Systemy Operacyjne SOP121

Plan wykładu

1. Wstęp; Unix/ perspektywa zwykłego użytkownika
2. Budowa “ogólnego” systemu operacyjnego
   (ilustrowane przykładami rozwiązań z konkretnych s.o.)
3. Konkretne systemy operacyjne
   (administrowanie, programowanie, budowa jądra itp)
   a) “Unix System V”
   b) “Linux”
   c) “Windows NT/2000/XP”

Systemy Operacyjne SOP121

Literatura

1. Literatura podstawowa:
   - Silberschatz, Galwin, "Podstawy systemów operacyjnych" (wydanie 3)
2. Literatura pomocnicza:
   - Stallings, "Operating Systems: Internals and Design Principles"
   - Bach, "Budowa systemu operacyjnego Unix" (opis "Unix-a System V")
   - Kaniewski, Wiermiejczyk, "Po prostu Unix"
   - Królikowski, Sajkowski, "System operacyjny Unix dla początkujących i zaawansowanych"

Systemy Operacyjne SOP121

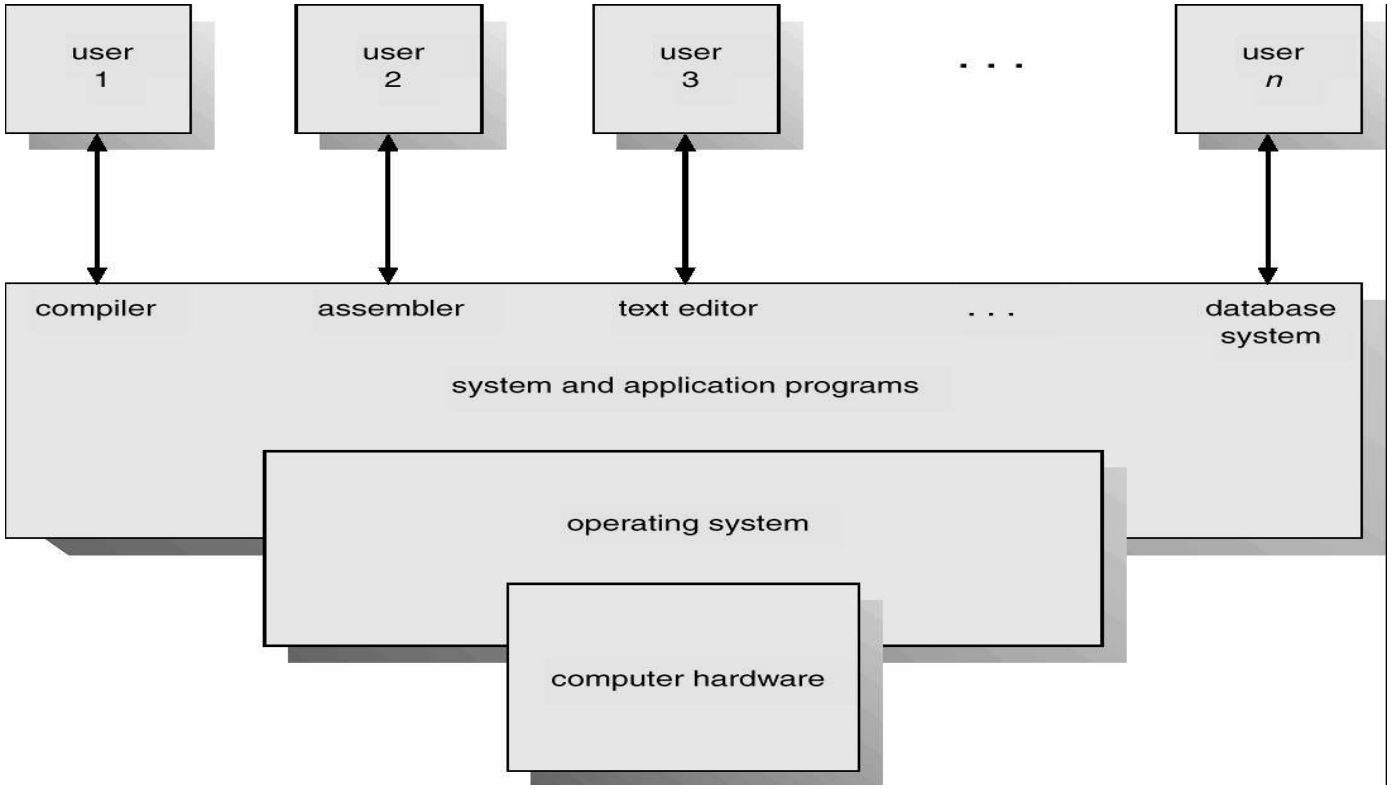
Literatura

3. Literatura pomocnicza c.d.
   - wielu autorów, "Linux Kernel – jądro systemu" (opis jądra Linux-a w wersji 2.0)
   - Tanenbaum, Woodhull, "Operating Systems: Design and Implementation" (opis ciekawego s.o. MINIX)
4. Dostępne w Internecie :
   - **http://main2.amu.edu.pl/~mhanckow** (* SOP121 *)
   - Linux-HOWTO
   - Linux-Mandrake Documentation
   - The Linux System Administrators' Guide
   - The Linux Programmer's Guide
   - Linux Installation and Getting Started
   - The Network Administrators' Guide

Definicje s.o.; podstawowe pojęcia

Co to jest system operacyjny?

- pośrednik między programami użytkowników a sprzętem ...



Definicje s.o.; podstawowe pojęcia

Co to jest system operacyjny?

- pośrednik między programami użytkowników a sprzętem ...
  - programy nie odwołują się bezpośrednio do sprzętu, a jedynie za pośrednictwem tzw *funkcji systemowych*
  - s.o. ukrywa szczegóły techniczne sprzętu przed programami
  - s.o. tworzy tzw *logiczna (lub wirtualną) maszynę*, która jest idealizacja maszyny fizycznej (=sprzętu)

Definicje s.o.; podstawowe pojęcia

Co to jest system operacyjny?

- tworzy *środowisko* w którym pracują programy użytkowników
- jest *dystrybutorem (lub alokatorem) zasobów*
  - co to są **zasoby**? np. czas procesora, obszar w pamięci operacyjnej lub dyskowej, urządzenia we/wy
  - s.o. przydziela zasoby działającym programom; może wtedy dojść do konfliktu, np. przy przydzielaniu czasu procesora gdy jest 1 procesor i >1 programów działających współbieżnie ...

Definicje s.o.; podstawowe pojęcia

Co to jest system operacyjny?

- jest *programem sterującym* :
  - nadzoruje działanie programów użytkowników (przeciwdziała błędom, niewłaściwemu użyciu komputera)
  - kontroluje i obsługuje urządzenia we/wy (np. nadzoruje przesyłanie danych między dyskiem magnetycznym a pamięcią operacyjną)
  - co to są **urządzenia we/wy** ? np. stacje CD, dyski magnetyczne, drukarki, ekran monitora/ terminal, mysz, klawiatura, ...
- jest programem, który działa bez końca ;-);
  - dokładniej tzw *jądro* systemu działa bez końca (s.o. składa się *z jądra* i z *programów systemowych*)

Definicje s.o.; podstawowe pojęcia

Najważniejsze cele s.o.

