# Systemy operacyjne 8

## Spis treści



# Konta użytkowników

## Pliki przechowujace konta

Ze względu na wieloużytkownikowość i wielodostęp, systemy UNIX posiadają bazę danych informacji o użytkownikach. W bazie tej zawarte są podstawowe dane o każdym z użytkowników, takie jak nazwa użytkownika, numer, grupa, do której należy, krótki opis, katalog domowy oraz powłoka, której używa. Baza użytkowników znajduje się w pliku ***/etc/passwd***, natomiast zaszyfrowane hasła użytkowników i informacje uzupełniające w ***/etc/shadow***. Wpisy do pliku ***/etc/passwd*** mają następującą strukturę:

```
Username:Password:UID:GID:Gecos:HomeDirectory:Shell
```

przykład:

```
root:x:0:0:root:/root:/bin/bash
```

username
: nazwa użytkownika w systemie.

haslo
: hasło kodowane DES, w systemach z shadow zazwyczaj x.

uid
: numer użytkownika.

gid
: numer głównej grupy.

gecos
: pole komentarza, zwane też polem GECOS zawiera dodatkowe informacje o użytkowniku, składa się z czterech pól oddzielonych przecinkami: imię z nazwiskiem, adres/położenie biura, telefon w pracy, telefon domowy. Pole GECOS nie jest obowiązkowe.

home
: katalog domowy użytkownika.

shell
: domyślny interpreter poleceń.

Wpisy w pliku */etc/shadow* mają następującą strukturę:

```
Username:haslo:t1:t2:t3:t4:t5:t6:zarezerwowane
```

przykład:

```
smithj:Ep6mckrOLChF.:10063:0:99999:7:::
```

username
: nazwa użytkownika.

haslo
: hasło, zakodowane w DES lub MD5; patrz crypt(3).

t1
: data ostatniej zmiany hasła, w dniach od 01.01.1970 GMT.

t2
: dni od zmiany hasła, po których można zmienić je ponownie.

t3
: dni od zmiany hasła, po których hasło musi być zmienione.

t4
: dni przed wygaśnięciem hasła, kiedy użytkownik jest ostrzegany.

t5
: dni po wygaśnięciu hasła, po których konto jest wyłączane.

t6
: data włączenia konta, w dniach od 01.01.1970 GMT.

zarezerwowane
: pozostawione dla przyszłego użytku.

Aby łatwiej zarządzać prawami dostępu do zasobów, użytkownicy należą do jednej lub więcej grup. W systemach UNIX przyjęto, że każdy użytkownik ma swój numer (UID), który jest jego identyfikatorem. Prawa dostępu do zasobów określane są względem tego numeru, a nie nazwy użytkownika. Nazwa jest jakby "aliasem" do numeru - można ją w każdej chwili zmienić bez żadnych konsekwencji - wszystkie prawa dostępu zostają zachowane, (bo przyznawane są według numerów, a nie nazw). Dzięki ***/etc/passwd*** możliwe jest "tłumaczenie" nazw na UIDy i odwrotnie.

Możliwe też jest stworzenie dwóch lub więcej kont z tymi samymi numerami UID pod różnymi nazwami. Wówczas użytkownicy, korzystający z tego samego UIDu mają identyczne prawa i ograniczenia w systemie, pomimo, że mogą posiadać różne hasła, katalogi domowe czy powłoki.

Grupy użytkowników funkcjonują na podobnych zasadach. Każda grupa posiada swój numer (GID) i to właśnie ten numer wykorzystywany jest przy określaniu praw dostępu.

Każdy z użytkowników musi należeć przynajmniej do jednej grupy. Główna grupa, do której należy użytkownik zapisana jest w ***/etc/passwd***. Pozostałe w ***/etc/group***.

```
nazwa_grupy:hasło:GID:lista_użytkowników
```

przykład:

```
gdftp::502:dave,nick,pete,ben,rwm
```

Aby przekonać się, w jakich grupach znajduje się użytkownik i jaki jest jego UID należy wydać komendę:

```
id [UŻYTKOWNIK]
```

lub:

```
groups [UŻYTKOWNIK]
```

Przykład:

```
user@ttyp0[linux]$ id
uid=1000(user) gid=1000(user) grupy=1000(usersi),20(dialout),21(fax),22(voice),
24(cdrom),25(floppy),26(tape),27(sudo),29(audio),30(dip),44(video),60(games),
100(users),106(usb)
```

Z powyższego zapisu wynika, że w systemie jest użytkownik **user** i ma nadany numer UID=1000, jego główną grupą jest **user** (GID=1000), a dodatkowo należy do grup "users" GID=100, "dialout" GID=20 i innych.

