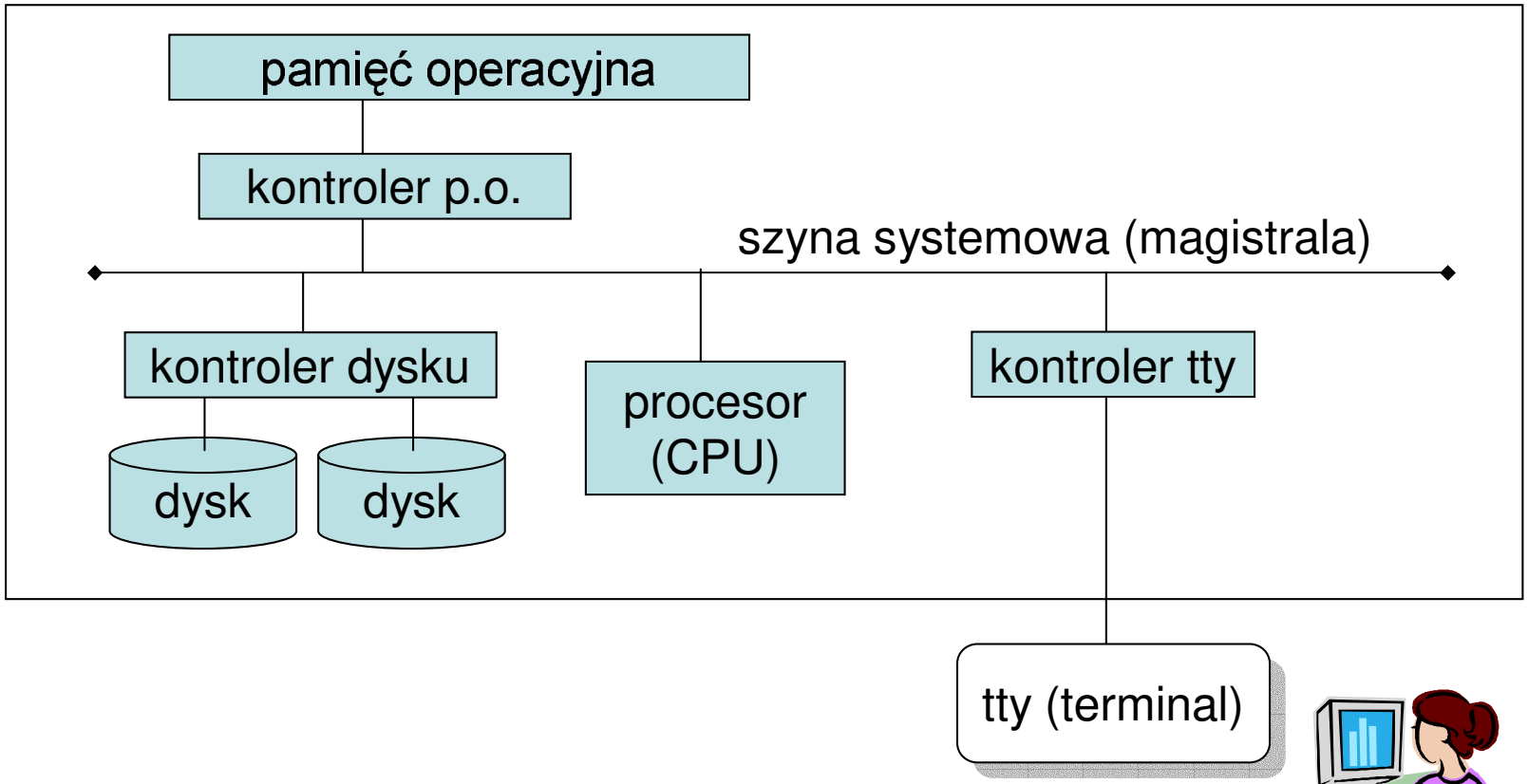
System wejścia/wyjścia

System wejścia/wyjścia

- **jak komputer rozmawia z urządzeniami we/wy (przypomnienie):**
  - kontroler urządzenia we/wy
    - pośredniczy między urządzeniem i procesorem
    - bezpośrednio połączony z szyną systemową i z urządzeniem we/wy (patrz: następny slajd)
  - port
    - "punkt styku" komputera i kontrolera urządzenia; zakres w prz. adr. we/wy; rozkazy in i out
  - odpytywanie
    - pętla w której sprawdza się czy są jakieś dane do odczytania z urządzenia; jeśli są to się je odczytuje ...
  - przerwanie
