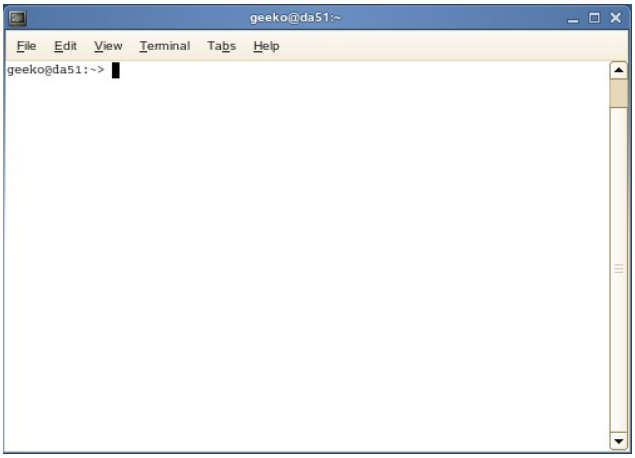
2. Powłoka (*shell*) i narzędzia Linuksa

Użytkownik systemu operacyjnego nie może komunikować się bezpośrednio z jądrem (kernelem) systemu. Potrzebny jest program pośredniczący – interfejs. Dla systemów rodziny Unix jest nim program zwany **powłoka** (*shell*).

By dostać się do programu powłoki – wybieramy **Gnome Terminal** lub **X Terminal** z głównego menu:

Gnome Terminal

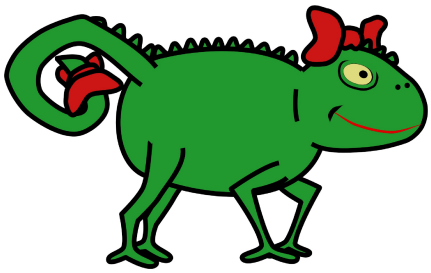


X Terminal

Możemy na komputerze uruchomić do sześciu wirtualnych terminali (F1 – F6). Przełączać się między nimi można kombinacją klawiszy: **Ctrl** + **Alt** + **F***x*; gdzie *x* to numer 1 do 6.

Naciskając kombinację klawiszy **Ctrl** + **Alt** +**F7** wracamy do graficznego interfejsu.

Zadaniem powłoki jest interpretacja poleceń i danych wprowadzanych przez użytkownika, przekształcanie ich na odpowiednie odwołania (funkcje) systemowe (*system calls*), a następnie przekazywanie odpowiedzi systemu (komunikatów systemowych) użytkownikowi. Powłoka często jest nazywana po prostu „*interpreterem poleceń*”.



Jest wiele rodzajów uniksowych powłok, większość z nich występuje w Linuksie.

Przykłady popularnych programów powłok:

- powłoka sh (Bourne shell),
- bash (Bourne Again shell),
- ksh (Korn shell),
- tsh (C shell),
- tcsh (Tenex C shell).

Powłoki te mają różną funkcjonalność i różną syntaktykę. Za najbardziej przyjazne dla użytkowników uznaje się **bash** oraz **tcsh**.

By pokazać różnice w syntaktyce obu programów powłok – uruchomiono w nich proste programy wyświetlające na ekranie liczby od 1 do 10:

bash

```
#!/bin/bash

declare -i maximum=10
declare -i current=0

while :; do
  if [ $current -eq $maximum ]; then
    break
  fi

  current=$current+1
  echo $current
done
```

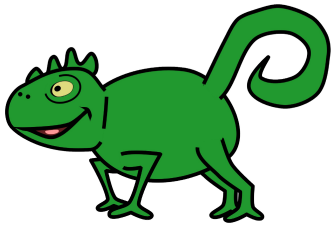
tcsh

```
#!/bin/tcsh

@ maximum = 10
@ current = 0

while (1)
  if ( $current == $maximum ) then
    break
  endif

  @ current = $current + 1
  echo $current
end
```



Najważniejsze zadania programów powłok, to:

- **Interpretacja wiersza poleceń**. Program odbiera wprowadzane przez użytkownika dane i interpretuje je.
- **Konfiguracja środowiska użytkownika**. Program powłoki przy uruchamianiu czyta pliki konfigurujące ustawiając odpowiednio na ich podstawie zmienne i odnośniki.
- **Programowanie powłoki, skrypty.** Automatyzacja złożonych zadań przez przygotowanie odpowiednich skryptów – ciągów poleceń powłoki.

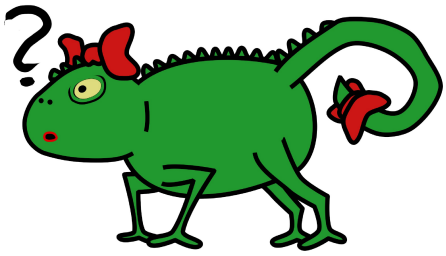
Standardową powłoką Linuksa jest ***bash***. Każda powłoka może być uruchomiona w dowolnym momencie, jak każdy inny program. Możesz przełączyć się na powłokę C shell poleceniem *tcsh*, na powłokę Korn shell wpisując *ksh*, a do powłoki bash wejdziemy poleceniem liniowym *bash*.

By wyjść z danej powłoki – należy użyć polecenia *exit*.

Ćwiczenie. Powłoki w Linuksie

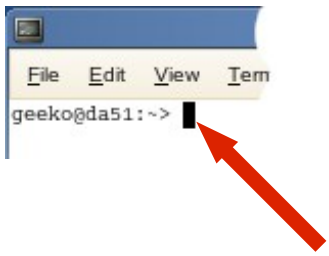
Znak zachęty wygląda inaczej w każdej powłoce. Połącz strzałkami nazwy powłok z odpowiadającymi im znakami zachęty (*prompt*):

Korn Shell	geeko@da51:~>
Bourne Again Shell	/home/geeko>
Tenex C Shell	geeko@da51:/home/geeko>



2.1 Zarządzanie systemem plików w wierszu poleceń powłoki

Po uruchomieniu programu powłoki – znak zachęty w wierszu poleceń informuje o gotowości powłoki do odbioru poleceń.



W powyższym przykładzie znak zachęty składa się z następujących elementów:

geeko – nazwa użytkownika (login),

da51 – nazwa komputera,

~ - bieżąca pozycja w drzewie katalogów.

Ścieżka dostępu do katalogu domowego może być skrócona do znaku tyldy („~”)

Jeżeli zalogujemy się jako *root*, nazwa użytkownika nie jest pokazywana, a znak zachęty ma następującą postać:

```
da51:~ #
```

2.1.1 Poruszanie się w systemie plików

Do zmiany pozycji w drzewie katalogów służy polecenie ***cd*** (*change directory*).

Poniżej przedstawiono kilka przykładów:

- cd Documents – wejście do podkatalogu Documents

```
geeko@da51:~> cd Documents
geeko@da51:~/Documents>
```

- cd /tmp - zmiana pozycji na katalog /tmp (ścieżka bezwzględna)

```
geeko@da51:~> cd /tmp
geeko@da51:/tmp>
```

- cd - podanie tego polecenia spowoduje powrót (z dowolnego miejsca) do katalogu domowego

```
geeko@da51:/tmp> cd
geeko@da51:~>
```

- cd .. przenosi o poziom wyżej w drzewie katalogów

```
geeko@da51:~> cd ..
geeko@da51:/home>
```

- cd . . / . . przenosi o dwa poziomy wyżej w drzewie katalogów
- cd - przenosi do poprzedniego katalogu

```
geeko@da51:/home> cd /tmp
geeko@da51:/tmp> cd -
/home
geeko@da51:/home>
```

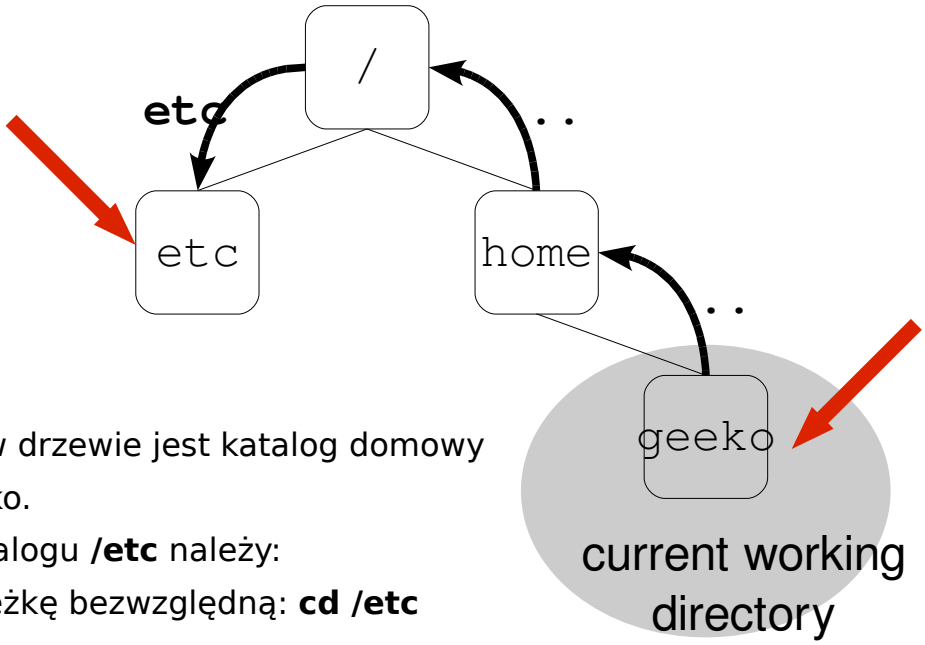
Nazwy katalogów w ścieżce dostępu oddzielone są znakiem ukośnika „/”(*slash*):

/usr/share/

Ścieżkę dostępu do pliku można podać na dwa sposoby:

- jako ścieżkę względną – względem aktualnej pozycji wdrzewie katalogów,
- jako ścieżkę bezwzględną – względem katalogu głównego (root) drzewa.