Inne polecenia pokrewne: users, who, whoami

## Własność zasobów

Użytkownicy i grupy ułatwiają zarządzanie dostępem do zasobów.

Dla przykładu:

```
user@ttyp0[tmp]$ ls -l kopia
-rw-r--r-- 1 user user 0 2004-04-19 21:43 kopia
user@ttyp0[tmp]$
```

Właścicielem pliku kopia jest użytkownik **user**. Prawa dla tego pliku, określane są dla właściciela, grupy i wszystkich pozostałych użytkowników. Aby optymalnie zarządzać tymi prawami, właściciel pliku może zmienić grupę względem której określane są prawa. Użytkownik **user**, zmieniając grupę np. na **users** spowoduje, że tylko użytkownicy tej grupy będą mogli przeglądać zawartość tego pliku. W tym celu posłużyć się należy poleceniem **chgrp** (**ch**ange **gr**ou**p**)o następującej składni:

```
chgrp GRUPA PLIK(I)
```

gdzie:

GRUPA
: grupa, na którą należy zmienić,

PLIK(I)
: lista plików do zmiany.

Przykład:

```
user @ttyp0[tmp]$ ls -l kopia
-rw-r--r-- 1 user user 0 2004-04-19 21:43 kopia
user @ttyp0[tmp]$ chgrp users kopia
user @ttyp0[tmp]$ ls -l kopia
-rw-r--r-- 1 user users 0 2004-04-19 21:43 kopia
```

Grupę może zmienić tylko właściciel pliku i to tylko i wyłącznie na grupę w której należy. Ograniczenia te nie dotyczą oczywiście użytkownika **root**. Dodatkowo administrator może zmienić właściciela pliku poleceniem **chown** (**ch**ange **own**er) o następującej składni:

```
chown [WLASCICIEL][:GRUPA] PLIK(I)
```

gdzie:

WLASCICIEL
: nowy właściciel pliku,

GRUPA
: nowa grupa,

PLIK(I)
: lista plików poddawanych operacji zamiany.

Inne polecenia związane z grupami: chmod, newgrp, groupadd, groupdel, groupmod

## Zarządzanie kontami

By użytkownik mógł pracować w systemie, Unix powinien mieć w nim założone konto.

Konto w tym przypadku rozumiemy, jako wszystkie zasoby, pliki i informacje o tym użytkowniku. Każde konto charakteryzują dwie podstawowe cechy - **hasło** i **login**. Dzięki loginowi komputer wie, z którym użytkownikiem aktualnie pracuje, a hasło służy do zabezpieczenia zasobów tego użytkownika.

Aby jednak było to możliwe, w systemie jest wbudowane konto **root** posiadająceuprawnieniado wykonywania czynności administracyjnych. Uwaga: cechą wyróżniającą konto administracyjne jest identyfikator UID równy 0. Konto administracyjne może mieć zmienioną nazwę, a nawet może istnieć kilka kont tego typu.

Konta wykorzystywane są również przez programy, które na każdym z tych kont mogą posiadać własne pliki konfiguracyjne, dostosowujące program do potrzeb danego użytkownika.

Polecenie służące do zakładania konta ma następującą postać:

```
useradd [-c komentarz] [-d katalog_domowy] [-e data_ważności] [-f dni_nieaktywności]
        [-g grupa_początkowa] [-G grupa [,...]] [-m [-k katalog_wzorców]] [-o] [-p hasło]
        [-s powłoka] [-u uid] login
```

Jak widać polecenie zawiera pokaźny zbiór opcji, w krańcowym przypadku wystarczy jednak ograniczyć się do tego co niezbędne: useradd *login.* Inne spotykane polecenia związane z grupami: userdel, usermod, chfn,

Często zachodzi potrzeba, aby wykonać pewne czynności administracyjne, do których zwykły użytkownik nie ma prawa. W tym celu istnieje możliwość zmiany tożsamości za pomocą polecenia **su** zarówno w odniesieniu do użytkownika **root** jak i pozostałych użytkowników.

