Prawa dostępu i prawa własności do plików w Linux.

W systemach sieciowych (czyli takich, do których dostęp ma wiele różnych osób), definiowanie praw dostępu do danych wpływa na bezpieczeństwo systemu oraz na prywatność danych użytkowników.

Graficzne menedżery plików również umożliwiają odczyt oraz ewentualne modyfikacje praw do plików.

Użytkownicy i grupy

Użytkownik to prywatne konto w systemie posiada własne pliki, na których pracuje.

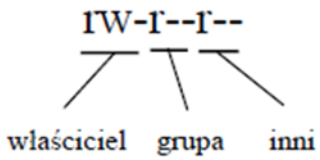
Grupa to pewien zbiór użytkowników, których łączy jakaś wspólna cecha (np. dostęp do Internetu, do drukarki czy dostępu do plików).

"Wszystko jest plikiem" - w myśl tej idei uważamy m.in. katalogi za pliki. Pliki (w potocznym znaczeniu tego słowa) nazywamy plikami zwyczajnymi.

Prawa dostępu

Prawa do plików możemy zdefiniować osobno dla właściciela pliku (z ang. user), grupy, do której plik należy (group) oraz pozostałych użytkowników (others). Prawa, które możemy nadawać to: prawo odczytu, zapisu oraz wykonania.

Uwaga - o ile w przypadku większości plików prawa dostępu działają całkowicie zgodnie ze swoimi nazwami, o tyle w przypadku katalogów jest trochę inaczej. Prawo do odczytu oznacza możliwość wyświetlenia zawartości folderu na ekranie, zapis oznacza tworzenie i modyfikowanie plików, natomiast prawo wykonania to nic innego jak możliwość wejścia do danego katalogu.



r – prawo czytania
w – prawo pisania
x – prawo wykonywania

Prawa te określone za pomocą kodu numerycznego mają postać:

	u	g	o
r	4	4	4
w	2	–	–
x	–	–	–
	6	4	4

⇒ 644

u = user – właściciel pliku
g = group – członkowie grupy
o = other – pozostali

r =4 nadanie praw, 0 – brak
w =2 nadanie praw, 0 – brak
x =1 nadanie praw, 0 – brak

< zapisać w zeszycie.

Sprawdzanie praw

Aby obejrzeć, w jaki sposób są przypisywane plikom należy posłużyć się komendą **ls -** potrafi ona wyświetlić zawartość danego folderu. Parametr -l włączy tryb szczegółowy, wyświetlający więcej informacji o plikach. Zatem komenda:

ls -l

może zwrócić na przykład:

-rw-r--r-- 1 uczen users 256 2015-12-16 dysk.sh

drwxr-xr-x 3 uczen users 4096 03-17 17:36 Obrazy

Wynik możemy podzielić na siedem kolumn rozdzielonych spacjami – opisze je zaczynając od końca. Ostatnia to nazwa pliku, szósta - data ostatniej modyfikacji, piąta - rozmiar interesującego nas obiektu. Kolumna trzecia i czwarta zawierają nazwę właściciela pliku oraz grupy, do której on należy.
W drugiej kolumnie podawana jest ilość odwołań do pliku - a żeby być bardziej precyzyjnym, liczba dowiązań twardych.

Dla zwykłego pliku ilość dowiązań twardych wynosi po prostu 1. W przypadku katalogów musimy wiedzieć, że oprócz samego folderu odnosi się do niego także wirtualny katalog o nazwie **.** (ten symbol oznacza po prostu katalog bieżący), umiejscowiony w jego wnętrzu. To oznacza, że dla zwyczajnego katalogu liczba dowiązań twardych wynosi przynajmniej 2. „Przynajmniej" – ponieważ każdy podkatalog to kolejne dowiązanie, powiększające wartość pokazywaną przez polecenie ls.

Z perspektywy praw dostępu najbardziej interesuje dziesięciu znaków pierwszej kolumny:

pierwszy znak - oznacza typ pliku, do najważniejszych należą:

– - zwykły plik

d - katalog

l - dowiązanie symboliczne

kolejne trzy bloki (po trzy znaki w każdym) oznaczają kolejno prawa dla: właściciela, grupy oraz pozostałych użytkowników - jeśli dane uprawnienie występuje jest oznaczone odpowiednią literą, ich brak przedstawiony jest znakiem –

read - prawa odczytu

write - prawa zapisu

execute - prawa wykonania

W przypadku skryptu dysk.sh, prawa do odczytu i zapisu ma właściciel pliku, wszyscy inni użytkownicy mogą go natomiast odczytać. W przypadku katalogu Obrazy pełne prawa ma właściciel, pozostali użytkownicy mogą jedynie obejrzeć jego zawartość i wejść do niego.

