WIELODOSTĘPNE SYSTEMY OPERACYJNE – LABORATORIUM

Opracował zespół pod kierownictwem dr inż. Lecha Krusia

2. Podstawy użytkowania systemu

2.1. Konto użytkownika

Unix jest systemem wielodostępnym, w którym jednocześnie może pracować wielu użytkowników.

- Każdy użytkownik na swoje **konto** w systemie (ang. ***account***).
- Konto zakłada administrator systemu.
- Z kontem związane są uprawnienia do korzystania z zasobów systemu.
- Dostęp do konta jest kontrolowany przez **hasło** (ang. ***password***).

Dwa podstawowe typy kont:

- konto użytkownika (ang. *user*)
  – przydzielane przez administratora systemu każdemu, kto chce pracować w systemie;
  – użytkownik ma ograniczony zakres działań, które może wykonywać;
  – jest właścicielem swoich plików i uruchomionych programów;
  – w systemie istnieje dowolnie wiele kont użytkowników tworzonych w miarę napływu zgłoszeń;
- konto uprzywilejowane (ang. *superuser account*, *root account*)
  – konto, którego właściciel ma dostęp do wszystkich działań, plików i uruchomionych programów;
  – używane jest do prowadzenia prac administracyjnych;
  – często określane jest mianem *konto super użytkownika* lub *konto administratora***.**

2.2. Otwieranie i zamykanie sesji

Przed rozpoczęciem pracy przez użytkownika administrator systemu musi mu założyć konto, czyli określić:

- identyfikator użytkownika (ang. *login id, login name, user name, account*)
- hasło (ang. *password*)
- katalog domowy użytkownika (ang. *home directory*)
- shell (ang. *login shell*) lub inny program obsługujący sesję użytkownika

Dodatkowo administrator określa środowisko pracy użytkownika (na przykład sposób sygnalizowania przez Unix gotowości przyjęcia kolejnego polecenia).

Otwieranie sesji (ang. *logging in*)

Pracę można rozpocząć, gdy na ekranie widać zgłoszenie gotowości przyjęcia użytkownika (ang. ***login prompt***):

```
login:
```

Należy wtedy wpisać **identyfikator**

```
login: user1
```

Następnie, gdy pojawi się pytanie o **hasło**, wpisać je (hasło nie pojawi się na ekranie)

```
Password:
```

Po poprawnym podaniu identyfikatora i hasła, system przedstawi się i pojawi się zgłoszenie gotowości shella obsługującego sesję (*shell prompt*):

```
$
```

Można teraz wpisywać polecenia.

Symbol zgłoszenia gotowości shella określany jest podczas defniowania środowiska pracy użytkownika.

Zamykanie sesji (ang. logging out, logging off)

Polecenie zamknięcia sesji związane jest z shellem obsługującym sesję.
Może to być polecenie:

```
/home/user1 $ exit            (shell Korn'a)
login:
```

lub

```
/home/user1 % logout    (shell C)
login:
```

Przykładowa sesja

```
Generic-Sys (generic) [HP Release B.10.1]
login: janusz
password:
/home/janusz $ clear
/home/janusz $ date
Wed Feb 21 13:01:51 MET 1996
/home/janusz $ who
root        console     Feb 21      11:38
sikorski    tty0p1      Feb 21      11:41
/home/janusz $ exit
logout

Generic-Sys (generic) [HP Release B.10.1]
login:
```

2.3. Zgłoszenie gotowości shella

- Użytkownik pracuje pod nadzorem programu interpretatora poleceń, nazywanego shellem. Program ten jest uruchamiany automatycznie w momencie rozpoczęcia sesji i pozostaje uruchomiony do momentu zakończenia sesji.
- Każdy shell posiada swój standard zgłoszenia
  $ shell Bourn'a, shell Korn'a
  % shell C
- Zgłoszenie może być inne dla zwykłego użytkownika i inne dla administratora
  $ użytkownik (lub inne, właściwe dla shella)
  # administrator (w każdym shellu)
- Użytkownik może dostosować zgłoszenie shella do swoich potrzeb i nadać mu inną postać
  `$` standard w shellu Korn'a
  `/home/user1 $` wybrane przez użytkownika, w zgłoszeniu widać katalog bieżący

2.4. Klawiatura

- UNIX jest dostosowany do pracy z różnymi terminalami - sposób działania terminala (i tym samym klawiatury) określany jest za pomocą polecenia:
  `stty`

Sprawdzenie obowiązującego przypisania klawiszy

`stty`	podstawowe ustawienia
`stty -a`	wszystkie ustawienia w HP-UX
`stty all`	wszystkie ustawienia w Sun OS

Przedefiniowywanie klawiatury

`stty` *`funkcja kombinacja-klawiszy`*

Przykład:

```
stty  erase  ^H
stty  erase  <Backspace>
```

Najczęściej przyjmowane przyporządkowanie klawiszy

Klawisz	Nazwa	Działanie
Backspace	***erase***	usuń znak na lewo od kursora
Ctrl-	***quit***	zakończ program, zachowaj obraz pamięci programu w pliku `core`
Ctrl-d	***eof***	zakończ sesję; zakończ wprowadzanie danych
Ctrl-s	***stop***	zatrzymaj wyświetlanie
Ctrl-q	***stop***	wznów wyświetlanie

2.5. Postać wiersza polecenia w UNIXie

- UNIX rozróżnia wielkie i małe litery w poleceniu
- Ogólna postać polecenia:

  polecenie [*opcja ...*] [*wyrażenie*] [*nazwa pliku ...*]

  - polecenie i jego argumenty rozdzielane są odstępami (spacja lub tabulacja)
  - pierwsze słowo w wierszu jest nazwą polecenia
  - opcja - modyfikuje polecenie, zpiasywana jest za pomocą litery poprzedzonej znakiem -
  - wyrażenie - opisuje dodatkowe dane wymagane przez polecenie
  - nazwa pliku - gdy pomijana najczęściej oznacza ekran lub klawiaturę

Przykład poleceń:

```
$ date        # wyświetla datę w postaci dzień i godzina (w DOSie potrzebne są dwa polecenia date i time)
$ ls -l /home       # wyświetla zawartość katalogu /home      (odpowiednik w DOSie - dir \users )

$ DATE
ksh: DATE: not found      (błędnie podane polecenie)

$ cp
Usage: cp [-f|i][-p] source_file target_file      (błędnie podane polecenie)
```

Polecenie z wieloma opcjami:

```
ls  -l  -a
ls  -la
```

Kilka poleceń w jednym wierszu:

```
polecenie1; polecenie2; polecenie3; polecenie4
```

Przykład:

```
date; ls -l
```

Polecenie nie mieszczące się w jednym wierszu:

- automatycznie zawijane i wyświetlane w następnym wierszu
- polecenie można pisać w kilku wierszach posługując się symbolem \ :

```
część polecenia \
kontynuacja polecenia
```

Przykład:

```
ls -l \
/users/user1
```

2.6. Zmiana hasła

Polecenie zmiany hasła:

passwd

Przykład:

```
$passwd
Changing password for janusz
Old password:
New password:
Re-enter new password:
$
```

O czym należy pamiętać:

- Minimalna długość hasła zależy od implementacji UNIX'a, zaleca się używanie dłuższych haseł.
- Jedynie pierwszych 8 znaków hasła jest branych pod uwagę.
- Program `passwd` może narzucać pewne specjalne wymagania, służące wzmocnieniu ochrony dostępu do systemu.
- Hasło jest przechowywane w postaci zaszyfrowanej.
- W wypadku zapomnienia hasła trzeba poprosić adminstratora o wpisanie nowego hasła.

Komunikaty wyświetlane podczas zmiany hasła (zależą od wersji Unix'a):

`Sorry`.	Wprowadzono niepoprawne hasło.
`Password is too short`	Za krótkie hasło (w HP-UX - musi mieć co najmniej 6 znaków).
`Please use a longer password`	Za krótkie hasło (Sun OS).
`Password must contain at least two alphabetic characters and at least one numeric or special character.`	Wymagania na hasło w HP-UX.
`Password unchanged`	Zrezygnowano ze zmiany hasła naciskając CTRL+C (Sun OS).

2.7. Środowisko pracy

Identyfikacja sprzętu i systemu operacyjnego

```
uname -a
```

Przykłady:

```
HP-UX unix B.10.01 B 9000/806 87564422 16-user licence
SunOS ibs 4.1.4 2 sun4m
```

Identyfikacja katalogu osobistego i shella obsługującego sesję

```
grep nazwa_użytkownika /etc/passwd
finger nazwa_użytkownika
```

Przykład:

```
$ grep janusz /etc/passwd
janusz:VCyulbFzRySlh:4001:101:Janusz Nowak, tel.156:/users/janusz:/bin/ksh
```

2.8. Tradycyjny podręcznik UNIXa

Składa się z 8 części zawierających:

Section 1	**User commands** (Polecenia użytkownika)
Section 2	**System calls** (Funkcje systemowe języka C)
Section 3	**Library functions** (Podprogramy biblioteczne języka C)
Section 4, 5, 7	**File Formats** (Formaty plików), **Miscellaneous Facilities** (Różne) **Device drivers** (Programy obsługi urządzeń),
Section 8 (lub 1m)	**System administration commands** (Polecenia dla administratora systemu)

W obrębie każdej części polecenia są umieszczane alfabetycznie.
W dokumentacji odwołanie do polecenia jest podawane w postaci:

polecenie (n)

gdzie ***polecenie*** jest nazwą polecenia, zaś ***n*** jest numerem części podręcznika. Zatem **date(1)** oznacza odwołanie się do polecenia **date** w części pierwszej podręcznika.

Układ strony w podręczniku

Strona ma układ standardowy. Nie każdy element musi występować w opisie danego polecenia.
W tabeli przedstawiono najczęściej używane elementy opisu.

NAME ***Nazwa***	Nazwa polecenia z krótkim opisem
SYNOPSIS ***Składnia***	Składnia polecenia. Elementy wyróżnione drukiem pogrubionym należy wpisać tak, jak podano w opisie. Opcje lub argumenty w nawiasach [] są opcjonalne. Pominięcie nawiasu przy argumencie oznacza, że jest on obowiązkowy. Symbol ... oznacza, że poprzedni argument może być powtórzony.
DESCRIPTION ***Opis***	Dokładny opis działania polecenia.
EXAMPLES ***Przykłady***	Tylko przy bardziej złożonych poleceniach.
FILES ***Pliki***	Wykaz plików, których używa polecenie.
DIAGNOSTICS ***Diagnostyka***	Opis komunikatów o błędach, mogących się pojawić po wydaniu polecenia.
WARNINGS ***Ostrzeżenia***	Opis warunków, mogących spowodować błędne działanie.
SEE ALSO ***Patrz również***	Odwołanie do innych stron podręcznika lub innego opracowania.
STANDARDS CONFORMANCE **Zgodność ze standardem**	Opis standardów, które spełnia polecenie

Korzystanie z podręcznika on-line

`man [X] polecenie` # pełny opis polecenia, X oznacza część podręcznika

`man -k słowo_kluczowe` # krótki opis poleceń, w których opisie występuje podane słowo kluczowe

Klawisze obsługujące polecenie man

Informacja wyświetlana na ekranie zostaje zatrzymana, gdy wypełni ekran.
Użyteczne klawisze:

- Return wyświetla następny wiersz
- Spacja wyświetla następny ekran
- q lub Q kończy wyświetlanie

Przykłady:

`man date`	wyświetla strony z opisem date (pierwsze znalezione)
man passwd	wyświetla pierwsze znalezione strony
`man 1 passwd`	wyświetla opis z części 1
`man 4 passwd`	wyświetla opis z części 4
`man -k passwd`	wyświetl wykaz wszystkich poleceń, w których w obszarze NAME opisu występuje słowo passwd

3. Wybrane podstawowe polecenia

3.1. Identyfikacja użytkownika

Konta użytkowników

- Każdy użytkownik systemu ma swój identyfikator (***login id***), za pomocą którego wchodzi do systemu.
- Pełna informacja o użytkowniku obejmuje:

```
ania:gdg4352ochxzq:124:11:Ania Kowalska, tel. 3322:/home/ania:/bin/ksh
```

identyfikator użytkownika login id — *hasło* — *numer użytkownika UID* — *numer grupy GID* — *informacje dodatkowe* — *katalog domowy home directory* — *shell sesji*

Identyfikator użytkownika (ang. *login ID*) - używany przy rozpoczynaniu pracy
Hasło (ang. *password*) - jest zakodowane
Numer użytkownika (ang. *user identifier*) - liczbowy identyfikator użytkownika w systemie
Numer grupy użytkownika (ang. *group identifier*) - użytkownicy systemu podzieleni są na ***grupy***; grupa może mieć podobne prawa dostępu do plików
Informacja o osobie - dowolny tekst komentarza
Katalog domowy (ang. *home directory*) - w tym katalogu użytkownik znajdzie się po rozpoczęciu sesji (pełna ***ścieżka***)
Shell - ten shell zostanie wywołany po rozpoczęciu sesji

3.1. Sprawdzanie informacji o użytkowniku

```
$ id
uid=5214(agata) gid=11(users)

$ finger agata
Login name: agata             In real life: Agata Kowalska
Directory: /home/agata        Shell: /bin/ksh
Last login: Thu Mar 11 13:42 on ttyp2.

$ who am i
agata       ttyp3 Mar 13 18:34
```

3.2. Kto pracuje w systemie -polecenie who

Składnia:

```
who [am i]
```
Informacja o użytkownikach prowadzących sesję

Przykład:

```
$ who am i
agata       ttyp3       Mar 13 18:34

$ who
agata       ttyp3       Mar 13 18:34
jacek       ttyp2       Mar 13 17:39
stach       ttyp3       Mar 13 18:34
```

3.3. Przesyłanie komunikatu do innego użytkownika - polecenie write

Składnia:

`write` *`nazwa_użytkownika`* [*`nazwa_terminala`*]

Przykład:

```
$ write ania <----------------  Nawiązanie połączenia
Co nowego? <RETURN>             Gdy naciśniesz klawisz RETURN komunikat pojawi się
To caly komunikat <RETURN>      na terminalu rozmówcy.
<CTRL-d> <----------------      Zamknięcie połączenia
$
```

- Jeżeli użytkownik pracuje na dwóch terminalach, należy podać, na który ma być przesłana wiadomość; na przykład: `write ania ttyp4`
- Napisany tekst zostanie przesłany do rozmówcy po naciśnięciu klawisza Return.
- Trzeba samemu ustalić protokół konwersacji, aby nie przerywać sobie nawzajem.
- Każda z osób, która zaczęła konwersację musi ją zakończyć naciskając CTRL+D.
- Użytkownik może zabronić wypisywania na swoim terminalu komunikatów.
- Nie można zabronić otrzymywania komunikatów od administratora systemu.

Możliwe komunikaty zwrotne:

- `Permission denied`. Użytkownik do którego chcesz wysłać komunikat, zabronił wypisywania komunikatów na terminalu.

- *`Nazwa_użytkownika`* `is not logged on.` Za pomocą write można przesyłać komunikat tylko do aktulanie pracującychużytkowników.

Przykład

terminal Janusza	**terminal Ani**
`$ write ania<RETURN>`	`Message agata(ttyp3)`
`Czy mialas czas zapoznac sie z moim raportem?<RETURN>`	
`OK<RETURN>`	`Czy mialas czas zapoznac sie z moim raportem?`
	`OK`
	`$ write agata<RETURN>`
`Message from ania (ttyp4)`	
	`Nie, nie mialam czasu. Zrobie to dzisiaj.`
`Nie, nie mialam czasu. Zrobie to dzisiaj. <RETURN>`	`OK <RETURN>`
`OK`	

	`OK<RETURN>`
	`OK<RETURN>`
	`CRL-d`

`OK`
`OK`
`EOF`
	`$`

`CTRL-d` `$`	`EOF`

3.4. Zabranianie wypisywania komunikatów na terminalu - polecenie mesg

Użytkownik może zabronić wypisywania na swoim terminalu komunikatów.

Składnia:

`mesg [y|n]` Zezwala/zabrania wypisywania komunikatów na ekranie

Przykład:

```
$ mesg        Sprawdzenie czy przyjmujesz komunikaty.
y
$ mesg n      Zabronienie przyjmowania komunikatów.
$ mesg        Sprawdzenie czy przyjmujesz komunikaty.
n
$mesg y       Zezwolenie na przyjmowanie komunikatów.
```

3.5. Komunikacja administrator - użytkownik: polecenie wall

Składnia:

`wall [nazwa_pliku]`

Przesłanie komunikatu do wszystkich użytkowników, którzy aktualnie pracują w systemie
Komunikat może być

- wpisany z klawiatury (pominięta nazwa pliku, tekst kończony jest przez CTRL+D)
- przesłany z pliku

Gdy komunikat jest przesłany przez:

- użytkownika - pojawi się tylko na tych termimalach, na których nie jest zabronione pisanie.
- administratora - komunikat pojawi się na wszystkich terminalach.