Ścieżka bezwzględna zawsze zaczyna się od znaku ukośnika („/”) - symbolu katalogu głównego (root) drzewa katalogów.



Na rysunku:

bieżącą pozycją w drzewie jest katalog domowy użytkownika geeko.

By przejść do katalogu **/etc** należy:

- podać ścieżkę bezwzględną: **cd /etc**

lub

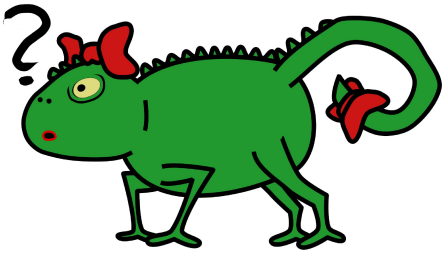
- podać ścieżkę względną: **cd ../../etc**

Maksymalna długość ścieżki dostępu nie może być dłuższa niż 4096 znaków, włącznie ze znakami ukośników.

Ćwiczenie. Przemieszczanie się w systemie plików

Wpisz w tabelce nazwę katalogu, do którego się przeniesiesz wprowadzając w wierszu poleceń odpowiednio:

Bieżąca pozycja	***Polecenie***	***Ostateczna pozycja***
/var/log/	cd cups	
/usr/X11R6/	cd bin	
/usr/X11R6/	cd /bin	
/usr/share/	cd . .	
/etc/postfix/	cd /	
~	cd . .	
/home	cd . .	
/media/cdrom/	cd ../ ../lib	
/sbin	cd	



2.1.2 Wyświetlanie zawartości katalogów

Polecenie **ls** (*list*) wyświetla zawartość katalogu.

Wprowadzenie w wierszu poleceń komendy **ls** bez podania żadnych opcji - wyświetli zawartość bieżącego katalogu:

```
geeko@da51:~> ls
bin  Desktop  Documents  public_html
geeko@da51:~>
```

Poniżej podano przykłady najbardziej przydatnych opcji (parametrów) polecenia **ls**:

- **bez podania żadnych opcji** – wyświetla w kilku kolumnach nazwy katalogów oraz plików znajdujących się w bieżącym katalogu:

```
geeko@da51:~> ls /var
adm  cache  games  lib  lock  log  mail  opt  run  spool  tmp  X11R6  yp
geeko@da51:~>
```

- **-a**

  wyświetla również ukryte pliki (na przykład **.bashrc**)

```
geeko@da51:~> ls -a /var
.   adm    games  lock  mail  run    tmp    yp
..  cache  lib    log   opt   spool  X11R6
geeko@da51:~>
```

- **-F**

  Po każdej wyświetlanej nazwie odpowiedni znak wskazuje na typ pliku:

  „**/**” to katalog, „*” plik wykonywalny (program), „**|**” plik FIFO, „**@**” dowiązanie symboliczne

```
geeko@da51:~> ls -F /var
adm/   games/  lock/  mail@  run/   tmp/   yp/
cache/  lib/   log/   opt/   spool/  X11R6/
geeko@da51:~>
```

- **-d**

  wyświetla tylko nazwy katalogów

```
geeko@da51:~> ls -d /var
/var
geeko@da51:~>
```

- -l („*long list*”)

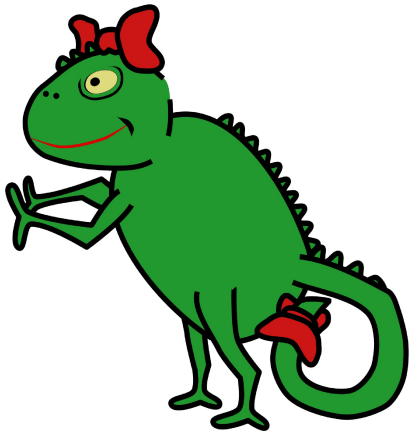
  Wyświetla szczegółowe informacje o plikach i katalogach. Są to informacje o uprawnieniach, czasie modyfikacji, wielkości pliku:

```
geeko@da51:~> ls -l /var
insgesamt 4
drwxr-xr-x  8 root  root   224 2006-09-04 10:21 adm
drwxr-xr-x 12 root  root   304 2006-09-04 10:26 cache
drwxrwxr-x  2 games games 2816 2006-09-04 10:37 games
drwxr-xr-x 39 root  root   968 2006-09-04 10:42 lib
drwxrwxr-t  4 root  uucp   128 2006-09-04 11:23 lock
drwxr-xr-x 11 root  root   936 2006-09-04 11:30 log
lrwxrwxrwx  1 root  root    10 2006-09-04 09:59 mail -> spool/mail
drwxr-xr-x  3 root  root    72 2006-09-04 10:06 opt
drwxr-xr-x 19 root  root   952 2006-09-04 11:30 run
drwxr-xr-x 11 root  root   296 2006-09-04 10:15 spool
drwxrwxrwt  3 root  root    80 2006-09-04 11:38 tmp
drwxr-xr-x  5 root  root   152 2006-09-04 10:13 X11R6
drwxr-xr-x  3 root  root   104 2006-09-04 10:15 yp
geeko@da51:~>
```

Rozpatrzmy dziesięć pierwszych znaków wiersza informacji: **- r w - r - - r - -**

Pierwszy znak z prawej wskazuje typ pliku:

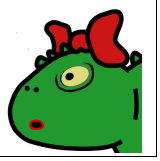
Znak	***Typ pliku***
-	zwykły plik
d	katalog
l	dowiązanie



Pozostałe dziewięć znaków pokazuje uprawnienia do pliku właściciela pliku, grupy oraz pozostałych użytkowników.

Można przydzielać trzy rodzaje uprawnień do pliku lub katalogu:

- **read (r)** pozwala na czytanie zawartości pliku oraz wyświetlanie zawartości katalogu,
- **write (w)** pozwala na modyfikację pliku oraz tworzenie i kasowanie plików w katalogu,
- **execute (x)** pozwala na wykonywanie pliku (gdy plik jest programem) oraz dokonywanie zmian w katalogu



Nadane uprawnienie reprezentowane jest przez odpowiedni znak (**rwx**), brak uprawnienia reprezentuje znak „-” (**---**).

Uprawnienia są pogrupowane (**rwx rwx rwx**):

- trzy pierwsze znaki reprezentują uprawnienia właściciela pliku (*file owner*),
- kolejne trzy – to uprawnienia grupy, która jest właścicielem (*owning group*),
- ostatnie trzy znaki reprezentują uprawnienia pozostałych użytkowników (*other users*).

Specjalne uprawnienia SUID , SGID, Sticky - reprezentowane są przez znaki „**s**” oraz ”**t**”.

- **SUID *set user ID*** – Ustawianie ID użytkownika. Program uruchamia się z uprawnieniami właściciela pliku

```
geeko@da51:~ > ls -l /usr/bin/passwd
-rwsr-xr-x 1 root shadow 79765 2006-03-24 12:19
/usr/bin/passwd
geeko@da51:~ >
```

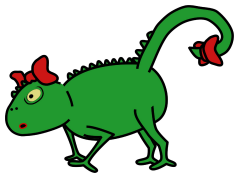
- **SGID set group ID** – Ustawianie ID grupy. Program uruchamia się za pozwoleniem powiązanej z nim grupy. Pliki znajdujące się w folderze przypisane są do grupy katalogu, a nie grupy użytkownika.

```
geeko@da51:~ > ls -l /usr/bin/wall
-rwxr-sr-x 1 root tty 10192 2006-03-22 05:24 /usr/bin/wall
geeko@da51:~ >
```

Sticky Sticky bit – bit „*lepkości*” -Użytkownik może usuwać pliki tylko w wypadku, gdy należą one do niego lub ma nadane bezpośrednio uprawnienie zapisu.

```
geeko@da51:~ > ls -ld /tmp
drwxrwxrwt 15 root root 608 2006-04-06 12:45 /tmp
geeko@da51:~ >
```

Liczba za uprawnieniami podaje ilość twardych dowiązań do pliku (w przykładzie jest ich 15). Dalej, kolejno podana jest nazwa właściciela pliku i nazwa grupy, do której plik należy; data i czas ostatniej modyfikacji oraz nazwa pliku (katalogu).



-t

Pliki są posortowane według daty ostatniej modyfikacji

```
geeko@da51:~> ls -t /var
tmp run log lock lib games cache adm spool yp X11R6 opt mail
geeko@da51:~>
```

dodanie opcji – r spowoduje, że będą posortowanie według daty, ale w odwrotnym porządku (najnowsze na końcu)

-R

listowana jest też zawartość wszystkich podkatalogów

```
geeko@da51:~> ls -R /var
/var:
adm
cache
games
lib
lock
log
mail
opt
run
spool
tmp
X11R6
```

```
yp
/var/adm:
autoinstall
backup
fillup-templates
mount
perl-modules
SuSEconfig
/var/adm/fillup-templates:
group.aaa_base
passwd.aaa_base
…
```

-u

pliki są posortowane według daty ostatniego dostępu

```
geeko@da51:~> ls -u /var
adm  cache  games  lib  lock  log  opt  run  spool  tmp  X11R6  yp  mail
geeko@da51:~>
```