    - wymusza je kontroler urządzenia gdy dane są gotowe do odczytania; istnieje procedura obsługi przerwania odczytująca dane ...
    - wymusza je kanał DMA gdy dane są już w pamięci operacyjnej
  - PIO [ang. Programmed Input Output]
    - odczytywanie danych przy pomocy rozkazów in i out (przez "port")
  - DMA [ang. Direct Memory Access]
    - kontroler urządzenia zapisuje dane do pamięci operacyjnej za pośrednictwem kanału DMA

System wejścia/wyjścia

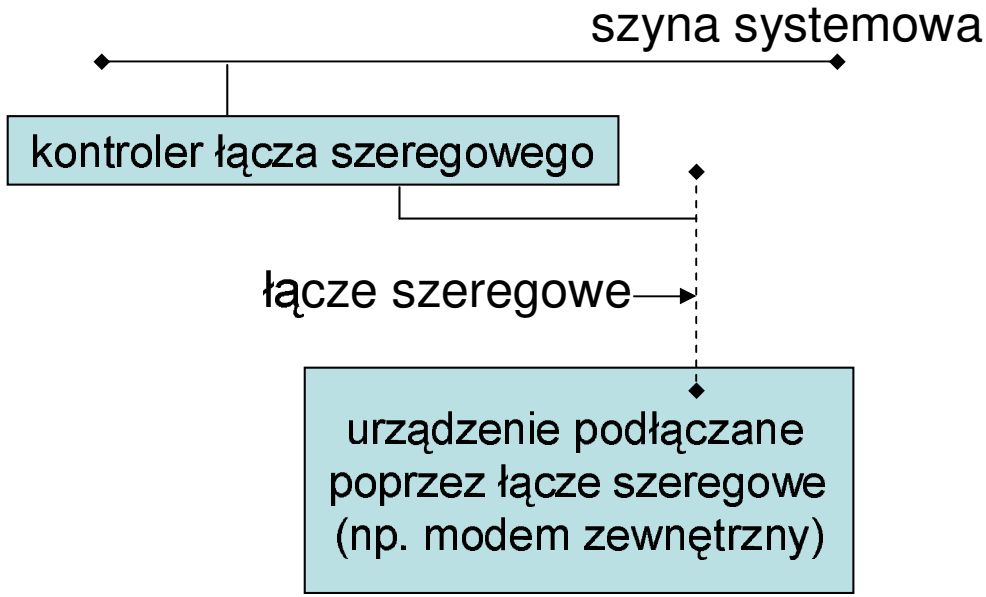
- **budowa systemu komputerowego (przypomnienie):**



sterownik
- sprzętowy = **kontroler** (ang. controller)
- programowy (ang. driver); lub "moduł obsługi urządzenia"; część s.o. (kod)