- wykonywanie programów użytkowników
- komputer ma być wygodny w użyciu
- komputer ma być wydajnie wykorzystywany
  - tzn każdy zasób ma powinien być możliwie często i sensownie wykorzystywany
  - np. czas procesora nie powinien być marnowany (kiedy jest marnowany? np. gdy procesor czeka na zakończenie operacji we/wy)
  - dawniej chodziło głównie o wydajność, dziś chodzi też o wygodę ...
  - *jeśli s.o. nie marnuje czasu procesora to nasze programy wykonają się szybciej !!!*

Unix/ perspektywa użytkownika

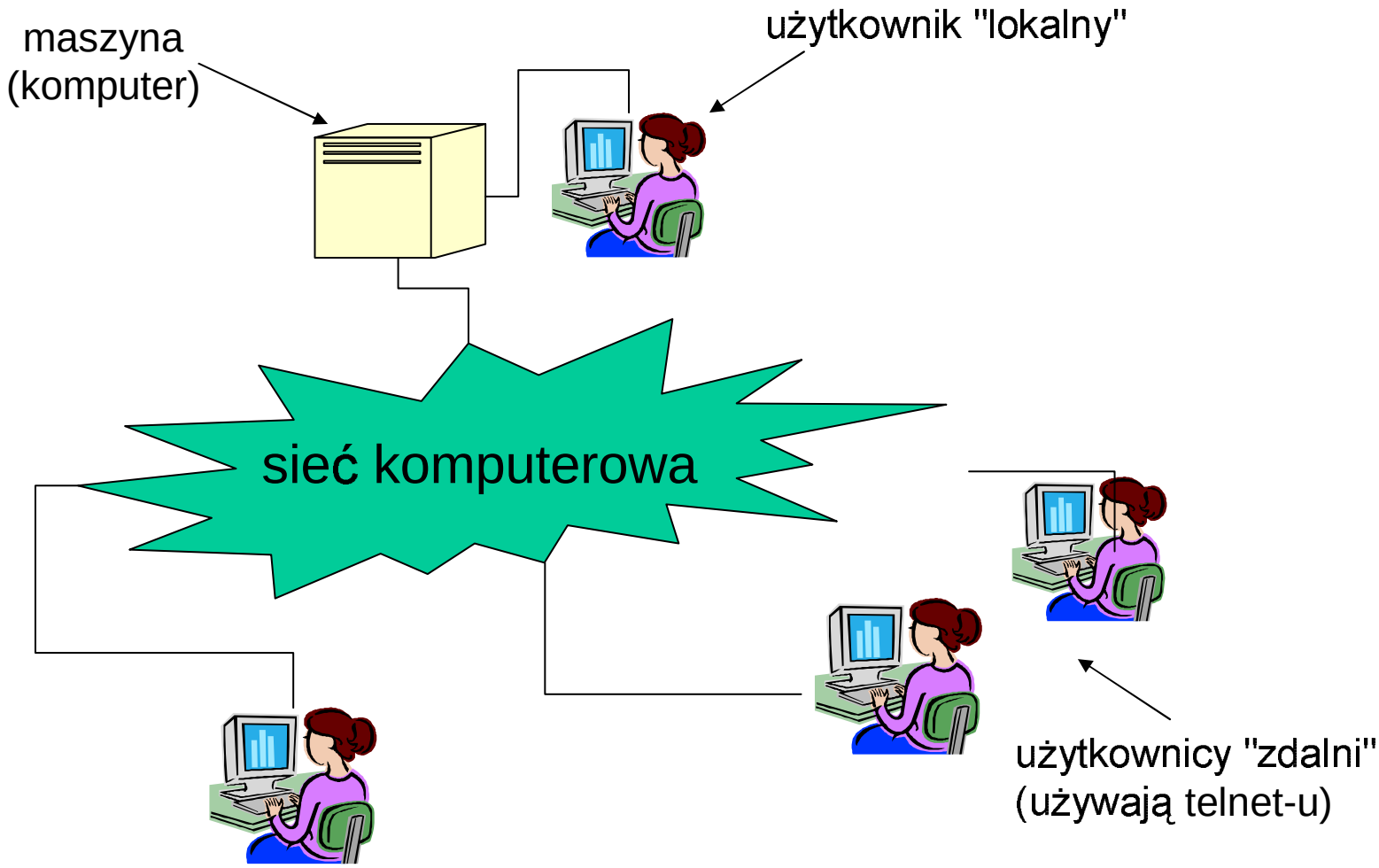
Logowanie

(oprócz "perspektywy użytkownika (zwykłego)" jest też perspektywa administratora i programisty ...)

- podczas logowania trzeba podać:
  - identyfikator użytkownika
  - hasło
  - ... i zostaniemy wpuszczeni do systemu
- maszyna na której pracujemy może być:
  - lokalna (= siedzimy bezpośrednio przy niej)
    - widzimy terminal tekstowy lub ...
    - jest włączone środowisko graficzne/okienkowe "X Windows"; wtedy można włączyć program **xterm** = emulator terminala tekstowego
  - zdalna
    - używamy usługi "telnet" (lub ssh); klient telnetu pokazuje nam okienko terminala tekstowego
- w *terminalu tekstowym* można wydawać *komendy*, które służą do uruchamiania programów ...

Unix/ perspektywa użytkownika

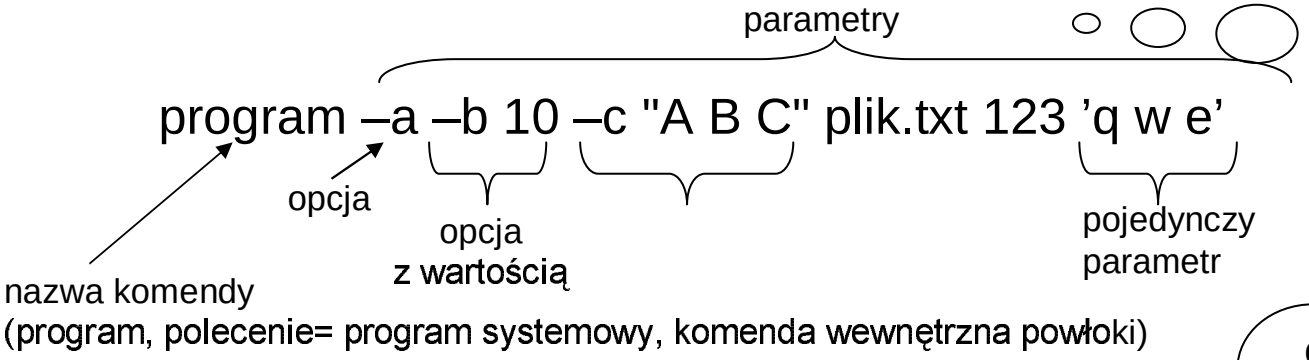
Logowanie



Unix/ perspektywa użytkownika

Uruchamianie programów przy pomocy komend

- ***komendy*** „wydaje się” w ***powłoce*** (ang. shell), która komunikuje się z użytkownikiem poprzez terminal tekstowy ...
- postać komendy:



Co „zależy” od jądra systemu a co od powłoki lub programu ?

- powłoki Unixowe: sh, ksh, csh, bash

odpowiedniki programu command.com DOS-u

- przykład "sesji" z powłoką w terminalu:

(pokazać "na żywo" w `xterm-ie`)