W poleceniu ls można podawać różne kombinacje opcji

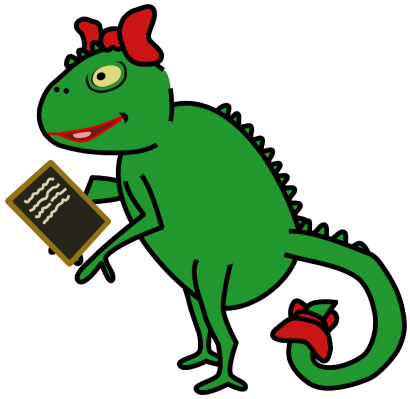
```
geeko@da51:~> ls -la /var
insgesamt 5
drwxr-xr-x 14 root  root   360 2006-09-04 10:04 .
drwxr-xr-x 22 root  root   520 2006-09-04 11:19 ..
drwxr-xr-x  8 root  root   224 2006-09-04 10:21 adm
drwxr-xr-x 12 root  root   304 2006-09-04 10:26 cache
drwxrwxr-x  2 games games 2816 2006-09-04 10:37 games
drwxr-xr-x 39 root  root   968 2006-09-04 10:42 lib
drwxrwxr-t  4 root  uucp   128 2006-09-04 11:23 lock
drwxr-xr-x 11 root  root   936 2006-09-04 11:30 log
lrwxrwxrwx  1 root  root    10 2006-09-04 09:59 mail -> spool/mail
drwxr-xr-x  3 root  root    72 2006-09-04 10:06 opt
drwxr-xr-x 19 root  root   952 2006-09-04 11:30 run
drwxr-xr-x 11 root  root   296 2006-09-04 10:15 spool
```

```
drwxrwxrwt  3 root  root    80 2006-09-04 11:38 tmp
drwxr-xr-x  5 root  root   152 2006-09-04 10:13 X11R6
drwxr-xr-x  3 root  root   104 2006-09-04 10:15 yp
geeko@da51:~>
```

Ćwiczenie. Wyświetlanie zawartości katalogu – część I

Ile podkatalogów jest w podanych katalogach:

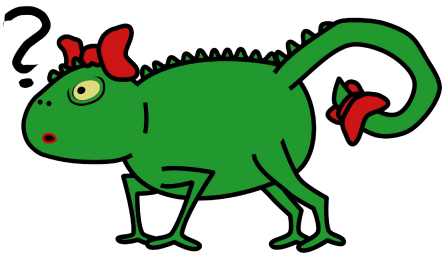
Katalog	*Liczba podkatalogów*
/	
/sbin	
/var/log/	
/etc/init.d/	
/usr/share/doc/	
/bin	



Ćwiczenie. Wyświetlanie zawartości katalogu – część II

Uzupełnij poniższą tabelkę o nazwy właścicieli - użytkownika i grupy oraz uprawnienia.

plik/katalog	*użytkownik*	*grupa*	*uprawnienia*
~/.nautilus/			
/tmp			
/var/spool/mail/			
/etc/init.d/autofs			
/usr/bin/passwd/			
/etc/passwd			
/etc/shadow			
/usr/bin/wall			



2.1.3 Przenoszenie pliku

Polecenie **mv** (*move*) służy do przenoszenia plików oraz do zmiany ich nazwy.

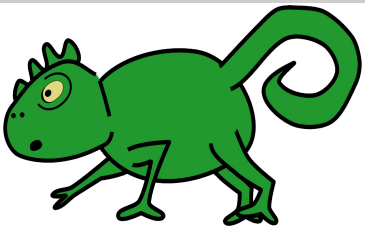
Składnia polecenia: **mv *źródło przeznaczenie***

Na przykład, by przenieść plik **index.html** do katalogu **/tmp**, napiszemy:

```
geeko@da51:~> mv index.html /tmp
geeko@da51:~>
```

Uwaga. Bash nie informuje o zakończeniu operacji sukcesem. Tylko informacje o błędach są wyświetlane.

```
geeko@da51:~> mv index.html /tmp
mv: cannot stat `index.html': No such file or directory
geeko@da51:~>
```



Uwaga. Jeżeli plik o tej samej nazwie co przenoszony już istnieje w docelowym katalogu – zostanie nadpisany bez informowania o tym!

Można użyć polecenia **mv** do zmiany nazwy pliku, na przykład:

```
geeko@da51:~> mv index.html index_en.html
geeko@da51:~>
```

Zostanie zmieniona nazwa pliku **index.html** na **index_en.html.**

Różnica w składni między oboma zastosowaniami polecenia **mv** polega na podaniu jako wartości drugiego argumentu polecenia - nazwy docelowego katalogu przy przenoszeniu pliku i nowej nazwy pliku przy zmianie nazwy.

Jeżeli chcemy przenieść więcej niż jeden plik, możemy użyć w podawanej wartości pierwszego argumentu - zmiennych globalnych „?” (za jeden znak) oraz „*” (za żaden, jeden, kilka znaków).

Przykład:

```
geeko@da51:~> mv *.html /tmp
geeko@da51:~>
```

Wszystkie pliki w bieżącym katalogu mające rozszerzenie **.html** zostaną przeniesione do katalogu **/tmp**.

Składnia polecenia **mv** ma kilka opcji. Najważniejsze dwie, to:

- **-i**

  wprowadza obowiązek konsultacji przy przenoszeniu lub zmianie nazwy pliku. Zapobiega to niebezpieczeństwu niekontrolowanego nadpisania pliku o tej samej nazwie.

```
geeko@da51:~> mv -i index.html /tmp
mv: overwrite `/tmp/index.html'? n
geeko@da51:~> mv index.html /tmp
geeko@da51:~>
```

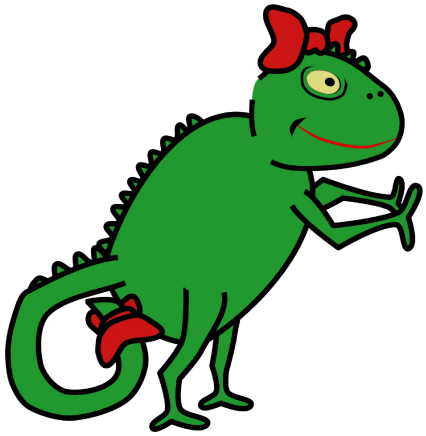
- **-u**

  przenosi tylko pliki nowsze niż te (o tej samej nazwie), które są już w katalogu docelowym:

```
1  geeko@da51:~> ls -l index.html
2  -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 index.html
3  geeko@da51:~> ls -l /tmp/index.html
4  -rw-r--r-- 1 geeko users 864 2006-09-04 11:26 /tmp/index.html
5  geeko@da51:~> mv -u index.html /tmp/
6  geeko@da51:~> ls -l index.html
7  -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 index.html
8  geeko@da51:~> ls -l /tmp/index.html
9  -rw-r--r-- 1 geeko users 864 2006-09-04 11:26 /tmp/index.html
10 geeko@da51:~> mv index.html /tmp/
11 geeko@da51:~> ls -l /tmp/index.html
12 -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 /tmp/index.html
13 geeko@da51:~> ls -l index.html
14 /bin/ls: index.html: No such file or directory
15 geeko@da51:~>
```

Wyjaśnienie. Plik ~/index.html został modyfikowany w grudniu 2005 roku i ma wielkość 26432 bajtów (wiersze 1-2).

Plik /tmp/index.html był modyfikowany we wrześniu 2006 i ma wielkość 864 bajty (wiersze 3-4).
Po przeniesieniu pliku ~/index.html do katalogu /tmp (wiersz 5) z opcją -u, plik ~/index.html istnieje nadal (wiersze 6-7) i plik przeznaczenia ma te same dane (864 bajty, wiersze 8-9), co oznacza, ze plik nie został przeniesiony.
Polecenie mv bez opcji -u (wiersz 10) nadpisuje plik przeznaczenia mimo tego, że jest on nowszy niż ~/index.html (wiersze 11-12).
Plik ~/index.html już nie istnieje (wiersze 13-14).



2.1.4 Kopiowanie pliku

Do kopiowania plików i katalogów służy polecenie **cp** (*copy*).

Składnia polecenia cp: **cp *źródło przeznaczenie***

Używając polecenia **cp** pamiętaj:

- polecenie **cp** nadpisuje istniejące pliki bez pytania o zgodę,
- jeżeli chcemy przekopiować zawartość katalogu (bez samego katalogu) – katalog docelowy musi już istnieć.

Polecenie **cp** bez dodatkowych opcji kopiuje pliki.

Przy kopiowaniu katalogów trzeba użyć

- opcji **-r**:

```
geeko@da51:~> cp proposals/ /tmp
cp: omitting directory `proposals/'
geeko@da51:~> cp -r proposals/ /tmp
geeko@da51:~>
```

W tym przykładzie katalog ~/proposals/ ze wszystkimi podkatalogami jest kopiowany do katalogu /tmp, wynikiem jest katalog /tmp/proposals/.

By skopiować wyłącznie zawartość katalogu proposals/ (włącznie z plikami ukrytymi oraz podkatalogami), napiszemy:

```
geeko@da51:~ > cp -r proposals/. /tmp
geeko@da51:~ >
```

Jeżeli nie chcemy przekopiować ukrytych plików, napiszemy:

```
geeko@da51:~ > cp -r proposals/* /tmp
geeko@da51:~ >
```

Poza **-r**, najważniejszymi opcjami polecenia **cp**, są:

- **-a, --archive**

  Kopiuje katalogi i podkatalogi (jak opcja -r); symboliczne dowiązania, uprawnienia plików, właściciele, znaczniki czasu pozostają niezmienione.

```
1 geeko@da51:~> ls -a index.html
2 -rw-r--r-- 1 geeko users 864 2006-09-04 11:26 index.html
3 geeko@da51:~> cp -a index.html /tmp/
4 geeko@da51:~> ls -l /tmp/index.html
5 -rw-r--r-- 1 geeko users 864 2006-09-04 11:26 /tmp/index.html
6 geeko@da51:~> cp index.html /tmp/
7 geeko@da51:~> ls -l /tmp/index.html
8 -rw-r--r-- 1 geeko users 864 2006-09-04 13:35 /tmp/index.html
9 geeko@da51:~>
```

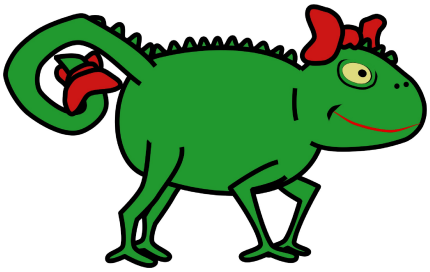
  Wyjaśnienie. W tym przypadku ostania zmiana pliku ~/index.html była o godzinie 11:26 (wiersze 1-2).
  Po przekopiowaniu pliku z opcją -a (wiersz 3), czas modyfikacji kopii to również 11:26 (wiersz 4-5).
  Gdy nie użyjemy opcji -a (wiersz 6) czas modyfikacji ustawiany jest na czas bieżący (tu: 13:35, wiersze 7-8).