3.6. Polecenia różne

Polecenie `echo`

Składnia:

`echo [arg...]` Wyświetla argumenty na ekranie

Przykład:

```
$ echo hej hej
```

Polecenie `date`

Składnia:

`date` Wyświetla datę i czas

Przykład:

```
$ date
Fri Jan 1 11:12 MET 1996
```

Polecenie `clear`

Składnia:

`clear` Czyści ekran

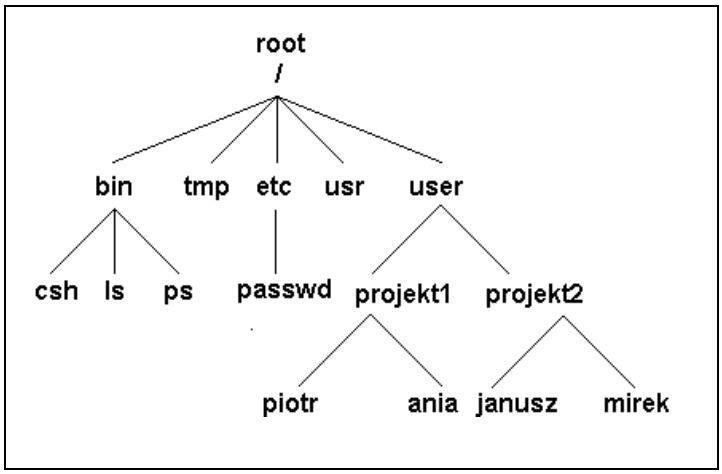
4. System plików

Przechowywana informacja umieszczana jest w **pliku**:
- **tekstowym** ciąg znaków oddzielanych znakiem nowej linii (LF, newline)
- **binarnym** sekwencja słów binarnych

Typy plików w systemie UNIX:
- **zwykłe pliki dyskowe** (***ordinary disk files***): służą do przechowywania danych użytkownika, programów aplikacyjnych lub systemowych; dla Unixa plik taki stanowi sekwencję bajtów i jest często nazywany s**trumieniem** (***stream***)
- **katalogi** (***directories***): służą do zorganizowania danych w hierarchiczne grupy plików; są to zwykłe pliki dyskowe ze specjalnymi uprawnieniami
- **pliki specjalne** (***special files***): służą do zarządzania urządzeniami we/wy
- **pliki FIFO** (first-in-first-out) : pliki dla potoków, mechanizmu przesyłania danych z jednego programu do innego

4.1. Co widzi użytkownik?

Użytkownik widzi- jednorodny system plików



- Pliki tworzą jednorodną strukturę nazywaną **systemem plików** (***file system***), fizyczne urządzenia, na których są przechowywane pliki nie są widoczne
- System plików ma strukturę hierarchiczną.
- Przedstawiany jest w postaci **drzewa katalogów** (***tree***).
- Składa się z **katalogów** i **plików**
- Podstawą podzielenia plików na katalogi jest ich funkcja w systemie.
- Najwyżej w hierarchii znajduje się katalog nazwany **katalogiem głównym** (ang. ***root***) oznaczony symbolem /
- Katalog, który zawiera inny katalog nazywany jest **katalogiem macierzystym** lub **katalogiem nadrzędnym** (ang. ***parent directory***)
- Katalog zawarty w innym katalogu nazywany jest **podkatalogiem** lub **katalogiem podrzędnym** (ang. ***child directory***, ***subdirectory***).
- Lista katalogów, które znajdują się na ścieżce prowadzącej do wybranego pliku nosi nazwę **ścieżki katalogowej** (ang. ***pathname***)

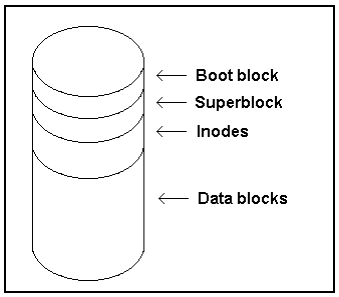
Uwaga: w języku angielskim występują dwie nie związane ze sobą nazwy

- root katalog
- root super użytkownik

4.2. Jak tworzony jest system plików

- Drzewo plików UNIXa zbudowane jest z części nazywanych **systemami plików** (***filesystems***).
- Każdy taki system plików składa się z katalogu, podkatalogów i plików.
- System plików przyłączany jest do drzewa plików za pomocą polecenia mount.
- W momencie inicjalizacji znany jest tylko **root file system**. Pozostałe systemy plików są przyłączane (ang. **mounted**) pod:
  - / (katalogiem głównym)
  - bieżącym katalogiem
  - dowolnym katalogiem znanym systemowi

4.3. Co widzi system - dysk logiczny



Dysk logiczny składa się z czterech części:
- blok 0 (***boot block***) - program ładujący i inicjujący
- blok nadrzędny (***super block***) - bieżące dane o systemie plików
- i-węzły (***inodes***) - metryki identyfikujące poszczególne pliki
- bloki przeznaczone na dane (***data blocks***)

i-te węzły (ang. ***inodes - information nodes*****) - metryki plików**
- i-węzeł opisuje zestaw bloków zajętych przez plik.
- Każdy i-węzeł ma numer za pomocą którego jest identyfikowany.
- W i-węźle umieszczonych jest kilkadziesiąt informacji, z których większość potrzebna jest tylko jądru.

Pozycja w katalogu
- Katalog przyporządkowuje i-węzłowi nazwę pliku. Każda pozycja katalogu to para (nazwa, numer i-węzła).
- Każdy katalog ma dwie pozycje specjalne:
  . (kropka) - numer i-węzła tego katalogu
  .. (dwie kropki) - numer i-węzła katalogu macierzystego

4.4. Standardowe katalogi

Wszystkie wersje UNIXa mają określone standardowe katalogi, w których umieszczone są polecenia i pliki pomocnicze wykorzystywane przez system.

`/`	katalog główny (ang. ***root directory of the file system***)
`bin`	podstawowe polecenia UNIXa
`dev`	pliki urządzeń (każde urządzenie ma swój plik)
`etc`	pliki i podkatalogi niezbędne w administrowaniu systemem
`lib`	pliki bibioteczne (np. dla kompilatorów)
`tmp`	pliki tymczasowe, tworzone przez niektóre polecenia i aplikacje
`usr`	polecenia systemowe i programy ogólnego przeznaczenia
`home`	katalogi do których przyłączane są systemy plików użytkowników

Informacje o systemie plików w danej wersji systemu Unix:

```
man hier
```

4.5. Katalog osobisty (domowy) użytkownika i katalog bieżący

Katalog osobisty (domowy) użytkownika (*home directory, login directory*)

- Jest to katalog zawierający katalogi i pliki uzytkownika
- W nim umieszczany jest użytkownik po rozpoczęciu sesji.

Informacja o katalogu osobistym:
na przykład polecenie finger *nazwa_użytkownika*

Katalog bieżący - *current (present) working directory*

Katalog, z którym aktualnie pracuje użytkownik.
Informacja o bieżącym katalogu -
polecenie `pwd`
Skrócona nazwa katalogu bieżącego:
. (kropka)

Składnia polecenia pwd**:**

`pwd` informacja o pełnej scieżce bieżącego, roboczego katalogu (*present working directory lub print working directory*)

Przykład:
```
$ pwd
/home/agata
```
Polecenie wyświetla pełną **ścieżkę katalogu**

4.6. Nazwy ścieżkowe (*ang. path names*)

Ścieżka służy do wyznaczenia jednoznacznego miejsca w systemie plików.
Nazwa **bezwzględna** ścieżki (***absolute pathname*** lub ***full pathname***):
- rozpoczyna się od znaku /
- wyznacza położenie względem katalogu głównego

Nazwa **względna** ścieżki (***relative pathname***)
- nie rozpoczyna się od znaku /
- wyznacza położenie względem bieżącego roboczego katalogu

Nazwa **prosta** pliku/ścieżki (***simple file/pathname***)
- nazwa pliku lub katalogu znajdującego się w bieżącym katalogu

W nazwach ścieżek mogą być używane skróty:

.	katalog bieżący
..	katalog macierzysty
~	katalog domowy

Przykłady: /home/user1/drzewo
drzewo/katalog1
../kurs5

4.7. Zmiana bieżącego katalogu - polecenie cd

Składnia**:**
`cd [`*`nazwa_ścieżkowa`*`]` Zmienia bieżący roboczy katalog na ***nazwa_ścieżkowa***.

Przykład**:**
```
$ pwd                    # gdzie jestem
/home/agata
$ cd /tmp                # zmiana na /tmp
$ pwd
/tmp
$ cd                     # powrót do katalogu domowego
$ pwd
/home/agata
$ cd ~kurs2              # zmiana na katalog domowy użytownika kurs2
$ pwd
/home/kurs2
```

4.8. Zawartość katalogu - polecenie ls

Składnia**:**

```
ls [-lFaRd] [nazwa]
```

Jeżeli nazwa jest nazwą katalogu, wyświetl jego zawartość.
Jeżeli nazwa jest nazwą pliku, podaj związane z nim informacje.
Jezeli nazwa jest pominięta, podaj informacje o bieżącym katalogu.

Wybrane opcje

`-l`	pełna informacja (***long format***)
`-F`	zaznacz za pomocą symbolu:
`/`	każdą nazwę katalogu,
`*`	plik z programem binarnym
`@`	dowiązania symboliczne
`-a`	wyświetl wszystkie nazwy, łącznie z nazwami rozpoczynającymi się od kropki (***dot files***) - inaczej nie są one wyświetlane
`-R`	uwzględnij pliki danego katalogu i jego wszystkich podkatalogów
`-d`	wyświetl informację o katalogu, bez jego zawartości

Przykłady:

```
$ ls -F
dmpl* list  raport/     roman

$ ls -a
.kshrc      .profile    dmpl  list  raport/     roman

$ ls -l raport
-rw-rw-rw-     1   janusz              27   Feb 5    10:25     czesc1
-rw-rw-rw-     1   janusz              37   Feb 5    10:25     czesc2

$ ls -R
dmpl        list  raport      roman
./raport:
czesc1      czesc2

$ ls -ld  raport
drw-rw-rw-     1    janusz              512   Feb 5    10:20     raport
```

4.9. Tworzenie i usuwanie katalogów - mkdir i rmdir

Składnia:

```
mkdir [nazwa_ścieżkowa]        Utwórz katalog  (make directory)
rmdir [nazwa_ścieżkowa]        Usuń katalog (remove directory)
```

Polecenie `rmdir` usuwa jedynie pusty katalog.

Przykłady:

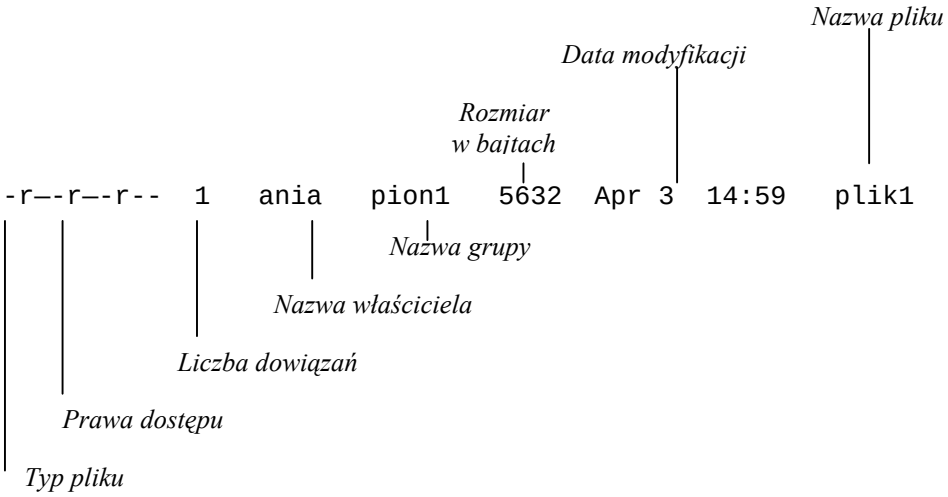
```
$ mkdir nowy
$ mkdir nowy/podkatalog
$ rmdir nowy
    rmdir: nowy not empty
$ rmdir nowy/podkatalog
$ rmdir nowy
```

5. Praca z plikami

5.1. Plik i jego atrybuty

Plik posiada następujące atrybuty:
- typ (***type***) - określa czy jest to plik (zwykły, specjlany) czy katalog
- prawa dostępu - określają kto może czytać, pisać lub wykonywać (***permission, mode***)
- liczbę dołączonych nazw (***number of links***)
- informację o właścicielu i grupie (***user login name***, ***group name***)
- rozmiar pliku (***size)*** - w bajtach
- datę ostatniej modyfikacji (***date & time of last modification***)
- nazwę (***filename***)

Przykład pliku i jego atrybutów



Jest to pełna informacja o pliku. Uzyskasz ją za pomocą polecenia:

```
ls -l
```
lub
```
ls –lg
```
(SUN)

gdzie:
`-l` pełna informacja o pliku (ang. *long*)
`-g` umieszczenie w pełnym opisie nazwy grupy (dotyczy SUNa)

Typy pliku:
- Pierwszy znak opisu określa typ pliku:

d	katalog (***directory***)
-	zwykły plik (***regular file***)
l	plik dowiązany symbolicznie (***symbolically linked file***)
c	plik urządzenia znakowego (***character device file***)
b	plik urządzenia blokowego (***block device file***)
p	plik typu FIFO (***named pipe***)

Nazwa pliku
- Długość nazwy pliku - określana jest w implementacji UNIXa. Przykładowo:
  HP-UX – krótka do 14 znaków, długa do 255 znaków
  AT&T - do 14 znaków
  Berkeley, SUN OS - do 255 znaków
- **Kropka** jako pierwszy znak nazwy oznacza tak zwany **plik ukryty** (***hidden file***). Nie jest wykazywany w wykazie plików uzyskiwanym za pomocą prostego polecenia `ls`.
- Nie ma podziału nazwy na główną i rozszerzenie. Aplikacje mogą używać rozszerzeń, pozornie więc nazwa może wyglądać tak, jak gdyby składała się z części.
- Można używać w nazwie dowolnego znaku ASCII. Powinno się unikać znaków mających specjalne znaczenie dla shella. Przykładowo

```
.    *    &    $
```

5.2. Informacja o pliku - polecenie ls jeszcze raz

Składnia:

```
ls [opcje] nazwa_pliku
```

Wybrane opcje:

`-l`	pełna informacja (**long format**)
`-g`	w uzupełnieniu do -l wyświetla grupę
`-F`	zaznacz za pomocą symbolu:
	`/` każdą nazwę katalogu
	`*` plik z programem wykonywalnym
	`@` łączniki symboliczne
`-a`	wyświetl wszystkie nazwy, również rozpoczynające się od `.`
`-d`	wyświetl tylko nazwy katalogów
`-R`	uzwględnij pliki podkatalogów
`-s`	wyświetl rozmiar plików w Kb
`-t`	wyświetl nazwy plików posortowane wg czasów modyfikacji (na początku najnowszy)

5.3. Informacja o pliku - polecenie file

Składnia**:**

```
file [opcje] nazwa_pliku
```

Polecenie bada zawartość pliku i próbuje określić typ danych w nim zawartych.

Przykład:

```
$ file  *
dane        ascii text
rysunek     PostScript document
raport92    directory
raport.c    c program text
dzisiaj     c-shell script
plik1       empty
```

5.4. Oglądanie zawartości pliku - polecenie cat

Składnia:

`cat [opcje] [plik ...]` wyświetla pliki (concatenate and print), również łączy pliki

Opcje:

`-v`	wyświetl znaki ukryte (bez TAB i NEWLINE)
`-t`	wyświetl znaki tabulacji w postaci ^I
`-e`	wyświetl znak końca wiersza w postaci $
`-n`	numeruj wiersze
`-b`	numeruj wiersze, pomiń puste

Przykłady:

```
$ cat plik1                     wyświetli plik
$ cat plik1 plik2 > plik3       połączy dwa pliki i zapisze do trzeciego
$ cat > plik1                   utworzy nowy plik o nazwie plik1 i zapisze w nim
zawartość pliku wiersz1         tekst wprowadzony z klawiatury
wiersz2
CTRL-d

$ cat plik1 >> plik2            dołączy plik1 na koniec pliku plik2
```

5.5. Oglądanie zawartości pliku - polecenie more

Składnia:

`more [opcje] [nazwa_pliku] ....` Wyświetla plik po stronie

Po zapełnieniu ekranu pojawia się napis:

```
---------------------- More ----- (11 %) -------------------------
```

Klawisze obsługi polecenia more

spacja	przewiń do następnej strony
Return	przewiń o jeden wiersz do przodu

Polecenia obsługi more

h	informacja o poleceniach (help)
q	zakończ polecenie
/***ciąg_znaków***	przewiń do tekstu ***ciąg_znaków***
b	przewiń o 1 ekran do tyłu
f	przewiń o 1 ekran do przodu
v	wywołaj edytor vi

i wiele innych patrz opis polecenia more

5.6. Inne polecenia oglądania zawartości pliku

Wyświetl:

`head [-N] nazwa_pliku`	pierwsze N wierszy pliku
`tail [-N] nazwa_pliku`	ostatnie N wierszy pliku

Jeżeli nie zostanie podane N, przyjmowane jest 10.

5.7. Tworzenie kopii pliku - cp

Składnia:

`cp [-i] plik1 plik2`	kopuj plik1 do pliku2
`cp -r [-ip] katalog1 katalog2`	kopiuj katalog1 z podkatalogami do katalogu2
`cp [-ir] plik1 .... katalog`	kopiuj wiele plików do katalogu

Opcje:

-i	potwierdź, że chcesz kopiować do istniejącego pliku
-r	kopiuj katalog z podkatalogami

- Polecenie cp ma co najmniej dwa argumenty
- Jeżeli istnieje plik o nazwie docelowej oraz nie podasz opcji -i, nie ma ostrzeżenia, że plik zostanie zastąpiony nową zawartością.

Przykłady:

`cp plik_stary plik_nowy`	kopiuje plik_stary do plik_nowy, jeśli plik_nowy istniał zostaje zamazany
`cp –r kat1 kat2`	kopiowana jest zawartość katalogu kat1, wraz ze wszystkimi podkatalogami do katalogu kat2
`cp –i plik1 plik2 kat3`	kpiuje pliki plik1 i plik2 do katalogu kat3, jeśli takie pliki już istnieją w kat3, to uzytkownik otrzyma ostrzeżenie i może wycofać się z operacji kopiowania

5.8. Przenoszenie i zmiana nazwy pliku - mv

Składnia:

```
mv [-fi]  plik1  plik2
mv [-fi]  katalog1  katalog2
mv [-fi]  plik1 ...  katalog
```

Opcje:

-i	potwierdź, że chcesz zmienić nazwę (przenieść) do istniejącego pliku (katalogu)
-f	pomiń zastrzeżenia opcji -i

- Jeżeli istnieje plik o nazwie docelowej oraz nie podasz opcji -i, nie ma ostrzeżenia, że plik zostanie zastąpiony nową zawartością.

Przykłady:

```
$ ls
raport
$ mv raport raport.96
# ls
raport.96
```

zmiana nazwy pliku raport na raport.96

```
$ ls –F
/kat1
$ mv kat1 kat2
$ ls –F
kat2/
```

zmiana nazwy katalogu kat1 na kat2

```
$ ls –F
p1 p2 p3 k1/ k2/ k3/ k4/
$ ls –F k4
p1
$ mv –i p1 p2 k3 k4
remove k4/p1? n
$ ls
p3 k1 k2 k4
$ ls –F k4
p1 p2 k3/
```

przeniesienie plików p1 i p2 oraz katalogu k3 do katalogu k4,
plik p1 nie zostanie przeniesiony, istniał już w katalogu k4, zostaliśmy o tym ostrzeżeni i zadecydowaliśmy o rezygnacji z przeniesienia

5.9. Usuwanie pliku - rm

Składnia:

```
rm [-irf] nazwa_pliku [nazwa_pliku ...]
```

Opcje:

-f	plik zostanie usunięty bez uwzględniania zastrzeżeń
-r	usunięcie zawartości katalogu i podkatalogów
-i	żądanie potwierdzenia chęci usunięcia pliku

Przykłady:

```
rm f1 f5 f6
```

usuwa pliki f1, f5 i f6 z bieżącego katalogu

```
rm –r gnioty
```

usuwa wszystkie pliki z katalogu gnioty i ze wszystkich jego podkatalogów oraz likwiduje sam katalog gnioty (uwaga! bardzo niebezpieczne polecenie)

```
rm –ir gnioty
```

przed usunięciem każdego pliku i katalogu jestesmy proszeni o potwierdzenie (wersja bezpieczna)

5.10. Nadanie drugiej nazwy plikowi - ln

Jeden plik może być widoczny w wielu katalogach pod tą samą lub inną nazwą. Czynność nadania innej nazwy plikowi nazywa się dowiązaniem.