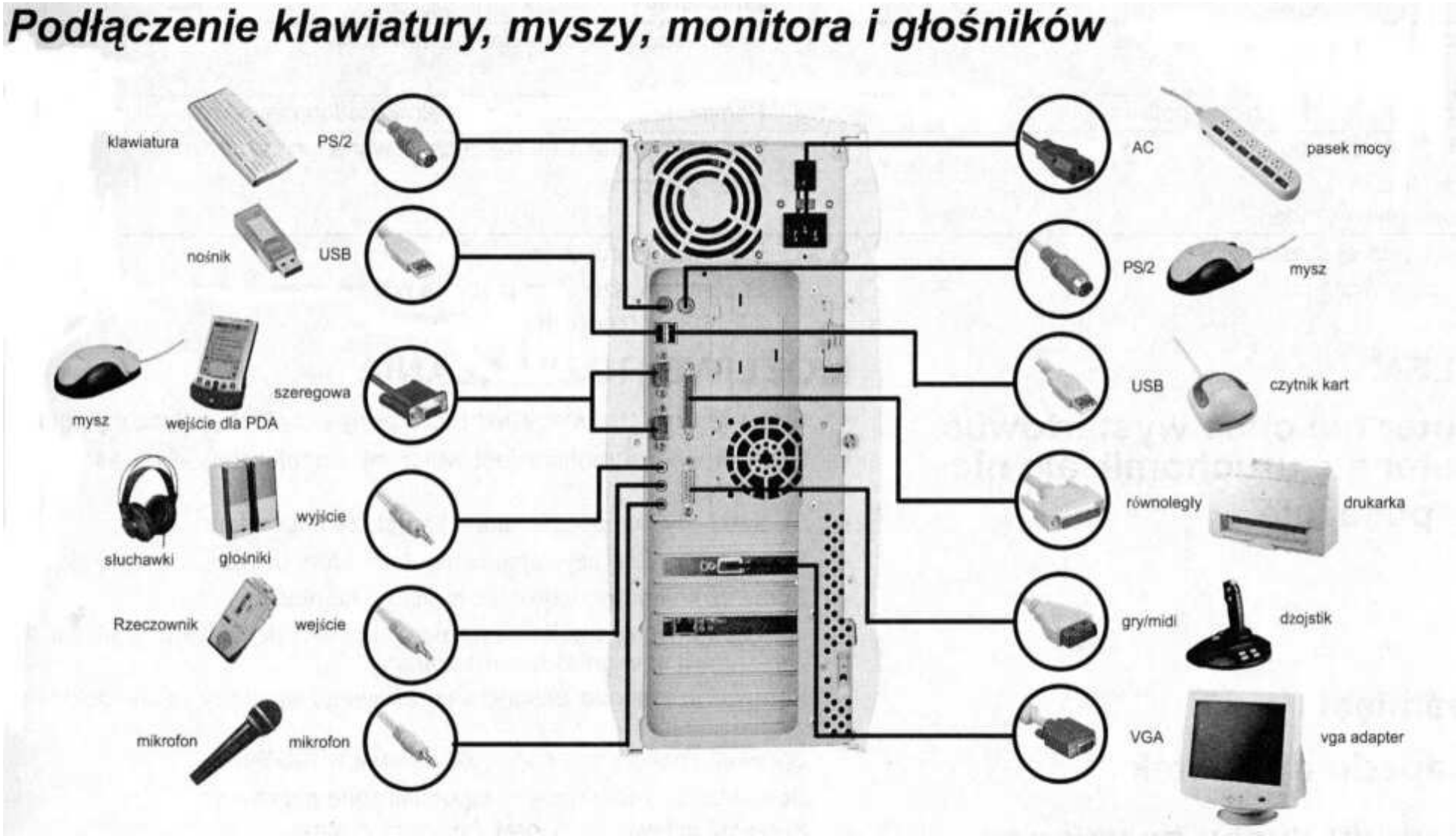
System wejścia/wyjścia

- **urządzenia we/wy:**
  - dysk magnetyczny (dysk twardy, HDD = Hard Disk Drive)
  - stacja dyskietek (FDD = Floppy Disk Drive), stacja CD, stacja DVD
  - monitor (kartę graficzną traktujemy jako kontroler monitora ?)
  - modem wewnętrzny
  - sieć komputerowa (kartę sieciową traktujemy jako kontroler sieci ?)
  - łącze równoległe/ szeregowe, USB (=Universal Serial Bus), PS2
    - niektóre urządzenia są podłączone **za pośrednictwem kontrolera** do szyny systemowej
    - są też urządzenia które podłącza się do innych "szyn"– takich jak np. "łącze szeregowe" czy USB ...
  - klawiatura, mysz
  - modem zewnętrzny
  - drukarka, skaner



System wejścia/wyjścia

- **podłączanie urządzeń we/wy do komputera (tych podłączanych przez "łącze szeregowe", USB, itp)**