- **-i, --interactive,** spyta przed napisaniem:

```
geeko@da51:~> cp -i index.html /tmp
cp: overwrite `/tmp/index.html'? n
geeko@da51:~> cp index.html /tmp
geeko@da51:~>
```

- **-s, --symbolic – link,** tworzy symboliczne dowiązanie zamiast kopii:

```
geeko@da51:~> cp -s /tmp/index.html .
geeko@da51:~> ls -l index.html
lrwxrwxrwx 1 geeko users 15 2006-09-04 13:37 index.html -> /tmp/index.html
geeko@da51:~>
```



-u, --update, kopiuje plik wtedy i tylko wtedy, gdy plik źródłowy jest nowszy niż plik docelowy lub plik w miejscu docelowym nie istnieje.

```
1  geeko@da51:~> ls -l index.html
2  -rw-r--r-- 1 geeko users 26432 2005-12-08 15:48 index.html
3  geeko@da51:~> ls -l /tmp/index.html
4  -rw-r--r-- 1 geeko users 864 2006-09-04 13:40 /tmp/index.html
5  geeko@da51:~> cp -u index.html /tmp/
6  geeko@da51:~> ls -l /tmp/index.html
7  -rw-r--r-- 1 geeko users 864 2006-09-04 13:40 /tmp/index.html
8  geeko@da51:~> cp index.html /tmp/
9  geeko@da51:~> ls -l /tmp/index.html
10 -rw-r--r-- 1 geeko users 26432 2006-09-04 13:42 /tmp/index.html
11 geeko@da51:~>
```

Wyjaśnienie. Plik ~/index.html był modyfikowany w grudniu 2005 i ma wielkość 26432 bajty (wiersze 1-2).
Plik /tmp/index.html był modyfikowany we wrześniu 2006 i ma wielkość 864 bajty (wiersze 3-4).
Po skopiowaniu pliku ~/index.html do katalogu /tmp z opcją – u (wiersz 5), plik przeznaczenia pozostaje ten sam (wiersze 6-7).
Gdy nie użyjemy opcji -u (wiersz 8) – polecenie cp nadpisze nowszy plik (wiersze 9-10).

Ćwiczenie: Przenoszenie i kopiowanie plików

Wykonaj, co następuje:

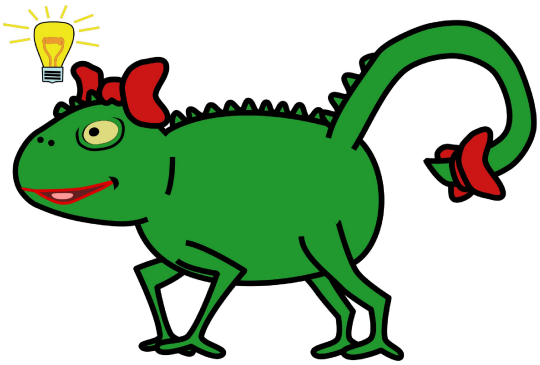
1.Przekopiuj plik /etc/init.d/syslog do swojego katalogu domowego.

Jak zmienią się uprawnienia?

_____

2.Zmień nazwę pliku ~/syslog na testfile.

3.Przekopiuj plik z dowolnej płytki CD do swojego katalogu domowego.



2.1.5 Tworzenie katalogów

Do tworzenia katalogów w wierszu poleceń można użyć polecenia **mkdir** (*make directory*):

```
geeko@da51:~> mkdir Proposal
geeko@da51:~>
```

Dodając do polecenia opcję **-p** – możemy tworzyć kompletną ścieżkę, jak w poniższym przykładzie:

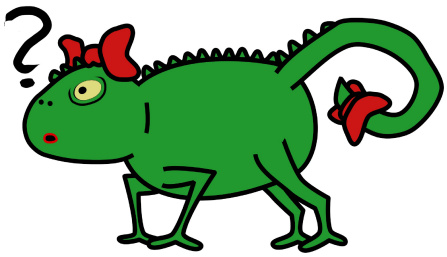
```
1  geeko@da51:~> ls
2  bin  Desktop  Documents  public_html
3  geeko@da51:~> mkdir Invoice/Customer
4  mkdir: cannot create directory `Invoice/Customer': No such file or directory
5  geeko@da51:~> mkdir -p Invoice/Customer
6  geeko@da51:~> ls
7  bin  Desktop  Documents  Invoice  public_html
8  geeko@da51:~> ls Invoice/
9  Customer
10 geeko@da51:~>
```

Wyjaśnienie. By utworzyć od razu katalog Invoice z podkatalogiem Customer (wiersze 3-4) – należy użyć opcji -p (wiersz 5).

Ćwiczenie: Tworzenie katalogu

Wykonaj poniższe polecenia:
1.Utwórz nowy katalog o nazwie Marketing w swoim katalogu domowym.
2.Przesuń plik testfile do katalogu Marketing.
3.Utwórz w katalogu /tmp nowy katalog Szkola z podkatalogiem Matma.
4.Przekopiuj katalog Marketing z katalogu domowego do /tmp/Szkola/Matma

5.Utwórz nowy katalog z podkatalogiem Angielski/lab w katalogu Szkola.



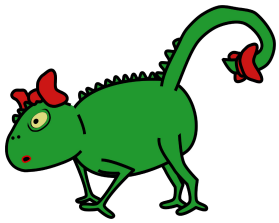
2.1.6Kasowanie katalogów i plików

Poleceniem **rmdir** (*remove directory*) można kasować katalogi. Przykładowo:

```
geeko@da51:~> rmdir Proposal
geeko@da51:~>
```

Katalogi muszą być puste by móc je skasować:

```
geeko@da51:~> rmdir Invoice
rmdir: Invoice: Directory not empty
geeko@da51:~> ls Invoice/
Customer
geeko@da51:~>
```



Do kasowania plików można użyć polecenia **rm** (*remove*), jak w poniższym przykładzie:

```
geeko@da51:~> rm file
geeko@da51:~>
```

Przy używaniu w poleceniu **rm** zmiennych globalnych („*”) należy bardzo uważać, by skasować te pliki, które naprawdę chcemy skasować.

```
geeko@da51:~> ls
bin      Documents  file2  file4  file6  Invoice
Desktop  file1      file3  file5  file7  public_html
```

```
geeko@da51:~> rm file*
geeko@da51:~>
```

W powyższym przykładzie zostaną skasowanie, bez pytania o potwierdzenie, WSZYSTKIE pliki, których nazwa zaczyna się od *file.*

Uwaga. Plików skasowanych poleceniem rm nie da się odtworzyć.

Jeżeli użytkownik nie ma uprawnienia do kasowania plików, - polecenie **rm** zostaje zignorowane i wyświetli się komunikat o błędzie.

```
geeko@da51:~> rm /etc/vimrc
rm: remove write-protected regular file `/etc/vimrc'? y
rm: cannot remove `/etc/vimrc': Permission denied
geeko@da51:~>
```

Najważniejsze opcje polecenia **rm**, to: -i, -r, -f.

- **-i** pyta się przed skasowaniem:

```
geeko@da51:~> rm -i file
rm: remove regular empty file `file'? y
geeko@da51:~>
```

- **-r** pozwala na skasowanie niepustych katalogów:

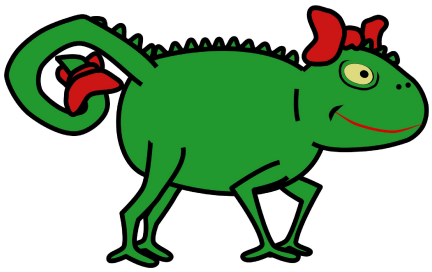
```
geeko@da51:~> rmdir Invoice
rmdir: Invoice: Directory not empty
geeko@da51:~> rm -r Invoice
geeko@da51:~>
```

- **-f** domyślnie **rm** pyta się przed skasowaniem plików z atrybutem **read only** (tylko do odczytu). Przy użyciu opcji **-f** – kasuje takie pliki bez uprzedniego zapytania.

```
1 geeko@da51:~> ls -l file
2 -r--r--r-- 1 geeko users 0 2006-09-04 11:44 file
3 geeko@da51:~> rm file
4 rm: remove write-protected regular empty file `file'? n
5 geeko@da51:~> rm -f file
6 geeko@da51:~>
```

Wyjaśnienie. Plik **file** ma atrybut *read-only* (tylko do odczytu) (wiersze 1-2). Jeżeli użyjemy polecenia rm bez żadnej opcji – pojawia się pytanie z prośbą

o potwierdzenie (wiersze 3-4).
Przy użyciu polecenia rm z opcją -f – pliki zostaną skasowane bez prośby o potwierdzenie (wiersz 5).



Ćwiczenie. Kasowanie katalogów i plików

Wykonaj, co następuje:

1.Skasuj plik /tmp/Szkola/Matma/Marketing/testfile.

2.Skasuj pusty katalog /tmp/Szkola/Angielski/lab/.

3.Z katalogu /tmp/ usuń katalog Szkola z całą zawartością.

2.1.7 Dowiązania plików

W uniksowych i linuksowych systemach plików dane i informacje administracyjne są przechowywane osobno. Dane są zorganizowane zgodnie z właściwym dla danego systemu plików formatem. Każdy plik jest opisywany przez tzw. **i-węzeł** (*inode: index node* lub *information node*).