Można tworzyć dowiązania:
- do plików znajdujących się tylko na tym samym dysku i w tej samej partycji – dowiązanie twarde, fizyczne (ang. *hard link*)
- do plików znajdujących się w dowolnym miejscu systemu plików – dowiązanie miękkie, symboliczne (ang. *soft link*).

Dowiązanie twarde (hard) powoduje utworzenie nazwy wskazującej na to samo miejsce na dysku, czyli i-węzeł).
Dowiązanie symboliczne (soft) powoduje utworzenie pliku pośredneigo, który wskazuje miejsce pliku źródłowego.

Przykład dowiązania twardego:
```
$ ls –i plik  plik_hl
790 plik
790 plik_hl
```

Przykład dowiązania symbolicznego:
```
$ ls –i plik  plik_sl
790 –rw-r—-r--  2  jurek  klasa  32  Apr 3 14:59  plik
791 lrw-r—-r--  1  jurek  klasa   4  Apr 4 10:07  plik_sl->plik
```

Wyświetlenie zawartości:
`cat plik` jest równoważne `cat plik_hl` i `cat plik_sl`.

Tworzenie dowiązania

Składnia:
```
ln [-s]  plik  plik_dowiązany
ln [-s]  plik1 [plik2...]  katalog
ln -s  katalog katalog_dowiązany
```

Opcje:
`-s` utwórz dowiązanie symboliczne (***soft link***)

- Dowiązanie nie zmienia własności, grupy, uprawnień.
- Przy próbie utworzenia dowiązania typu **hard** na innym dysku lub partycji pojawi się komunikat o błędzie.

Przykłady:

Dowiąż plik `p1` do katalogu `/home/kasia` (utwórz dowiązanie twarde):

```
$ ls –l p1
-rw-rw-r--   1  zosia  rok1  45 Feb 16 14:45  p1
$ ln p1 /home/kasia
$ ls –l p1
-rw-rw-r--   2  zosia  rok1  45 Feb 16 14:45  p1
 ls –l /home/kasia/p1
-rw-rw-r--   2  zosia  rok1  45 Feb 16 14:45  p1
```

Dowiąż symbolicznie katalog /aplikacje/examples/netlib do katalgu netlib

```
$ ln –s /aplikacje/examples/netlib netlib
$ ls –l netlib
lrwxrwxrwx  1 zosia  17 Jan 24 10:03  netlib->/aplikacje/examples/netlib
```

5.11. Modyfikacja czasów związanych z plikiem

Czasy związane z plikiem można modyfikować za pomocą polecenia touch. Polecenie to pozwala zmienić czas modyfikacji i dostępu.

Składnia:

```
touch [-opcje]  nazwa_pliku...
```

Działanie tego polecenia zależy od tego, czy plik będący jego argumentem istnieje czy nie:

- jeżeli plik nie istnieje, to jest tworzony plik o podanej nazwie, będzie on miał zerową długość,
- jeżeli plik istnieje, to będzie zmieniony jego czas modyfikacji na czas autualny (czas modyfikacji pliku jest wyświetlany po wydaniu polecenia ls –l).

Przykład:

```
$ ls –l
-rw-rw-r--  1 zosia  klasa     45  Feb 16 14:45  p1
-rw-rw-r--  1 zosia  klasa   3245  Jan 10  7:50  p2
-rw-rw-r--  1 zosia  klasa    123  Feb 10 10:12  p3

$ touch p1 p4
$ ls –l
-rw-rw-r--  1 zosia  klasa     45  Feb 24 14:30  p1
-rw-rw-r--  1 zosia  klasa   3245  Jan 10  7:50  p2
-rw-rw-r--  1 zosia  klasa    123  Feb 10 10:12  p3
-rw-rw-r--  1 zosia  klasa      0  Feb 24 14:30  p4
```

6. Prawa dostępu do plików

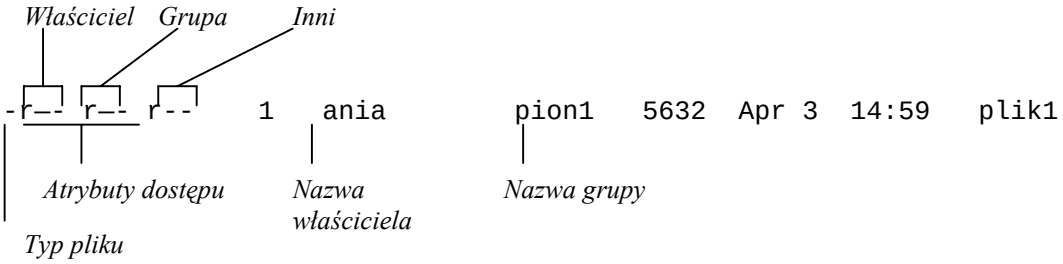
Unix rozróżnia trzy typy dostępu do pliku:
- prawo do czytania
- prawo do zapisywania
- prawo do wykonywania

Dodatkowo użytkownicy plików podzieleni są na trzy kategorie:

- właściciel (***user***) — ma pełne prawa do pliku:
  może ustalać typ dostępu dla pozostałych użytkowników,
  może przekazać prawo własności innemu użytkownikowi
- grupa (***group***) — zbiór użytkowników wyróżniony w celu nadania im określonych praw dostępu
- inni (***other***) — pozostali użytkownicy, nie będący właścicielem lub członkami grupy

Uprawnienia związane z dostępem do pliku wynikają z przynależności użytkownika do określonej kategorii oraz z ustanowionego przez właściciela dostępu do pliku dla danej kategorii użytkowników.

Przykład:



6.1. Podstawowe prawa dostępu do pliku (*permission, mode*)

Podstawowe prawa dostępu obejmują:
`r` prawo do czytania (*read*)
`w` prawo do zapisu (*write*)
`x` prawo do wykonywania (*execute*)

Przykład: **`rwxr-xr--`**
właściciel **`rwx`**
grupa **`r-x`**
inni **`r--`**

Symbol `-` oznacza brak prawa dostępu określonego typu.

Prawa dostępu można sprawdzić za pomocą polecenia ls:

`ls -l` *`nazwa_pliku`*	lub	`ll` *`nazwa_pliku`*	HP-UX
`ls -lg` *`nazwa_pliku`*			SUN OS
`ls -ld` *`nazwa_katalogu`*	lub	`ll -d` *`nazwa_katalogu`*	HP-UX
`ls -ldg` *`nazwa_katalogu`*			SUN OS

Interpretacja praw dostępu

	r	w	x
pliki	Można otworzyć plik i przeczytać zawartość (more, cat, lp, cp)	Można zmieniać zawartość pliku (vi)	Można wykonywać program
katalogi	Można przeglądać zawartość katalogu (ls)	Można zmieniać zawartość katalogu (rm, cp, mv, mkdir, rmdir, ln)	Uruchomiony program lub wydane polecenie ma dostęp do pliku w katalogu (cd, ll nazwa_pliku)

Ochrona katalogu:

Prawa dostępu	**Wynik**
Brak praw dostępu	Katalog ani jego podkatalogi nie są dostępne
Prawo do wykonywania	Pozwala użytkownikom pracować z tymi programami w katalogu, które są im znane; pozostałe programy są ukryte
Prawo do czytania i wykonywania	Pozwala użytkownikom pracować z programami w katalogu, wyświetlać zawartość katalogu, ale nie pozwala tworzyć lub usuwać plików z katalogu
Prawo do czytania, pisania i wykonywania	Pozwala użytkownikom pracować z programami w katalogu, wyświetlać zawartość katalogu, tworzyć lub usuwać pliki z katalogu

6.2. Zmiana praw dostępu

Składnia:

`chmod [-R] prawa_dostępu nazwa_pliku [nazwa_pliku]`
gdzie: ***prawa_dostępu*** określają nowe prawa

Tylko właściciel lub administrator mogą zmieniać prawa dostępu.
Prawa dostępu można określać za pomocą:

- wyrażenia w systemie ósemkowym (zmiana bezwzględna)
  `chmod 764 plik`
- kodów mnemotechnicznych (zmiana bezwzględna lub względna)
  `chmod u=rwx,g=rw,o=r  plik`
  `chmod u-x  plik`

Przykład 1: Uprawnienia podawane w systemie ósemkowym

```
$ ls -l plik1
-rw-rw-rw-  1     ania  pion1 1123 Feb 2 9:12 plik1
$ chmod 700 plik1
$ ls -l plik1
-rwx------  1     ania  pion1 1123 Feb 2 9:12 plik1
$ chmod 644 plik1
$ ls -l plik1
-rw-r--r--  1     ania  pion1 1123 Feb 2 9:12 plik1
```

Przykład 2: Uprawnienia podawane za pomocą kodów mnemotechnicznych

```
$ ls -l plik1
-rw-rw-rw-  1     ania  pion1 1123 Feb 2 9:12 plik1
$ chmod a=rwx plik1
$ ls -l plik1
-rwxrwxrwx  1     ania  pion1 1123 Feb 2 9:12 plik1
$ chmod u=rw,go-wx plik1
$ ls -l plik1
-rw-r--r--  1     ania  pion1 1123 Feb 2 9:12 plik1
```

6.3. Użyteczne tabele

Interpretacja kodów ósemkowych

Prawa pliku	Właściciel	Grupa	Inni
`rwxr-xr--`	`rwx` **111** **7**	`r-x` **101** **5**	`r--` **100** **4**

Tabela z interpretacją kodów ósemkowych

0	Brak praw	`-`
1	Wykonywanie	`x`
2	Pisanie	`w`
3	Wykonywanie i pisanie	`wx`
4	Czytanie	`r`
5	Czytanie i wykonywanie	`rx`
6	Czytanie i pisanie	`rw`
7	Czytanie, pisanie, wykonywanie	`rwx`

Kody mnemotechniczne

Klasa użytkownika	
`u`	Właściciel
`g`	Grupa
`o`	Inni
`a`	Wszyscy

Uprawnienie	
`+`	Dodaj
`-`	Odbierz
`=`	Nadaj

Do	
`r`	Czytania
`w`	Pisania
`x`	Wykonywania

6.4. Uprawnienia nadawane nowemu plikowi/katalogowi i ich modyfikacja

Każdy nowo utworzony plik/katalog otrzymuje standardowy zestaw uprawnień (***default permission***).
Najczęściej są to:

`rw-r--r--` c	czyli	**644**	dla pliku
`rwxr-xr-x`	czyli	**755**	dla katalogu

Użytkownik może modyfikować początkowe prawa za pomocą polecenia `umask` (*user file-creation mode mask*). Polecenie to określa prawa, które mają być odebrane nowo tworzonym plikom / katalogom.

Składnia:

`umask`	wyświetl obowiązującą wartość maski
`umask` *`nowa_wartość`*	ustaw wartość maski uprawnień nadawanych nowo utworzonemu plikowi/katalogowi

Przykład:

```
umask 022
```

czyli

- nie zmieniaj uprawnień właściciela (plik `rw-`, katalog `rwx`)
- nadaj grupie uprawnienie dla pliku `r--`, dla katalogu `r-x`
- nadaj innym uprawnienie dla pliku `r--`, dla katalogu `r-x`

```
$ touch nowy_plik
$ mkdir nowy_katalog
$ ll  nowy_plik
-rw-r--r--  1  ania   pion1   1667   Feb   2  9:30   nowy_plik
$ ll  -d  nowy_katalog
-rwxr-xr-x  2  ania   pion1      24   Feb   2  9:32   nowy_katalog
```

Prawa własności nowego pliku/katalogu

Właścicielem tworzonego pliku/katalogu jest osoba, która wydaje polecenie.
Plik/katalog zostaje przyporządkowany grupie, do której właściciel należał w chwili wydawania polecenia.

6.5. Zmiana właściciela pliku

Składnia:
```
chown    [-R]   nowy_właściciel    nazwa_pliku
```

Możliwość wydawania polecenia chown zależy od wersji systemu Unix:
- W HP-UX każdy właściciel pliku może oddać go innej osobie.
- W SunOS właściciela może zmieniać tylko administrator (użytkownik root).

6.6. Zmiana grupy

Składnia:
```
chgrp    nowa_grupa    nazwa_pliku
```

Możliwość wydawania polecenia chown zależy od wersji systemu Unix:
- W HP-UX każdy właściciel pliku może zmienić grupę na dowolnie inną.
- W Sun OS właściciel pliku może zmienić grupę tylko taką, do której on sam należy.

6.7. Jak sprawdzić nazwę użytkownika / grupy?

Identyfikacja użytkownika w systemie

- Każdy użytkownik w systemie posiada swój identyfikator:
  uid (ang. *user id*) **liczbowy**, np. 7007
  tekstowy, np. ania
- Użytkownik wchodzi w skład grupy, która również ma swój identyfikator:
  gid (ang. *group id*) **liczbowy**, np. 22
  tekstowe, np. pion1

Użytkownik może należeć do kilku grup.
Opis użytkownika zawarty jest w plikach:

`/etc/passwd`	identyfikator użytkownika, jego grupa w momencie otworzenia sesji
`/etc/group`	identyfikatory grup istniejących w systemie i przynależnych do nich użytkowników

Polecenia do identyfikacji użytkownika

`who am i`	wyświetla nazwę użytkownika, który rozpoczął sesję (*login name*) i inne informacje z nim związane
`whoami`	wyświetla bieżącą nazwę użytkownika
`id`	wyświetla wszystkie identyfikatory użytkownika
`groups` *`nazwa_użytkownika`*	wyświetla nazwę/nazwy grup, do których należy użytkownik

Przykład:
```
$ whoami
ania
$ who am i
ania   tty0p9   Feb 29   16:47
$ id
uid=7007(ania)  gid=201(pion1)
$ groups ania
pion1   ztsw
```

6.8. Zmiana identyfikatora użytkownika podczas sesji

Można chwilowo uzyskać uprawnienia innego użytkownika za pomocą zmiany:
- identyfikator użytkownika polecenie `su`
- identyfikator grupy polecenie `newgrp`

Skladnia:
```
su   [-]   [nazwa_użytkownika]
newgrp [nazwa_grupy]
```

- `su` bez argumentu wybiera użytkownika root
- `newgrp` bez argumentu przywraca grupę początkową użytkownika.
- `exit` przywraca poprzedni identyfikator użytkownika

Przykład:

```
$ whoami
ania
$ who am i
ania   tty0p9   Feb 29   16:47
$ su janusz
$ logname
ania
$ whoami
janusz
$ who am i
ania   tty0p9   Feb 29   16:47
$ id
uid=7008(janusz)  gid=203(pion3)

$ exit

$ whoami
$ ania
```

6.9. Mechanizm ACL (*Access Control List*)

Mechanizm ten rozszerza możliwości określenia praw dostępu do pliku: właściciel może selektywanie nadać określone prawa wybranym użytkownikom.
Mechanizm ACL nie jest dostępny w każdej wersji systemu Unix. Jest dostępny na przykład w HP-UX.

Rozróżnia się dwa typy praw ACL:
- prawa podstawowe - jest to odwzorowanie zwykłych praw dostępu;
- prawa dodatkowe - nadawane za pomocą specjalnego polecenia

Prawa dostępu ACL można definiować dla:
- określonego użytkownika w określonej grupie (użytkownik.grupa)
- określonego użytkownik w dowolnej grupie (użytkownik.%)
- dowolnego użytkownika w określonej grupie (%.grupa)
- dowolnego użytkownika w dowolnej grupie (%.%)

Sprawdzanie listy praw listy ACL w HP-UX

```
lsacl   [-l]  nazwa_pliku ...
```

Nadawanie praw listy ACL w HP-UX

```
chacl  '(użytkownik.grupa,prawa)'  nazwa_pliku ...
chacl  '(użytkownik.grupa  operator  prawa)'  nazwa_pliku ...
```

Polecenia, które uwzględniają prawa ACL (HP-UX)

```
ll  plik_z_nadanymi_prawami_ACL
chmod  -A  plik_z_nadanymi_prawami_ACL
```

7. Edytor vi - visual display editor

- Podstawowy edytor tekstowy systemu UNIX.
- Może być używany na dowolnym terminalu.
- vi pracuje z kopią pliku umieszczoną w buforze pamięci.

7.1. Rozpoczęcie pracy

Edytor vi uruchamia się za pomocą polecenia vi, które jako argument otrzymuje nazwę pliku.

Przykład 1:
Jako argument polecenie vi ma nazwę nowego, jeszcze nie istniejącego pliku:

```
vi plik1
■
~
~
~
~
~
~
"plik1" [New File]
```

Przykład 2:
Jako argument polecenie vi ma nazwę pliku już istniejącego:

```
vi plikstary
W tym pliku jest zapisana informacja o ....
~
~
~
~
~
~
"plikstary" 1 line, 39 characters
```

Symbol ~ na początku wiersza oznacza pusty wiersz (za koncem pliku).

Tryby pracy edytora vi

Tryb poleceń (***command mode***)	Każde naciśnięcie klawisza jest poleceniem edytora vi. Wpisywane z klawiatury polecenia nie są widoczne na ekran Są od razu wykonywane. Polecenie błędne jest sygnalizowa sygnałem dźwiękowym
Tryb wprowadzania*(input mode*)	Każde naciśnięcie klawisza wprowadza tekst do pliku . Wyj naciśnięcie klawisza ESC. Można pracować w trybie: **wstawiania** (***insert mode***) **nadpisywania** (***replace mode***) **zamiany** (***change mode***)
Tryb edytora ex (***ex mode***)	Każdy napisany tekst jest uważany za polecenie edytora ex.

- **Uwaga**: Edytor vi rozróżnia w poleceniach małe i duże litery.

Zmiana trybów pracy

- Rozpoczęcie pracy: automatycznie wybierany jest **tryb poleceń**
- Przejście do trybu edycji: za pomocą jednego z poleceń związanych z dodawaniem lub zmienianiem tekstu np. polecenie **i** (insert) - wstaw tekst w miejscu kursora
- Przejście do trybu poleceń: za pomocą klawisza ESC
- Przejście do trybu edytora ex : polecenia rozpoczynające się dwukropkiem (**:**), ukośnikiem (/) lub znakiem zapytania (**?**).

Jak sprawdzić tryb pracy?

Naciśnij klawisz ESC :

- Jeżeli już jesteś w trybie poleceń, usłyszysz sygnał dźwiękowy.
- Jeżeli jesteś w trybie wprowadzania, przejdziesz w tryb poleceń.

Ustaw opcję wyświetlania trybu wprowadzania:

```
:set showmode
```

Po ustawieniu tej opcji, gdy jesteś w trybie wprowadzania w prawym dolnym rogu ekranu wyświetlana jest informacja o wybranym trybie (insert, replace itd.)