Unix/ perspektywa użytkownika

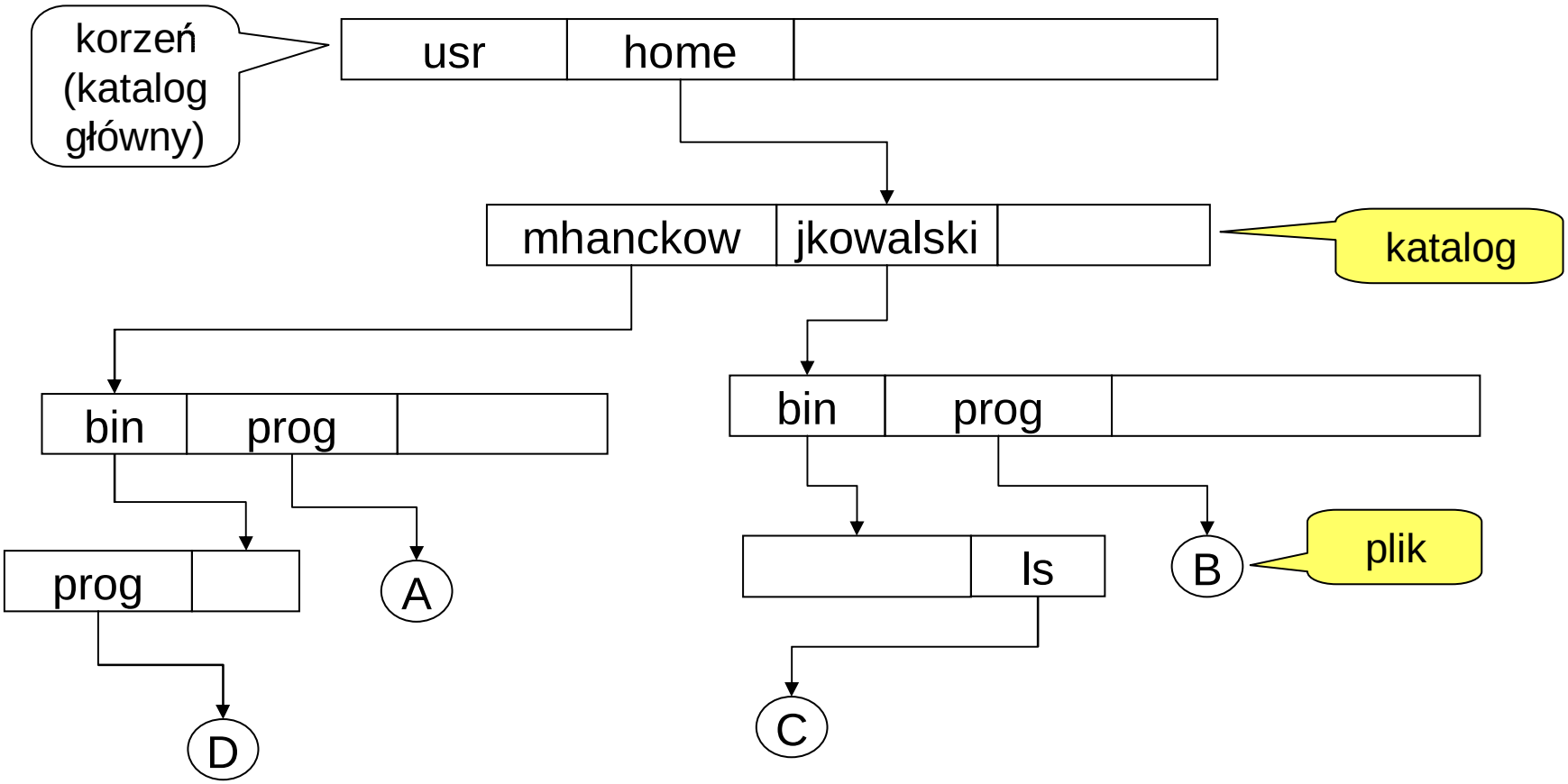
Pliki i katalogi

- *def **pliku***:
  - plik to ciąg bajtów (0..255) trwale zapisany np. na dysku magnetycznym
  - posiada *nazwę*
  - plik może zawierać dane lub program
- *def **katalogu***:
  - katalog to tablica której elementami są nazwy plików i katalogów (tzw pod-katalogów);
  - katalog także jest trwale zapisany
  - posiada nazwę
- *nazwa pliku i katalogu powinna sugerować jego zawartość !!!*
  - np. katalog o nazwie "moje obrazki" powinien zawierać pliki z grafiką

Unix/ perspektywa użytkownika

Pliki i katalogi

- rysunek przedstawiający przykładową ***strukturę katalogów*** (w tym wypadku - *drzewo katalogów !!!*)



Unix/ perspektywa użytkownika

Pliki i katalogi

- ***ścieżki*** do plików lub katalogów:
  - bezwzględne

    nazwa pliku (w katalogu macierzystym)

    `/home/mhanckow/bin/prog`

    bezwzględna ścieżka do pliku (od "korzenia" czyli "/")
  - względne
    - każdy proces (=uruchomiony program) ma ***katalog bieżący ...***
    - ścieżka która NIE zaczyna się od "/" jest względna
    - ścieżka względna "jest obliczana" od katalogu bieżącego
- pozycje "." i ".." w katalogu X
  - "." oznacza katalog **X**
  - ".." oznacza rodzica katalogu **X**
  - przykład prawidłowej ścieżki używającej "." i ".."

    `/home/mhanckow/../jkowalski/bin/../../mhanckow/bin/prog`

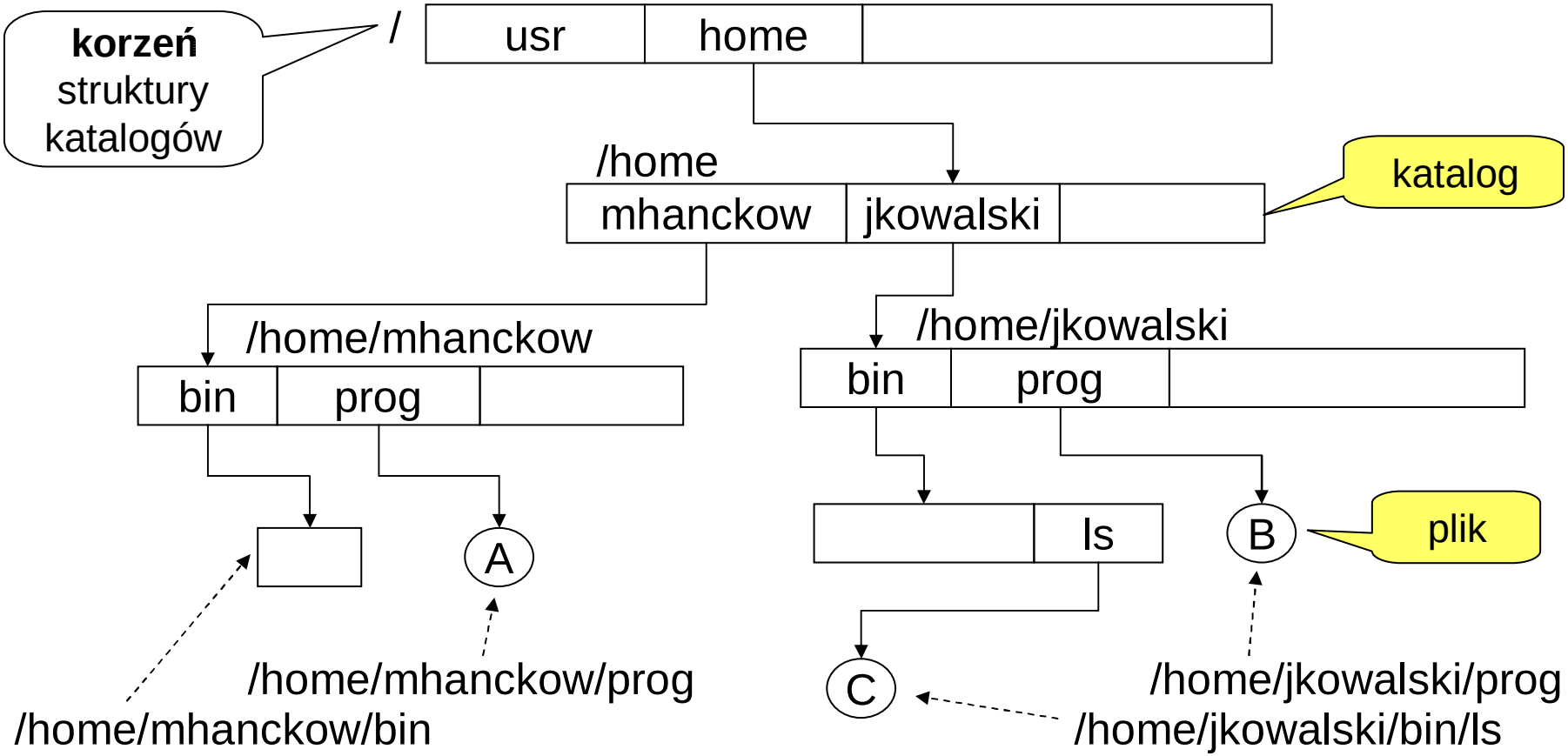
    `jkowalski/./../bin/../../mhanckow/bin/prog`

    (w drugim przykładzie katalog bieżący= /home)
  - samoistne "." oznacza katalog bieżący
  - ścieżki zaczynające się od "." są ścieżkami względnymi