Zmiana praw

Ponieważ potrafimy już sprawdzić prawa dostępu dla danego pliku, czas zająć się ich modyfikowaniem. Używamy do tego komendy chmod, a całe polecenie tworzymy podając:

symbol oznaczający użytkowników, których mają dotyczyć zmiany

- u - właściciel pliku
- g - grupa pliku
- - pozostali użytkownicy
- a - wszyscy użytkownicy, alternatywny zapis: ugo

znak definiujący zmianę praw

- + - nadanie praw

- – - usunięcie praw
- = - ustawienie jedynie praw podanych, pozostałe są usuwane

prawa, które będą zmienione

- r - prawo odczytu
- w - prawo zapisu
- x - prawo wykonania

Na końcu polecenia znaleźć się musi lista plików, których zmiany dotyczą.

Przyklad 1. Przykład zastosowanie komendy chmod

chmod u=rwx,go=rx ~

Czasem potrzebujemy zmienić prawa dla katalogu wraz z całą jego zawartością - warto wtedy posłużyć się parametrem -R, który wprowadza zmiany rekursywnie. Jak już wcześniej pisaliśmy prawo wykonania działa zupełnie inaczej dla katalogów i plików zwyczajnych. W tych pierwszych jest on jak najbardziej pożądany, z kolei dla tych drugich znacznie rzadziej (uruchamiamy tylko pliki binarne oraz skrypty, muzyki czy zdjęć już nie). Wyobraźmy sobie sytuację, kiedy chcielibyśmy udostępnić katalog wraz z jego zawartością grupie po minimalnych prawach. g=rx nie zda w takiej sytuacji egzaminu - każdy plik otrzyma prawa uruchomienia dla grupy, do której należy. Rozwiązaniem jest zastosowanie g=rX – wielkie "X". Takie polecenie nada prawa execute jedynie dla katalogów, pomijając pliki.

Po takiej porcji teorii czas na kilka przykładów:

Przykład 2.

chmod ug=rw .bashrc

Komenda nadaje właścicielowi i jego grupie prawa odczytu i zapisu do pliku .bashrc, pozbawiając równocześnie innych użytkowników jakichkolwiek praw.

Przykład 3.

chmod go= -R /boot

Tutaj z kolei odebraliśmy użytkownikowi i jego grupie wszelkie prawa do folderu /boot oraz jego zawartości.

Przykład 4.

chmod g-w,o+r .bashrc

W tym przykładzie grupa właściciela .bashrc straciła prawo zapisu tego pliku, natomiast pozostali użytkownicy zyskali możliwość odczytu tego pliku.

chmod - składnia absolutna (numeryczna)

Oprócz przedstawionej powyżej formy zapisu praw, polecenie chmod rozpoznaje także składnię numeryczną. Najbardziej popularny przykład to nieszczęsne chmod -R 777 /, przy pomocy którego początkujący użytkownicy próbują nadać sobie prawa administratora.

Najogólniej rzecz ujmując, trzy kolejne cyfry oznaczają prawa dla właściciela, grupy oraz pozostałych użytkowników, a ich wartość to cyfrowe oznaczenie odpowiadające poszczególnym prawom. Przykładowy plik dysk.sh miał prawa:

rw-r--r--

Żeby było prościej, rozbijmy to na trzy części odpowiadające uprawnieniom właściciela, grupy i pozostałych użytkowników:

rw- r-- r--

Wystarczy teraz wiedzieć, że do prawa wykonywalności przypisana jest liczba 1, do kolejnych praw (a więc po kolei - zapisu i odczytu) przypisana jest liczba dwukrotnie wyższa od poprzedniej, czyli:

odczyt – 4

zapis – 2

uruchomienie – 1

Jeśli równocześnie pojawia się więcej niż jedno prawo dostępu (np. odczyt i zapis, jak w tym przykładzie), wartości liczbowe po prostu sumujemy. Tak więc w przypadku opisywanego tu do bólu pliku dysk.sh, prawa w postaci absolutnej to 644.

W przypadku zastosowania takiej metody zapisu, poleceniu z Przykładu 2. równoważny będzie Przykład 5.