Część z tych zasobów użytkownik może zmienić podczas pracy w systemie. Przede wszystkim użytkownik może zmienić powłokę, której używa poprzez polecenie **chsh** (**ch**ange **sh**ell). Przykładowa zmiana powłoki wygląda następująco:

```
user @ttyp0[linux]$ chsh
Password:
Zmieniam powłokę logowania dla user
Wpisz nową wartość lub wciśnij ENTER by przyjąć wartość domyślną
        Powłoka logowania [/bin/bash]: /bin/sash
```

Innym poleceniem umożliwiającym takie zmiany jest **chfn** (**ch**ange **f**inger i**n**formation), które pozwala na dokonanie zmiany pełnej nazwy użytkownika, ewentualnie na podanie miejsca pracy i telefonów.

Wielodostępna natura systemów Unix/Linux oznacza, że z usług systemu może korzystać większa liczba użytkowników jednocześnie. Aby dowiedzieć się, kto jest aktualnie zalogowany do systemu należy wydać komendę **users**. Oto przykładowy wynik działania tej komendy:

```
user @ttyp0[linux]$ users
user root root
user @ttyp0[Linux]$
```

Wynika z tego, że zalogowanych jest 2 użytkowników: **user** i **root**. Innym poleceniem sprawdzającym, kto jest aktualnie zalogowany jest: **w** oraz **who**. Działanie tej ostatniej wygląda następująco:

```
barakuda@angband ~ $ who
barakuda tty1         2006-10-31 23:34
test     tty3         2006-11-04 17:06
root     tty2         2006-11-03 15:46
barakuda@angband ~ $
```

W wyniku działania tej komendy stwierdzić można, z jakiej konsoli logowali się użytkownicy oraz od kiedy są zalogowani. Konsole oznaczone jako **ttyX** (X – numer konsoli) są to wirtualne konsole, natomiast **pts** to pseudoterminale, czyli użytkownicy zalogowali się do serwera poprzez sieć i usługę telnet.

Większą porcję danych o użytkownikach zalogowanych w systemie dostarcza polecenie **w**. Przykładowe działanie wygląda następująco:

```
barakuda@angband ~ $ w
 17:07:58 up 3 days, 17:34,  3 users,  load average: 0,71, 0,59, 0,41
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
barakuda tty1      Tue23   25:19m  0.09s  0.00s /bin/sh /usr/bin/startx
test     tty3      17:06    1:31   0.00s  0.00s -bash
root     tty2      Fri15   21:46m  0.16s  0.16s -bash
barakuda@angband ~ $
```

Pierwsza linia wyświetlana przez to polecenie zawiera ogólne informacje o systemie - godzinę (21:53:34), jak długo serwer pracuje (19 minut), ilość zalogowanych użytkowników (3 users) oraz obciążenie serwera w ciągu ostatniej minuty, w ciągu ostatnich 5 minut i ostatnich 15 minut (load average: 0.19, 0.18, 0.18). Następne linie zawierają tabelaryczne zestawienie danych dotyczących zalogowanych użytkowników. W pierwszej kolumnie znajduje się identyfikator użytkownika. Druga kolumna zawiera nazwę terminala, z którego zalogowany jest dany użytkownik. Trzecią kolumnę stanowi nazwa hosta, z którego użytkownik się zalogował (przydatne, gdy użytkownik korzysta z pseudoterminala). Dalsze kolumny to kolejno: informacja, o której godzinie nastąpiło logowanie, jak długo użytkownik jest bezczynny, ile czasu procesora zajmuje obsługa danego terminala, ile czasu procesora zajmuje obsługa zadania wyświetlanego w ostatniej kolumnie oraz aktualnie wykonywane zadanie przez użytkownika.