Unix/ perspektywa użytkownika

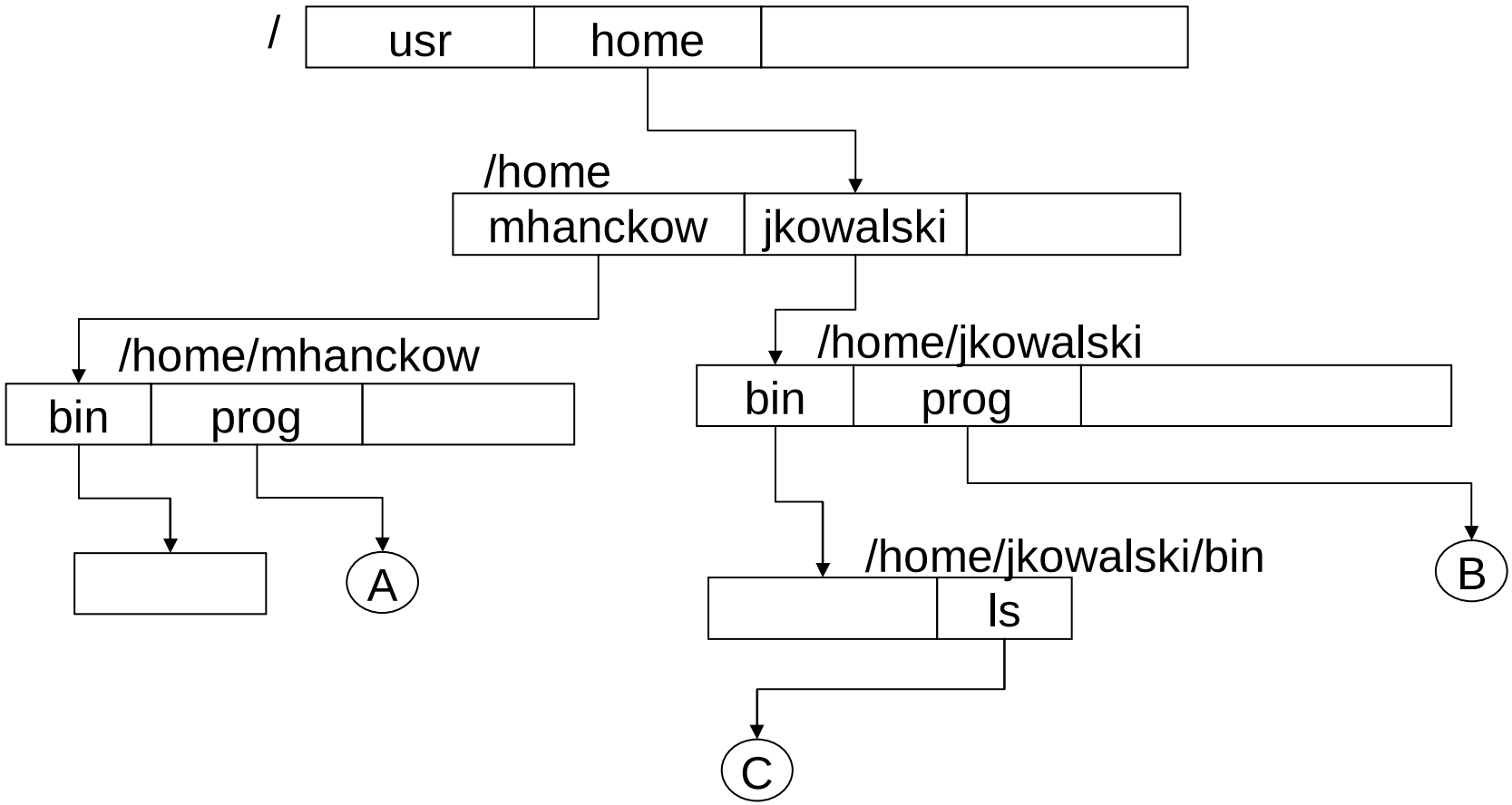
Pliki i katalogi

- pokazane są ***ścieżki bezwzględne*** do plików i katalogów ...



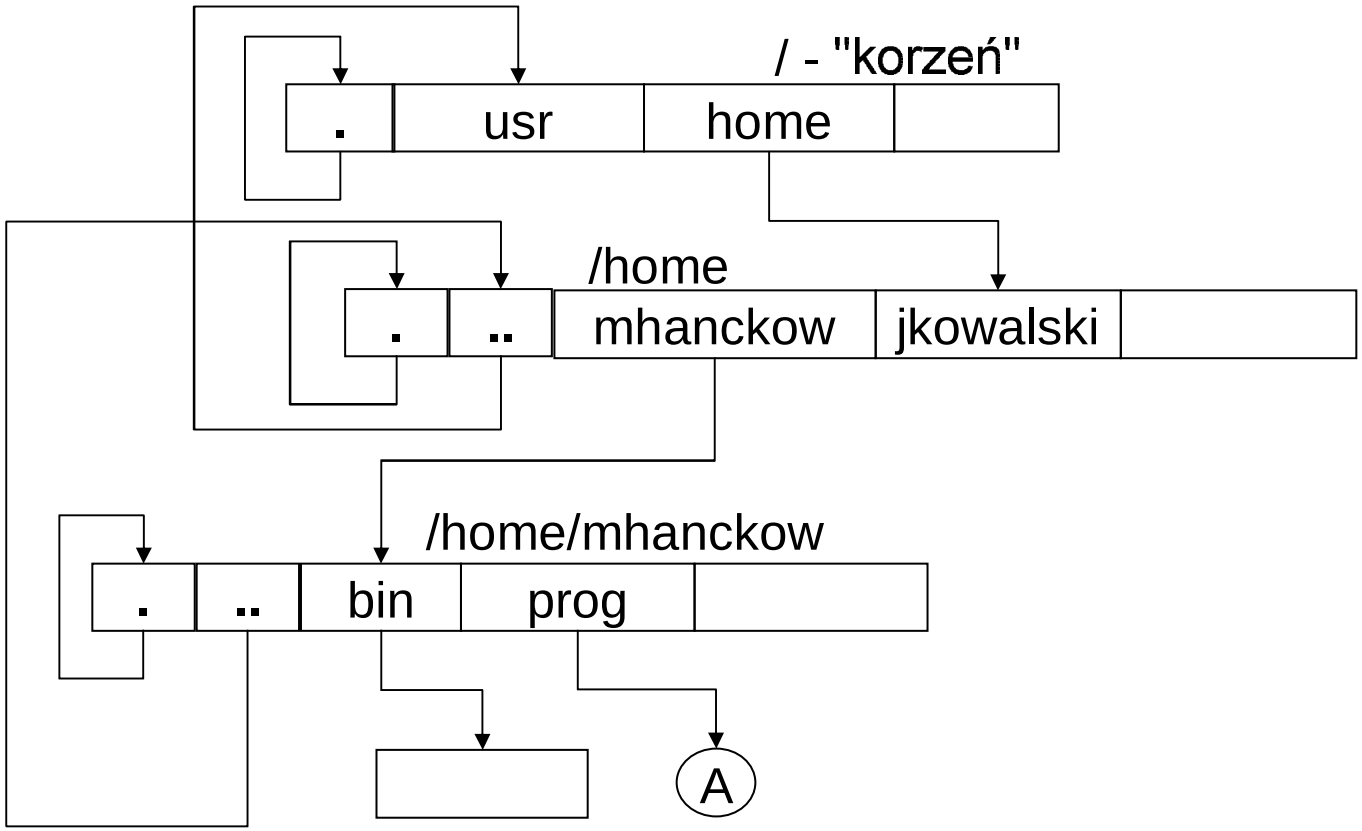
Unix - pliki i katalogi

- katalogiem bieżącym jest /home/jkowalski/bin
- jak wygląda ścieżka **względna** do pliku A ???