7.2. Kończenie pracy w edytorze vi

Tryb poleceń vi:

`ZZ`	zapisz bufor do pliku o nazwie podanej podczas rozpoczynania pracy z vi i zakończ pracę

Częściej wydaje się polecenia w trybie edytora ex:

`:w`	zapisz do pliku, pozostań w edytorze
`:w` *`nazwa_pliku`*	zapisz do pliku *nazwa_pliku*, pozostań w edytorze
`:q`	zakończ bez zapisywania
`:wq`	zapisz i zakończ pracę (odpowiednik ZZ)
`:q!`	zakończ natychmiast, nie zapisuj zmian do pliku

7.3. Polecenia wpisywania tekstu

- Miejsce wstawienia tekstu określane jest położeniem kursora.
- W zależności od tego, gdzie w stosunku do kursora ma być wstawiany tekst, można posłużyć się jednym z poleceń:

`a`	dołącz tekst za kursorem (*append*)
`A`	dołącz tekst na końcu wiersza
`i`	wstaw tekst przed kursorem (*insert*)
`I`	wstaw tekst na początku wiersza
`o`	otwórz nowy, pusty wiersz pod bieżącym wierszem (*open*)
`O`	otwórz nowy, pusty wiersz nad bieżącym wierszem

- Każde z tych poleceń powoduje przejście w tryb wprowadzania (*input mode*). W trybie tym każde naciśnięcie klawisza wprowadza tekst do pliku.
- Powrót do trybu poleceń wymaga naciśnięcia klawisza ESC.

Usuwanie wpisywanego tekstu w trybie wprowadzania:

`Backspace`	usuwa jeden znak do tyłu
`^w`	usuwa znaki do początku słowa

Uwaga: znaki są usunięte mimo, że są wyświetlane.

Przykład wprowadzania tekstu

1. Wywołaj edytor vi podając jako parametr nazwę nowego pliku:
   `$vi  nowy_plik`
2. Naciśnij klawisz **`i`** aby przejść w tryb wstawiania (*insert*) i móc wprowadzać tekst.
3. Wpisz tekst tak, jak go widzisz. Błędy są celowe. Nie poprawiaj ich. Możesz poprawiać tylko swoje błędy (inne niż w przykładzie), o ile je w porę zauważysz, za pomocą klawiszy Backspace oraz ^w .

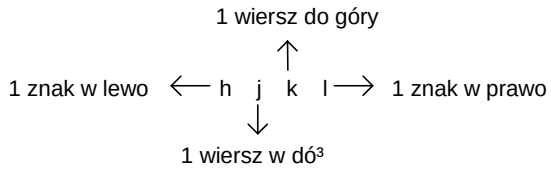
```
Wpisz nowy tekst.
W drugim wiersz jest jeden blad.
Trzeci wiersz jest poprawny.
czwartym wieszu sa dwa bledy.
```

4. Naciśnij klawisz **ESC** aby wrócić do trybu poleceń. Sprawdź, ponownie naciskając klawisz **ESC**, że jesteś w trybie poleceń.
5. Zakończ pracę za pomocą polecenia **`:wq`** . Jest to polecenie zapisania pliku na dysku i zakończenia pracy z vi.

7.4. Polecenia poruszania się po tekście

Podstawowe ruchy kursorem

- Wykonywane są za pomocą klawiszy alfanumerycznych - nie zależnie od tego, czy terminal posiada czy nie posiada klawiatury ze strzałkami



`h`	(Backspace)	jeden znak w lewo
`j`	(Return)	jeden wiersz w dół
`k`		jeden wiersz do góry
`l`	(Spacja)	jeden znak w prawo

- Każde polecenie ruchu kursora może być zwielokrotnione
  `l` przesuń o jeden znak w prawo
  `5l` przesuń o 5 znaków w prawo

Ustawienie w określonym wierszu

`G`	ostatni wiersz pliku
n`G`	wiersz o numerze *n*

Poruszanie się po wierszu

`$`	ostatni znak w bieżącym wierszu
`0`	pierwszy znak w bieżącym wierszu
`^`	pierwszy różny od spacji znak w bieżącym wierszu
`w`	początek następnego słowa
`W`	początek następnego słowa, ignoruj znaki przestankowe
`b`	początek poprzedniego słowa
`B`	początek poprzedniego słowa, ignoruj znaki przestankowe
`e`	koniec następnego słowa
`E`	koniec następnego słowa, ignoruj znaki przestankowe

Przewijanie pliku

`^d`	przewiń w dół (*down*) o połowę ekranu
`^f`	przewiń w dół (*forward*) o cały ekran
`^u`	przewiń do góry (up) o połowę ekranu
`^b`	przewiń do góry (backward) o cały ekran

Poruszanie się po ekranie

`L`	ostatni wiersz na ekranie
`M`	środkowy wiersz na ekranie
`H`	pierwszy wiersz na ekranie

Odświeżanie ekranu

`^L`	odświeżanie ekranu

Przykład poprawiania tekstu

1. Wywołaj edytor vi podając jako parametr nazwę poprawianego pliku:

```
$ cat  nowy_plik
Wpisz nowy tekst.
W drugim wiersz jest jeden blad.
Trzeci wiersz jest poprawny.
czwartym wieszu sa dwa bledy.
$vi nowy_plik
```

2. Ustaw opcję wyświetlania informacji o wybranym trybie (o ile nie jest ona już ustawiona)

```
:set showmode
```

3. Popraw błąd w drugim wierszu :

3.1. Za pomocą klawisza **j** ustaw się w drugim wierszu.
3.2. Za pomocą klawisza **l** (oraz **h**) ustaw się na literze **z** w słowie **wiersz**.
3.3. Naciśnij **a** (*append*)
3.4. Popraw tekst dopisując **u**
3.5 Naciśnij **ESC**.

4. Popraw pierwszy błąd w czwartym wierszu:

4.1. Za pomocą klawisza **j** ustaw się w czwartym wierszu.
4.2. Naciśnij **I** (duża litera i - wstaw na początku wiersza).
4.3. Popraw tekst wpisując **W** oraz spację.
4.4. Naciśnij **ESC**.

5. Popraw drugi błąd w czwartym wierszu:

5.1. Za pomocą klawiszy **l** oraz **h** ustaw się na literze **s** w **wieszu**.
5.2. Naciśnij **i** (wstaw przed kursorem).
5.3. Popraw tekst wpisując **r**
5.4. Naciśnij **ESC**.

6. Zakończ pracę za pomocą polecenia **:wq**

7.5. Polecenia usuwania tekstu

`x`	usuń znak w miejscu kursora
`X`	usuń znak na lewo od kursora
`d`*`obiekt`*	usuń obiekt (słowo, wiersz, fragment tekstu)

- Polecenie może być zwielokrotnione
  `5x` usuń 5 znaków

- Obiekt może być definiowany za pomocą polecenia przesunięcia kursora:

`dw`	usuń do początku następnego słowa
`dG`	usuń do końca pliku
`d$`	usuń do końca wiersza
`d^`	usuń od kursora do początku wiersza
`5dw`	usuń 5 słów

- Obiektem może być cały wiersz (powtarzana jest wtedy nazwa polecenia):

  `dd` usuń cały wiersz

- Liczba określająca ilość powtórzeń może być umieszczona przed poleceniem usuwania lub między poleceniem usuwania i obiektem

  `5dw` usuń 5 słów
  `d5w` usuń 5 słów

7.6. Zaniechanie zmian

`u`	zaniechaj (*undo*) ostatniej zmiany w tekście
`U`	zaniechaj (undo) wszystkich zmian w bieżącym wierszu(aby polecenie zadziałało, nie przesuw kursora!)
`:q!`	zakończ pracę edytora vi bez zapisywania zmian do pliku

7.7. Przenoszenie i kopiowanie tekstu

- Polecenia przenoszenia i kopiowania tekstu posługują się buforem. Tekst jest w nim przechowywany zanim zostanie wstawiony w nowym miejscu.

Polecenia wykorzystywane przy przenoszeniu i kopiowaniu

`d`*`obiekt`*	usuń (*delete*) obiekt do bufora bez nazwy
`y`*`obiekt`*	skopiuj (*yank*) obiekt do bufora bez nazwy
`p`	wstaw (*put*) zawartość bufora bez nazwy za kursorem
`P`	wstaw zawartość bufora bez nazwy przed kursorem

Schemat postępowania:

1. Umieść kursor na początku tekstu, który chcesz skopiować lub przenieść.
2. Gdy chcesz kopiować, wydaj polecenie `y`*`obiekt`*.
   Gdy chcesz przenosić wydaj polecenie `d`*`obiekt`*.
3. Przesuń kursor do miejsca, w którym chcesz umieścić kopiowany lub przenoszony tekst.
4. Gdy chcesz wstawić za kursorem, wydaj polecenie p
   Gdy chcesz wstawić przed kursorem, wydaj polecenie P.

Znak nowej linii (*new line*)

- Gdy tekst umieszczony w buforze zawiera znak nowej linii:
  - polecenie p umieści tekst z bufora w wierszu poniżej kursora
  - polecenie P umieści tekst z bufora powyżej kursora
- Gdy tekst umieszczony w buforze nie zawiera znaku nowej linii:
  - polecenie p umieści tekst z bufora z prawej strony kursora
  - polecenie P umieści tekst z bufora z lewej strony kursora

Przykłady:

`dd`	usuń cały wiersz, wstaw do bufora
`yy`	skopiuj cały wiersz do bufora
`d$`	usuń tekst do końca wiersza, wstaw do bufora
`y$`	skopiuj tekst do końca wiersza do bufora
`dw`	usuń słowo, wstaw do bufora
yw	skopiuj słowo do bufora
`4yy`	skopiuj 4 wiersze do bufora

Przykład przenoszenia tekstu

1. Wywołaj edytor vi podając jako parametr nazwę poprawianego pliku:

```
$ cat  plik_przenies
To jest wiersz pierwszy.
To jest wiersz trzeci.
To jest wiersz czwarty.
oT wiersz drugi.
$ vi  plik_przenies
```

2. Przenieś wiersz 4 za wiersz 1:
   - 2.1. Umieść kursor w dowolnym miejscu w wierszu 2.
   - 2.2. Napisz **dd** (usuń wiersz do bufora)
   - 2.3. Przesuń kursor do wiersza 1.
   - 2.4. Napisz **p** (wstaw wiersz z bufora poniżej kursora)

3. Popraw czeski błąd **oT** w wierszu drugim :
   - 3.1. Umieść kursor na literze **o**.
   - 3.2. Napisz **xp** (x - usuwa jeden znak do bufora, p wstawia zawartość bufora za kursorem)

4. Zakończ pracę za pomocą polecenia :**wq**

Przykład kopiowania tekstu

1. Wywołaj edytor vi podając jako parametr nazwę poprawianego pliku:

```
$ cat  plik_kopiuj
To jest pierwszy wiersz tekstu do kopiowania.
To jest wiersz drugi.
To jest wiersz trzeci.
To jest ostatni wiersz tekstu do kopiowania.
$ vi  plik_kopiuj
```

2. Powiel cztery wiersze tekstu umieszczając je za ostatnim wierszem tekstu oryginalnego.
   - 2.1. Umieść kursor w dowolnym miejscu w wierszu 1.
   - 2.2. Napisz **4yy** (skopiuj 4 wiersze do bufora)
   - 2.3. Przesuń kursor do oststniego wiersza.
   - 2.4. Napisz **p** (wstaw wiersz z bufora poniżej kursora)

3. Zakończ pracę za pomocą polecenia **:wq**

7.8. Inne przydatne polecenia

`.`	powtórz ostatnią zmianę tekstu
`J`	połącz (*join*) bieżący wiersz z następnym

Dzielenie wierszy

Nie ma specjalnego polecenia podziału wiersza. Aby podzielić wiersz należy:

1. Ustaw kursor w miejscu, w którym chcesz podzielić wiersz.
2. Przejdź w odpowiedni tryb wstawiania, np. poleceniem **i**
3. Naciśnij Return

7.9. Zmiana tekstu

`rX`	zastąp bieżący znak znakiem X
`R`	przejdź do trybu nadpisywania (*replace*)
`cobiekt`	zmień nazwany obiekt (usuwa nazwany obiekt, przechodzi w tryb wstawiania, należy wstawić nowy tekst)
`s`*`nowy_tekst`*`ESC`	zastąp bieżący znak tekstem *nowy_tekst*
`S`	usuń zawartość całego wiersza i przejdź w tryb wstawiania

Przykłady:

`cw`	zmień do początku następnego słowa
`cG`	zmień do końca pliku
`cc`	zmień cały wiersz

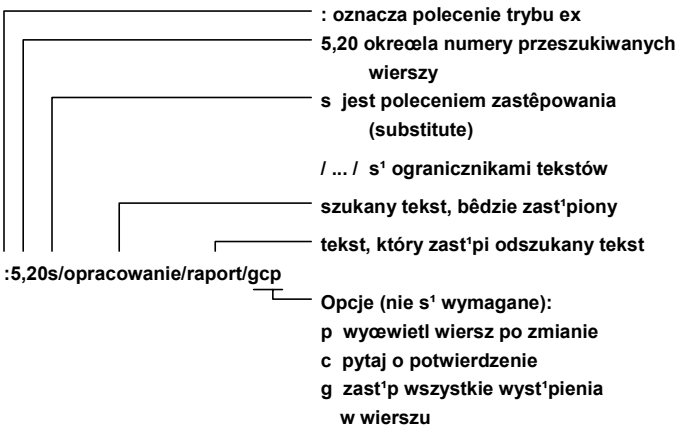
7.10. Poszukiwanie tekstu

`/tekst`	szukaj tekstu w dół, po napotkaniu końca pliku kontynuuj od początku pliku
`?tekst`	szukaj tekstu do góry, po napotkaniu początku pliku kontynuuj od końca pliku
`n`	powtórz poprzednie szukanie w tym samym kierunku
`N`	powtórz poprzednie szukanie w odwrotnym kierunku

7.11. Działania na plikach

`:r` *`nazwa_pliku`*	czytaj plik *nazwa_pliku* i umieść po bieżącym wierszu
`:e` *`nazwa_pliku`*	zastąp bieżący plik plikiem *nazwa_pliku*
`:w >>` *`nazwa pliku`*	dołącz zawartość buforu z plikiem do pliku *nazwa_pliku*
`:`*`M`*`,`*`N`*`w` *`nazwa_pliku`*	zapisz wiersze *M* do *N* do pliku *nazwa_pliku*

7.12. Globalne poszukiwanie i zastępowanie tekstu w trybie edytora ex



Przykłady:

- Zastąp wszystkie wystąpienia tekstu *stary_tekst* tekstem *nowy_tekst* :

```
:1,$s/stary_tekst/nowy_tekst/g
```

- Zastąp pierwsze wystąpienie w wierszu tekstu *stary_tekst* tekstem *nowy_tekst*, działanie wykonaj w wierszach *m,n* :

```
:m,ns/stary_tekst/nowy_tekst/
```

Wyrażenia regularne (regular expressions)

Szukając tekstu można posłużyć się wyrażeniem regularnym.

Przykłady prostych wyrażeń regularnych :

`^tekst`	tekst na początku wiersza
`tekst$`	tekst na końcu wiersza
`[ ]`	klasa znaków, z których każdy może wystąpić
`.`	dowolny znak
`znak*`	powtórz znak 0 lub dowolną liczbę razy

Przykłady wykorzystania wyrażeń regularnych do szukania tekstu w edytorze vi:

`/^po`	szukaj tekstu po na początku wiersza
`/koniec$`	szukaj wiersza zakończonego tekstem koniec
`/[Ss]tary_tekst`	szukaj tekstu "Stary_tekst" lub "stary_tekst"
`/^[abc].*`	szukaj wiersza rozpoczynającego się od znaku a,b lub c

7.13. Dopasowywanie edytora vi

Opcje pozwalają dopasować edytor do potrzeb użytkownika.

Polecenia ustawiania opcji:

`:set all`	wyświetl obowiązujące ustawienie
`:set opcja`	włącz opcję
`:set noopcja`	wyłącz opcję

Przykłady:

`:set autoindent`	rozpoczynaj następny wiersz od tej samej kolumny, co wiersz poprzedni; po ustawieniu tej opcji, aby rozpocząć od pierwszej kolumny, musisz nacisnąć CTRL+D
`:set number`	wyświetlaj numery wierszy
`:set nonumber`	nie wyświetlaj numerów wierszy
`:set showmode`	wyświetlaj informację o trybie pracy
`:set tabstop=5`	ustaw tabulację co 5 znaków

Plik `.exrc`

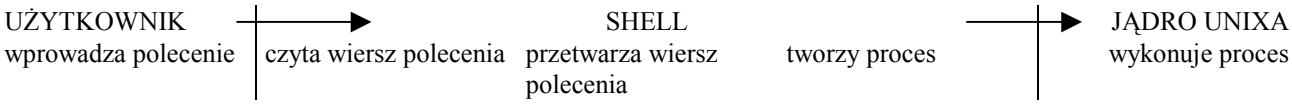
Aby ustawienia obowiązywały przy każdym wywołaniu, należy umieścić je w pliku .exrc w katalogu osobistym.
Polecenie ustawienia opcji umieszczane w pliku .exrc ma postać:

`set opcja` (bez dwukropka)

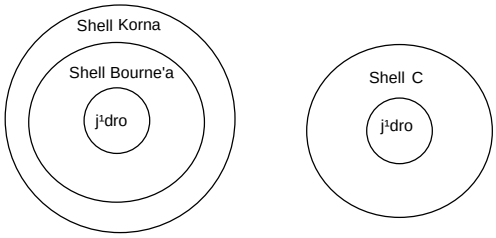
8. Podstawy korzystania z shella

8.1.Co to jest Shell?

Jest to program, który pośredniczy między jądrem systemu (***kernel***), systemem plików (***file system***)i programami usługowymi (***utilities***).



Podstawowe shelle



Podstawowe funkcje shella

- Przekazywanie sterowania do programu wybranego poleceniem użytkownika
- Wykonywanie wbudowanych poleceń
- Dostarczenie języka do pisania skryptów
- Ustawianie środowiska pracy
- Przywoływanie i edycja uprzednio wydanych poleceń
- Przeadresowywanie wejścia - wyjścia poleceń
- Generowanie nazw plików
- Umożliwienie łączenia poleceń w potok
- Umożliwienie przetwarzania w drugim planie (nie interakcyjnie)

8.2. Shell jako interpretator poleceń

Shell interpretuje pierwsze słowo w wierszu polecenia jako nazwę polecenia.

