
| 6 | 110 | **rw**  **-** |
| 7 | 111 | **rw**  **x** |

Наприклад, команда **chmod 751 my\_file**

встановлює такі права доступу до файлу my\_file: власнику – зчитування, записування, виконання, групі – зчитування і виконання, а всім іншим – лише виконання. Команда ls -l покаже для цього файлу таку інформацію про права доступу: -rwxr-x--x (перший “мінус” означає, що це звичайний файл).

За умовчанням для всіх нових файлів, що створюються, встановлюються такі права: для файлів 666, тобто rw-rw-rw-, для каталогів – 777, тобто rwxrwxrwx (ми вже бачили вище, що наявність права на виконання дуже важлива для каталогів). Є особлива команда – umask, яка дозволяє зменшити права доступу, що встановлюються для нових файлів. Параметром цієї команди є восьмерична маска, аналогічна тій, що використовується при числовому способі задавання параметрів команди chmod, але у випадку команди umask права доступу, що задані нею, будуть відніматись від прав доступу, що були задані по замовчуванню для вже утвореного файлу. Команда umask не може додавати прав, тому нові файли ніколи автоматично не отримають право на виконання, в разі необхідності його треба буде додати вручну. Дія команди umask розповсюджується на всі файли всіх типів, що утворюються протягом поточної сесії користувача після цієї команди. Наприклад, після команди umask123 всі файли будуть утворюватись з параметрами доступу rw-r--r-- (644), а всі каталоги – rw-r-xr-- (654). За замовчуванням рекомендується використовувати команду umask 22 (інтерпретується як umask 022).

Для зміни власника файлу існує команда chown. З міркувань безпеки, ця команда дозволяє встановлювати власником файла будь-кого лише системному адміністратору (**root**). Інші користувачі можуть лише привласнити файл собі, якщо вони мають на це права, встановлені правами доступу до файлу і каталогу.

#### С писки ACL

Стандартні права доступу в UNIX і Linux не мають достатньої гнучкості і в деяких випадках не дозволяють встановити потрібні правила доступу. Розглянемо приклади.

Припустимо, у нас є власник файлу з повними правами доступу і потрібно надати доступ на зчитування і записування у файл ще одному користувачеві (всім решті — лише на зчитування). Для цього можна створити нову групу і вказати, що ця група володіє файлом.

##### addgroup <new\_group> chgrp <new\_group> <this\_file> chmod g=rw <this\_file>

Далі треба відредагувати файл /etc/group, додавши у нову групу потрібного нам користувача.

А тепер припустимо, що нам треба надати права зчитування, записування та запуску на виконання двом різним користувачам, ще двом – права зчитування й записування, ще одному – тільки зчитування, а усім решті – взагалі не дати доступу до цього файлу. Класична система розмежування доступу UNIX не дозволяє налаштувати такі права.

Але на певному етапі розвитку в UNIX-подібних системах (спочатку в Solaris, далі – у FreeBSD, а згодом і в Linux) впровадили потрібний механізм. Це списки керування доступом до об’єктів файлової системи, access control lists (ACLs). Іноді їх називають списками контролю доступу, що є неточним перекладом з англійської. За допомогою ACL можна призначити як права доступу власника файлу, групи файлу і усіх інших, так і окремі права для певних користувачів і груп, а також максимальні права доступу за умовчанням, які дозволено мати будь-якому користувачу.

Далі ми будемо називати ACL для файлів і каталогів «розширеними правами доступу». Розширені права доступу встановлюються командою setfacl. Аргументи команди задаються як ряд полів, що розділені двокрапками:

##### <entry\_type>:[<uid>|<gid>]:<perms>

Тут <entry\_type> – тип розширеного права доступу (**u** – user, **g** – group, **o** – other, **m** – mask), <uid> і <gid> – ідентифікатори користувача і групи, відповідно, <perms> – права доступу, що призначають (**r** – read, **w** – write, **x** – execute).