By zobaczyć numery i-węzłów dla konkretnych plików, należy w wierszu poleceń użyć polecenia **ls -i** :

```
geeko@da51:~> ls -li
insgesamt 0
84800 drwxr-xr-x 2 geeko users 48 2006-09-22 11:20 bin
84811 drwxr-xr-x 2 geeko users 80 2006-09-22 11:20 Desktop
84801 drwx------ 2 geeko users 80 2006-09-22 11:20 Documents
84808 drwxr-xr-x 2 geeko users 80 2006-09-22 11:20 public_html
geeko@da51:~>
```

Każdy z i-węzłów ma wielkość 128 bajtów i zawiera wszystkie informacje opisujące plik z wyjątkiem jego nazwy. I-węzeł zawiera szczegóły dotyczące właściciela pliku, uprawnień, wielkości, parametry czasu i daty (czas modyfikacji, dostępu, modyfikacji i węzła) oraz dowiązania (*links*) do bloków danych pliku.

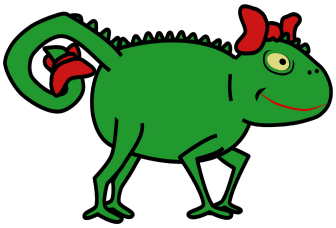
Polecenie liniowe **ln** tworzy *dowiązanie* do pliku. *Dowiązanie* to odnośnik - odwołanie do pliku. Dzięki dowiązaniom – można dostać się do pliku z każdego

miejsca w systemie plików używając różnych nazw pliku. Oznacza to, że sam plik (bloki danych) istnieje tylko w jednym egzemplarzu w systemie plików, natomiast może być udostępniany pod różnymi nazwami z różnych miejsc w logicznym drzewie katalogów i plików.

W Linuksie dostępne są dwa typy dowiązań:

- twarde dowiązania (*hard links*),
- symboliczne dowiązania (*symbolic links*).

- **Twarde dowiązanie**, wskazujące bezpośrednio na i-węzeł konkretnego pliku, tworzymy poleceniem **ln**. Do pliku możemy mieć dostęp zarówno przez nazwę samego pliku, jak i nazwę jego dowiązania. Nie widać różnicy między plikiem a twardym dowiązaniem do niego.



Przykład zastosowania polecenia **ln** do tworzenia twardego dowiązania:

```
1 geeko@da51:~/Proposal > ls -li
2 total 4
3 88658 -rw-r--r-- 1 geeko users 82 2004-04-06 14:21 old
4 geeko@da51:~/Proposal > ln old new
5 geeko@da51:~/Proposal > ls -li
6 total 8
7 88658 -rw-r--r-- 2 geeko users 82 2004-04-06 14:21 old
8 88658 -rw-r--r-- 2 geeko users 82 2004-04-06 14:21 new
9 geeko@da51:~/Proposal >
```

Wyjaśnienie. ls –i wyświetla i-węzeł pliku (wiersze 1-3). W powyższym przykładzie i-węzeł pliku old ma numer 88658. Poleceniem ln tworzymy twarde dowiązanie (wiersz 4). Nowe dowiązanie ma ten sam i-węzeł co oryginalny plik (wiersze 5-8).

Polecenie ls -i wyświetla wyłącznie numery i-węzłów:

```
macbook-malgosia:~ malgosia$ ls -i
1595516 Załączniki do umowy.pdf
```

```
1224524 ptun.odp
 964832 testycert1.png
```

Jeżeli użyjemy polecenia z opcjami l oraz i, czyli l**s -li**; wyświetlone zostanie o wiele więcej informacji:

```
macbook-malgosia:~ malgosia$ ls -li
total 2200
1224524 -rw-r--r--@   1 malgosia  staff  1082034 22 sie 12:14 ptun.odp
 964832 -rwxrwxrwx    1 malgosia staff    36519 23 cze 13:10 testycert1.png
macbook-malgosia:~ malgosia$
```

Twarde dowiązania mogą być użyte wyłącznie wewnątrz jednego systemu plików (i jednej partycji dysku), ponieważ numery i-węzłów są unikalne tylko wewnątrz jednego systemu plików.

Liczba twardych dowiązań do i-węzła danego pliku może być wyświetlona poleceniem ls -li. Jest to liczba widoczna w wierszu dotyczącym danego pliku - za uprawnieniami.

Przykład:

```
macbook-malgosia:~ malgosia$ ls -li
 964832 -rwxrwxrwx    1 malgosia staff    36519 23 cze 13:10 testycert1.png
macbook-malgosia:~ malgosia$
```

- **Dowiązanie symboliczne** ma przydzielony swój własny i-węzeł (odnośnik do pliku) tak, że zawsze można rozróżnić dowiązanie od pliku, którego dotyczy.

Można dowiązanie symboliczne utworzyć poleceniem **ln** z opcją **-s**.

Przykład tworzenia dowiązania symbolicznego:

```
1 geeko@da51:~/Proposal > ls -li
2 total 4
3 88658 -rw-r--r-- 1 geeko users 82 2004-04-06 14:21 old
4 geeko@da51:~/Proposal > ln -s old new
5 geeko@da51:~/Proposal > ls -li
6 total 4
7 88658 -rw-r--r-- 1 geeko users 82 2004-04-06 14:21 old
8 88657 lrwxrwxrwx 1 geeko users 3 2004-04-06 14:27 new -> old
```

```
9 geeko@da51:~/Proposal >
```

Wyjaśnienie. W przykładzie plik old ma i-węzeł o numerze 88658 (wiersze 1-3). Poleceniem ln -s tworzymy dowiązanie symboliczne o nazwie new do pliku old (wiersz 4).
Nowe dowiązanie ma inny i-węzeł, o numerze 88657 co oryginalny plik (wiersze 7-8).

Wadą dowiązań symbolicznych jest to, że nie są automatycznie kasowane w przypadku usunięcia oryginalnego pliku; mogą więc wskazywać na nieistniejący obiekt.

W omawianym przykładzie – po usunięciu pliku **old**, dowiązanie **new** istnieje nadal – wskazując na nieistniejący plik. Polecenie **ls** nie informuje o tym:

```
geeko@da51:~/Proposal > rm old
geeko@da51:~/Proposal > ls -li
total 0
88657 lrwxrwxrwx 1 geeko users 3 2004-04-06 14:27 new -> old
geeko@da51:~/Proposal >
```

Symboliczne dowiązania mogą być tworzone również do katalogów.

Ćwiczenie. Dowiązania do plików

Wykonaj, co następuje:

1.Odpowiedz na pytanie: Jak dużo dowiązań ma poniższy katalog?

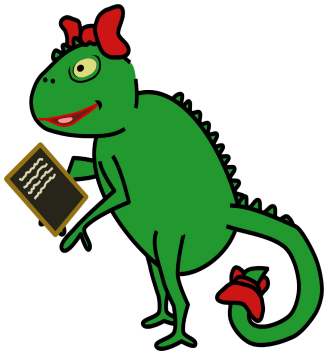
/boot/boot/boot/boot/boot/boot/boot/boot/boot/

2.Utwórz symboliczne dowiązanie o nazwie symblink w swoim domowym katalogu - do pliku Marketing/testfile.

3.Utwórz twarde dowiązanie o nazwie hardlink w swoim domowym katalogu – wskazujące na plik Marketing/testfile.

4.Usuń plik Marketing/testfile

5.Sprawdź dowiązania poleceniem cat *nazwa_pliku*. Polecenie **cat** wyświetla zawartość pliku.



2.1.8 Wyszukiwanie plików

Każda powłoka (*shell*) dostarcza poleceń do wyszukiwania plików w systemie plików.

Najważniejsze polecenia do wyszukiwania plików, to:

- **find,**
- **locate**.

Wyszukiwanie plików za pomocą polecenia find

Składnia polecenia find ma postać: **find** *ścieżka kryterium działanie*

Przykład:

```
geeko@da51:~> find . -name "File?"
./File1
./File2
geeko@da51:~>
```

Polecenie **find** ma dużo różnych opcji, argumentów i parametrów. Poniżej wyjaśniono tylko niektóre ważniejsze.

- Argument ***ścieżka*** definiuje część systemu plików do przeszukania (określony katalog oraz wszystkie jego podkatalogi). Jeżeli nic nie jest podane – bieżący katalog i jego podkatalogi przyjmowane są za wartość argumentu.
- Argument ***kryterium*** definiuje cechy, które wyszukiwane pliki mają mieć. Możliwe są, między innymi:

  -ctime [+/-] *dni* wyszukiwane są pliki, których zawartość zmieniono nie później (lub nie wcześniej) niż podaną ilość dni.

  -group *nazwa* wyszukiwane są pliki, których właścicielem jest grupa o podanej nazwie. Zamiast nazwy można użyć identyfikatora grupy GID.

  -name *wzorzec* wyszukiwane są pliki, których nazwa jest zgodna z podanym w poleceniu wzorcem. Jeżeli wzorzec zawiera znaki globalne (*, ?) - musi być zawarty w cudzysłowie. Jeżeli nie będzie w cudzysłowie

– zostanie przekazany do interpretacji powłoce, a nie w ramach polecenia **find.**

-size [+/-] *rozmiar* wyszukuje pliki większe lub mniejsze od podanej argumentem *rozmiar* wielkości. Argumentem jest liczba będąca wielokrotnością bloku 512 bajtów.

Przyrostek „**c**” pozwala na podawanie ilości bajtów (np. **-size -260c** → pliki mniejsze od 260 bajtów), a przyrostek „**k**” - kilobajtów (przykład **-size +600k** → pliki większe od 600 kilobajtów).

-type *typ_pliku* wyszukuje pliki o podanym typie. Możliwe wartości argumentu typ_pliku, to: **d** → katalog; **f** → plik; **l** → dowiązanie symboliczne.

-user *użytkownik* wyszukuje pliki należące do określonego użytkownika. Zamiast nazwy można użyć identyfikatora użytkownika UID.