System wejścia/wyjścia

- **podłączanie urządzeń we/wy do komputera za pomocą USB ...**

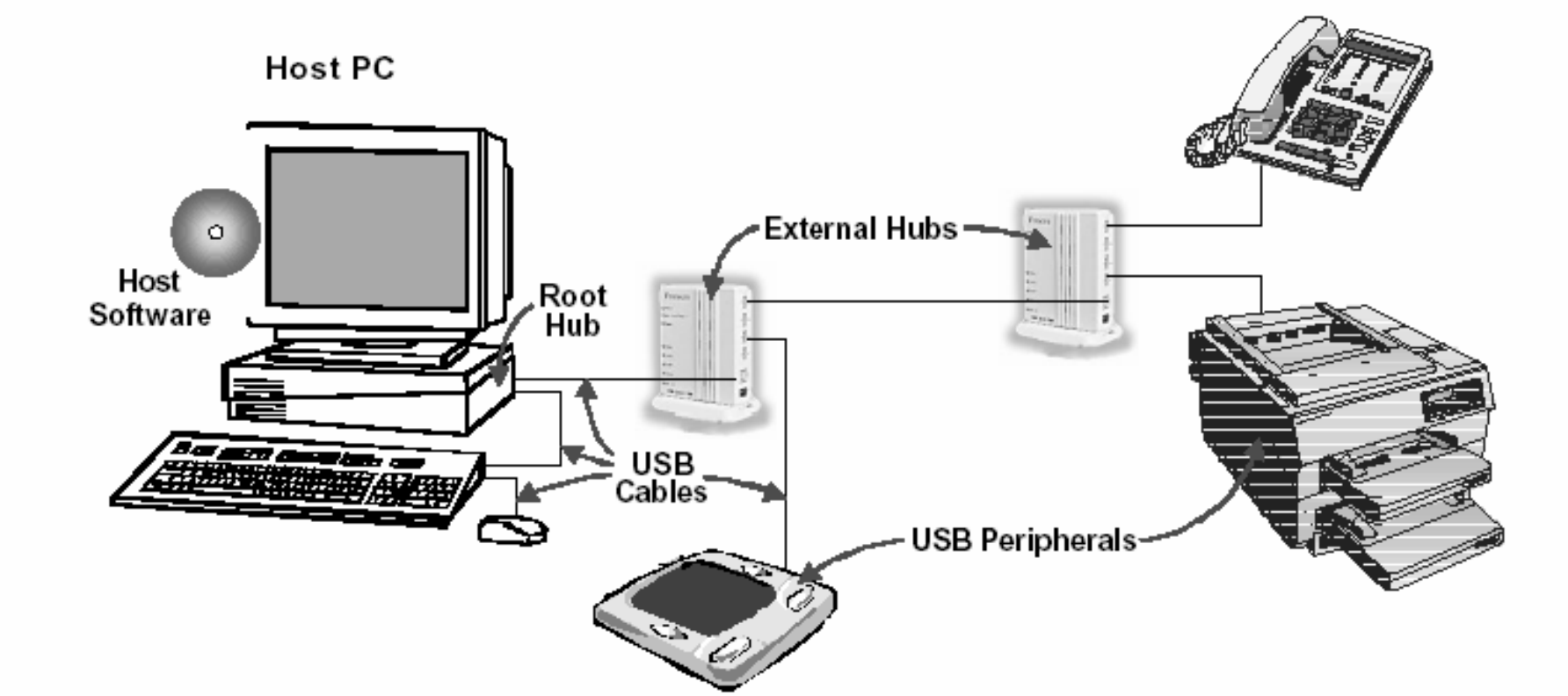


Figure 1. Example USB 1.1 System Configuration

System wejścia/wyjścia
struktura we/wy w s.o.

- s.o. ukrywa szczegóły urządzeń we/wy przed programami
  - przykład/Unix: jednolity dostęp do różnego typu urządzeń we/wy za pomocą plików specjalnych (np. "surowy" dostęp do partycji dysku twardego poprzez plik /dev/hda1)
- jak obsługuje się urządzenia we/wy ?
  - wydzielony składnik s.o. służący do obsługi urządzenia we/wy to tzw "**moduł obsługi urządzenia**"= "**sterownik (programowy)"**= [ang.] **driver**
  - dla każdego typu urządzenia jest osobny sterownik
  - s.o. ma zazwyczaj wbudowane sterowniki wielu typowych urządzeń we/wy (np. stacji dyskietek czy dysku twardego)
  - każdy sterownik "implementuje pewien standardowy interfejs", czyli musi dostarczyć standardowy zestaw procedur do wykonywania operacji na urządzeniu we/wy
  - sterownik zawiera m.in. procedury obsługi przerwań generowanych przez urządzenie we/wy (tylko sterownik wie jak odczytać dane z urządzenia !)
- jak obsługuje się "nowe" urządzenia we/wy ?
  - można dodać nowy sterownik do s.o. (zazwyczaj jest on wczytywany przy uruchamianiu s.o.)
  - Linux: sterowniki mogą być w tzw "modułach ładowalnych" ...
  - DOS: pliki *.sys definiowane w pliku "config.sys"