Poleceniem, które dostarcza informacji o konkretnym użytkowniku jest **finger**, np. finger alvin

```
barakuda@angband ~ $ finger
Login     Name       Tty      Idle  Login Time   Office     Office Phone
barakuda            *tty1       1d  Oct 31 23:34
root      root      *tty2    21:46  Nov  3 15:46
test                *tty3        1  Nov  4 17:06
barakuda@angband ~ $
```

W odróżnieniu od polecenia **w**, wyświetla dane w nieco innym formacie i dołącza dodatkowo pełną nazwę użytkownika. Dodatkowo **finger** wyświetla informacje o tym, czy jesteśmy w stanie z danym użytkownikiem porozmawiać (dokładniej: użyć polecenia **write** bądź **talk**). Jeśli nie ma takiej możliwości, wówczas w kolumnie **Tty** pojawia się gwiazdka obok nazwy terminala (jak w przypadku użytkownika **root**).

Unix pozwala także sprawdzić nie tylko, kto aktualnie jest, zalogowany ale również, kto pracował w systemie w przeszłości (w określonym przedziale czasu). Do tego służy polecenie:

```
last [-NUMER] [UŻYTKOWNIK]
```

gdzie:

NUMER
: ogranicza rozmiar wyniku do podanej ilości linii

UŻYTKOWNIK
: podanie użytkownika oznacza wyświetlenie statystyk tylko dla podanego użytkownika. Zamiast użytkownika można wpisać nazwę terminala bądź słowo **reboot**, aby uzyskać informację o czasie pracy systemu.

Przykład:

aby zobaczyć 10 ostatnich logowań i restartów systemu:

```
barakuda@angband ~ $ last -n 10
root     tty2                         Sat Nov  4 17:11   still logged in
root     tty2                         Sat Nov  4 17:11 - 17:11  (00:00)
barakuda tty2                         Sat Nov  4 17:11 - 17:11  (00:00)
barakuda tty2                         Sat Nov  4 17:11 - 17:11  (00:00)
test     tty2                         Sat Nov  4 17:11 - 17:11  (00:00)
test     tty2                         Sat Nov  4 17:11 - 17:11  (00:00)
test     tty3                         Sat Nov  4 17:06   still logged in
test     tty3                         Sat Nov  4 17:06 - 17:06  (00:00)
barakuda pts/2        :0.0            Sat Nov  4 16:47 - 16:50  (00:03)
barakuda pts/2        :0.0            Sat Nov  4 16:34 - 16:37  (00:03)
wtmp begins Wed Nov  1 15:29:23 2006
barakuda@angband ~ $
```

## Hasła

Dostęp do konta każdego użytkownika chroniony jest zwykle jedynie hasłem. Zwykły użytkownik może zmienić wyłącznie hasło własnego konta, użytkownik **root** może zmieniać hasła dowolnych kont, natomiast administrator grupy może zmienić hasło tej grupy. Należy pamiętać o podstawowych zasadach związanych z bezpieczeństwem haseł, tzn. hasło powinno mieć minimum kilkanaście znaków zawierających małe i duże litery, cyfry i znaki przestankowe. Nie powinno być „słownikowe” oraz powinno być łatwe do zapamiętania.

Zmiana konta dokonywana jest za pomocą polecenia: **passwd**. Po czym użytkownik zostaje poproszony o podanie starego hasła i dwukrotne wprowadzenie nowego.

Dobrym pomysłem przy tworzeniu haseł wydaje się tworzenie skrótów z dłuższych fraz. np. "k2Bwh,pBt" utworzone jest z łatwego do zapamiętania zdania "kupiłem 2 bułki w hipermarkecie, ponieważ były tanie".

## Zdalny dostęp

Wszystkie systemy uniksowe pozwalają na obsługę komputera poprzez sieć. Najczęściej spotyka się trzy metody dostępu do zdalnej konsoli:

- rsh
- telnet
- ssh

Jednakże ze względów bezpieczeństwa administratorzy preferują program ssh i często blokują dostęp pozostałymi metodami.