Argument ***działanie*** wpływa na podane warunki lub określa przeznaczenie całego polecenia wyszukiwania, przykładowo:

-print (default)

-exec *polecenie* W ten sposób można wywołać inne polecenie. Jest to czasem stosowane przy łączeniu polecenia find z poleceniem grep, przykładowo:

```
1 geeko@da51:~ > find ~ -name "letter*" -type f -exec grep appointment {} \;
2 appointment for next meeting: 23.08.
3 /home/geeko/letters/letter_Smith
4 geeko@da51:~ >
```

Wyjaśnienie. Polecenie **find** ma wyszukać pliki, których nazwa zaczyna się od **letter**, a następnie przekazać znalezioną listę do polecenia **grep appointment {};** nawiasy klamrowe wskazują miejsce na wstawienie listy; średnik **;** zamyka polecenie **exec**, a ponieważ jest to znak specjalny – poprzedza go ukośnik **** (wiersz 1).

Polecenie **grep** wyszukuje określone wyrażenie w podanym pliku. **grep** razem z **find** pozwala na odnalezienie wyrażenia w pliku, którego lokalizacja jest nieznana.

Ćwiczenie. Wyszukiwanie plików

Odpowiedz na poniższe pytania:

1.Ile plików w katalogu /bin jest ma wielkość większą od 1024kB?

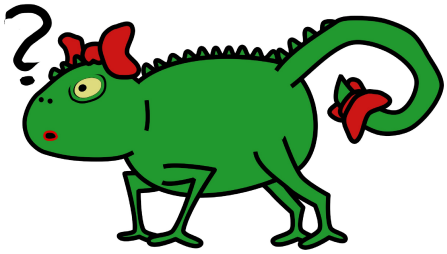
_____

2.Ile plików w katalogu /etc należy do grupy lp ?

_____

3.Gdzie w systemie plików znajduje się polecenie grep?

_____



Polecenie locate

Uwaga. Polecenie **locate** nie jest dostępne po standardowej instalacji systemu. Należy zainstalować pakiet **findutils-locate**, by móc używać tego polecenia.

Polecenie **locate** odpowiada poleceniu **find -name.**

Ponieważ polecenie **find** przeszukuje wskazany podsystem plików – może to trwać dłuższą chwilę. Polecenie **locate** przeszukuje wyłącznie bazę danych stworzoną specjalnie do tego celu (**/var/lib/locatedb**), co znacznie przyśpiesza proces wyszukiwania.

W systemie SLED baza ta jest tworzona automatycznie i aktualizowana codziennie. Niezależnie można w dowolnym momencie przeprowadzić ręczną aktualizację poleceniem **updatedb**.

Uwaga. Po zainstalowaniu pakietu **findutils-locate** należy bazę utworzyć ręcznie poleceniem **updatedb** (na koncie roota), w przeciwnym wypadku będą wyświetlane błędy po wydaniu polecenia **locate**:

```
geeko@da51:~> locate File?
locate: /var/lib/locatedb: No such file or directory
```

Polecenie **updatedb** można skutecznie wykonywać w wierszu poleceń wyłącznie na konto użytkownika **root**. W tym celu należy poleceniem **su** przełączyć się na użytkownika **root**.

By zrobić to jedynie w celu wydania pojedynczego polecenia, możemy użyć opcji -c :

su -c *polecenie*

a następnie potwierdzić tożsamość użytkownika root podając jego hasło.

```
geeko@da51:~> su -c updatedb
```

```
Password:
geeko@da51:~>
```

Poniżej pokazano efekt przykładowego polecenia **locate**:

```
geeko@da51:~> locate File?
/home/geeko/File1
/home/geeko/File1a
/home/geeko/File1b
/home/geeko/File2
geeko@da51:~>
```

locate wyświetla wszystkie pliki, których nazwy zawierają poszukiwany ciąg znaków.

By nauczyć się więcej o poleceniu **locate** – użyj, jak zresztą w każdym innym przypadku, polecenia **man locate** by wywołać szczegółowy opis polecenia z bazy wiedzy (*manual*).

Ćwiczenie. Lokalizowanie plików

Upewnij się, że pakiet findutils-locate jest zainstalowany i zaktualizowana jest baza używana przez polecenie locate.

Użyj polecenia locate do odszukania lokalizacji następujących plików:

updatedb:_____

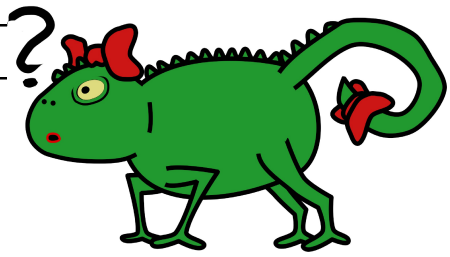
XF86Config:_____

smb.conf:_____

hardlink:_____

boot.log:_____

installkernel:_____



Odnajdowanie plików wykonywalnych za pomocą polecenia which

Polecenie **which** przeszukuje wszystkie ścieżki dostępu podane w zmiennej PATH dla danego polecenia i jako wynik polecenia zwraca pełną ścieżkę

dostępu do pierwszego wystąpienia poszukiwanego pliku.

Zmienną nazywa się oznaczoną etykietą przestrzeń w pamięci, gdzie można przechowywać określone informacje.

W zmiennej PATH przechowywane są najważniejsze katalogi, które sprawdza powłoka wyszukując programy – pliki wykonywalne.

By sprawdzić zawartość zmiennej, użyj polecenia **echo** z przedrostkiem „**$**” przed nazwą zmiennej.

Przykłady:

```
macbook-malgosia:~ malgosia$ echo $PATH
/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/X11/bin
```

```
geeko@da51:~> echo $PATH
/home/geeko/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/gnom
e/bin:/opt/kde3/bin:/usr/lib/jvm/jre/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin
geeko@da51:~>
```

Jeżeli kilka wersji polecenia jest w różnych miejscach systemu plików, a my chcemy dowiedzieć się, która konkretnie wersja (plik) jest uruchamiany gdy podajemy nazwę programu bez podania ścieżki dostępu; należy użyć polecenia **which**.

Przykłady:

```
geeko@da51:~ > which find
/usr/bin/find
geeko@da51:~ > which cp
/bin/cp
geeko@da51:~> which locate
/usr/bin/locate
geeko@da51:~ >
```

By nauczyć się więcej o poleceniu **which** – użyj polecenia **man which.**

Ćwiczenie. Lokalizowanie plików wykonywalnych

Użyj polecenia which do odszukania lokalizacji następujących programów:

grep:_____

which:_____

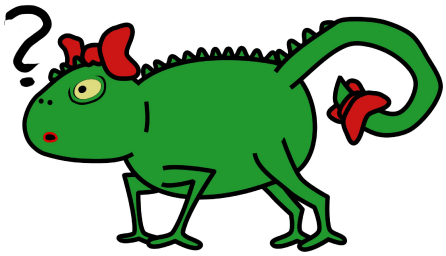
mkdir:_____

nautilus:_____

OOo-writer:_____

ktab:_____

fvwm2:_____



Określenie rodzaju polecenia

Polecenie **type** umożliwia określenie rodzaju wywoływanego polecenia:

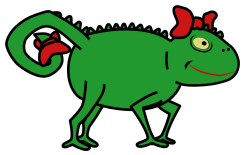
- wbudowane polecenie powłoki (w kodzie źródłowym powłoki),
- polecenie zewnętrzne (wywołane przez powłokę),
- alias innego polecenia,
- funkcja.

Opcja **-a** wyświetla wszystkie instancje polecenia z nazwą systemu plików.

Przykład użycia polecenia **type**:

```
geeko@da51:~ > type type
type is a shell built in
geeko@da51:~ > type grep
grep is /usr/bin/grep
geeko@da51:~ > type -a grep
grep is /usr/bin/grep
grep is /bin/grep
geeko@da51:~ >
```

Ćwiczenie. Typy poleceń



Jakiego rodzaju są podane pliki:

echo:_____

ls:_____

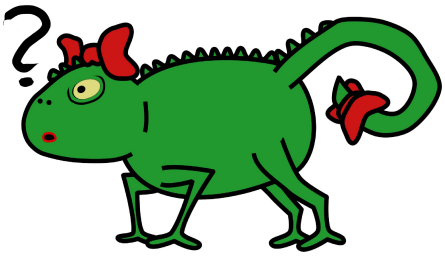
which:_____

test:_____

grep:_____

type:_____

man:_____



2.1.9 Archiwizacja plików

tar

tar (*tape archiver*) jest najczęściej używanym narzędziem do backupu (kopii bezpieczeństwa) danych. Tar archiwizuje pliki w specjalnym formacie; albo bezpośrednio na odpowiednim medium (takim, jak taśma magnetyczna) lub w tzw. pliku archiwum. Domyślnie dane są nieskompresowane (niespakowane).

Pliki zarchiwizowane mają rozszerzenie **.tar**. Gdy są dodatkowo skompresowane (zwykle poleceniem **gzip**), rozszerzenie pliku ma postać **.tar.gz** lub **.tgz**.

Składnia polecenia: **tar *opcja pliki***

Argument ***opcja*** określa dokładnie co i w jaki sposób ma być zarchiwizowane. Następnie musi być określone co (nazwa katalogu) ma być zarchiwizowane. Będą również zachowane wszystkie katalogi i pliki poniżej.

Bardzo ważną opcją jest **-f** (*file*)

- **-f *filename,*** określa nazwę archiwum (lub plik urządzenia).

Najczęściej archiwizuje się katalogi poleceniem **tar** w postaci:

```
geeko@da51:~ > tar -cvf /tmp/backup.tar /home/geeko
```

Wyjaśnienie. Polecenie powyższe archiwizuje pełną zawartość katalogu domowego użytkownika geeko i przechowuje archiwum w pliku /tmp/backup.tar.

- Opcja **– c** (*create*) tworzy archiwum.
- Opcja **-v** (*verbose*) dostarcza szczegółowy wykaz (nazwy) przenoszonych

do archiwum plików.

Po stworzeniu archiwum, ścieżka bezwzględna jest zmieniana na względną poprzez usunięcie wiodącego ukośnika „/” ze ścieżki:

```
tar: Removing leading `/' from member names
```

By rozpakować archiwum, należy również użyć polecenia **tar**:

```
geeko@da51:~ > tar -xvf /tmp/backup.tar
```

Wyjaśnienie. To polecenie rozpakuje wszystkie pliki przechowywane w archiwum - do bieżącego katalogu. Struktura katalogów zostanie utworzona w oparciu o specyfikacje względnych ścieżek zapamiętane w archiwum tar.