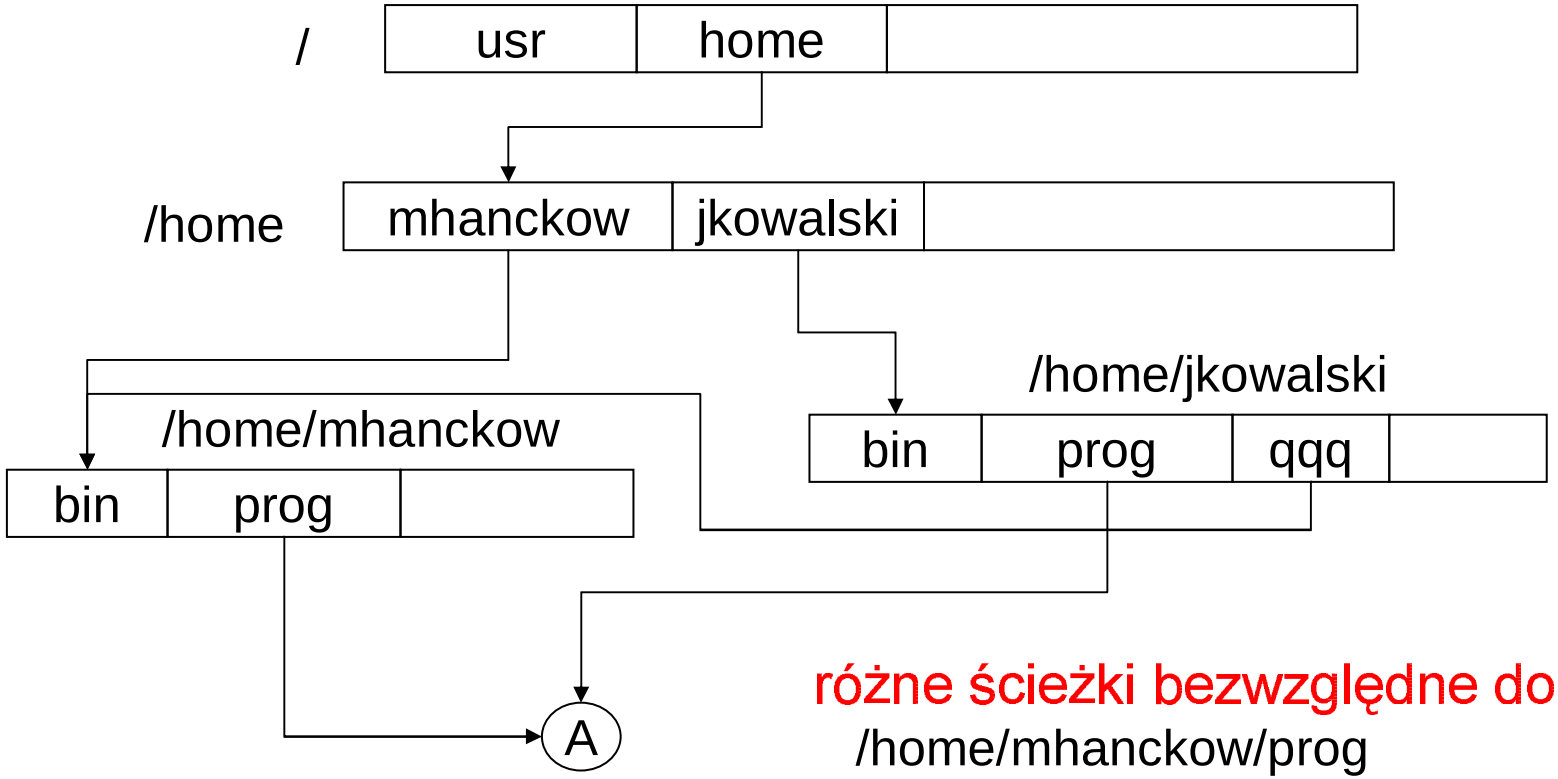
Unix - pliki i katalogi

- struktura katalogów uwzględniająca pozycje "." i ".."



Struktura katalogów w Unixie

- struktura katalogów nie musi być drzewem !!!
  (dlatego używamy słowa "struktura" a nie "drzewo")



różne ścieżki bezwzględne do (A):

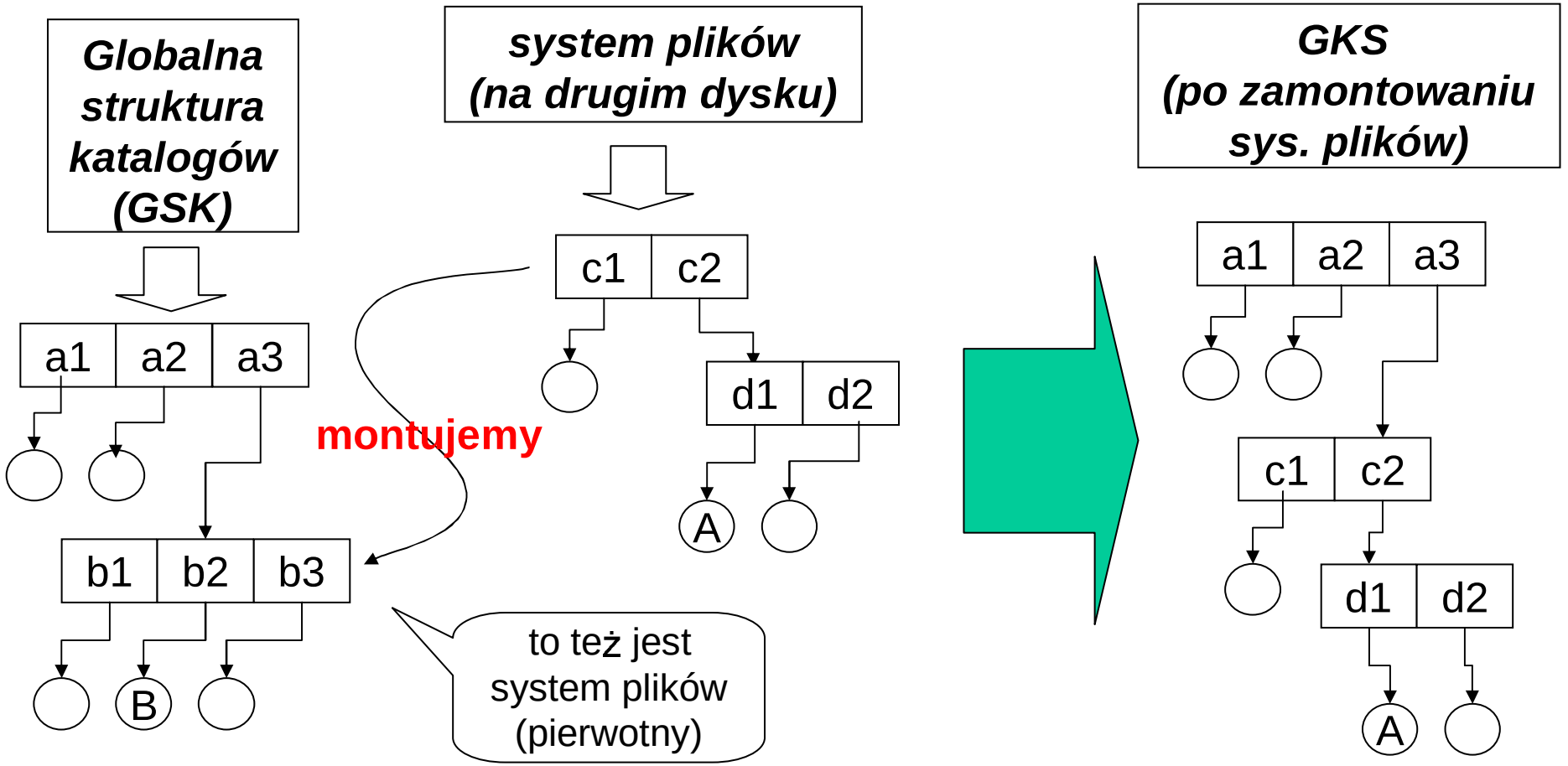
/home/mhanckow/prog
/home/jkowalski/prog
/home/jkowalski/qqq/prog

Montowanie systemu plików w Unixie

- ***system plików*** = strukturakatalogów(i plików) "rezydująca" na pojedynczym dysku
- ***globalna struktura katalogów (GSK*** – terminologia MH !***)*** to strukturakatalogów poprzez którą sądostępnewszystkie systemy plików naszej maszyny
  - początkowo GSK jest pojedynczym systemem plików w którego katalogach montuje się kolejne systemy plików
  - montowanie odbywa się m.in. podczas boot- owania (uruchamiania systemu)
- ***montowanie*** to umieszczenie jednego systemu plików w pewnym (pod)katalogu globalnej struktury katalogów (GSK)

Montowanie systemu plików w Unixie

- zmiana struktury katalogów po zamontowaniu nowego systemu plików ...
- (po zamontowaniu) dostęp do /a3/c2/d1 oznacza dostęp do pliku "A" nowego systemu plików; plik "B" jest niedostępny !