Przykład 5.

chmod 600 .bashrc

Prawa specjalne

Linux oferuje możliwość nadawania praw specjalnych. Do tej kategorii należą:

SetUID - ustaw identyfikator użytkownika - jeśli takie prawo zostało zdefiniowane dla pliku wykonalnego, uruchomiony w ten sposób proces będzie miał takie prawa jak właściciel programu,

SetGID - ustaw identyfikator grupy - analogicznie do wcześniejszego prawa, tutaj plik wykonalny zostanie uruchomiony z prawami grupy, która jest jego właścicielem,

sticky bit - bit zaczepienia - używany dla zabezpieczenia zawartości katalogu w sytuacji, w której prawa zapisu ma większa grupa użytkowników. Jeśli zdefiniujemy dla folderu takie prawo, zawartość tego katalogu będzie mógł modyfikować bądź usuwać tylko i wyłącznie jego właściciel.

W klasycznym zapisie informującym o prawach dostępu, prawa specjalne pojawiają się zawsze w miejscu normalnie definiującym prawa dotyczące wykonywania danego pliku. SetUID zajmie miejsce w informacji o prawach wykonania dla właściciela, SetGID – grupy, sticky bit – innych użytkowników. Ponieważ prawa specjalne „zajęły” nam miejsce, w którym znajdowała się informacja o prawie wykonania (lub jego braku), o tej kwestii dowiemy się odpowiednio dzięki małej lub wielkiej literze odpowiadającej za prawo specjalne. I tak, poniższy zapis:

-rws--------- 1 uczen users 256 2015-12-16 dysk.sh

oznacza, że dla pliku dysk.sh SetUID jest aktywny (odpowiada za to literka „s” w wyniku), a właściciel ma pełne prawa do pliku - prawo odczytu, zapisu oraz wykonania (o tym ostatnim informuje nas mała litera s).

Sytuacja w kolejnym wpisie będzie wyglądać nieco inaczej:

drwxrwxrwT 1 uczen users 256 2015-12-16 Obrazy

Jeśli chodzi o właściciela i jego grupę, wszystko jest jasne - posiadają pełne prawa. Dla innych użytkowników dostęp do zawartości folderu ogranicza sticky bit (tak, to ta litera „T”) i nie mają oni prawa wykonania (wejścia do katalogu), o czym informuje wielka litera definiująca sticky bit.

Znak	Numer	Nazwa	Pliki	Katalogi
t	1	Sticky bit bit „lepkości”	Nie dotyczy	Użytkownicy mogą kasować pliki tylko wtedy, gdy są ich właścicielami, użytkownikiem root albo właścicielem katalogu. Zwykle stosuje się do katalogu /tmp.
s	2	SGID (set GroupID) ustaw ID grupy	Kiedy program startuje, GroupID procesu ustawiany jest na GID grupy pliku.	Pliki tworzone w tym katalogu należą do grupy katalogu a nie użytkownika. Nowe katalogi dziedziczą bit SGID.
s	4	SUID(set UserID)	Kiedy program startuje, UserID	Nie dotyczy
		ustaw ID użytkownika	procesu ustawiany jest na UserID właściciela pliku.	

< zapisać w zeszycie.

Jak modyfikować prawa specjalne.

Podobnie jak w przypadku zwykłych praw, możemy użyć składni zwykłej:

Przykład 6.

chmod u+s system.sh

aktywuje SetUID dla skryptu system.sh

Przykład 7.

chmod g+s system.sh

oznacza nadanie SetGID.

Prawa specjalne można definiować także przy pomocy postaci absolutnej. W takiej sytuacji „zwyczajowe” trzy cyfry należy poprzedzić jeszcze jedną, będącą sumą praw specjalnych: dla SetUID będzie to 4, dla UserUID - 2, natomiast dla sticky bit - 1. O „sumie” praw specjalnych piszemy tylko dla

porządku – w praktyce używa się ich pojedynczo. Jednoczesne stosowanie SetUID i SetGID dla jakiegoś pliku wykonalnego nie ma sensu, a sticky bit jest wykorzystywany w przypadku katalogów.