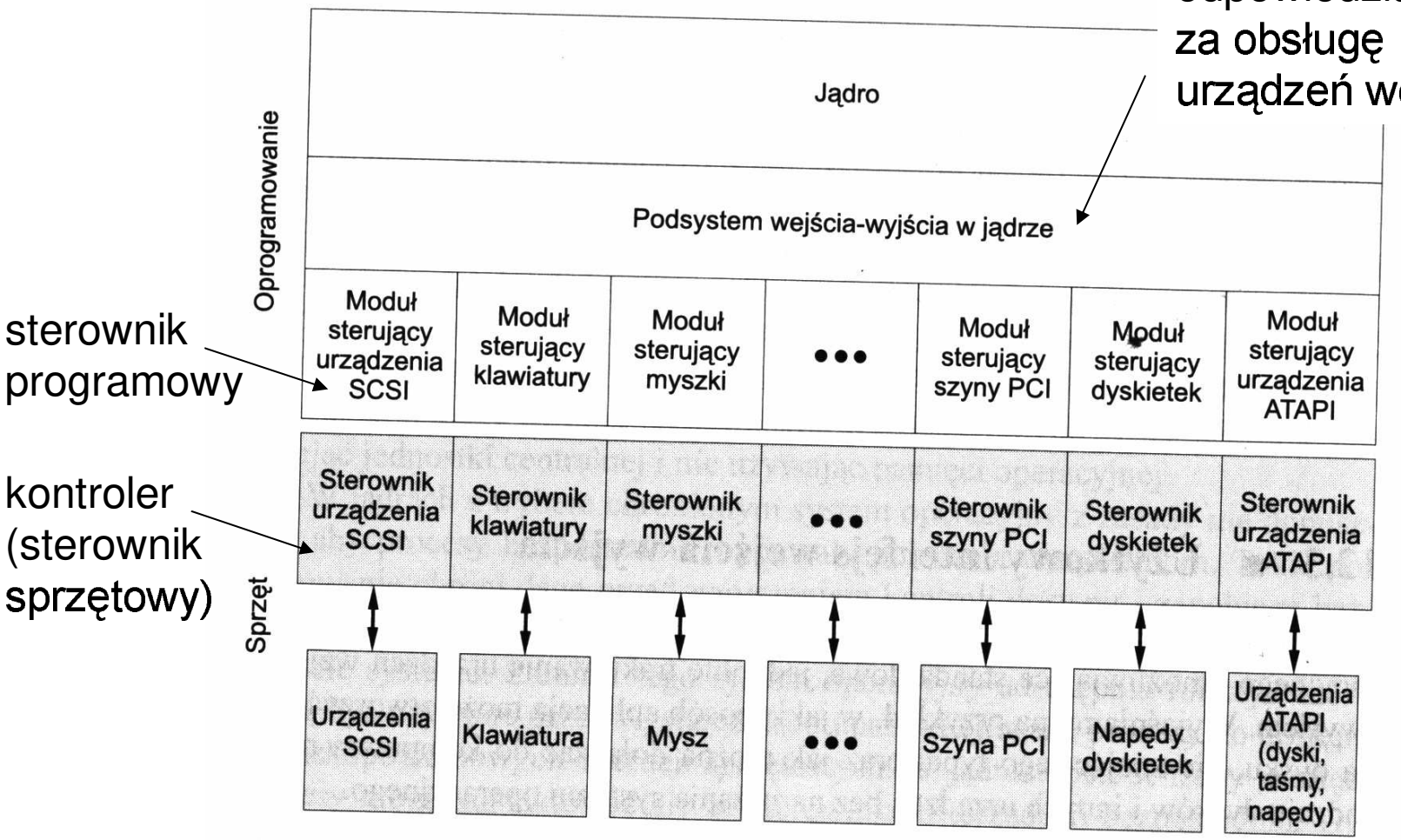
System wejścia/wyjścia

- miejsce "sterownika programowego" w jądrze s.o.

część s.o. odpowiedzialna za obsługę urządzeń we/wy

sterownik programowy

kontroler (sterownik sprzętowy)