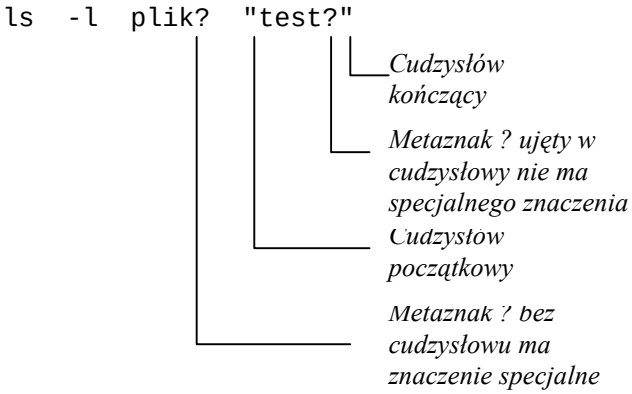
```
$ ls  -l  plik?  "test?"
```

1. Shell wczytuje polecenie do wewnętrznego bufora.
```
ls  -l  plik?  "test?"
```

2. Polecenie jest dzielone na części nazywane słowami. Shell określa znaczenie każdego słowa.

`ls`	`-l`	`plik?`	`"test?"`

3. Shell szuka i przetwarza znaki specjalne.



4. W wyniku dokonanych przekształceń Shell zleca wykonanie polecenia o postaci:
```
ls -l plik1 plik2 plik3 test?
```

5. Shell "usypia" i czeka na zakończenie wykonywania polecenia.

6. Po zakończeniu wykonywania polecenia zgłasza gotowość przyjęcia nowego polecenia.

8.3. Który shell?

Gdzie wybierany jest shell?
Shell przydzielany użytkownikowi w momencie otwierania sesji jest określany przez administratora podczas zakładania konta nowego użytkownika (plik /etc/passwd)

Jak rozpoznać przydzielony shell?
Każdy shell ma swój standardowy znak zgłoszenia gotowości:
- `$` Shell Bourne'a (sh), Shell Korna (ksh), Shell Posix (sh)
- `%` shell C (csh)

Można również sprawdzić nazwę shella poleceniami:
```
finger -m nazwa_użytkownika
more /etc/passwd
echo $SHELL
```

Jak zmienić shell podczas sesji?
Należy **w**pisać nazwę wywoływanego shella. W systemie HP-UX 11.x będzie to polecenie:

`/usr/bin/csh`	shell C
`/usr/old/bin/sh`	shell Bourne'a
`/usr/bin/ksh`	shell Korna
`/usr/bin/sh`	shell Posix

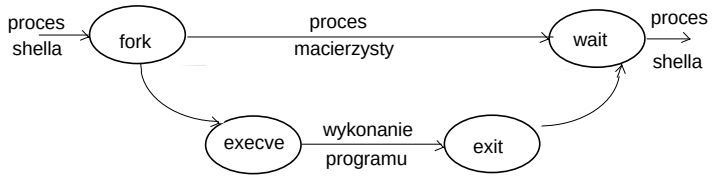
Jak porozumiewać się z shellem
- za pomocą poleceń - tryb interakcyjny
- za pomocą plików wsadowych zwanych skryptami ***(scripts***)

Jak powrócić do poprzedniego shella?
- polecenie `exit`
- CTRL-d

8.4. Jak jest wykonywane polecenie

Shell dopuszcza trzy typy poleceń:
- plik wykonywalny, który zawiera kod wynikowy skompilowanego programu
- plik wykonywalny, który zawiera ciąg poleceń dla shella (nazywany *skryptem*)
- wewnętrzne polecenia shella

Schemat wykonywania poleceń w systemie Unix



Przed wykonaniem polecenia shell musi wiedzieć
- kto wykonuje polecenie (czy ma prawo do wykonania)
- gdzie umieszczone jest polecenie do wykonania

Polecenia wbudowane (wewnętrzne) i zewnętrzne

Polecenie wewnętrzne ksh	Polecenie zewnętrzne Unixa
cd	ls
exit	more

Polecenia wbudowane

- ich zestaw zależy od shella (mogą być inne w shellu ksh i na przykład w shellu csh)
- wykonywane wewnątrz shella - nie tworzą nowego procesu

Polecenia zewnętrzne

- realizowane przez oddzielne programy
- wymagają utworzenia nowego procesu
- wymagają określenia katalogu, w którym są umieszczone

Skrypty

- plik skryptu musi mieć prawo do czytania i wykonywania

8.5. Środowisko shella

Podstawowe środowisko shella tworzą:
- opcje- mogą być włączone lub wyłączone, modyfikują zachowanie shella
- zmienne- można im przypisywać wartości określające zachowanie shella i innych programów

Przykład opcji:
`ignoreeof` ignoruj CTRL-d, jeśli wydane w celu zakończenia sesji; sesja będzie zakończona tylko za pomocą polecenia exit
`vi` używaj edytora vi do edycji uprzednio wydanych poleceń

Przykład zmiennych:
HOME katalog osobisty (domowy) użytkownika
PATH ścieżka dostępu do programów

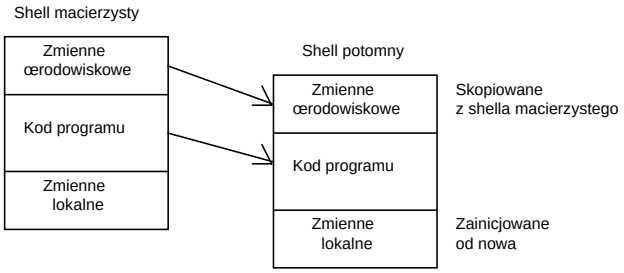
Shell macierzysty i shell potomny

W momencie otworzenia sesji każdy użytkownik uruchamia swoją kopię shella. Działa ona do zamknięcia sesji.
W niektórych sytuacjach tworzony jest dodatkowo shell potomny (podshell):

- wykonywanie poleceń ze skryptu
- grupowanie poleceń - (cd /tmp; pwd; ls)

Uruchamianie shella potomnego przebiega tak samo, jak uruchamianie zwykłego programu: tworzony jest proces potomny.

Przekazywanie informacji z shella macierzystego do shella potomnego



Zmienne shellowe

Zmienne shellowe mogą być deklarowane jako

- **lokalne** dostępne tylko w bieżącym shellu
- **środowiskowe** przekazywane do każdego uruchomionego polecenia lub podshella

Nazwa zmiennej rozpoczyna się od litery. Rozróżniane są małe i wielkie litery.
Składnia polecenia określającego typ zmiennej oraz przypisującego jej wartość zależy od shella.
Każdy shell ma wbudowany zestaw zmiennych środowiskowych i lokalnych. Niektóre z nich mają nadane wartości domyślne.
Przyjęło się, że nazwy zmiennych środowiskowych są pisane wielkimi literami.

Przypisywanie wartości zmiennym w shellu Korna

nazwa_zmiennej=wartość	zmienna lokalna
export nazwa_zmiennej	zmiana typu na środowiskowy

Usuwanie zmiennej

unset *nazwa_zmiennej*

Sprawdzenie wartości zmiennych

`set`	wszystkie zmienne
`env`	zmienne środowiskowe
`echo $`***`nazwa_zmiennej`***	konkretna zmienna

Użycie zmiennej

```
PATH=$PATH:$HOME
```

Zmienne wstępnie zdefiniowane (predefined shell variables)

Shell ma wbudowaną listę nazw zmiennych, które rozpoznaje i wykorzystuje w działaniu.
Niektóre z nich mają wstępnie ustaloną wartość.

Zmienne ustawione w momencie rozpoczynania sesji:

HOME	katalog osobisty, wg /etc/passwd
SHELL	shell użytkownika, wg /etc/passwd
TERM	typ terminala
LOGNAME	identyfikator użytkownika, wg /etc/passwd
PATH	ścieżka przeszukiwanych katalogów
PWD	nazwa bieżącego katalogu
PS1	tekst zgłoszenia gotowości shella
PS2	tekst zgłoszenia kontynuacji polecenia
MAIL	katalog ze skrzynką pocztową

8.7. Techniki wykonywania skryptów shellowych

W celu wykonania skryptu można:
- nadać plikowi ze skryptem prawo do czytania i wykonywania - wtedy wystarczy wpisać nazwę skryptu
- nadać plikowi ze skryptem prawo do czytania i wykonać go za pomocą uruchmionego shella

Przykład:
```
$ chmod u+x skrypt
$ skrypt

$ ksh skrypt
```

Jeśli chce się uruchmić skrypt w bieżącym shellu trzeba użyć polecenia (dotyczy sh, ksh, posix):

```
$ .  skrypt
```

8.8. Pliki konfiguracyjne shella

- Środowisko shella można określić w pliku konfiguracyjnym.
- Plik taki powinien znajdować się w katalogu osobistym użytkownika.
- Zawiera przypisania wartości zmiennych i polecenia, które muszą być wykonane podczas otwierania sesji lub ponownego uruchomienia shella.

Każdy shell ma sobie właściwe pliki konfiguracyjne.

Nazwa shella	**Pliki konfiguracyjne**
Bourne	.profile - wykonywany podczas otwierania sesji
Korn	.profile - wykonywany podczas otwierania sesji .kshrc - wykonywany jeśli podczas sesji zostanie uruchomiona nowa wersja shella Korna; wymaga ustawienia odpowiedniej zmiennej
Posix	.profile - wykonywany podczas otwierania sesji .kshrc - wykonywany jeśli podczas sesji zostanie uruchomiona nowa wersja shella Posix; wymaga ustawienia odpowiedniej zmiennej
C	.login - wykonywany podczas otwierania sesji .cshrc - wykonywany jeśli podczas sesji zostanie uruchomiona nowa wersja shella C .logout - wykonywany podczas kończenia sesji

9. Środowisko użytkownika w HP-UX

Zalecanym shellem użytkownika w HP-UX jest shell Korna lub shell Posix.
Podstawowe środowisko użytkownika jest zdefiniowane przez:
- wartości domyślne opcji shella
- wartości domyślne zmiennych shella
- wbudowane synonimy poleceń

Jest ono określone w:
- programie login, który otwiera sesję użytkownika
- shellu obsługującym sesję użytkownika

Administrator może zmodyfikować środowisko użytkownika za pomoca pliku `/etc/profile`.
Użytkownik może zmodyfikować środowisko za pomocą pliku *katalog_osobisty_użytkownika*/.profile

Polecenia do sprawdzenia środowiska użytkownika:

- `set -o`
- `env`
- `alias`
- `umask`

9.1. Przykładowy plik .profile

```
# Ustaw typ terminala
eval `tset -s -Q -h`
# Mozna wpisac na stale
#TERM=vt320
#
# Ustaw ścieżki przeszukiwania katalogó
PATH=/bin:/usr/bin:. ; export PATH
#
# Ustaw zgloszenie gotowosci systemu
PS1='$PWD $' ; export PS1

#
# Ustaw domyslne prawa dostepu nowo tworzonych plikow
umask 022
#
# Przypisz wartosci zmiennym wykorzystywanym przez mechanizm historii polecen
EDITOR=/usr/bin/vi ; export EDITOR
FCEDIT=/usr/bin/vi; export FCEDIT
HISTFILE=$HOME/.sh_history ; export HISTFILE
HISTSIZE=50; export HISTSIZE
```

9.2. Zmienna środowiskowa PATH

Zmienną tą posługuje się shell, gdy wydane zostanie polecenie bez pełnej nazwy ścieżkowej. Określa przeszukiwane przez shell katalogi w momencie wydania polecenia.

Przykład:
```
$ echo $PATH
PATH=/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:.
```

Symbol dwukropka : rozdziela nazwy katalogów.

Oznaczenie katalogu bieżącego:
- . w dowolnym miejscu zamiast nazwy katalogu
- : na początku listy lub na końcu listy

9.3. Poszukiwanie poleceń

Podstawowym poleceniem, które pozwoli znaleźć polecenie w systemie Unix jest polecenie `whereis`.
Składnia:

```
whereis [-bms] polecenie
```

Opcje:

`-b` programy binarne
`-m` strony podręcznika
`-s` kody źródłowe

Przykład:

```
$ whereis vi
vi :  /usr/bin/vi  /usr/man/man1.Z/vi.1
```

Polecenie whereis przeszukuje tylko *określone* katalogi (patrz opis polecenia w dokumentacji)

Inne polecenia:

`which, find, whence (whence –v, type)`

9.4. Ustawianie symbolu zgłoszenia gotowości systemu

Domyślnie symbol zgłoszenia gotowości systemu określany jest przez shell.
W shellu ksh można go zmienić przypisując nową wartość zmiennej PS1.

Przykłady:

```
PS1='[!]$'
PS1='$PWD $'
PS1='[!]$PWD $'
PS1='$(hostname) $PWD >'
```

10 Mechanizmy w shellu

10.1. Generowanie nazw plików

Metaznaki używane do generowania nazw plików

*	0 lub dowolnie wiele dowolnych znaków
?	jeden dowolny znak
[***lista***]	dowolny znak z listy
[!***lista***]	dowolny znak nie z listy
~	katalog osobisty
~***nazwa***	katalog osobisty użytkownika ***nazwa***

- **Metaznaki nie zastąpią kropki rozpoczynającej nazwę**. Musi być ona podana jawnie.

Lista znaków może być podana z użyciem symbolu myślnika, na przykład a-c

- Generowanie nazw plików odbywa się zanim polecenie zostanie przesłane do wykonania.
- Jeśli shell nie znajdzie pliku zgodnego ze wzorcem, do polecenia przekazywany jest wzorzec.

Przykłady:

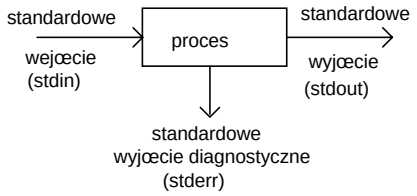
`cd ~user10` przejdź do katalogu osobistego użytkownika `user10`; jeśli katalogiem osobitym tego użytkownika jest `/home/user10`, to polecenie jest równoważne `cd /home/user10`

`ll ~user10/programy` wyświetl zawartość katalogu `/home/user10/programy`

Użycie w poleceniu wyrażenia spowoduje dopasowanie:

Wyrażenie	Dopasowanie
*	wszystkie nazwy w bieżącym katalogu (oprócz rozpoczynających się od kropki)
prog	wszystkie nazwy zawierające "prog"
*.c	wszystkie nazwy kończące się na ".c"
[ab]*	wszystkie nazwy rozpoczynające się od "a" lub "b"
[A-Z]*	wszystkie nazwy rozpoczynające się od dużych liter alfabetu
[testuj,proba]*	wszystkie nazwy rozpoczynające się od "testuj" lub "proba"
[!0-9]*	wszystkie nazwy nie rozpoczynające się od cyfry
/usr/bin/?	wszystkie jednoznakowe nazwy w katalogu /usr/bin

10.2 Przeadresowywanie wejścia - wyjścia



Dane w Unixie nie mają narzuconej struktury i są uważane za strumień znaków.
Każde polecenie może mieć standardowo przyporządkowane trzy strumienie:

- wejścia (*input*) czytanie danych
- wyjścia (*output*) wyprowadzanie wyników
- diagnostyczny (*error*) wyprowadzanie komunikatów o błędach

Są one domyślnie przyporządkowane terminalowi.
Przyporządkowanie to może być zmienione za pomocą odpowiedniego metaznaku.

Metaznaki przeadresowania (*redirection*)

<	czytaj z pliku
>	prześlij wyjście standardowe do *pliku_wy,* jeżeli *plik_wy* nie istnieje zostanie utworzony; jeżeli *plik_wy* istnieje, jego poprzednia zawartość zostanie usunięta
2>	prześlij wyjście diagnostyczne do *pliku_wy*; jeżeli *plik_wy* nie istnieje zostanie utworzony; jeżeli *plik_wy* istnieje, jego poprzednia zawartość zostanie usunięta
>>	dołącz wyjście standardowe do *pliku_wy*; jeżeli *plik_wy* nie istnieje zostanie utworzony
2>>	dołącz wyjściediagnostyczne do *pliku_wy*; jeżeli *plik_wy* nie istnieje zostanie utworzony;

Po metaznaku należy podać nazwę pliku, który ma zastąpić standardowe urządzenie.

Przykład:

```
$ date  >  wykaz
$ ls  >  wykaz          (zastąpi poprzednią zawartość pliku)
$ date >> wykaz         (dopisze wynik polecenia date do pliku)
$ mail  user1  <  list.23.02.06
```

Rozdzielanie wyjścia standardowego i diagnostycznego:

W jednym poleceniu można przeadresowywać obydwa wyjścia, rozdzielając w ten sposób strumienie.

```
$ program   > wyniki   2> bledy
```

Opcja noclobber i mechanizm przeadresowywania

Za pomocą tej opcji można zabezpieczyć się przed przypadkowym usunięciem plików podczas przeadresowywania. Opcja ta zmienia działanie metaznaków przeadresowywania wejścia - wyjścia.

>	prześlij wyjście standardowe do *pliku_wy,* jeżeli *plik_wy* nie istnieje zostanie utworzony; **jeżeli *plik_wy* istnieje, polecenie NIE zostanie wykonane**
2>	prześlij wyjście standardowe i wyjście diagnostyczne do *pliku_wy*; jeżeli *plik_wy* nie istnieje zostanie utworzony; **jeżeli *plik_wy* istnieje, polecenie NIE zostanie wykonane**
>>	dołącz wyjście standardowe do *pliku_wy*; **jeżeli *plik_wy* nie istnieje polenie NIE zostanie wykonane**
2>>	dołącz wyjście standardowe i diagnostyczne do *pliku_wy*; **jeżeli *plik_wy* nie istnieje polecenie NIE zostanie wykonane**

Sprawdzenie ustawienia opcji noclobber

```
$ set -o
```

Jeśli w opisie opcji występuje:

`on` - opcja jest włączona

`off` - opcja jest wyłączona

Ustawienie opcji noclobber

```
$ set -o noclobber
```

Przykład działania po ustawieniu opcji noclobber:

```
$ date > wykaz1
$ ls > wykaz1
wykaz1: File exists              noclobber nie pozwala zmienić zawartości pliku
```

Ignorowanie opcji noclobber

Można ignorować ustawienie opcji noclobber. Służy do tego symbol |

>|	prześlij wyjście standardowe do *pliku_wy,* jeżeli *plik_wy* nie istnieje zostanie utworzony; jeżeli *plik_wy* istnieje, jego poprzednia zawartość zostanie usunięta; przykład: ls >! wykaz
2>|	prześlij wyjście standardowe i wyjście diagnostyczne do *pliku_wy*; jeżeli *plik_wy* nie istnieje zostanie utworzony; jeżeli *plik_wy* istnieje, jego poprzednia zawartość zostanie usunięta
>>|	dołącz wyjście standardowe do *pliku_wy*; jeżeli *plik_wy* nie istnieje zostanie utworzony;
2>>|	dołącz wyjście standardowe i diagnostyczne do *pliku_wy*; jeżeli *plik_wy* nie istnieje zostanie utworzony;

Przykład:

```
$ date > wykaz1
$ ls > wykaz1
wykaz1: File exists     noclobber nie pozwala zmienić zawartości pliku
$ ls >| wykaz1          wymuszamy zapisanie do pliku przy ustawionej opcji noclobber
```

10.3 Podstawianie poleceń *(command substitution)*

Wyjście polecenia może być podstawione jako argument innego polecenia.

Składnia:

- w znakach akcentu accent grave ` `$ echo Dzisiaj mamy `date``
- z użyciem nawiasu `$ echo Dzisiaj mamy $(date)`

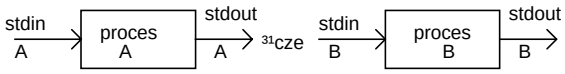
Uwaga: w shellu Bourne'a rozpoznawane są tylko znaki akcentu.