Przykładowa komenda, która nadaje plikowi „skrypt” SetUID, pełne prawa dla właściciela i prawo wykonania dla grupy oraz pozostałych użytkowników:

Przykład 8.

chmod 4711 skrypt

Prawa specjalne nie są może często wykorzystywane, jednak niektóre pliki systemowe wykorzystują ich możliwości. Przykładem dotyczącym każdej instalacji Ubuntu może być chociażby wynik poniższego polecenia:

Przykład 9.

ls -l /usr/bin/passwd

-rwsr-xr-x 1 root root 37140 2010-04-20 13:14 /usr/bin/passwd

Zmiana właściciela i grupy pliku

Zmiany właściciela pliku może dokonać tylko użytkownik z prawami administratora (a więc root, względnie normalny użytkownik mogący korzystać z sudo). Właściciel pliku może co najwyżej zmienić grupę, do której plik należy - o ile sam jest członkiem nowej grupy, której chce przypisać dany plik.

Przeprowadzenie takiego zabiegu dla pliku „test” pokazują poniższe przykłady:

Przykład 10a.

chown użytkownik:grupa test

Przykład 10b.

chown użytkownik test

Przykład 10c.

chown :grupa test

W przykładzie 10a. zmodyfikowaliśmy zarówno właściciela jak i grupę, do której należy plik test. W kolejnych dwóch przykładach skupiliśmy się na zmianie tylko jednej informacji (albo właściciela pliku - przykład 10b., albo też grupy, do której dany obiekt należy - przykład 10c.). W takiej sytuacji w składni polecenia pomijamy po prostu zapis „użytkownik” względnie „: grupa”. Jeśli interesuje nas jedynie zmiana grupy, do której należy plik, możemy skorzystać z polecenia:

Przykład 11.

chgrp grupa test

Ta komenda służy wyłącznie do zmiany grupy, do której należy dany obiekt. Przykład 11. jest zatem równoważny poleceniu z przykładu 10c.

Jeżeli chcemy użyć któregoś z powyższych poleceń dla folderu wraz z jego zawartością, stosujemy parametr -R, dokładnie tak samo jak w przypadku komendy chmod

Definiowanie domyślnych praw dla plików

Każdy nowo utworzony plik czy katalog posiada z góry zdefiniowane wartości właściciela oraz grupy, a także prawa. W przypadku dwóch pierwszych właścicielem zostaje użytkownik, który stworzył plik lub katalog, natomiast grupą - domyślna grupa właściciela.
Prawa dostępu przypisane automatycznie świeżo tworzonym plikom, definiuje je maska praw.
Aktualną wartość maski uzyskamy poleceniem

umask

0022

Jak to odczytać? Na chwilę zapomnijmy o pierwszej cyfrze (odpowiada ona za nieczęsto wykorzystywane prawa specjalne, a w przypadku niektórych dystrybucji Linuksa w ogóle nie jest wyświetlana) i zajmijmy się resztą wyniku. Są to prawa postaci absolutnej, które należy odjąć od pełnych praw katalogu (777) oraz pliku (666 – wykonanie w przypadku plików jest zdecydowanie rzadziej wykorzystywane). Świeżo utworzone katalogi będą miały zatem pełne prawa dla jego właściciela oraz prawa odczytu (wylistowania zawartości) oraz wejścia do folderu dla pozostałych użytkowników.

Wróćmy na chwilę do pominiętego powyżej pierwszego znaku w czterocyfrowym wyniku umask. Wiemy już, że odpowiada on za prawa specjalne - tyle że trudno wyobrazić sobie sytuację, w której ktoś chciałby uaktywnić któreś z tych praw dla wszystkich nowych plików tworzonych w systemie. Zapamiętaj, że wartość „0” oznacza tutaj brak zdefiniowanych praw specjalnych.

Jeśli komuś postać absolutna i odejmowanie praw sprawiają problemy można wyświetlić umask w zdecydowanie przyjemniejszej dla oka postaci:

umask -S

u=rwx,g=rx,o=rx

Modyfikowanie wartość umask. Wystarczy, że jako argument polecenia podamy nową wartość maski.

Aby nadać pełne prawa nowo tworzonych plików dla właściciela, zaś dla reszty prawa zerowe użyjemy polecenia:

Przykład 12a.

umask 0077

Możemy użyć formy „standardowej”:

Przykład 12b.

umask -S u=rwx,g=,o=

Wydanie powyższych poleceń jest jednak jednorazowe - po wylogowaniu i ponownym zalogowaniu umask wróci do domyślnych ustawień. Zaradzić temu możemy, dodając stosowny wpis (np. umask 0077) do pliku konfiguracyjnego powłoki.
Dla Ubuntu Basha ten plik to ~/.bashrc.