Jeżeli chcemy rozpakować z archiwum tylko jeden plik, należy podać jego nazwę:

```
geeko@da51:~ > tar -xvf /tmp/backup.tar home/geeko/.bashrc
```

Użyteczne opcje polecenia tar

- **-C** archiwum zostanie rozpakowane do katalogu określonego opcją:

```
geeko@da51:~ > tar -xvf /tmp/backup.tar -C /data
```

- **-d** zostaną porównane pliki archiwum i systemu plików:

```
geeko@da51:~ > tar -dvf /tmp/backup.tar -C /home/geeko
```

- **-j** kompresuje lub dekompresuje archiwum **tar** programem **bzip2**:

```
geeko@da51:~ > tar -cjvf /tmp/backup.tar.bz2 /home/geeko
```

- **-r** dodaje pliki do archiwum:

```
geeko@da51:~ > tar -vf /tmp/backup.tar -r new_file
```

- **-t** wyświetla zawartość archiwum:

```
geeko@da51:~ > tar -tvf /tmp/backup.tar
```

- **-u** dodaje do archiwum tylko pliki nowsze od wersji tychże przechowywanych w archiwum (aktualizacja - *update*):

```
geeko@da51:~ > tar -uvf /tmp/backup.tar -C /home/geeko
```

- **--exclude** wyłącza podane pliki z archiwizacji:

```
geeko@da51:~ > tar -cvf /tmp/backup.tar /home/geeko --exclude="*.iso"
```

Wyjaśnienie. Zostanie zarchiwizowana zawartość katalogu domowego użytkownika geeko z wyłączeniem wszystkich plików posiadających rozszerzenie .iso.

- **-z** kompresuje (pakuje) lub dekompresuje (rozpakowuje) archiwum **tar** programem **gzip**.

```
geeko@da51:~ > tar -czvf /tmp/backup.tar.gz /home/geeko
```

Ćwiczenie. Tworzenie archiwum tar

Utwórz archiwum tar **/tmp/bin.tar** z całej zawartości katalogu **/bin**. Następnie przejdź do katalogu **/tmp/** i rozpakuj tam tylko jeden plik z archiwum – program o nazwie **chown**. Porównaj właściciela i datę utworzenia pliku rozpakowanego i oryginalnego.

- By spakować archiwum **tar** można użyć opcji **–j** lub **-z**.

Ponadto, w Linuksie, mamy kilka specjalnych narzędzi do pakowania (kompresji) oraz rozpakowywania plików.

gzip

Można użyć programu (polecenia) **gzip**. Spakowane pliki mają rozszerzenie **.gz**. Program ten jest użyteczny tylko dla pakowania pojedynczych plików.

Składnia polecenia: **gzip [options]** ***file***

Przykład:

```
1 geeko@da51:~> ls
2 backup.tar bin Desktop Documents public_html
3 geeko@da51:~> gzip backup.tar
4 geeko@da51:~> ls
5 backup.tar.gz bin Desktop Documents public_html
6 geeko@da51:~>
```

By spakować kilka plików lub całe katalogi do jednego pliku – użyj polecenia **tar**.

Użyteczne opcje programu gzip

- **-c** kompresuje plik bez zmiany pliku oryginalnego. Wynik jest wysyłany na standardowe wyjście (zwykle ekran monitora).

- **-d** rozpakowuje (dekompresuje) podany plik.

  Zamiast **gzip -d** można użyć polecenia **gunzip**. To polecenie rozpakowuje plik spakowany poleceniem **gzip** i usuwa rozszerzenie **.gz**.

```
geeko@da51:~> ls
backup.tar.gz  bin  Desktop  Documents  public_html
geeko@da51:~> gunzip backup.tar.gz
geeko@da51:~> ls
backup.tar  bin  Desktop  Documents  public_html
geeko@da51:~>
```

- **-1** do **-9** kontroluje szybkość kompresji.

  -1 kompresuje najszybciej, ale w rezultacie otrzymujemy stosunkowo duże pliki, natomiast **-9** kosztem większego czasu przetwarzania tworzy mniejsze pliki. Ustawienie domyślne opcji to **-6**.

- -r pakuje i rozpakowuje pliki we wszystkich podkatalogach.

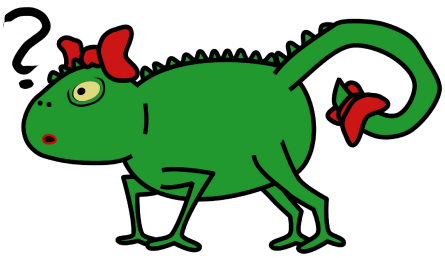
Ćwiczenie. Kompresja pliku – część I

Spakuj archiwum /tmp/bin.tar programem gzip w trzech różnych szybkościach kompresji. Zapisz poniżej rozmiar otrzymanego w każdym przypadku pliku.

-1 _____

-6 _____

-9 _____



bzip2

bzip2 również kompresuje pliki. Tworzy pliki zwykle około 20-30% mniejsze niż przy kompresji programem **gzip,** ale wymaga więcej czasu.

Pliki spakowane programem **bzip2** mają rozszerzenie **.bz2**.

Składnia polecenia: **bzip2 [options]** ***file***

Przykład:

```
geeko@da51:~> ls
backup.tar  bin  Desktop  Documents  public_html
```

```
geeko@da51:~> bzip2 backup.tar
geeko@da51:~> ls
backup.tar.bz2  bin  Desktop  Documents  public_html
geeko@da51:~>
```

Użyteczne opcje programu bzip2

- **-c** kompresuje plik bez zmiany pliku oryginalnego. Wynik jest wysyłany na standardowe wyjście (zwykle ekran monitora).

- **-d** rozpakowuje (dekompresuje) podany plik.

  Zamiast **bzip2 -d** można użyć polecenia **bunzip2**. To polecenie rozpakowuje plik spakowany poleceniem **bzip2** i usuwa rozszerzenie **.bz2**.

- **-1** do **-9** kontroluje szybkość kompresji.

  -1 kompresuje najszybciej, ale w rezultacie otrzymujemy stosunkowo duże pliki, natomiast **-9** kosztem większego czasu przetwarzania tworzy mniejsze pliki. Ustawienie domyślne opcji to **-6**.

Ćwiczenie. Kompresja pliku – część II

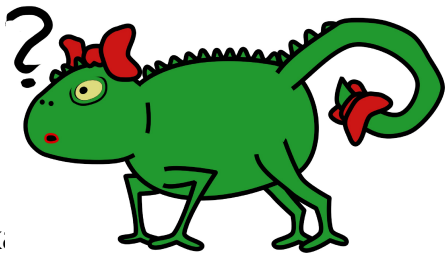
Spakuj archiwum /tmp/bin.tar programem bzip2 w trzech różnych szybkościach kompresji. Zapisz poniżej rozmiar otrzymanego w każdym przypadku pliku.

-1 _____

-6 _____

-9 _____

Skopiuj archiwum tar do swojego domowego katalogu i rozpakuj do nowego katalogu Nauk



2.1.10Zarządzanie uprawnieniami do plików i statusem właściciela

Można użyć polecenia **ls -l** celem wyświetlenia pełnej zawartości bieżącego katalogu z przydzielonymi do każdego pliku i podkatalogu uprawnieniami.

Przykład:

```
geeko@da51:~> ls -l MyFile
-rw-r--r-- 1 geeko users 0 2006-02-07 19:10 MyFile
geeko@da51:~>
```

chmod

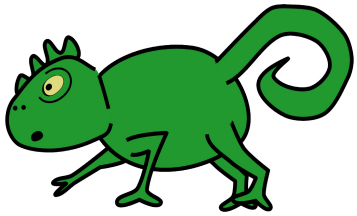
Można użyć polecenia **chmod** do zmiany uprawnień do pliku. Programu **chmod** może używać zarówno użytkownik **root**, jak i właściciel pliku.

Niektóre opcje zmiany uprawnień do pliku poleceniem **chmod**:

- **u** – zmiana uprawnień właściciela,
- **g** – zmiana uprawnień grupy,
- **o** – zmiana uprawnień pozostałych użytkowników,
- **a** – zmiana uprawnień wszystkich użytkowników,
- **+** dodawanie uprawnień,
- **-** odejmowanie (usuwanie) uprawnień.

Przykłady użycia polecenia **chmod**:

przykład	***wynik***
chmod u+x	Właściciel pliku może go uruchomić.
chmod g=rw	Wszyscy członkowie grupy mogą czytać i pisać w pliku.
chmod u=rwx	Właściciel ma pełne uprawnienia.
chmod u=rwx , g=rw , o=r	Właściciel ma pełne prawa, grupa – czytania i zapisu, inni – tylko czytania.
chmod +x	Wszyscy użytkownicy (właściciel, grupa, inni) mogą uruchamiać plik.
chmod a+x	Wszyscy użytkownicy (właściciel, grupa, inni) mogą uruchamiać plik.



W poniższym przykładzie, użytkownik geeko zezwala innym członkom grupy **users** na pisanie do pliku **hello.txt**:

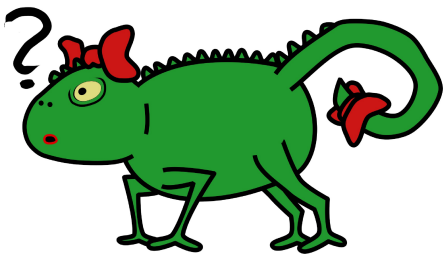
```
geeko@da51:~ > ls -l hello.txt
```

```
-rw-r--r-- 1 geeko users 0 2006-04-06 12:40 hello.txt
geeko@da51:~ > chmod g+w hello.txt
geeko@da51:~ > ls -l hello.txt
-rw-rw-r-- 1 geeko users 0 2006-04-06 12:40 hello.txt
```

Opcja **-R** (*recursive*) przy podaniu nazwy katalogu pozwala na zmianę uprawnienia do wszystkich plików i podkatalogów określonego w poleceniu katalogu.