**Розширені права доступу до файлу можуть бути такими, як вказано у Таблиці:**

|  |  |
| --- | --- |
| **u[ser]::<perms>** | права доступу власника файлу |
| **g[roup]::<perms>** | права доступу групи файлу |
| **o[ther]:<perms>** | права усіх інших |
| **m[ask]:<perms>** | маска ACL |
| **u[ser]:<uid>:<perms>** | права доступу вказаного користувача; в якості uid можна |
|  | вказати і UID, і ім’я (login) користувача |
| **g[roup]:<gid>:<perms>** | права доступу вказаної групи; в якості gid можно вказати і GID, і ім’я групи |

Маска задає максимальні права доступу для усіх користувачів, за винятком власника і групи. Призначення маски є найшвидшим способом обмежити фактичні (ефективні) права доступу.

Наприклад, маска **r — —** вказує, що користувачі і групи не можуть мати більших прав, ніж просто зчитування, навіть якщо їм призначені права доступу на зчитування й записування.

Розглянуті вище розширені права доступу можуть застосовуватись і до файлів, і до каталогів. Крім них, існують специфічні права доступа до каталогу, а саме права доступу за умовчанням. Файли і підкаталоги, що будуть створювати у цьому каталоге, будуть автоматично отримувати такі ж росширені права доступу, які задані в розширених правах доступу за умовчанням до цього каталогу.

### Розширені права доступу до каталогів за умовчанням

|  |  |
| --- | --- |
| **d[efault]:u[ser]::<perms>** | права власника файлу за умовчанням |
| **d[efault]:g[roup]::<perms>** | права групи файлу за умовчанням |
| **d[efault]:o[ther]:<perms>** | права решти користувачів за умовчанням |
| **d[efault]:m[ask]:<perms>** | маска ACL за умовчанням |
| **d[efault]:u[ser]:<uid>:<perms>** | права доступу за умовчанням для вказаного користувача; в якості uid можна вказати і UID, і ім’я (login) користувача |
| **d[efault]:g[roup]:<gid>:<perms>** | права доступу за умовчанням для вказаної групи; в якості gid можна вказати і GID, і ім’я групи |

Встановлюючи права доступу за умовчанням для конкретних користувачів і груп (два останніх рядка таблиці), необхідно також встановити права доступу за умовчанням для власника і групи файлу, усіх інших, а також маску ACL (тобто, чотири перших рядка таблиці у цьому випадку є обов’язковими).

Встановлення розширених прав доступу здійснюється командою setfacl з ключем –s (set). Ключ -m вказує на додавання прав доступа замість їх заміны.

Ефективні (тобто ті, що будуть насправді застосовані) права доступу користувачів визначаються як їхніми персональними правами доступу до цього файлу, так і встановленою маскою. З персональних прав і маски обираються найбільш строгі обмеження.

Команда ls –l не показує розширені права доступу, але показує наявність створеного ACL – символ «+» (плюс) після стандартних прав доступу.

Переглянути розширені права доступу можна командою getfacl.

Детальнішу інформацію про параметри команд setfacl і getfacl можна отримати з man.

## Завдання до виконання

1. Створіть каталог lab\_2.
2. Скопіюйте в каталог lab\_2 файл /bin/cat під назвою my\_cat.
3. За допомогою файлу my\_cat, що знаходиться в каталозі lab\_2, перегляньте уміст файлу .profile (ви знаходитесь у домашньому каталозі).
4. Перегляньте список файлів у каталозі lab\_2. Потім перегляньте список усіх файлів, включаючи приховані, з повною інформацією про файли. Зверніть увагу на права доступу, власника, дату модифікації файлу, що ви тількино скопіювали. Потім перегляньте цю інформацію про оригінальний файл (той, який копіювали) і порівняйте два результати.
5. Змініть права доступу до файлу my\_cat так, щоб власник міг тільки читати цей файл.
6. Переконайтеся в тому, що ви зробили ці зміни і повторіть п.3.
7. Визначте права на файл my\_cat таким чином, щоб ви могли робити з файлом усе, що завгодно, а всі інші — нічого не могли робити.
8. Поверніться в домашній каталог. Змініть права доступу до каталогу lab\_2 так, щоб ви могли його тільки читати.
9. Спробуйте переглянути простий список файлів у цьому каталозі. Спробуйте переглянути список файлів з повною інформацією про них. Спробуйте запустити і видалити файл my\_cat з цього каталогу.
10. Поясніть отримані результати. Результати виконання