Pliki specjalne w Unixie

- umożliwiają (surowy) dostęp do urządzeń wejścia/wyjścia
- znajdują się w katalogu /dev *(pokazać "na żywo")*
- przykłady:
  - /dev/tty01 – dostęp do terminala (*zapis do tego pliku powoduje zapis na terminalu*)
  - /dev/fd0 – dostęp do dyskietki (*dyskietka jest traktowana jako ciąg bajtów a nie jako struktura katalogów*)
  - /dev/hda1 – dostęp do 1 partycji pierwszego dysku twardego
  - /dev/hdb3 – dostęp do 3 partycji drugiego dysku twardego
  - /dev/cdrom – dostęp do stacji CD
- zastosowania plików specjalnych:
  - polecenie "fsck" (=file system check); naprawianie uszkodzonego systemu plików poprzez modyfikowanie pliku spec partycji
  - dostęp do terminala to zapis/odczyt z pliku /dev/tty01
  - podczas montowania systemu plików podaje się plik specjalny identyfikujący dysk/partycje na której znajduje się system plików

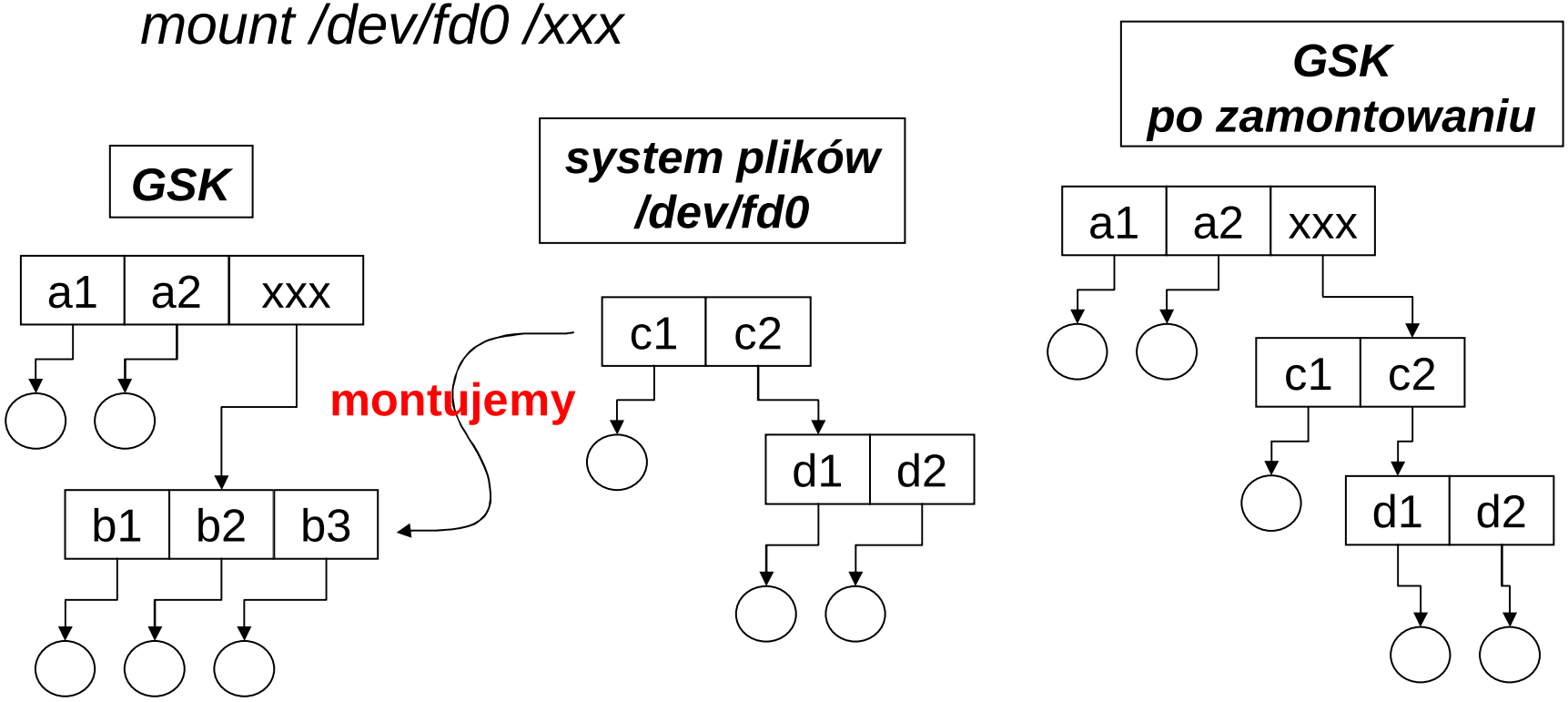
Montowanie systemu plików w Unixie (c.d)

- "montuje" się przy pomocy polecenia:

  mount plik_specjalny_sys_plików katalog_montowania

  przykład:

  mount /dev/fd0 /xxx



Programy, procesy, polecenia w Unixie.

- program = plik zawierający (m.in) kod programu
  - ***kod programu*** = ciąg elementarnych rozkazów typu:

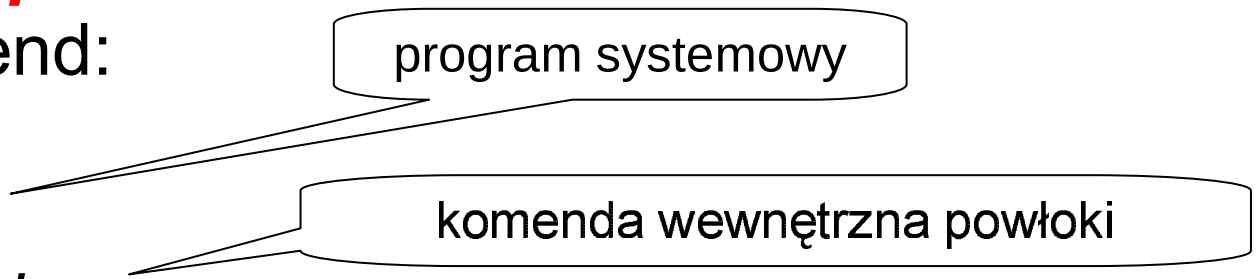
    mov AX, [1000] ;; odczytaj komórkę o adresie 1000
    mov BX, [2000]
    add AX, BX ;; AX:=AX + BX
    mov [3000], AX

  - ***pamięć operacyjna*** = ciąg ponumerowanych komórek pamięci zawierających liczbę (1 bajt = 0..255); numery komórek nazywamy adresami komórek pamięci
- proces = uruchomiony program