Przykład:

```
$ kat_biezacy=`pwd`
$ echo $kat_biezacy
/users/user1

$ set x =$(ls)
$ echo $x
plik1 plik2 plik3
```

10.4 Potoki (*pipelines*)



- Wyjście jednego polecenia może być skierowane na wejście innego polecenia.
- Utworzony w ten sposób ciąg poleceń nosi nazwę *potoku*.
- Symbolem łączącym jest znak pionowej kreski |

Zamiast pisać:

```
ls -l > wykaz
lp wykaz              można        ls -l | lp
rm wykaz
```

Przykład:

```
$ who | wc -l
$ echo   W systemie pracuje  $(who | wc -l)  uzytkownikow
$ echo   W systemie pracuje  `who | wc -l`  uzytkownikow
$ alias  who=who|sort
```

10.5. Mechanizm cytowania (quoting mechanism)

Shell wiele znaków uważa za znaki specjalne i dokonuje ich interpretacji, przed przesłaniem polecenia do wykonania. Aby zapobiec interpretacji znaków specjalnych trzeba posłużyć się mechanizmem cytowania.

Jak działa mechanizm cytowania?

- \ lewy ukośnik (*backslash*) usuwa specjalne znaczenie wszystkich znaków
- ' apostrof usuwa specjalne znaczenie wszystkich znaków
- " cudzysłów usuwa specjalne znaczenie wszystkich znaków oprócz:
  \ " $ `
  (wtedy aby usunąć znaczenie ", $ lub ` można posłużyć się \ )

Przykład:

```
echo 'cena w $'
echo "Data: $(date)"
stty erase ^\?        (kasowanie klawiszem Delete)
```

10.6. Synonimy (*ang. aliases*)

Dowolne polecenie można zastąpić jego synonimem. Służy do tego polecenie:

```
alias synonim=polecenie
```

Przykład:

```
$ alias  dir=ls
$ alias rm='rm -i'
$ alias  ls='ls -l'
$ alias  kto='
> date
> echo Sesje prowadza:
> who | sort'
```

Typy synonimów
- wbudowane w shell
- definiowane przez użytkownika
- ścieżkowe (*ang. tracked*)
- eksportowane

Jak uzyskać informację o synonimach?

```
alias
```

Jak usunąć synonim?

```
unalias nazwa_synonimu
```

Jak wykonać polecenie w wersji oryginalnej?

Poprzedzić synonim:
- znakiem \ , jeśli polecenie jest zewnętrzne
- znakami "" , jeśli polecenie jest wbudowane

Przykład:

```
$ alias rm =  'rm -i'
$ rm test
rm: remove test? n          przed usunięciem pyta, bo alias dla rm -i
$ \rm test                  wykonuje standardowe polecenie rm
$
```

10.7. Uzupełnianie nazw plików (*filename completion*)

Sposób uzupełniania nazw plików zależy od shella. Podany poniżej mechanizm dotyczy shella Korna.

Zacząłeś wydawać polecenie i nie pamiętasz nazwy pliku:

ESC ESC shell uzupełni nazwę pliku do miejsca, które jest jednoznaczne
ESC= wyświetli listę wszystkich plików o nazwach rozpoczynających się od podanego tekstu

Przykład 1.
W katalogu znajduje się tylko jeden plik o nazwie rozpoczynającej się od p :

```
$ rm -i pESC ESC
```

Shell uzupełni polecenie do postaci:

```
rm -i proba _
```

Przykład 2.
W katalogu znajdują się pliki o nazwach `test1`, `test2` i `test3` :

```
$ cat tESC ESC
test1 test2 test3 test4
$ cat test_                    shell stwierdził niejednoznaczność nazwy
```

Możemy teraz sprawdzić jakie pliki są w katalogu:

```
ESC=
```

1. test1
2. test2
3. test3

i dopisać odpowiednią końcówkę.

10.8. Historia poleceń – czyli przywoływanie, edycja i powtarzanie uprzednio wydanych poleceń

- Shell pamięta uprzednio wprowadzone polecenia. Można je przywołać i powtórnie uruchomić w postaci takiej jak uprzednia lub po dokonaniu odpowiednich modyfikacji.
- Sposób korzystania z mechanizmu historii poleceń zależy od używanego shella.

11 Wybrane polecenia działania na plikach tekstowych

wc	zliczanie wierszy, słow, znaków
sort	sortowanie
tr	zamiana znaków na inne
cut	wyci nanie kolumn z pliku
paste	łączenie rozdzielonych kolumn w pojedynczy plik
join	łączenie wierszy z kilku plików w oparciu o wspólne pole
nl	wyświetlanie tekstu z numerowaniem wierszy
uniq	usuwanie identycznych wiersze
od	wyświetlanie pliku w ósemkowo, dziesiętnie, szestnastkowo i znakowo
split	rozdzielanie plików na mniejsze

Większość tych poleceń ma zastosowanie jako filtry w potokach.
Polecenia te mogą mieć różne zestawy opcji w różnych wersjach Unixa. W materiałach podano podstawowe opcje występujące w większości wersji Unixa.

11.1. Zliczanie wierszy, słów i znaków w pliku - wc (word count)

Składnia:
```
wc [-cwl] [nazwa_pliku ...]
```

gdzie:
`-c` licz znaki (*characters*)
`-w` licz słowa (*words*); słowo jest to ciąg znaków rozdzielonych spacją, TAB, NEWLINE
`-l` licz wiersze (*lines*)

Przykład:

```
$ wc   plik1
23  207  1123  plik1

$ wc -c *
1123  plik1
12    list
1135  total
```

11.2. Sortowanie pliku tekstowego - sort

Składnia:
```
sort [opcje] [plik_we ...]
```

Domyślnie: sortuje znakowo, wg kodu ASCII, rosnąco, wg całych wierszy.
Wiersz jest dzielony na pola rozdzielane spacją lub tabulacją.

Grupy opcji

Opcje określające klucz sortowania:
`-t`*znak* gdzie *znak* określa separator pól
`-k`*n* gdzie *n* określa numer pola (numeracja w tym wypadku rozpoczyna się od 1)

Opcje sposobu przetwarzania plików:
`-o plik_wy` przydatna, jeśli plik posortowany ma zastąpić plik nie posortowany
`-m` nie sortuj ale łącz posortowane pliki
`-u` pomijanie w pliku wynikowym powtórzonych wierszy

Opcje kolejności sortowania:
domyślnie sortowanie leksykograficzne (ASCII)
`-d` słownikowe (pominięcie znaków, które nie są literami, cyframi, odstępami)
`-n` sortowanie numeryczne

11.3. Zamiana zestawu znaków - tr (translate characters)

Polecenie kopiuje standardowe wejście na standardowe wyjście dokonując zamiany lub usunięcia znaków.

Składnia:

```
tr [-ds] [napis_1 [napis_2]]
```

gdzie:

napis_1	lista znaków poszukiwanych w strumieniu wejściowym
napis_2	lista znaków, którymi należy zastąpić znalezione znaki

Opcje:

d	usuń znaki zawarte w napis_1 (*delete*)
s	usuń powtórzenia znaków (*squeeze*)

Przykład:

```
$ cat wiadomosc
Dzisiaj ostatni dzien kursu.
$ tr "abcdef" "fedcba" <wiadomosc  >wiadomosc_zakodowana
$ cat wiadomosc
Dzisifj ostftni czibn kursu.

$ tr -d "aeiou" <plik1  >plik2

$ pwd | tr '/' ' '  | wc -w

$ who | tr -s ' '
```

11.4. Podział pliku na części - cut i paste

Polecenie `cut` widzi plik w postaci tabeli o określonej liczbie wierszy i kolumn. Umożliwia wycięcie tych pól i kolumn, które są potrzebne.

Polecenie `paste` również widzi plik w postaci tabeli i pozwala scalać w poziome wiersze.

Składnia:

```
$ cut -flista [opcje] [nazwa_pliku ...]
$ cut -clista [nazwa_pliku ...]

$ paste nazwa_pliku  nazwa_pliku ...
$ paste -dlista  nazwa_pliku nazwa_pliku ...
```

gdzie:

-flista	wycinaj pola wg listy
-clista	wycinaj kolumny wg listy
-dc	zmień standardowy znak rozdzielający pola (TAB) na znak c

Pola mogą być określone jako:

```
-f1-3                       -f-3-
-f1,2,3                     -f1,2-5,8
```

Przykład:

Wytnij z pliku /etc/passwd identyfikator oraz informacje o użytkowniku

```
cut -d: -f1,5  /etc/passwd
```

11.5. Usuwanie identycznych wierszy -uniq

Składnia:
```
uniq [-cdu +|-n] [nazwa_pliku_we [nazwa_pliku_wy]]
```

Opcje:

`-c`	poprzedź wiersz informacją o ilości identycznych wierszy
`-d`	wyświetl po jednym wierszu, zignoruj wiersze identyczne
`-u`	wyświetl tylko wiersze, które się nie powtarzają
`-n`	zignoruj pierwsze n pól (rozdzielane spacją lub tab)
`+n`	zignoruj pierwsze n znaków

Warunek działania: plik posortowany

Przykład:

```
sort wykaz | uniq -c
who | cut -f1 -d' ' | sort | uniq -c
```

11. 6. Łączenie plików poprzez wspólne pole - join

Składnia:
```
join [opcje] plik1 plik2
```

Jest to implementacja instrukcji join relacyjnej bazy danych.

Przykład:

```
$ cat osoby
a Bury
c Kowalski
d Nowak
$ cat zakłady
a ZTSW
b LKSW
c ZBO
d ZTS
$ join osoby zaklady
a Bury ZTSW
c Kowalski ZBO
d Nowak ZTS
```

11. 7. Numerowanie wierszy w pliku - nl

Składnia:
```
nl  [opcje] nazwa_pliku
```

Opcje:
`-v` *numer_początkowy*
`-i` *co_ile*

Przykład:

```
$ nl -v100 -i10 osoby
100 a Bury
110 c Kowalski
120 d Nowak
```

11.8. Dzielenie pliku na części - split

Składnia:
```
split [-l liczba] [nazwa_pliku_we [nazwa_wy]]
```

gdzie:
`-liczba` po ile wierszy dzielić

Pliki wyjściowe otrzymują rozszerzenie aa, ab itd.

```
$ split -l100 plik_we plik_wy
$ ll plik*
plik_we
plik_wyaa
plik_wyab
plik_wyac
```

11.9. grep - Global Regular Expression Printer

Polecenie grep wyszukuje wiersze zawierające podany wzorzec.
Wzorzec jest zdefiniowany za pomocą **wyrażenia regularnego** (***regular expression***).

Składnia:
```
grep [-opcje] 'wyrażenie regularne' [nazwa_pliku]
```

Opcje

`-c`	wyświetl tylko liczbę wierszy odpowiadających wzorcowi
`-v`	wyświetl wiersze, które nie odpowiadają wzorcowi
`-l`	wyświetl nazwy plików, które zawierają wiersze z wzorcami
`-i`	ignoruj wielkość liter
`-n`	poprzedź każdy wiersz jego numerem w pliku
`-s`	nie wyświetlaj informacji o błędach

Rodzina poleceń wyszukiwania

`grep`	we wzorcu można umieścić podzbiór znaków używanych w wyrażeniach regularnych, szuka względnie szybko
`egrep`	we wzorcu można umieścić każdy ze znaków używanych w wyrażeniach regularnych, szuka wolno (*extended grep*)
`fgrep`	szybkie poszukiwanie wg stałego tekstu (*fixed-string grep*)

O czym należy pamiętać:
- wybór polecenia zależy od wzorca użytego do poszukiwania
- istnieje wiele znaków (np. kropka), które mają znaczenie specjalne w wyrażeniu regularnym, chcąc je użyć we wzorcu należy je poprzedzić znakiem \

Przykłady:

```
fregp janusz /etc/passwd

grep  '^user'  /etc/passwd
grep  '/bin/sh$'  /etc/passwd
grep  '^user[1-5]:'  /etc/passwd
```

Znaki specjalne w wyrażeniach regularnych polecenia grep

`^`	dopasuj do początku wiersza
`$`	dopasuj do końca wiersza
`[lista]`	dopasuj do jednego ze znaków listy w nawiasach
`[^lista]`	dopasuj bez dowolnego znaku z listy w nawiasach
`.`	dopasuj do dowolnego pojedynczego znaku
`*`	powtórz jedno-znakowe wyrażenie regularne umieszczone przed znakiem gwiazdki 0 lub więcej razy
`.*`	powtórz dowolny znak 0 lub więcej razy

Przykłady:

```
grep  '.S'  wykaz
grep  '63-.....2'  wykaz
grep  '^[ST]'  wykaz
grep  '[2-5]$'  wykaz
grep '^S.* C' wykaz
```

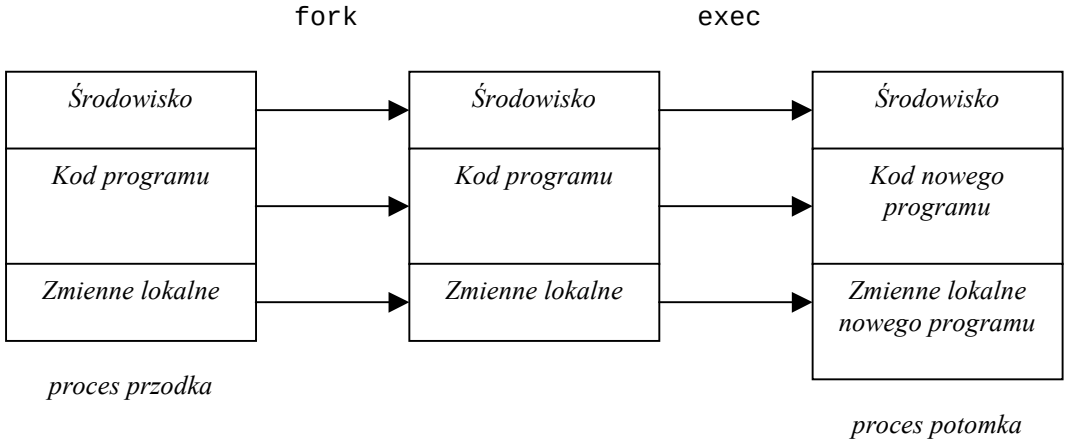
12. Nadzorowanie wykonywania poleceń

12.1. Wykonywanie polecenia zewnętrznego

Proces - każdy program razem ze swoim środowiskiem

Polecenie wykonywane jest dwuetapowo:

- tworzony jest nowy proces (**proces potomka**) , który ma wykonać polecenie; jest to kopia bieżącego procesu; mechanizm ten nosi nazwę **rozwidlania** (ang. ***fork, forking a process***)
- nowy proces jest jest przekształcany w proces wykonujący polecenie; mechanizm nosi nazwę **wykonywania** (ang. ***execing a command***)
- przenoszone środowisko obejmuje informację o właścicielu i grupie, o standardowych strumieniach, do których przyłączony jest terminal, otwartych plikach, bieżącym katalogu i przenoszonych zmiennych środowiskowych



Procesy

- Proces jest to jednostka wykonywania w systemie Unix.
- Zawiera instrukcje programu i jego środowisko.
- Jądro identyfikuje proces za pomocą numeu nazywanego identyfikatorem procesu PID (*process identifier*).
- Zinicjowanie wykonania procesu jest zawsze wynikiem działania innego procesu.
- Proces, który utworzył dany proces jest dla niego procesem *macierzystym* lub procesem *przodkiem*.
- Identyfikator procesu przodka danego procesu oznaczany jest w skrócie PPID (*parent process identifier*).
- Przodkiem wszystkich procesów jest proces `init`, którego PID jest równy 1.
- Każdy proces ma jeden proces macierzysty (oprócz procesu init), lecz może mieć wiele procesów potomnych.
- W momencie rozpoczęcia sesji jądro przydziela użytkownikowi proces interpretatora poleceń - *login shell*.
- Proces systemowy, który jest wykonywany w tle – nie ma żadnego przydzielonego terminala, ani nie jest związany z żadnym shellem nazywany jest *demonem*.

12.2. Podglądanie procesów - polecenie ps (process status)

Składnia:

```
ps [opcje] [-t terminal] [num]
```

Polecenie `ps` bez żadnej opcji pozwala obejrzeć skróconą informacje o procesach mających efektywny ID użytkownika wydającego polecenie i dołączonych do terminala, z którego wydawane jest polecenie.

Opcje pozwalają uzyskać
- informację o różnych atrybutach procesów
- informację o innych procesach

Zestaw opcji zależy od wersji systemu.

Wybrane opcje:

`-f`	pełna informacja o procesach
`-u` *`nazwa_użytkownika`*	informacja ukierunkowana na użytkownika o określonej nazwie lub UID
`-e`	lista wszystkich procesów w systemie
`-t`	informacja o procesach związanych z danym terminalem

Przykład 1

```
$ ps
PID         TTY         TIME        COMMAND
128         ttypa       0:00        ksh
135         ttypa       0:00        ps
```

gdzie:

PID	identyfikator procesu
TTY	nazwa terminala związanego z procesem
TIME	czas wykonywania procesu (minuty:sekundy)
COMMAND	nazwa polecenia

Przykład 2

```
$ ps -ef

UID    PID   PPID    C       STIME      TTY     TIME   COMMAND
root   1     0       0       Dec 27     ?       0:00   init
root   706   1       0       Dec 27     ?       0:01   /usr/sbin/inetd
root   8896  706     0       15:20:40   ttypa   0:00   telnetd
user5  8897  8896    6       15:20:41   ttypa   0:00   -ksh
```

W przykładzie pokazano jedynie fargment długiel listy wszystkich procesów.
Poszczególne wartości oznaczają:

UID	identyfikator użytkownika
PID	identyfikator procesu
PPID	identyfikator procesu przodka
C	parametr określający wykorystanie procesora
STIME	czas uruchomienia procesu
TTY	nazwa terminala związanego z procesem
TIME	czas wykonywania procesu (minuty:sekundy)
COMMAND	nazwa polecenia związanego z procesem

12.3. Kończenie wykonywania procesów

Wykonywanie procesu można przerwać wysyłając do niego odpowiedni sygnał. Do wysyłania sygnałów do procesu służy polecenie `kill`.

Składnia:

```
kill [-s nazwa sygnału | -numer sygnału ] PID [PID...]
```

gdzie:PID oznacza identyfikator procesu

- Zestaw rozpoznawanych sygnałów zależy od wersji Unixa.
- Polecenie `kill -l` wyświetla obowiązujący zestaw.
- Polecenie `kill` bez podania numeru/nazwy przerwania przyjmuje przerwanie numer 15 (SIGTERM) - software termination signal. Przerywa ono proces tylko w okreslonych warunkach (może być przechwycone, ignorowane, lub blokowane przez proces, do którego zostało wysłane).
- Sygnałem, którego nie można przechwycić, zignorować ani zablokować jest sygnał o numerze 9 (SIGKILL).