Ćwiczenie. Zarządzanie uprawnieniami do pliku – część I

Odbierz uprawnienie x (wykonywania) grupie oraz innym do katalogu Nauka i jego całej zawartości.



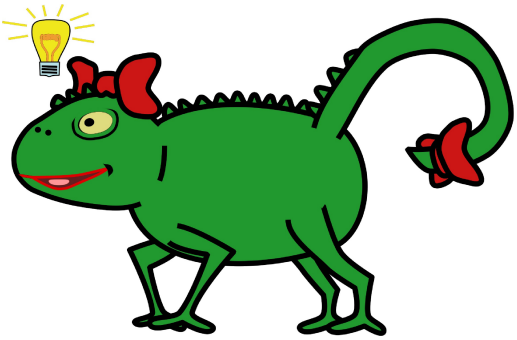
Zamiast używania liter (**rwx**), można używać odpowiednich wartości liczbowych.

Każdy plik oraz katalog w linuksowym systemie ma uprawnienia określone w postaci numerycznej – liczby trzycyfrowej.

Pierwsza cyfra reprezentuje uprawnienia przydzielone właścicielowi pliku czy katalogu, druga reprezentuje uprawnienia grupy skojarzonej z plikiem lub katalogiem, a trzecia odpowiada uprawnieniom innych użytkowników.

Każda cyfra odpowiada liczbie będącej sumą uprawnień wyrażonych odpowiednio:

- czytanie (*read*): 4
- pisanie (*write*): 2
- wykonywanie (*execute*): 1

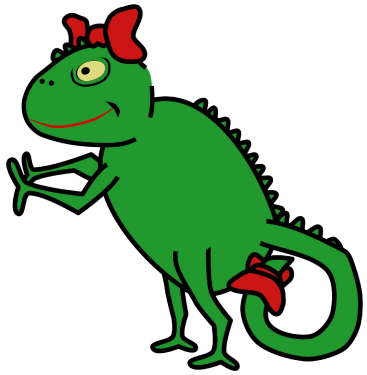


Przykłady:

Plik o nazwie MyFile.txt ma uprawnienia 754.

To znaczy że właściciel pliku może czytać, pisać i wykonywać plik (4+2+1=7), skojarzona z plikiem grupa może plik czytać i wykonywać (4+1=5), a inni mogą tylko czytać (4).

Właściciel	***Grupa***	***Inni***
rwx	r-x	r--
421(4+2+1=7)	4 -1 (4+1=5)	4-- (4)



przykład	***Wynik***
chmod 754 hello.txt	Wszystkie prawa dla właściciela, czytanie i wykonywanie dla grupy, czytanie dla innych (rwx r-xr--).
chmod 777 hello.txt	Wszystkie prawa dla wszystkich (rwx rwx rwx).

Poleceniem **chmod** można też ustawić specjalne uprawnienia.

Przedstawiono to w poniższej tabeli:

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.	

Można ustawić „*sticky bit*” poleceniem **chmod**, używając liter (**chmod o+t /tmp**) lub cyfr (**chmod 1777 /tmp**). „*sticky bit*” zostanie wyświetlony w grupie uprawnień „innych użytkowników”:

```
geeko@da51:~ > ls -ld /tmp
drwxrwxrwt 15 root root 608 2006-04-06 12:45 /tmp
geeko@da51:~ >
```

Poniżej przedstawiono przykład SUID:

```
geeko@da51:~ > ls -l /usr/bin/passwd
-rwsr-xr-x 1 root shadow 79765 2006-03-24 12:19 /usr/bin/passwd
geeko@da51:~ >
```

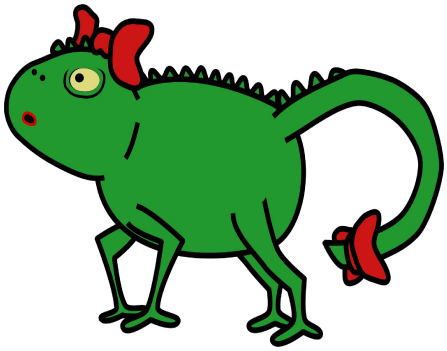
Każdy użytkownik ma prawo zmienić swoje hasło, ale trzeba mieć uprawnienia użytkownika root by pisać do pliku **/etc/shadow**.

Przykład SGID:

```
geeko@da51:~ > ls -l /usr/bin/wall
-rwxr-sr-x 1 root tty 10192 2006-03-22 05:24 /usr/bin/wall
geeko@da51:~ >
```

Można użyć polecenia **wall** do wysłania komunikatu na wszystkie wirtualne terminale. Wykonywane to być może tylko na prawach grupy **tty**.

Uwaga. Te rodzaje uprawnień należy stosować bardzo ostrożnie. Wyjaśniono je tutaj bardzo pobieżnie.



Ćwiczenie. Zarządzanie uprawnieniami do pliku – część II

Co znaczą praktycznie następujące wartości uprawnień?

Uprawnienie 777

właściciel _____

grupa_____

inni _____

Uprawnienie 755

właściciel _____

grupa_____

inni _____

Uprawnienie 600

właściciel _____

grupa_____

inni _____

Uprawnienie 644

właściciel _____

grupa_____

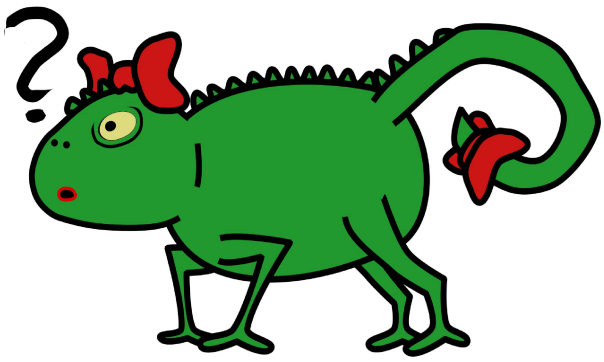
inni _____

Uprawnienie 2755

właściciel _____

grupa_____

inni _____



Użytkownik **root** może używać polecenia **chown** do zmiany przypisań użytkownika i grupy do pliku.

Składnia polecenia: **chown *new_user.new_group file***

Przykład:

```
da51:~ # chown geeko.users MyFile
da51:~ #
```

By zmienić właściciela nie zmieniając grupy, należy użyć składni:

Składnia polecenia: **chown *new_user file***

Przykład:

```
da51:~ # chown geeko MyFile
da51:~ #
```

By zmienić grupę nie zmieniając użytkownika, należy użyć składni:

Składnia polecenia: **chown *.new_group file***

Przykład:

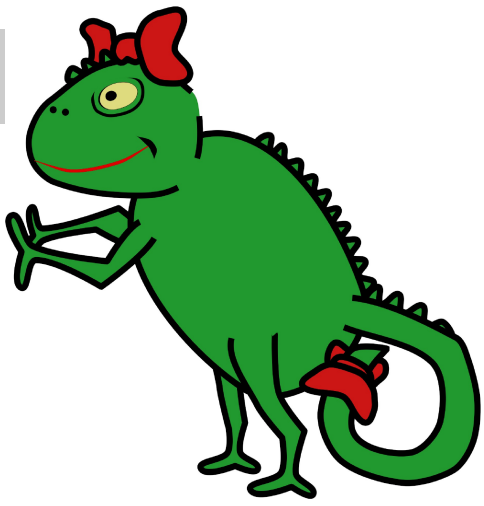
```
da51:~ # chown .users MyFile
da51:~ #
```

Użytkownik root może zmienić przypisanie grupy poleceniem chgrp.

Składnia polecenia: **chgrp *new_group file***

Przykład:

```
da51:~ # chgrp users MyFile
da51:~ #
```



Zwykły użytkownik może użyć polecenia **chown** do przydzielenia swojego pliku nowej grupie.

Składnia polecenia: **chown *.new_group file***

Przykład:

```
geeko@da51:~> chown .dialout MyFile
geeko@da51:~>
```

Może też użyć polecenia **chgrp**.

Składnia polecenia: **chgrp *new_group file***

Przykład:

```
geeko@da51:~> chgrp dialout MyFile
geeko@da51:~>
```

Możesz zmienić przydzielenie pliku do grupy tylko w przypadku, gdy jesteś członkiem tej grupy.

W poniższym przykładzie użytkownik **root** użył polecenia **chown** to zmiany właściciela pliku **hello.txt** z **geeko** na **tux**:

```
da51:~ # ls -l hello.txt
-rw-r--r-- 1 geeko users 0 2006-04-06 12:43 hello.txt
da51:~ # chown tux.users hello.txt
da51:~ # ls -l hello.txt
-rw-r--r-- 1 tux users 0 2006-04-06 12:43 hello.txt
da51:~ #
```

W poniższym przykładzie **root** użył polecenia **chown** do ograniczenia dostępu do pliku **list.txt** do członków grupy **training**:

```
da51:~ # ls -l list.txt
-rw-r----- 1 geeko users 0 2006-04-06 12:43 list.txt
da51:~ # chown .training list.txt
```

```
da51:~ # ls -l list.txt
-rw-r----- 1 geeko training 0 2006-04-06 12:43 list.txt
da51:~ #
```

Zarówno root, jak i właściciel pliku mają nadal dostęp do pliku. Mimo że grupa została zmieniona, uprawnienia właścicieli pozostają te same.

- Najważniejszą opcją dla poleceń **chown** oraz **chgrp** jest opcja **-R**, pozwalająca na zmianę właściciela pliku oraz przypisanej grupy we wszystkich plikach danego katalogu.

Ćwiczenie. Zarządzanie przypisaniami użytkownika i grupy

Zmień grupę przypisaną do katalogu Nauka i jego zawartości na grupę *dialout*.