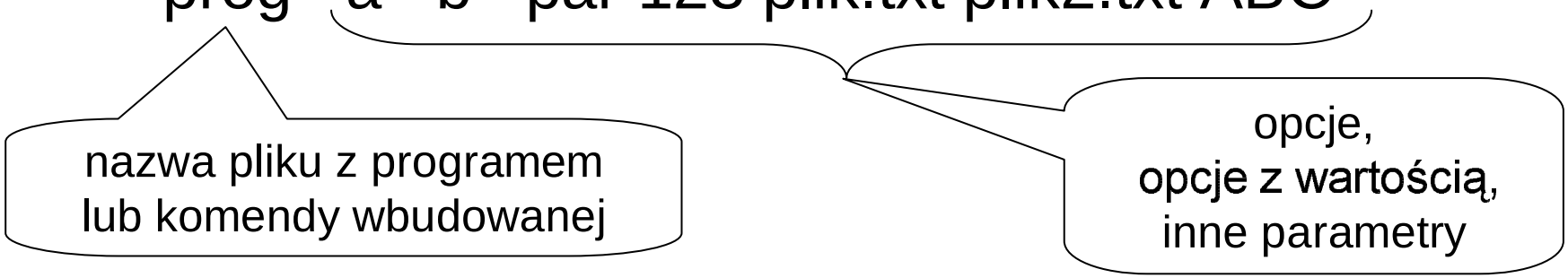
  uruchomienie programu oznacza:
  - przydzielenie programowi obszaru w pamięci operacyjnej i skopiowanie tam kodu programu
  - przydzielenie czasu procesora ... oraz innych zasobów

Uruchamianie programów w Unixie

- Programy uruchamia się przy pomocy ***komend*** wydawanych w ***powłoce***...
  Przykłady komend:
  prog 1 2 3
  ls –l plik txt
  cd mój_katalog



- Powłoki unixa (od najstarszej):
  sh, csh, ksh, bash
- Parametry (argumenty) komend:
  prog –a –b –par 123 plik.txt plik2.txt ABC



Rola zmiennej ***PATH*** przy uruchamianiu programów

- każdy proces posiada *zmienne środowiska*; zmienne te posiadają *nazwę* i *wartość*
- zmienne środowiska są „dziedziczone” przez procesy potomne (czyli programy uruchomione przez nasz proces)
- zmienne środowiska w powłoce (bash, ksh) ...
  tworzy się tak:
  export PATH=.:/bin:/usr/bin
  odczytuje się tak:
  echo $PATH
- PATH to zmienna środowiska zawierająca listę katalogów (oddzielonych znakiem ":")
- Gdy uruchamiamy program NIE podając ścieżki do pliku z programem to katalogi PATH są przeszukiwane ...
- Częsty błąd: brak kropki na zmiennej PATH gdy uruchamiamy program z bieżącego katalogu !
  PATH=$PATH**:.**

... pokazać przykład; **./prog** kontra **pro**

Polecenia (=programy systemowe) Unixa

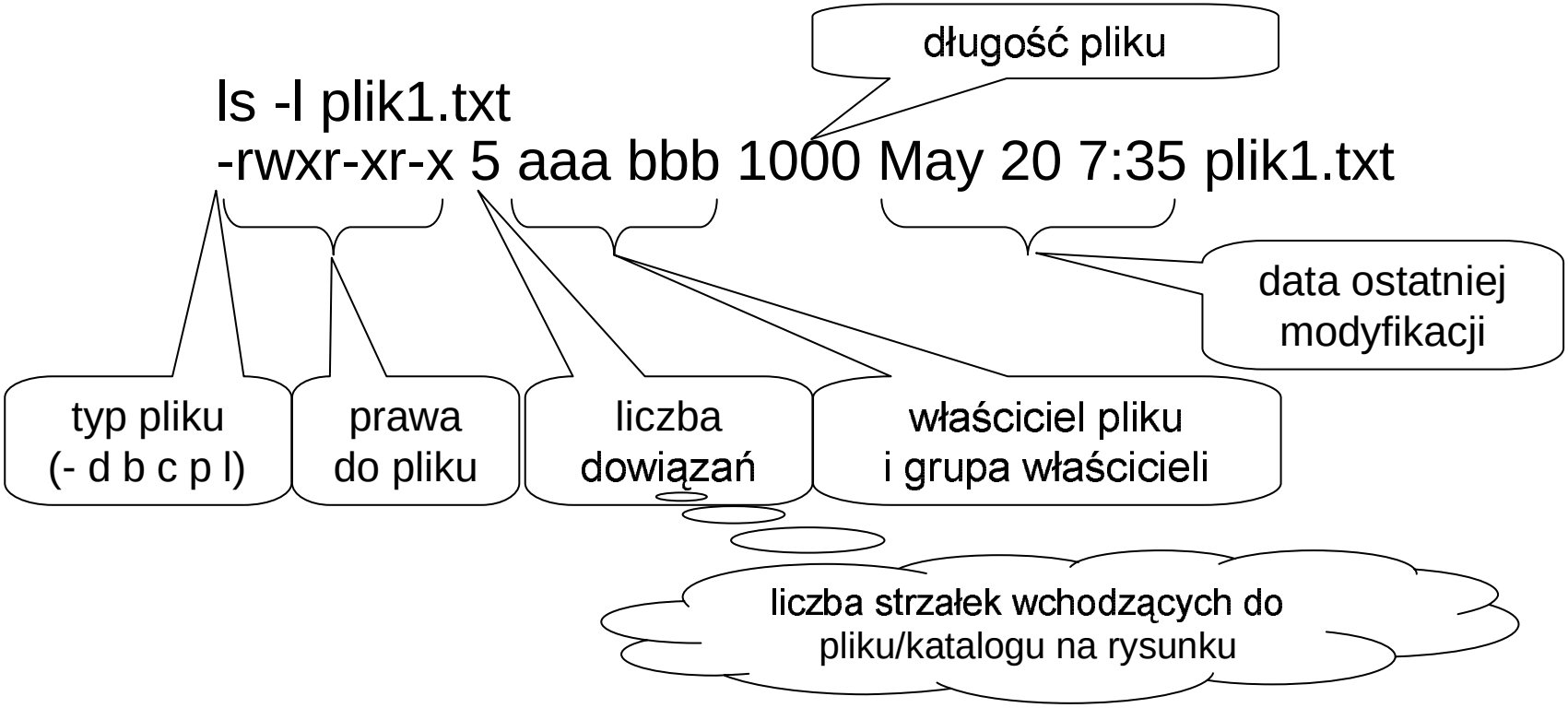
- dotyczące plików:

  ls, pwd, cd, cp, mv, rm, mkdir, cat, chmod, ln

- dotyczące procesów:

  ps, kill, tty

- polecenie "ls -l" wyświetla szczegółowe informacje o plikach



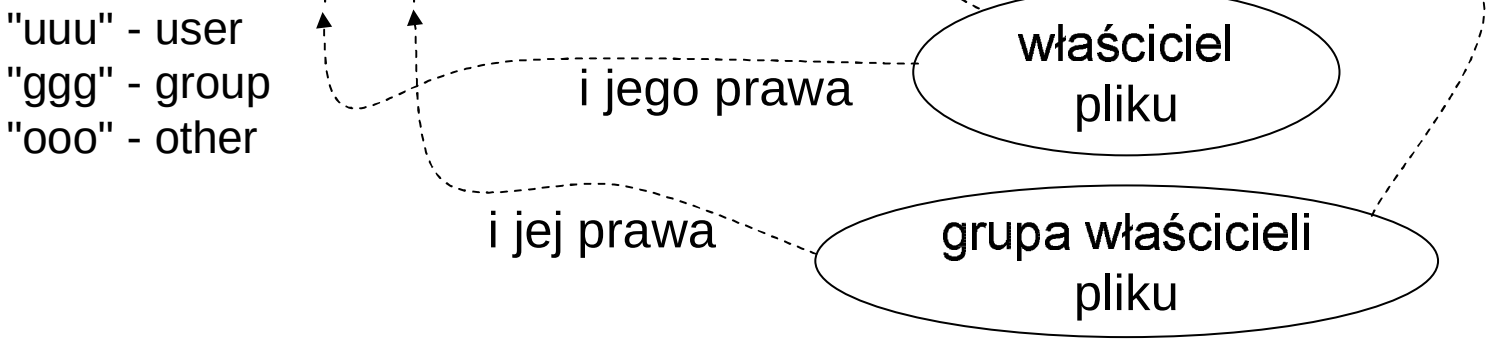
Polecenie "ls –l"