Aby zalogować się poprzez protokół ssh należy wydać polecenie:

```
ssh -l użytkownik nazwa_komputera
```

lub

```
ssh użytkownik@nazwa_komputera
```

gdzie:

użytkownik
: to nazwa użytkownika na którego konto chcemy się zalogować

nazwa_komputera
: to nazwa domenowa hosta lub jego adres IP

Po wydaniu polecenie zostaniemy zapytani o hasło (lub nie, jeśli tak został skonfigurowany serwer). W chwili połączenia uzyskujemy dostęp do konsoli zdalnego hosta z pełnymi uprawnieniami danego użytkownika. Aby zerwać połączenie należy wydać polecenie:

```
exit
```

## Zadania

- Utwórz użytkowników test1, test2 oraz test3
- Ustal hasła dla użytkowników test1 i test2 (identyczne jak nazwy)
- Zaloguj się na użytkownika test1
- Będąc zalogowanym na użytkownika test1 ustaw hasło użytkownikowi test3. Jest to możliwe? Skorzystaj z narzędzia nadającego Ci prawa administratora.
- Utwórz grupy grupa1 i grupa2
- Przypisz użytkowników test1 i test2 do grupy grupa1.
- Przypisz użytkowników test2 i test3 do grupy grupa2. Czy użytkownik test2 może należeć do dwóch grup? Do jakiej grupy należą pliki zakładane przez użytkownika test2?
- Ustaw grupę domyślną użytkownika test2 na grupa2. Do jakiej grupy należą pliki tworzone przez tego użytkownika?
- Zmień powlokę użytkownika test2 na /bin/sh. Umiał byś to zrobić edytując pliki konfiguracyjne?

- Zmień powlokę użytkownika test3 na /bin/passwd. Jest to możliwe? Jeśli zajdzie potrzeba zmodyfikuj odpowiednie pliki. W jakich okolicznościach takie ustawienie może mieć sens?
- Przekształć konto test1 w „zapasowe" konto administracyjne.
- Zaloguj się zdalnie do komputera osoby siędzącej obok.

# Prawa dostępu do plików i katalogów

Prawa dostępu do plików i katalogów są jednymi z najważniejszych mechanizmów bezpieczeństwa systemu. Uniemożliwiają one innym użytkownikom przeglądanie naszych zasobów. Prawa dostępu podzielone są na trzy sekcje:

- właściciel pliku lub katalogu
- grupa związana z plikiem lub katalogiem
- wszyscy inni użytkownicy systemu

Prawa dostępu można odczytać wydając polecenie *ls -l*:

```
angband ~ # ls -l
razem 5016
-rw-r--r--   1 root root   31624 kwi  7  2006 agpgart.ko
drwxr-xr-x   3 root root    4096 lut 11  2005 app-portage
drwxr-xr-x   2 root root    4096 gru 11  2005 arsene
-rw-r--r--   1 root root       0 maj 26 20:39 CHG
-rw-r--r--   1 root root       0 maj 26 20:39 CHGCAR
```

znaki minus (-) i litery występujące na początku każdej linii reprezentują typ pliku i prawa dostępu.

## Prawa dostępu do pliku

Każdy plik ma ściśle określone prawa dostępu stwierdzające, czy określony użytkownik jest uprawniony do odczytania lub zapisania pliku bądź do jego wykonania. Każdy użytkownik może mieć dowolną kombinację tych praw. Są one całkowicie niezależne i posiadanie jakiegokolwiek z nich nie jest warunkiem posiadania innego. W przypadku pliku prawa są interpretowane w następujący sposób:

- r – prawo czytania umożliwia oglądanie zawartości pliku, oznacza jednocześnie prawo do kopiowania,
- w – prawo pisania oznacza zezwolenie na modyfikację zawartości pliku,
- x – prawo do uruchomienia pliku wykonywalnego.

## Prawa dostępu do katalogu

Te same kategorie praw - czytania, pisania i wykonywania odnoszą się do katalogów:

- r – prawo czytania umożliwia przeszukiwanie zawartości katalogu, jest interpretowane jako prawo wypisywania zawartości (komenda *ls*),
- w – prawo pisania daje możliwość modyfikowania zawartości katalogów umożliwia dodawanie nowych oraz usuwanie dotychczasowych plików z katalogu,
- x – prawo wykonywania w stosunku do katalogu pozwala na dostęp do plików zapisanych w nim oraz na wejście do danego katalogu uczynienie go katalogiem bieżącym (polecenie: *cd katalog*).