Przykład:

```
$ps
PID  TTY       TIME COMMAND
8923 ttypa     0:00 ps
8897 ttypa     0:00 ksh
8896 ttypa     0:00 telnetd
$ kill 8897
$ ps                                  # shell ksh ignoruje sygnał 15
PID  TTY       TIME COMMAND
8923 ttypa     0:00 ps
8897 ttypa     0:00 ksh
8896 ttypa     0:00 telnetd
$ kill -9 9987                        # zawsze zostanie wykonane
połączenie zostaje zamknięte
```

12.4. Dwa sposoby wykonywania poleceń

- synchronicznie (przetwarzanie w pierwszym planie - *foreground*)

Shell (proces przodka):

1. Shell zgłasza gotowość przyjęcia polecenia.
2. Znajduje/ładuje/przekazuje do wykonania polecenie.
3. Przechodzi w stan spoczynku, czekając na zakończenie polecenia. Nadzór przekazywany jest do procesu polecenia.

Proces zwiazany z poleceniem (proces potomka):

4. Proces polecenia się wykonuje.
5. Kończąc się budzi shell.
6. Zakończony proces znika.

Shell (proces przodka):

7. Shell przejmuje nadzór i wyświetla znak gotowości przyjęcia nowego polecenia.

- asynchronicznie (przetwarzanie w drugim planie - *background*)

Shell (proces przodka):

1. Shell zgłasza gotowość przyjęcia polecenia.
2. Znajduje/ładuje/ przekazuje do wykonania polecenie.
3. Shell wyświetla znak gotowości przyjęcia nowego polecenia.

Proces zwiazany z poleceniem (proces potomka):

1. Proces polecenia się wykonuje jako niezależny proces.
2. Zakończony proces znika.

12.4. Przetwarzanie w pierwszym i drugim planie w shellu

Aby można było wykonywać polecenia w pierwszym i drugim planie shell wprowadza dodatkową w stosunku do procesu jednostkę wykonywania: **zadanie** (***job***).
Każdy wiersz polecenia tworzy zadanie (**job**).
Wiersz polecenia może być:

- poleceniem prostym, na przykład `ls -l`
- poleceniem złożonym z kilku poleceń połączonych w potok, na przykład: `ls -s | sort -n | head -5`
- zestawem poleceń rozdzielonych średnikiem, na przykład `cd drzewo; du >wykorzystanie`

Zadanie może być wykonywane jako

- **pierwszoplanowe** (***foreground***) shell czeka na zakończenie zadania
- **drugoplanowe** (***background***) shell zleca wykonanie polecenia i wyprowadza zgłoszenie gotowości przyjęcia następnego polecenia

Uruchamianie zadania

- pierwszoplanowe domyślnie każde zadanie
- drugoplanowe na końcu wiersza polecenia umieszczany jest znak &

Przykład:

`$ sleep 600 &`	zlecenie zadania
`[1] 1234`	informacja o przyjęciu zadania, otrzymało ono numer 1; związany z nim proces numer 1234
`$`	można wydać nowe polecenie

Po pewnym czasie na ekranie pojawi się komunikat o zakończeniu zadania:

```
[1] + Done        sleep 600
```

O czym należy pamiętać
Polecenie wykonania zadania w drugim planie nie przeadresowuje automatycznie standardowych we/wy. Należy to zrobić samemu.

Przykład:

```
find  /bin /etc -name passwd -print > lista  2> /dev/null &
```

Lista uruchomionych zadań

Składania:

```
jobs
```

Obok numeru zadania pojawi się znak:

+	**zadanie bieżące** (***current job***) czyli ostatnie zatrzymane zadanie lub gdy nie ma zadań zatrzymanych ostatnie uruchomione zadanie w drugim planie
-	**zadanie poprzednie**

Zadanie może być:

	pierwszoplanowe	drugoplanowe
zatrzymywane	`CTRL-Z`	`stop [%zadanie]`
usuwane ("zabijane")	`kill nr_procesu`	`kill nr_procesu` `kill [%zadanie]`
wznawiane w pierwszym planie	`fg [%zadanie]` * `%zadanie`	`fg [%zadanie]` `%zadanie`
wznawiane w drugim planie	`bg [%zadanie]` * `%zadanie&`	`bg [%zadanie]` `%zadanie&`

* dotyczy zadań zatrzymanych

Odwołanie do zadania

`%`	lub %+ zadanie bieżące
`% -`	zadanie poprzednie
`%j`	gdzie j jest numerem zadania

Różne

- Można zlecić shellowi, aby czekał na zakończenie wszystkich procesów rozpoczętych za pomocą & lub bg. Służy do tego polecenie wait.
- Jeżeli po wydaniu polecenia `exit` istnieją zatrzymane zadania, wyświetlony zostanie komunikat:
  ```
  There are running jobs.
  ```
  oznacza to, ze nie zostały jeszcze zakończone procesy uruchomione w tle.
- Można wtedy
  - zakończyć zadania
  - ponownie wydać polecenie `exit` - procesy automatycznie zostaną zakończone.
- W shellu ksh można zadanie uruchamiane w drugim planie poprzedzić poleceniem nohup. Wtedy nie zakończone prcesy będą wykonywane również po zakończeniu sesji. Są one wtedy adoptowane przez proces `init`.

Jednorazowe wykonanie polecenia o określonej godzinie - polecenie at

- Polecenie to czyta polecenia ze standardowego wejścia i umieszcza je w kolejce do wykonuje je o określonej godzinie.
- Wyjścia standardowe i diagnostyczne są wysyłane pocztą (mail) do użytkownika.
- Obsługa polecenia at jest możliwa tylko wtedy, kiedy uruchomiony jest odpowiedni demon, obsługujący kolejkę zleceń.
- Zlecanie poleceń do wykonania o określonej godzinie jest możliwe tylko dla tych użytkowników, którym na to zezwoli administrator.

Składnia:

```
at godzina [data] [+ przyrost][skrypt]
```

gdzie:

godzina: 1,2 cyfry - godzina
4 cyfry - godzina i minuty (1234 i 12:34 równoważne)
domyślnie czas 24-godzinny
nazwy specjalne: midnight, noon, now, next

data: miesiąc dzień [,rok] (Jan 24)
dzień_tygodnia (Friday)
nazwy specjalne: today, tomorrow

przyrost:+ liczba { określenie jednostki }
jednostki: minutes, hours, days, weeks, months, years

Przykład 1:

```
$ at  9:20
at> date
at> /usr/bin/banner "Przerwa" > /dev/ttyp5
at>CTRL-D
job  8914  at  Wed Mar  17  9:20:00   1998
```

Przykład 2:

```
at now + 5 hours ...
at 23:00 Friday next week ...
```

Polecenia pomocnicze:

- polecenie wyświetlające kolejkę zadań oczekujących do realizacji
- polecenie usuwające określone zadanie z kolejki

W zależności od wersji systemu Unix może to być polecenie at z odpowiednimi opcjami lub inne polecenie.

Cykliczne wykonywanie poleceń określonego dnia, o określonej godzinie

- Jeśli w systemie zostanie uruchomy odpowiedni demon (cron), to możliwe jest przekazywanie do wykonania zleceń cyklicznych. Zlecenia te umieszczane są w pliku zleceń.
- Każdy użytkownik ma jeden plik zleceń.
- Administrator określa, którzy użytkownicy mogą zlecać zadania do wykonania demonowi cron.

Format pliku zleceń

Każdy wiersz pliku zleceń składa się z sześciu pól.
Pierwszych pięć pól zawiera informacje o tym kiedy polecenie ma być wykonywane, w kolejności:

- minuty (0-59)
- godziny (0-23)
- dni miesiąca (1-31)
- miesiące (1-12)
- dni tygodnia (0-6, przy czym 0 oznacza niedzielę)

Ostatnie, szóste pole zawiera polecenie do wykonania.
Pola oddzielan są znakami spacji lub tabulacji.
"*" zastępuje dowolną dopuszczalną wartość.
Liczby podawane w jednym polu są rozdzielane przecinkami.
Znak "-" jest używany do wskazania zakresu wartości dopuszczalnych.

Przykład:

```
5,35  9     *     *     1-5   /bin/who > kto_jest
30    15    *     *     5     /bin/lp     zestawienie_tygodniowe
```

Przygotowanie pliku zleceń i przekazanie do wykonania

- Plik zleceń można przygotować za pomocą edytora i następnie przekazać go do wykonania za pomocą polecenia:

  ```
  $ crontab plik_zlecen
  ```

- Można również otworzyć plik do edycji za pomocą polecenia:
  ```
  $ crontab –e
  ```

  Uruchamiany jest wtedy automatycznie edytor wskazany przez zmienną EDITOR, np. edytor vi. Zakończenie przygotowania pliku równoważne jest z przesłaniem go do wykonywania.

Sprawdzanie zleceń zaplanowanych do wykonania

```
$ crontab -l
```

Usuwanie zleceń zaplanowanych do wykonania

```
$ crontab –r
```

13 Archiwizacja plików

13.1. Informacja o ilości miejsca na dysku

Polecenia, które pozwalają wyświetlić informację o ilości miejsca na dyskach to:

`df` - system SunOS, Linux
`bdf` – system HP-UX

Polecenia te dostarczają informacji zbiorczej, związanej z wykorzystaniem partycji dysków lub i ich logicznych odpowiedników.

Jeśli chcemy uzyskać informację o zajętości dysku przez poszczególne katalogi, możemy się posłużyć poleceniem

`du`

Przykład:

```
$ du –s .   # wyświetl infomację sumaryczną o bieżącym katalogu
129 .
$ du –s /home/user5
225 /home/user5
```

13.2. Poszukiwanie pliku w systemie - polecenie find

Składnia:

`find` *gdzie_rozpocząć* *czego_szukać* *co_zrobić*

Gdzie rozpocząć ?
Wykaz katalogów podany w postaci nazw prostych, nazw względnych lub bezwzględnych.

Czego szukać?
Kryteria poszukiwań mogą dotyczyć:

- nazwy, typu pliku
- praw własności
- praw dostępu
- daty ostatniej modyfikacji lub ostatniego korzystania z pliku
- rozmiaru pliku

Co zrobić, gdy plik zostanie odszukany?
Jeżeli chcesz

- wyświetlić odszukane pliki użyj opcji `–print`; w wielu systemach Unix jest to działanie domyślne polecenia `find`
- użyć bardziej złożonej czynności, użyj opcji `-exec`

Przykład

```
find  .  –print  # Wyświetlone zostaną wszystkie pliki począwszy od katalogu bieżącego.
                   Pominięty został parametr czego_szukać
```

`find . -print`	wyświetli nazwy względne katalogów (plików)
`find ~ -print`	wyświetli nazwy bewzględne
`find /home/user5 -print`	wyświetla nazwy bezwzględne

Polecenie find – niektóre kryteria wyboru pliku

Wg nazwy lub typu pliku

`-name` *`nazwa_pliku`* znaleźć wszystkie pliki *nazwa pliku*
`-type typ_pliku` znaleźć wszystkie pliki typu *typ_pliku*

gdzie typ pliku:

`f`	zwykły plik
`d`	katalog
`c`	plik urządzenia znakowego
`b`	plik urządzenia blokowego

Przykłady:

```
find /home -name plik1 -print
find /home -name  "plik*" -print
find /home -type d -print
find /home -type d -name "usera*" -print
find /home -type d -name "usera*" -print > /dev/ttya  >& /dev/null
```

Wg właściciela i uprawnień

`-user` *`nazwa_użytkownika`*	nazwa użytkownika lub jego ID
`-group` *`nazwa_grupy`*	nazwa grupy lub jej ID
`-perm` *`uprawnienia`*	określone prawa dostępu, ósemkowo

Przykład:

```
find /bin -user bin -group bin -perm 555 -print
```

Wg czasu

`-atime` *`n`*	czas ostatniego dostępu - czytanie pliku (***last access time***)
`-mtime` *`n`*	czas ostatniej modyfikacji - zmiana zawartości (***modification time***)
`-ctime` *`n`*	czas ostatnie zmiany statusu pliku - zmiana uprawnień, właściciela (***status change time***)

gdzie `n` określna liczbę dni i może być poprzedzone znakiem

`+`	więcej niż
`-`	mniej niż

Przykład:

```
find /bin -atime +90 -print
```

Wg rozmiaru

`-size n` `w blokach (512 bajtów)` (dla urządzeń znakowych w znakach)

Przykład:

```
find /home -size +1000 -print
```

13.3. Archiwizacja plików - tar

Program tar (tape archiver) tworzy kopię danych, plik po pliku, zachowując hierarchiczną strukturę katalogów, tak, aby można było odzyskać pliki wraz z ich ścieżkami.

Składnia:

```
tar fukcja [modyfikatory] [lista_plików]
```

Funkcja

c twórz nowe archiwum (***create***)
u zaktualizuj wg podanej listy plików (***update***) ; uwaga: pliki nie są usuwane, nowa wersja jest dodawana na końcu
r dodaj na końcu archiwum (***write***) (nie działa z taśmą 1/4")
x wybierz pliki z archiwum i zapisz w systemie plików UNIXa (***extract***); pliki zostaną zapisane zgodnie z tym, jak zostały wpisane do archiwum:
- ścieżka pełna - dokładnie w tym samym miejscu
- ścieżka względna - względem bieżącego katalogu
- nazwa prosta - w bieżącym katalogu

t wyświetl zawartość archiwum (***table of contents)***

Modyfikatory

f (***filename***) kolejny argument będzie nazwą pliku (zwykłego lub specjalnego) ,
v (***verbose***) informuj o przebiegu archiwizacji lub wyświetlaj więcej informacji o plikach

Przykłady

```
tar cf model.tar  ~/*.f  utwórz archiwum wszystkich plików *.f  z katalogu osobistego
                         i umieść w pliku model
tar tf model.tar         wyświetl całą zawartość archiwum model
tar xvf model.tar        odtwórz całą zawartość archiwum
```

13.4. Archiwizacja – cpio (*CoPy files In from and Out to an archive*)

Można za pomocą cpio, nie można za pomocą tar (lub jest to trudne):
- archiwum na wielu taśmach
- zapis przyrostowy - w połączeniu z poleceniem find można archiwizować tylko pliki, które powstały lub zmieniły się w ciągu podanego okresu czasu
- można archiwizować pliki specjalne (urządzeń znakowych i blokowych)

Można za pomocą tar, nie można za pomocą cpio (lub jest to trudne):
- dodawać pliki na koniec archiwum
- aktualizować archiwum (opcja u w tar)

Składnia:

```
cpio -o[opcje]                 czytaj ze standardowego we nazwy plików  i kopiuj do archiwum
cpio -i[opcje] [wzorzec]       czytaj ze standardowego wejścia i wybierz pliki, które odpowiadają wzorcowi
cpio -p[opcje] katalog         czytaj ze standardowego we nazwy plików i kopiuj do wybranego katalogu
```

Wybrane opcje
- v informuj o przebiegu archiwizacji (***verbose***)
- t wyświetl zawartość archiwum (***table of contents***)
- d twórz katalogi, gdy jest to potrzebne (***directory***)

Przykłady:

```
ls   ~/*.f   | cpio -o  > model.cpio
cpio   -i   dane.f   < model.cpio
find . -name '*.f' -print | cpio -pd programy
```

13.5. Kompresja i dekompresja pliku - compress

Składnia:

```
compress  [-v]  nazwa_pliku ...
uncompress   [-v]  nazwa_pliku ...
```

gdzie:

-v wyświetl % redukcji dla każdego pliku

W wyniku kompresji tworzony jest plik o nazwie zakończonej .Z Plik oryginalny jest usuwany.
W wyniku dekompresji, odtwarzany jest plik oryginalny zaś plik *nazwa.Z* jest usuwany.

Przykład:

```
$ compress –v model.tar
model.tar: Compression: 82.28% - replaced with model.tar.Z
```

Inne metody

Dodatkowe programy służące do kompresji i archiowizowania – nie zawsze są dostarczane z firmowym systemem UNIX.
Lokalny administrator instaluje je zwykle w jednym z dwóch katalogów:

```
/usr/local/bin
/usr/contrib/bin
```

Programy GNU-zip:

gzip – kompresja
gunzip – dekompresja

Programy inne

zip tworzenie archiwum z jednoczesnym stosowaniem kompresji
unzip odtwarzanie archiwum

Zaleta: zgodność z programami PKZIP/PKUNZIP używanym jako programy archiwuzujące MS-DOS.

Ćwiczenia

2. Podstawy użytkowania systemu

1. Spróbuj rozpocząć pracę w systemie Unix celowo robiąc błędy:
   - wpisz nieprawidłowy identyfikator (nazwę) użytkownika
   - wpisz nieprawidłowe hasło
   - użyj dużej litery w identyfikatorze
   - wpisz wszysto prawidłowo

2. Zakończ pracę za pomocą polecenia **exit**. W jaki sposób poznasz, że zakończyłeś sesję?

3. Rozpocznij nową sesję. Na jakim komputerze i z jakim systemem operacyjnym pracujesz?

4. Zmień hasło. Pamiętaj o wymaganiach, które musi spełniać.

5. Sprawdź za pomocą polecenia stty -a funkcje przypisane twojej klawiaturze.

6. Za pomocą jakiego klawisza można skasować cały wiersz (fukcja **kill**) ? Sprawdź jak działa ta funkcja.

Zmień przyporządkowanie klawisza funkcji erase na inny klawisz. Sprawdź czy działa. Zakończ pracę za pomocą polecenia: exit i ponownie rozpocznij sesję. Czy ustawione w poprzedniej sesji przyporządkowanie klawisza erase nadal obowiązuje?

3. Wybrane podstawowe polecenia

1. Sprawdź jaki masz identyfikator lizcbowy i do jakiej grupy należysz?

2. Sprawdź ilu użytkowników aktualnie pracuje w systemie. Czy jest wśród nich użytkownik "user5"? Na którym terminalu pracuje?

3. Użyj polecenia finger, aby uzyskac informacje o użytkowniku user3.

4. Sprawdź za pomocą polecenia **mesg**, czy inny użytkownik może przesyłać do ciebie komunikaty.

5. Sprawdź działanie opcji –H, -T polecenia who.

6. Sprawdź do ilu użytkowników możesz pisać komunkaty.

6. Za pomocą polecenia **write** napisz do kolegi jak się nazywasz, gdzie pracujesz i dlaczego uczysz się Unixa. Aby sobie wzajemnie nie przerywać, zaproponuj protokół komunikacji - jakieś słowo oznaczające koniec twojego komunikatu i oczekiwanie na komunikat kolegi.

7. Zabroń za pomocą polecenia **mesg** wyświetlanie komunikatów na twoim terminalu. Poproś kolegę o próbę nawiązania z tobą łączności za pomocą polecenia **write**.