- typ pliku:
  - "-" pliki zwykłe; przechowywanie danych i programów
  - "d" katalogi
  - "b", "c" pliki specjalne; pozwalają na *surowy* dostęp do urządzeń
    b -blokowych, c -znakowych
  - "p" łącza nazwane; służą procesom do komunikowania się
  - "l" dowiązania symboliczne (miękkie); jest to plik który zawiera ścieżkę do pliku docelowego
  - *(pokazać "na żywo" pliki różnego typu w katalogu test)*

Polecenie "ls –l"

- prawa do pliku:
  - "r" czytanie (katalogi: przeglądanie zawartości)
  - "w" zapis (katalogi: dodawanie, usuwanie pozycji; zmiana nazwy)
  - "x" wykonanie programu (katalogi: przechodzenie przez katalog)
- właściciel i grupa właścicieli pliku:
  - każdy plik ma właściciela i grupę właścicieli
  - nadaje się prawa osobno dla właściciela, grupy właścicieli i pozostałych

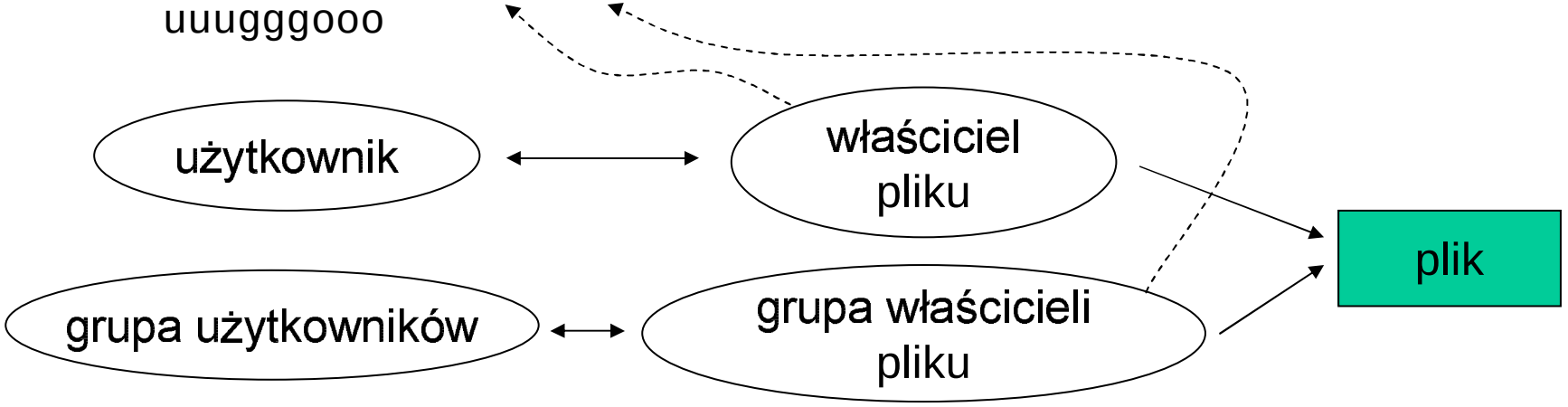
```
ls -l plik1.txt
-rwxr-xr-x 5 aaa bbb 1000 May 20 7:35 plik1.txt
 uuugggooo
```



Prawa do plików w Unixie

- istnieją *użytkownicy* i *grupy użytkowników*
  (każdy użytkownik należy do pewnej grupy użytkowników)
- mamy następujący plik:

```
ls -l plik1.txt
-rwxr-xr-x 5 aaa bbb 1000 May 20 7:35 plik1.txt
 uuugggooo
```



- obliczanie praw do pliku danego użytkownika:
  - jeśli jestem właścicielem pliku to mam prawa "uuu"
  - jeśli nie jestem właścicielem ale należę do grupy właścicieli to mam prawa "ggg"
  - w przeciwnym wypadku mam prawa "ooo"

UWAGA: prawa grupy mogą być większe niż właściciela a wtedy ...

Polecenie "chmod"

- służy do modyfikowania praw do plików i katalogów:

chmod { u, g, o, a } { +, -, = } **prawa plik**

"u" – user
"g" – group
"o" – other
"a" - all

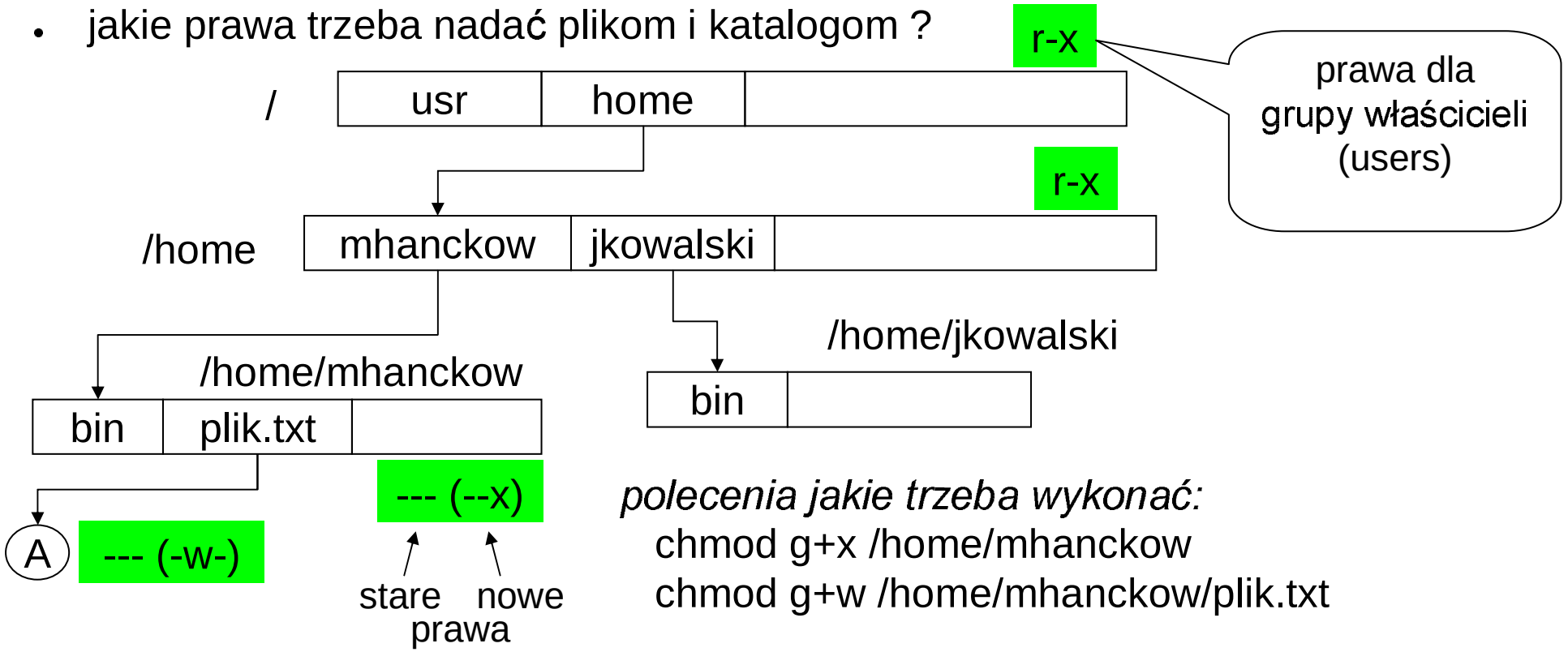
"+" – dodaj prawa
"-" – usuń prawa
"=" – dokładnie takie prawa

przykłady:

```
chmod u=rw plik.txt
chmod u=rw,g-r,o-rw plik.txt
chmod u+x,go-x moj_katalog
```

Zastosowania praw do plików

- mamy 2 *użytkowników*: mhanckow, jkowalski
- mhanckow, jkowalski $\in$ users, gdzie users jest *grupą użytkowników*
- wszystkie pliki i katalogi mają grupę właścicieli users
- chcemy aby jkowalski miał prawo zapisu do (A) ...
- jakie prawa trzeba nadać plikom i katalogom ?



polecenia jakie trzeba wykonać:

chmod g+x /home/mhanckow

chmod g+w /home/mhanckow/plik.txt