Wydając polecenie ls –l otrzymujemy na ekranie:

```
-rwxr--r-- 1 root root 497 May 23 11:57 index.html
drwxr-xr-x 5 root root 512 Sep 17 12:47 www
```

znaki minus (-) i litery występujące na początku każdej linii reprezentują typ pliku i prawa dostępu. Przyjmują one postać:

```
krwxrwxrwx
```

Wyjaśnienie tego zapisu jest następujące:

- *k* – jest to identyfikator typu, gdzie:
  - - zwykły plik
  - b specjalny plik blokowy
  - c specjalny plik znakowy
  - d katalog
  - l link symboliczny
  - p potok
  - s gniazdo
- prawa dostępu:
  - Pierwsza trójka **rwx** – oznacza uprawnienia dla właściciela (u – user)
  - Druga trójka **rwx** – oznacza uprawnienia dla grupy (g – group)
  - Trzecia trójka **rwx** – oznacza uprawnienia dla pozostałych użytkowników (o – others)

| Prawo | Plik | Katalog |
|---|---|---|
| r | czytania zawartości | przeszukania zawartości |
| w | zmiany zawartości | zmiany zawartości |
| x | Wykonywanie | przejścia do tego katalogu |

## Przypisywanie uprawnień

Do zmiany uprawnień użytkowników w stosunku do katalogu lub pliku służy polecenie *chmod*. Wymaga ono określenia, czyje uprawnienia należy zmienić, na jakie oraz jakiego pliku lub katalogu ta zmiana będzie dotyczyć. Prawa dostępu mogą być podane na dwa sposoby:

- przy pomocy systemu kodów numerycznych (w formacie ósemkowym): 4 2 1
- przy pomocy systemu kodów znakowych: r w x

Oto porównanie tych systemów (w nawiasach przy zapisie numerycznym podano zapis binarny uprawnień):

| Prawa dostępu | Zapis numeryczny | Zapis znakowy |
|---|---|---|
| Tylko do czytania | 4 (100) | r-- |
| Tylko do pisania | 2 (010) | -w- |
| Tylko do wykonywania | 1 (001) | --x |
| Do czytania i pisania | 6 (110) | rw- |

| | | |
|---|---|---|
| Do czytania i wykonywania | 5 (101) | r-x |
| Czytania, pisania i wykonywania | 7 (111) | rwx |

## Kody numeryczne praw

Jeśli prawa dostępu do danego katalogu lub pliku będziemy podawać numerycznie, wówczas składnia polecenia *chmod* ma postać:

```
chmod numeryczny_kod nazwa_zasobu
```

Przykład:

```
chmod 644 plik.txt
```

zapis ten oznacza:

- prawo dostępu właściciela **6 = 4 + 2 + 0 czyli rw**
- prawo dostępu grupy **4 = 4 + 0 + 0 czyli r**
- prawo dostępu pozostałych użytkowników **4 = 4 + 0 + 0 czyli r**

```
chmod 701 katalog
```

zapis ten oznacza:

- prawo dostępu właściciela **7 = 4 + 2 + 1 czyli rwx**
- prawo dostępu grupy **0 = 0 + 0 + 0 czyli brak uprawnień**
- prawo dostępu pozostałych użytkowników **1 = 0 + 0 + 1 czyli x**

```
chmod 755 plik2.txt
```

zapis ten oznacza:

- prawo dostępu właściciela **7 = 4 + 2 + 1 czyli rwx**
- prawo dostępu grupy **5 = 4 + 0 + 1 czyli r-x**
- prawo dostępu pozostałych użytkowników **5 = 4 + 0 + 1 czyli r-x**

## kody symboliczne praw

Jeśli prawa dostępu do danego katalogu lub pliku będziemy podawać symbolicznie, wówczas składnia polecenia *chmod* ma postać:

```
chmod kto_oper_prawo nazwa_zasobu
```

gdzie:

kto
: określa komu nadawane są prawa i może być jednym, bądź kilkoma, spośród symboli:

  - a - wszyscy użytkownicy,
  - u - właściciel pliku
  - g - grupa pliku,
  - o - inni użytkownicy.