8. Odczytaj aktualną datę i czas.

4. System plików

1. Które z podanych napisów mogą być nazwami katalogów:

Pogramy	RAPORTY
moje dane	list.*
bardzodluganazwakatalogu	/

   Sprawdź za pomocą polecenia mkdir co uzyskasz, gdy użyjesz każdego z napisów jako nazwy katalogu.

2. Utwórz w katalogu osobistym (domowym) katalog **smieci.** Uczyń go katalogiem bieżącym. Jakiego użyłeś polecenia? Jaka jest pełna nazwa ścieżkowa tego katalogu?

3. Wróć do katalogu domowego. Na ile sposobów możesz to zrobić? Jak sprawdzisz czy jest to już twój katalog domowy?

4. Pozostając w katalogu domowym utwórz za pomocą jednego polecenia katalogi:

   smieci/listy/prywatne
   smieci/listy
   smieci/listy/sluzbowe
   smieci/listy/prywatne/ola

   Czy miałeś z tym kłopoty? Jeżeli tak, usuń wszystkie katalogi, które powstały podczas próby i wydaj polecenia ponownie.

5. Pozostając w katalogu domowym usuń za pomocą jednego polecenia wszystkie podkatalogi katalogu **listy**.

5 Praca z plikami

1. Utwórz plik (użyj polecenia touch) i sprawdź ile ma dowiązań.

2. Utwórz katalog i sprawdź ile ma dowiązań.

3. Obejrzyj atrybuty pliku pierwszy znadującego się w twoim katalogu domowym. Czy jest to plik zwykły czy katalog? Jaki jest rozmiar tego pliku? Jaka jest liczba dowiązań?

4. Jakie pliki ukryte znajdują się w twoim katalogu osobistym?

5. W katalogu osobistym utwórz podkatalog A i przejdź do niego. W katalogu A utwórz plik `skrypt` zawierający polecenie "echo To jest skrypt".

6. Do pliku skrypt w katalogu A dopisz na końcu zawartość pliku pierwszy (jest w twoim katalogu osobistym).

7. Użyj polecenia cat do obejrzenia pliku linie w katalogu osobistym. Czy jest to wygodny sposób oglądania pliku na ekranie? Obejrzyj 5 pierwszych i 8 ostatnich wierszy tego pliku. Obejrzyj plik "strona po stronie" poleceniem more.

8. Skopiuj plik pierwszy z katalogu osobistego do katalogu A.

9. Utwórz w katalogu A podkatalog podA i skopiuj cały katalog A wraz z zawartością do katalogu B w twoim katalogu sobistym (katalog B nie istnieje).

6 Prawa dostępu do plików i katalogów

1. W poniższych ćwiczeniach ważne jest ustawienie maski praw dostępu. Sprawdź jaka jest aktualna wartość maski. Jeżeli jest różna od 0, ustaw ją na 0.

2. Sprawdź jakie prawa dostępu ma twój katalog osobisty. Kto jest jego właścicielem? Do jakiej grupy należy? Czy jest to grupa, do której ty należysz?

3. Sprawdź domyślnie przyjmowaną postać uprawnień dla nowo tworzonych plików i katalogów, zakładając nowy podkatalog **B** i tworząc w nim (nie kopiując!) plik o nazwie **nowy**.

4. Znajdź w katalogu osobistym plik **skrypt** . Sprawdź jakie ma prawa dostępu. Zmień je na rwxr-x--- a następnie przywróć prawa oryginalne, posługując się raz poleceniem **chmod** z argumentami w formie liczbowej a drugi raz w formie symbolicznej.Oceń wygodę stosowania obu form polecenia **chmod** .

5. Utwórz plik **raport** o prawach dostępu rwx r-- r--. Sprawdź, że nie masz pliku o nazwie **raport96**. Skopiuj plik **raport** do pliku o nazwie **raport96**. Jakie prawa dostępu ma plik **raport96** ? Utwórz plik **raport_nowy** o prawach dostępu rwx r-- ---. Skopiuj go do istniejącego pliku o nazwie **raport96**. Czy plik **raport96** zachował swoje prawa dostępu?

   Na podstawie powyższego ćwiczenia odpowiedz na pytanie: Jakie są prawa dostępu pliku docelowego po skopiowaniu do niego pliku źródłowego jeżeli:
   - plik docelowy nie istnieje
   - plik docelowy istnieje

   (Zakładamy, że umask wynosi 0)

6. Wypełnij poniższą tabelę:

Polecenie	**Minimalne prawa dostępu**	
	dla pliku	dla katalogu z plikiem
cd /users/user10		
ls /users/user10/*.c		
ls -s /users/user10/*.c		
cat program		
cat >> program		
program (plik binarny)		
program (skrypt)		
rm program		

7. Ustaw maskę praw dostępu na **077**. Utwórz nowy plik o nazwie **plik077** i nowy katalog o nazwie **kat077**. Jakie mają prawa dostępu?

8. Nie zmieniaj maski. Niech nadal ma wartość 077. Utwórz plik **raport_01** o prawach dostępu rw- r-- r--. Skopiuj go do nie istniejącego pliku o nazwie **raport_koncowy**. Czy plik **raport_końcowy** ma takie same prawa jak plik źrółowy? Porównaj z wynikami zadania 5.

9. Zmień chwilowo swój identyfikator podając identyfikator innego uczestnika kursu. Zwróć uwagę na zmiany wykazywane przez polecenie **id** . Sprawdź, czy zmienił się katalog bieżący oraz porównaj wyniki wykonania poleceń **logname**, **id, whoami** oraz **who am i** i **groups** po zmianie identyfikatora. Powróć do swojego identyfikatora.

10. Dobierz argument dla **umask** w taki sposób, żeby nowo tworzony katalog miał prawa dostępu rwx--x--x . Jakie uprawnienia zyskuje wtedy nowo tworzony plik ?

11. W jaki sposób uzyskać prawa rwx--x--x dla nowotworzonego pliku?

12. Wypełnij poniższą tabelę:

Prawa dostępu dla nowotowrzonego pliku i katalogu przy podanych wartościach umask

	Właściciel		Grupa		Inni	
	plik	katalog	plik	katalog	plik	katalog
000						
002						
007						
022						
037						
077						

7 Podstawy edycji tekstów

1. Wprowadź poniższy tekst do pliku o nazwie **wiersz** (polskie litery zastąp zbliżoną literą łacińską).

```
Skąd przybywasz krokodylu?
 Ja? Znad Milu.
Wypuść mnie na parę dni,
To zabiorę cię na Nil.
```

   Przesuń kursor do słowa "pare". Usuń je i w to miejsce wpisz "kilka".

2. Popraw resztę wierszyka. Zamień:
   "dni" na "chwil"
   "przybywasz" na "ty jestes"
   "Milu" na "Nilu"
   "zabiore" na "zawioze"

   Zapisz tekst do pliku nie kończąc pracy z edytorem (polecenie :w)

3. Umieść kursor w trzecim wierszu, na początku słowa "chwil". Usuń wszystkie znaki od tego miejsca do:
   - końca wiersza
   - końca pliku

   Jakich użyłeś poleceń? Zakończ pracę z vi nie zapisując uszkodzonego tekstu do pliku (polecenie :q!)

4. Otwórz do edycji plik **nowy_wiersz.** W pliku tym:
   Przenieś wiersz "Budza sie" pod wiersz "Gdy".
   Przenieś dwa ostatnie wiersze pomiędzy wiersze "Budza sie" oraz "Gdy".
   Połącz trzy pierwsze wiersze.
   Wstaw "," na końcu pierwszego wiersza.
   Skopiuj "Gdy wiatr" na początek drugiego wiersza.
   Rozdziel wiersz 2 przed słowem "Budza".
   Wstaw na koniec drugiego wiersza "zaswista, "
   Połącz wiersze 3,4 i 5.

5. Otwórz do edycji plik **wiersz_oli.** Za pomocą poleceń zmiany tekstu (r, R, c, s, S) utwórz tekst:
   Jeden, dwa
   Jeden, dwa
   Pewna pani
   miala psa.

6. Otwórz plik o nazwie **samochody**. Zmień wszystkie nazwy "fiat" na "Fiat".

7. Otwórz plik o nazwie **pracownicy.** Przepisz pierwsze 5 wierszy z pliku **pracownicy** do pliku **nowi_pracownicy.**

8 Podstawy korzystania z shella

1. Jakich poleceń użyjesz, aby wyświetlić wartości zmiennych lokalnych i środowiskowych. Wydaj te polecenia.

2. Wprowadź zmienną lokalną **zlok** i nadaj jej wartość Unix. Wprowadź zmienną środowiskową **zglob** i nadaj jej wartość Shell. Sprawdź wykaz zmiennych lokalnych i środowiskowych. Wywołaj nowego shella i sprawdź wartości zmiennych lokalnych i środowiskowych. Czy są wśród nich zmienne **zlok** i **zglob**?

Nadaj zmiennej **zglob** nową wartość równą Lekcja po czym wróć do shella początkowego i ponownie sprawdź wartości zmiennych. Zinterpretuj wyniki.

3. Zdefiniuj znak gotowości shella tak, aby wyświetlał tekst: nazwa_użytkownika: .

4. Przypisz zmiennej dir wartość /bin/ls. Jak można posłużyć się tą zmienną, aby wykonać polecenie ls? Czy można w tak utworzonym poleceniu podać argumenty z nazwą katalogu lub pliku?

5. W którym katalogu znajdziesz się po wydaniu polecenia (cd /bin; ls).

9 Środowisko użytkownika

1. Sprawdź poleceniem set -o czy jest ustawiona opcja ignoreeof. Jeżeli nie jest,,
ustaw ją poleceniem set -o ignoreeof. Sprawdź, czy można zakończyć sesję za pomocą naciśnięcia klawiszy CTRl-d.

2. Ustaw w pliku .profile ścieżkę tak, aby do aktualnie obowiązującej ścieżki na końcu został dodany twój podkatalog z programami (*katalog_osobisty_użytkownika/bin*). Wykonaj plik .profile. Czy trzeba w tym celu kończyć sesję?

3. Znajdź katalog, w którym znajduje się polecenie man.

4. Zmodyfikuj swoje środowisko tak, aby po otworzeniu sesji wyświetlana była zawartość skrzynki pocztowej.

5. Zmodyfikuj swoje środowisko tak, aby po otworzeniu sesji wyświetlana była informacja o użytkownikach aktualnie prowadzących sesje (polecenie finger lub who z odpowiednimi parametrami).

6. Co należy zrobić, aby w zgłoszeniu gotowości shella wyświetlana była informacja: *numer_polecenia nazwa_użytkownika:*

10 Mechanizmy w shellu (ksh)

Uwaga: w ćwiczeniach potrzebna jest znajomość mechanizmu historii poleceń. Jest on opisany na stronach poświęconych odpowiedniemu shellowi.

1. Zdefniuj następujące synonimy:

a) **rm** usuwanie pliku wymagające potwierdzenia

b) **dir** wypisywanie strona po stronie zawartości katalogu łącznie z wszystkimi atrybutami plików (typ, prawa dostępu, liczba dowiązań itp.)

c) **h** wypisywanie strona po stronie do 30 ostatnio wykonanych poleceń.

2. Sprawdź jakie polecenia ostatnio wydałeś. Powtórz ostatnie polecenie. Powtórz np. polecenie piąte od końca.

3. Wykonaj następujące polecenia:

```
ls /bin
ls -al
more .profile
who
```

Teraz wykonaj jeszcze raz polecenie **ls -al**, pamiętając, aby jak najmniej klawiszy nacisnąć. A jak należy wywołać polecenie **ls /bin**? Wykonaj polecenie **ls /usr/bin** wykorzystując do tego mechanizm historii.

4. W bieżącym katalogu znajdują się między innymi następujące pliki:

```
abcdxefghxijkl    abcdxefghy  abcdy
```

Jak korzystając z polecenia **ls -l** oraz mechanizmu dopełniania nazw plików wyświetlić atrybuty pliku **abcdxefghxijkl**.

5. W pewnym katalogu znajdują się następujące pliki:

```
.plab       pl11        pl.exe.     plik.
.plaa       pl12        p..
pl1         pl111       p.?
pl2         pl.exe      p??
```

Wyświetl atrybuty (polecenie **ls -l**):

- wszystkich plików o nazwach co najmniej czteroliterowych zaczynajacych się od **pl1**,
- wszystkich plików o nazwach trzyliterowych
- wszystkich plików o nazwach kończących się znakiem **?**,
- wszystkich plików ukrytych,
- wszystkich plików, których nazwy zawierają kropkę w dowolnym miejscu, ale nie mogą się od niej zaczynać,
- wszystkich plików, których nazwy zawierają kropkę w dowolnym miejscu.

6. Wydaj polecenie wyświetlające wszystkie nieukryte pliki w katalogu osobistym. Użyj polecenia **ls** oraz **echo**. Zinterpretuj otrzymane wyniki.

7. Utwórz plik **wykaz.kto**, w którym pierwszy wiersz zawiera dzisiejszą datę a nastepne wiersze nazwy aktulanie pracujących użytkowników.

8. Utwórz plik **bez.vi** (nie używaj edytora **vi**) zawierający zdanie :

> Proste pliki przygotowywujemy
> nawet wtedy, gdy nie znamy edytora vi.

9. Utwórz plik **konkat** jako połączenie plików **bez.vi** oraz **wykaz.kto**.

10. Spróbuj utworzyć plik **testuj** jako połączenie nieistniejącego pliku **bb** z plikiem **bez.vi** . Gdzie przesłany zostanie komunikat o błędzie? Co należy zrobić, aby komunikat o błędzie:

- był zapisywany w pliku **testerr**,

- nie pojawiał się nigdzie.

11. Wyświetlając na ekranie plik **/etc/passwd** skopiuj go jednocześnie do pliku **konta**. (Skorzystaj z polecenia **tee**).

12. Zinterpretuj wynik polecenia **ls | date** .

11 Wybrane działania na plikach tekstowych

1. Oblicz ilu użytkowników korzysta w tej chwili z serwera?
2. Wyświetl listę terminali, na których aktualnie prowadzone są sesje.
3. Wyświetl listę zawierającą prawa dostępu do plików, ich nazwy a także właścicieli plików z własnego katalogu.
4. Utwórz plik `telefony` o postaci:

   `nazwisko imię  nr_telefonu`

   `Kowalski Adam  34-78-89`

   Wpisywane pola oddziel znakami TAB. Zmień następnie małe litery na duże.
5. Skopiuj do swojego katalogu osobistego plik **/etc/passwd**. Wytnij z niego informację o nazwach użytkowników, numerach identyfikacyjnych grupy i użytkownika. Posortuj wg numerów użytkowników.
6. Sprządź wykaz użytkowników aktualnie pracujących. Posortuj go alfabetycznie. Zlicz ile sesji otworzył każdy z użytkowników.
7. Wybierz dzień, miesiąc i rok z polecenia date. Wypisz komunikat: `Dzisiaj mamy dd.mm.rr.`

12 Nadzorowanie wykonywania poleceń

1. Wyświetl informacje o uruchomionych przez siebie procesach, o procesach innych użytkowników, a także o procesach, które nie mają przydzielonego terminala (demonach). Policz te ostatnie.

2. Wydaj polecenie
   `$ sleep liczba_sekund`
   tak aby było wykonywane w tle. Przyjmij, że liczba_sekund jest równa 1000+UID. Sprawdź następnie PID oraz PPID procesu uruchomionego w celu wykonania tego polecenia. Zakończ tę sesję, rozpocznij nową i wydaj polecenie:
   `$ ps –ef | grep sleep`
   Czy znalazłeś swój proces?

3. (shell ksh) Tym razem wydaj polecenie
   `$nohup sleep liczba_sekund`

   Powtórz kroki opisane w zadaniu 3 i odpowiedz na to samo pytanie.

4. Rozpocznij wykonywanie polecenia
   `$ sleep 900`
   jako zadanie pierwszoplanowe. Zatrzymaj to zadanie i umieść je w drugim planie.
   Rozpocznij wykonywanie następnego polecenia:
   `$ sleep 800`
   tym razem jako zadanie drugoplanowe. Wyświetl listę wykonywanych zadań. Przerwij wykonywanie pierwszego zadania. Drugie zadanie przenieś na piwerszy plan.

5. Zaplanuj wykonanie polecenia **ls** w katalogu osobistym i w katalogu głównym (/) za, odpowiednio, trzy i dwie minuty, wykonując przeadresowanie wyników tych poleceń do plików **rap.home** i **rap.root** . Sprawdź kolejkę zaplanowanych zadań.
   Powtórz planowanie tych samych zadań, ale bez kierowania wyników poleceń **ls** do pliku. Gdzie powinny trafić rezultaty wykonania tych poleceń ?

6. Zaplanuj wykonanie skryptu **plan** na na godz. 14.45 dzisiejszego dnia, oraz na godz 23.59, 13 grudnia tego roku.
   Sprawdź numery obu tych zadań, a następnie usuń te zadania z kolejki.

13 Archiwizacja plików

1. Odszukaj, gdzie znajduje się polecenie vi, wystartuj z katalogu /. Co należy zrobić, aby uniknąć wyswietlania na ekranie komunikatów o niedostępnych katalogach?

2. Wyświetl nazwy plików, które rozpoczynają się od "pl":
   a) poprzedzone pełnymi nazwami ścieżkowymi
   b) poprzedzone względnymi nazwami ścieżkowymi

3. Załóż podkatalog **$HOME/dane/A** i utwórz w nim kilka "dużych" plików będących np. efektem działania poleceń **ls** , **ps -ef** czy też kopiowania plików systemowych takich jak **/etc/passwd** . Zapamiętaj (z grubsza) rozmiary utworzonych plików. Wykonaj kompresję wszystkich plików zawartych w katalogu **A**, zlecając dodatkowo wyświetlanie informacji o przebiegu kompresji. Porównaj rozmiary skompresowanych plików z ich rozmiarami oryginalnymi.

4. Wykonaj dekompresję plików z katalogu **A** . Skopiuj katalog **A** do (nowych) podkatalogów **B** i **C** , a do katalogu **dane** skopiuj plik **$HOME/.cshrc** - uzyskując w efekcie końcowym strukturę plików:
   ($HOME) -> (dane) -> ((A), (B), (C), .profile).
   W pliku **$HOME/ARCH.tar** utwórz archiwum zawierające poddrzewo rozpoczynające się od katalogu **dane** .

5. Wyświetl spis zawartości utworzonego pliku archiwum i jeśli jest poprawny, wykonaj kompresję tego pliku

6. Usuń podkatalog **dane** . Utwórz w innym miejscu podkatalog **kopia** . Wczytaj do niego zawartość archiwum **ARCH.tar** (po wcześniejszym wykonaniu dekompresji tego pliku).

7. Wyświetl pełne informacje o wykorzystaniu dysku przez katalog **kopia** oraz zapisz do pliku **my.space** analogiczne wyniki dla swojego katalogu osobistego. Jak przedstawia się zajętość partycji dyskowych w systemie?