Systemy operacyjne

Laboratorium (Zabawy z prawami dostępu)

Polecenie **ls -l** wyświetla dla każdego obiektu na początku linii ciąg 10 atrybutów w ciągu. Pierwszy znak w tym ciągu określa typ pliku:

-	plik zwykły
d	katalog
p	łącze nazwane
l	dowiązanie symboliczne
b	urządzenie blokowe
c	urządzenie znakowe
s	gniazdo

i kilka innych.

Pozostałe znaki po ciągu atrybutach (jest ich 9) tworzą 3 trójki postaci:

rwxrwxrwx

gdzie na każdej pozycji może wystąpić podana litera lub znak -. Znaki te określają prawa dostępu do pliku. Wystąpienie litery oznacza, że dane prawo jest nadane, zaś znak **-** oznacza, że dane prawo jest odebrane. Pierwsza trójka odnosi się do praw właściciela, druga do grupy właściciela, zaś trzecia do wszystkich innych użytkowników danego systemu.

W odniesieniu do pliku powyższe litery oznaczają:

r	prawo do odczytu (read)
w	prawo do zapisu (write)
x	prawo do wykonywania (execute)

W odniesieniu do katalogu powyższe litery oznaczają:

r	prawo do zapoznania się z zawartością katalogu
w	prawo do zmiany zawartości katalogu
x	prawo do wejścia do danego katalogu

Poleceniem służącym nadawać oraz odbierać prawa dostępu do pliku lub katalogu jest program **chmod**.

Jego składnia jest następująca:

$$chmod < komu >< jakieprawa >< nazwaobiektu >$$

Dla parametru $< komu >$ stosowane są następujące oznaczenia:

u	właściciel (user)
g	grupa (group)
o	inni (other)

Przykłady:

- chmod g+x plik - nadanie grupie właściciela praw do wykonywania programu

- chmod o-w plik - odebranie innym użytkowników praw do zapisu w pliku
- chmod u=rw plik - właściciel może czytać i pisać do pliku ale nie go wykonywać

Innym sposobem jest podawanie cyfr oktalnych (trójka bitów). Ich implementacja jest następująca: pierwszy bit odnosi się do właściciela, drugi do grupy a ostatni do pozostałych użytkowników. Przykład:

- chmod 705 plik - właściciel otrzymuje wszystkie prawa, grupa żadne zaś pozostali użytkownicy - prawo do odczytu i wykonywania pliku.

Zadanie 1 (0.2 pkt)**.** Utwórz w swoim katalogu domowym katalog o nazwie **SO_imie_nazwisko**. Następnie utworzyć pliki **dom.txt kot.txt main.c styl.java**. Przydziel do każdego pliku następujące prawa:

- dom.txt - grupa może czytać, pisać i wykonywać - pozostali nie
- kot.txt - właściciel może wykonywać, grupa pisać, zaś pozostali czytać
- main.c - właściciel może czytać i pisać, grupa wykonywać
- styl.java - wszyscy mogą pisać, czytać i wykonywać

Napisz w pliku **ODPSO.txt** jakie polecenia oraz jakie prawa są przydzielone tym plikom. Użyj obydwu wersji: oktalnej oraz literalnej.

Zadanie 2 (0.1 pkt)**.** Będąc na serwerze **SIGMA** spróbuj wejść do katalogu domowego innego użytkownika (swojego sąsiada). Spróbuj utworzyć u niego plik. Pamiętaj że musi ci na to pozwolić!
UWAGA!
Nie należy odbierać samemu sobie praw dostępu do swojego katalogu domowego! (grozi wizytą u administratorów systemu).

Zadanie 3 (0.1 pkt)**.** W swoim katalogu z zadania pierwszego utwórz katalog o nazwie **podkatalog**. Utwórz w nim plik podmain.c ala.txt. Przeprowadź eksperyment:

- Usuwając prawa dostępu do katalogu z zadania 1.
- Usuwając prawa dostępu do katalogu bieżącego.

W jakim przypadku możemy użyć polecenia **cd** oraz **chmod**?
Czy możemy bezpośrednio przeskoczyć do podkatalogu kiedy nie mamy prawa wstępu do katalogu nadrzędnego? Odpowiedź uzasadnij. Następnie rekurencyjnie przydziel tylko **sobie** prawa do katalogu z zadania z 1 wraz z zawartością.

Zadanie 4 (0.2 pkt)**.** Przeczytać opis polecenia **umask** i wypróbować jego działanie tworząc a następnie zmieniając swoją maskę trybu pliku. Jaka operacja logiczna na bitach domyślnych praw dostępu oraz maski jest wykonywana? Jaką maskę trzeba ustawić aby:

- Właściciel mógł tylko tworzyć i czytać pliki?
- Tylko właściciel może pisać do pliku, pozostali mogą czytać?
- Wykluczyć pozostałych użytkowników?

Zadanie 5 (0.3 pkt)**.** Poza zwykle stosowanymi prawami dostępu r,w,x możliwe jest też stosowanie innych atrybutów modyfikujących działanie nadanych praw. Jednym z nich jest atrybut **s**, mogący powodować, że proces wykonujący program zawarty w pliku o takim atrybucie będzie działał nie z prawami swojego rzeczywistego właściciela ale z prawami właściciela pliku z programem.

1. Przeczytać opis atrybutu **s** zawarty w opisie polecenia zewnętrznego chmod. Wypróbować działanie poleceń: **echo $UID** oraz **echo $EUID**.

2. Utworzyć plik tekstowy i nadać mu uprawnienia 600. Następnie przepisać i skompilować następujący program w języku **ANSI C**

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(){
  int c;
  FILE *f;
  printf("REAL UID=%u\n",getuid());
  printf("EFFECTIVE UID= %u\n",geteuid());
  f = fopen("sciezka do pliku tekstowego","r");
  if(f == NULL){
    perror("Cannot open file");
    exit(1);
  }
  while((c = fgetc(f)) != EOF){
    putchar(c);
  }
  fclose(f);
  return EXIT_SUCCESS;
}
```

   W zespołach 2- lub 3- osobowych przeprowadzić eksperyment:

   - Przydzielić odpowiednie prawa dostępu tak, aby umożliwić pozostałym członkom zespołu wejście do katalogu, w którym jest plik tekstowy oraz skompilowany program.
   - Przydzielić plikowi zawierającemu skomplikowany program prawa 755, wykonać program samemu i zalecić to samo pozostałym członkom zespołu.
   - Podać polecenie:

     chmod u=rws (nazwa pliku ze skompilowanym programem)

     i zalecić powtórne uruchomienie programu. Jaka jest różnica? Jak przydzielić te same prawe używając notacji oktalnej? Jakie wartości przyjmuje bajt odpowiadający za prawa specjalne?

Zadanie 6 (0.1 pkt)**.** W zespołach 2 lub 3-osobowych utworzyć katalog **sticky**. Ustawić tzw. bit sticky w tym katalogu. Każdy członek zespołu niech utworzy pliki w tym katalogu i usunie pliki innych uzytkowników. Co robi **sticky bit**?