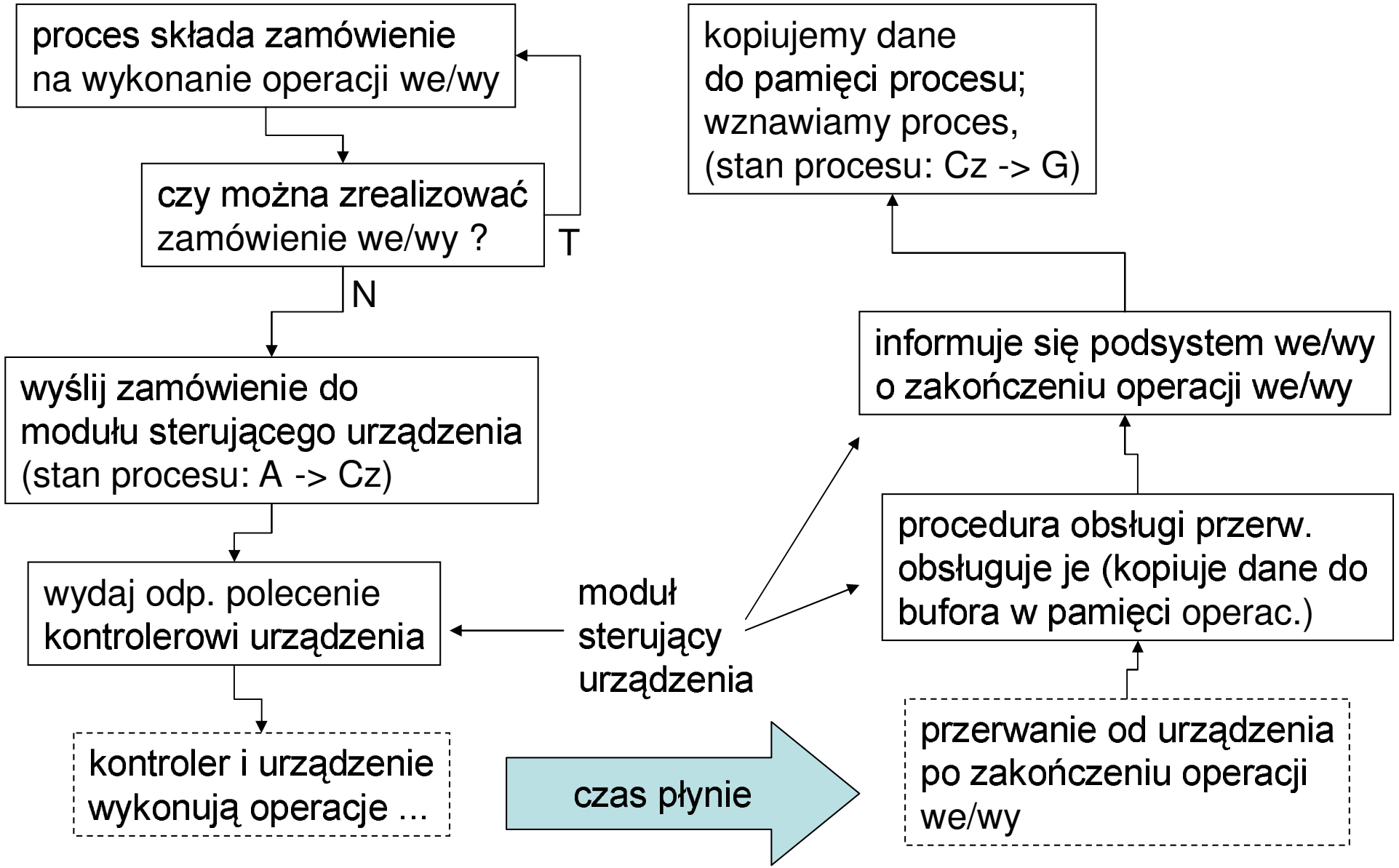
Rys. 12.6 Struktura oprogramowania wejścia-wyjścia w jądrze

System wejścia/wyjścia

- klasyfikacja urządzeń we/wy:
  - urządzenia blokowe/ znakowe
    - blokowe: np. dysk
    - znakowe: np. klawiatura (czyta się po 1 znaku)
  - dostęp sekwencyjny/ swobodny
    - sekwencyjny: np. łącze szeregowe
    - swobodny: np. dysk
  - synchroniczne/ asynchroniczne
    - synchroniczne: przesyłają dane w przewidywalnym czasie; np. dysk
    - asynchroniczne: np. klawiatura (nie wiadomo kiedy ktoś naciśnie klawisz)
  - dzielenie/ wyłączność
    - dzielenie: np. dysk może być używany równocześnie przez wiele procesów
    - wyłączność: np. drukarka (jeśli nie używa się spool-ingu), stacja taśm magnet.
  - szybkość działania
    - szybkie: dysk, karta sieciowa
    - wolne: klawiatura
  - tryb dostępu (czytanie, pisanie, czytanie i pisanie)
    - czytanie: CD-ROM, pisanie: drukarka, czytanie i pisanie: dysk magnet.