Pominięcie symbolu kategorii nadaje wszystkim (właścicielowi, grupie, pozostałym) takie same prawa.

oper
: jest jednym z następujących symboli:

  - – oznacza odebranie prawa,
  - + oznacza dodanie prawa,
  - = ustala nowe uprawnienia niezależnie od stanu poprzedniego.

prawo
: typ praw dostępu, który jest jednym, bądź kilkoma, spośród symboli: r, w, x,

Przykład:

```
chmod u-w plik1.txt
```

zapis ten zabiera właścicielowi prawo pisania do pliku plik1.txt,

```
chmod a=rw plik1.txt
```

zapis ten nadaje wszystkim prawo rw do pliku plik1.txt,

```
chmod u+w, og+r-x plik1.txt
```

zapis ten nadaje właścicielowi prawo pisania do pliku plik1.txt, a członkom grupy oraz pozostałym użytkownikom systemu nadaje prawo czytania i odbiera prawo wykonania pliku plik1.txt,

**Porównanie obu zapisów:**

Prawo do czytania dla właściciela pliku

```
chmod 400 nazwa_pliku
chmod u+r nazwa_pliku
```

Wszystkie prawa dla właściciela pliku i prawo do czytania dla grupy

```
chmod 740 nazwa_pliku
chmod u+rwx,g+r nazwa_pliku
```

## Domyślne prawa dostępu przy tworzeniu plików i katalogów

Domyślne prawa dostępu dla plików i katalogów nadawane są podczas ich tworzenia. Zmianę tych praw uzyskujemy poleceniem *umask*. Jeśli chcielibyśmy, aby tworzone pliki miały domyślne prawa 644, które zezwalają właścicielowi na czytanie i pisanie, a reszcie tylko na czytanie to od wartości 777 należy odjąć 644, a wynik podać jako parametr polecenia *umask*

*777-644 = 133*

```
umask 133
```

## suid i sgid - pożyteczne i niebezpieczne narzędzie

Zadaniem tego potężnego, a zarazem niebezpiecznego narzędzia jest uruchamianie programu (nie skryptu) z prawami właściciela lub grupy przypisanej temu programowi, a nie z prawami użytkownika, który ten program uruchamia.

Zagrożenie z używania tych flag może wynikać z możliwości przejęcia kontroli nad systemem. Jeśli zwykłemu użytkownikowi uda się tak zawiesić program (którego właścicielem jest użytkownik root i który ma ustawioną flagę suid lub sgid), aby dostać się do powłoki to otrzyma on prawa właściciela programu (czyli w tym przypadku użytkownika root) co stanowi ogromne zagrożenie dla systemu.

Dlatego należy z dużym rozsądkiem używać tych flag.

Nadawanie plikom **suid** lub **sgid** wygląda następująco:

**suid**

```
chmod u+s nazwa_plku
chmod 2*** nazwa_pliku
```

**sgid**

```
chmod g+s nazwa_pliku
chmod 4*** nazwa_pliku
```

**suid** i **sgid**

```
chmod 6*** nazwa_pliku
```

gdzie *** oznacza dowolne prawa dla właściciela, grupy i innych użytkowników.

Flaga **suid** w listingach plików reprezentowana jest przez literkę s w prawach dla właściciela pliku :

```
rws r-x r-x
```

Flaga **sgid** w listingach plików reprezentowana jest przez literkę s w prawach dla grupy:

```
rwx r-s r-x
```

## sticky bit

Dla pliku ustawienie sticky bitu oznacza, że program, który on przechowuje będzie po jego zakończeniu nadal przechowywany w pamięci komputera. Dla katalogów sticky bit oznacza, że tylko właściciel może go usunąć mimo ustawienia praw na przykład na 777.