п.8 можуть бути різними в різних версіях UNIX, зокрема, Linux і FreeBSD. Прокоментуйте отримані результати у висновках.

1. За допомогою команди su <user name>, завантажтесь в систему, користуючись обліковим записом іншого користувача. (Вам потрібно знати пароль цього користувача.) Спробуйте отримати доступ до Вашого каталогу lab\_2. Перевірте, чи правильно зроблено завдання попереднього пункту. Створіть каталог lab\_2\_2.
2. Знову завантажтесь в систему, користуючись своїм обліковим записом[[3]](#footnote-3). Спробуйте зробити власником каталогу lab\_2 іншого користувача. Спробуйте зробити себе власником каталогу lab\_2\_2. Поясніть результати.
3. Зайдіть у каталог lab\_2. Зробіть так, щоб нові створені файли і каталоги діставали права доступу згідно Таблиці. Створіть новий файл і каталог і переконайтеся в правильності ваших установок.

|  |  |  |
| --- | --- | --- |
| **варіант** | **Права для файлів** | **Права для**  **каталогів** |
| 1 | **644** | **754** |
| 2 | **664** | **774** |
| 3 | **6-4** | **7-5** |
| 4 | **62-** | **73-** |
| 5 | **644** | **745** |
| 6 | **664** | **764** |
| 7 | **6-4** | **715** |
| 8 | **62-** | **63-** |
| 9 | **644** | **744** |
| 10 | **664** | **765** |

1. Поверніть собі права читати, писати, та переглядати вміст каталогів.
2. Створіть у каталозі lab\_2 каталог acl\_test та у ньому файли file1, file2. Після час створення file1 додайте у нього довільний текст.
3. Виведіть ACL для file1
4. Змінить права доступу на file1 так, щоб тільки власник мав право на читання.
5. Увійдіть до системи під іншим обліковим записом та спробуйте прочитати вміст file1. Що отримаємо?

Поверніться до свого облікового запису.

1. За допомогою команди setfacl додайте право на читання іншому обраному користувачу для file1. Перевірте, що створився новий ACL для file1.
2. Увійдіть до системи під іншим обліковим записом та спробуйте прочитати вміст file1. Що отримаємо?

Поверніться до свого облікового запису.

1. За допомогою команди setfacl встановіть значення маски таким чином щоб дозволити читати вміст file1 іншому користувачу. Виведіть ACL для file1
2. Увійдіть до системи під іншим обліковим записом, та спробуйте прочитати вміст file1. Ви повинні мати таку змогу.

1. Для команди **umask** не існує окремої сторінки **man**, оскільки це – вбудована команда **shell**, тому її опис (хоча й не повною мірою детальний) міститься в сторінках **man** для кожного з **shell** (**sh**, **bash**, тощо). [↑](#footnote-ref-1)
2. Більш докладну інформацію про моделі розмежування доступу ви отримаєте з курсів “Основи кібербезпеки” і “Захист інформації в

   інформаційно-комунікаційних системах” [↑](#footnote-ref-2)
3. Ви можете одночасно користуватись різними обліковими записами, використовуючи для цього різні віртуальні консолі в текстовому режимі, або різні вікна терміналів в графічній багатовіконній системі. Віртуальні консолі в текстовому режимі переключаються комбінаціями клавіш Alt+F1 – Alt+F8. [↑](#footnote-ref-3)