System wejścia/wyjścia

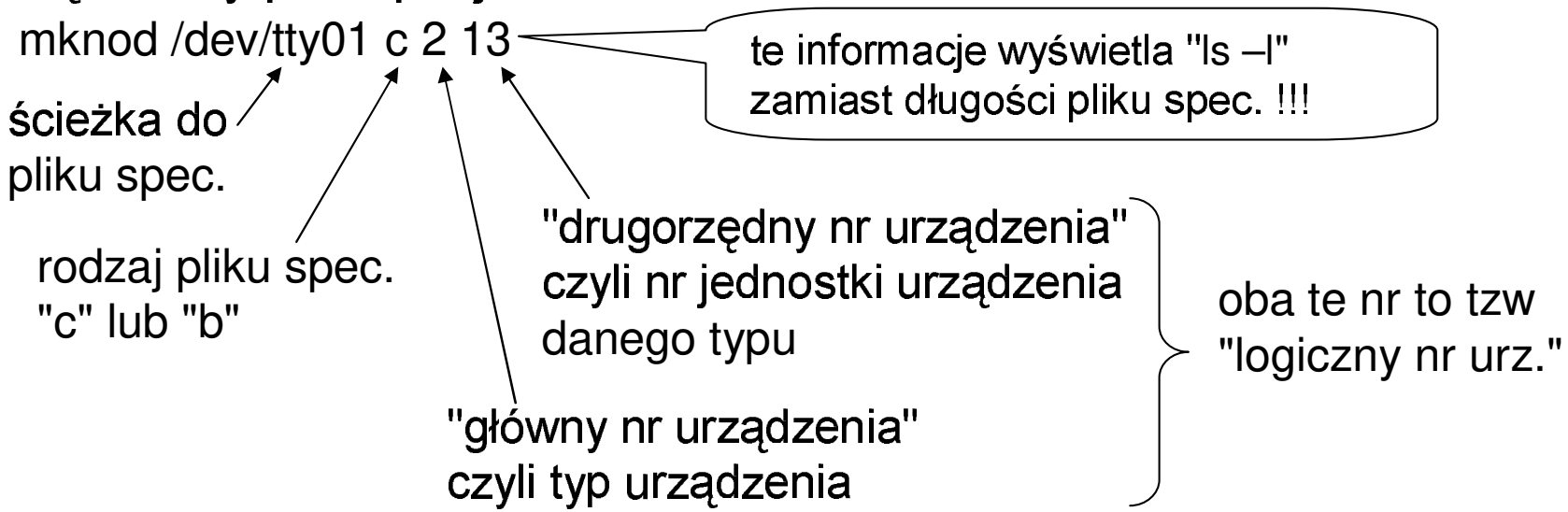
- jak działa podsystem we/wy w s.o. ?



System wejścia/wyjścia

podsystem we/wy w Unix-ie

- urządzenia są dostępne poprzez pliki specjalne w kat /dev
  - /dev/tty01 – dostęp do terminala tekstowego
  - /dev/fd0 – "surowy" dostęp do dyskietki (surowy = dyskietka jest traktowana jako ciąg bloków/sektorów)
- pliki specjalne
  - znakowe "c" (np. /dev/tty01)
  - blokowe "b" – używa się pamięci podręcznej ("podręcznej pamięci buforowej") przy dostępie do urządzenia poprzez ten plik (np. /dev/fd0)
- jak się tworzy pliki specjalne ?