Ustawienie sticky bitu dla plików wygląda następująco:

```
chmod 1*** nazwa_pliku_katalogu
chmod +t nazwa_pliku_katalogu
```

gdzie *** oznacza dowolne prawa dla właściciela, grupy i innych użytkowników.

Reprezentantem tego bitu w listingach katalogów jest literka **t** w sekcji dotyczącej reszty użytkowników :

```
rwx r-x r-t
```

## Zmiana właściciela i grupy pliku

Zmiana właściciela i grupy pliku możliwa jest przy użyciu poleceń: *chown, chgrp*. Rzecz się ma następująco:

```
chown nowy_właściciel nazwa_pliku(ów)
chgrp nowa_grupa nazwa_pliku(ów)
```

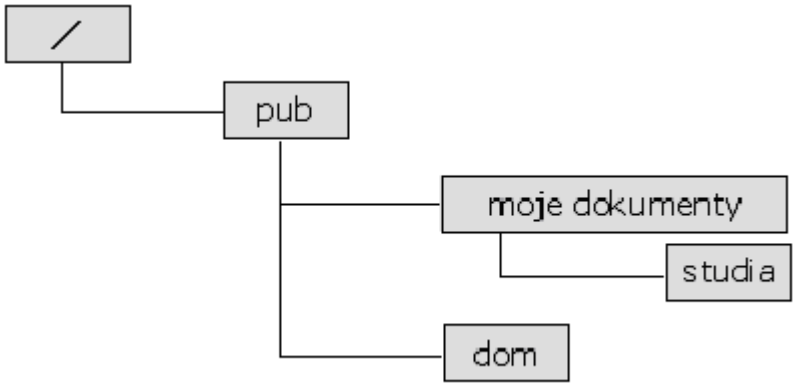
lub w połączeniu

```
chown nowy_właściciel:nowa_grupa nazwa_pliku(ów)
```

Używając opcji *-R* polecenia *chown*, możesz zmienić właściciela wszystkich plików w danym katalogu i wszystkich jego podkatalogach. Wówczas zamiast nazwy pliku podaje się nazwę żądanego katalogu. Opcja ta jest również ważna dla polecenia *chgrp*.

## Zadania

- Utwórz następującą strukturę katalogów:




Rys. 1. Struktura katalogów do "Sesji przy terminalu 2"

- Skopiuj wszystkie polecenia dwuliterowe zaczynające się od litery *d* z katalogu */bin* do katalogu *moje dokumenty*
- Utwórz w katalogu *dom* dwa pliki tekstowe *plik1.txt* oraz *plik2.txt*. Pliki te powinny zawierać przykładowy tekst.
- Jakie uprawienie domyślne mają *pliki1.txt* oraz *plik2.txt*.
- Ustaw dla pliku plik1.txt uprawnienia rw-r--rw-.
- Ustaw dla pliku plik2.txt następujące uprawnienia właściciel: odczyt, zapis, wykonanie, grupa: tylko zapis, pozostali: odczyt, zapis. Jak to zrobić korzystając z zapisu numerycznego?
- Wyświetl uprawnienia plików plik1.txt oraz plik2.txt.
- Wydaj polecenie chmod 574 dla plik1.txt? Jakie zostaną nadane mu prawa?
- Wydaj polecenie chmod u=rw, g=r, o-r dla pliku plik2.txt. Jakie zostaną nadane mu prawa?
- Wyświetl uprawnienia plików plik1.txt oraz plik2.txt. Które polecenie (pkt. 8 i 9) nadpisuje uprawnienia, a które je modyfikuje?

- Utwórz plik mojedane.txt w katalogu studia i umieść w nim swoje dane.
- Załóż plik o nazwie *email.txt*, w katalogu studia i umieść swój adres e-mail
- Ustaw dla katalogu moje dokumenty i wszystkich podkatalogów następujące uprawnienia właściciel: odczyt, zapis, wykonanie, grupa: brak uprawnień, pozostali: odczyt, zapis.
- Sprawdź, czy możesz nie nadawać plikowi żadnych praw?
- Pracując jako zwykły użytkownik utwórz plik moj.txt. Następnie zaloguj się na konto root i przejmij ten plik na własność.
- Kto jest właścicielem plików znajdujących się w katalogu dom?