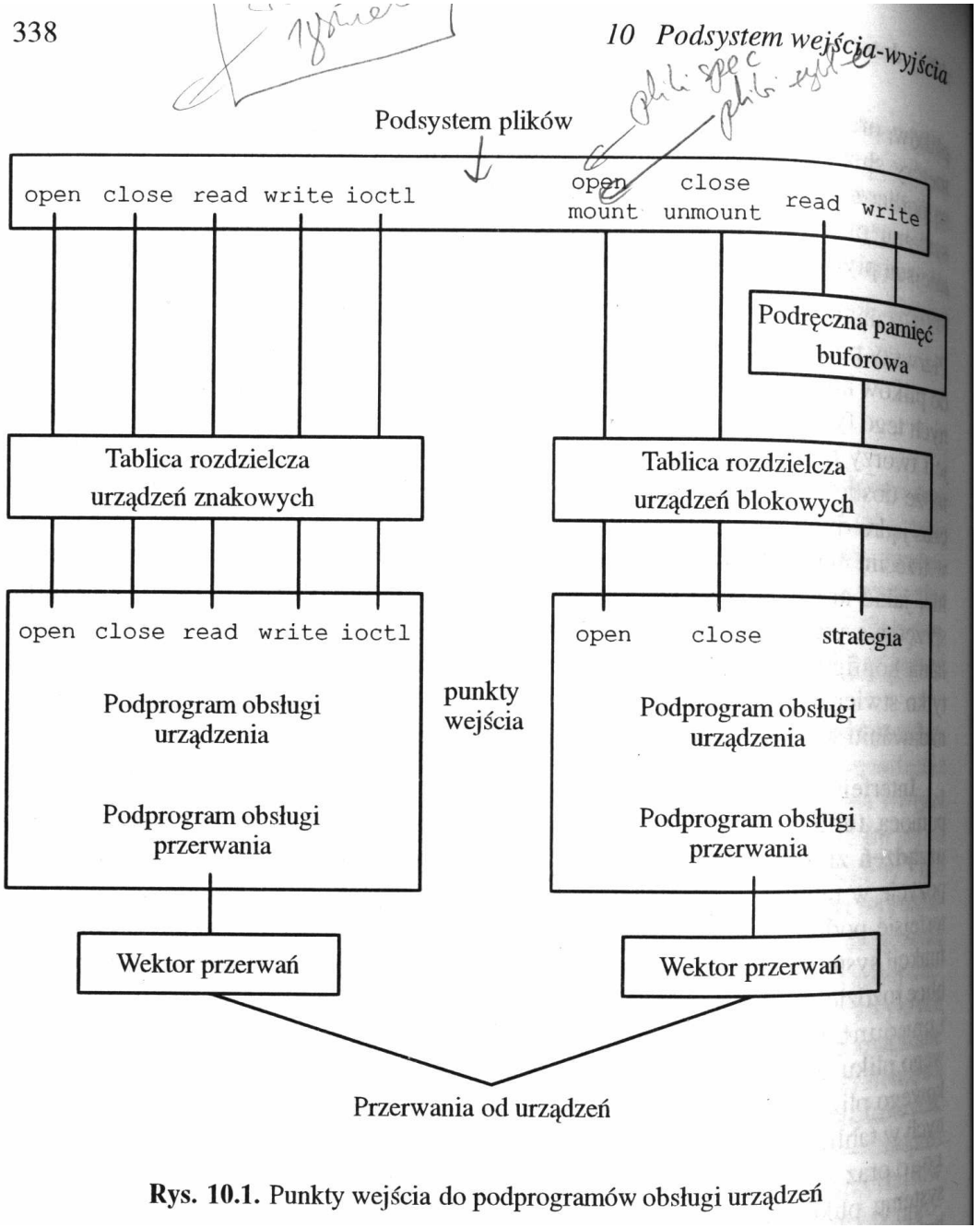
System wejścia/wyjścia

podsystem we/wy w Unix-ie c.d.

- **co się dzieje przy wykonaniu kodu:**

```
d=open("/dev/tty01", ...);
  // tty01 to plik spec. znakowy
read(d, buf, 10);
```

- odp: wywoływanie fun. sys. open() i read() powoduje uruchomienie odp. funkcji których adresy są w tablicy rozdzielczej urz. znakowych (główny nr urz. jest indeksem do tej tablicy !!!)
- są dwie tablice rozdzielcze:
  - urządzeń znakowych
  - urządzeń blokowych
- dostęp do plików zwykłych oraz do plików spec. blokowych odbywa się poprzez tablicę urządzeń blokowych ! (bloki są odczytywane za pośrednictwem pamięci podręcznej; procedura "strategia" zajmuje się kopiowaniem bloków miedzy urz. a pam. podr. !)

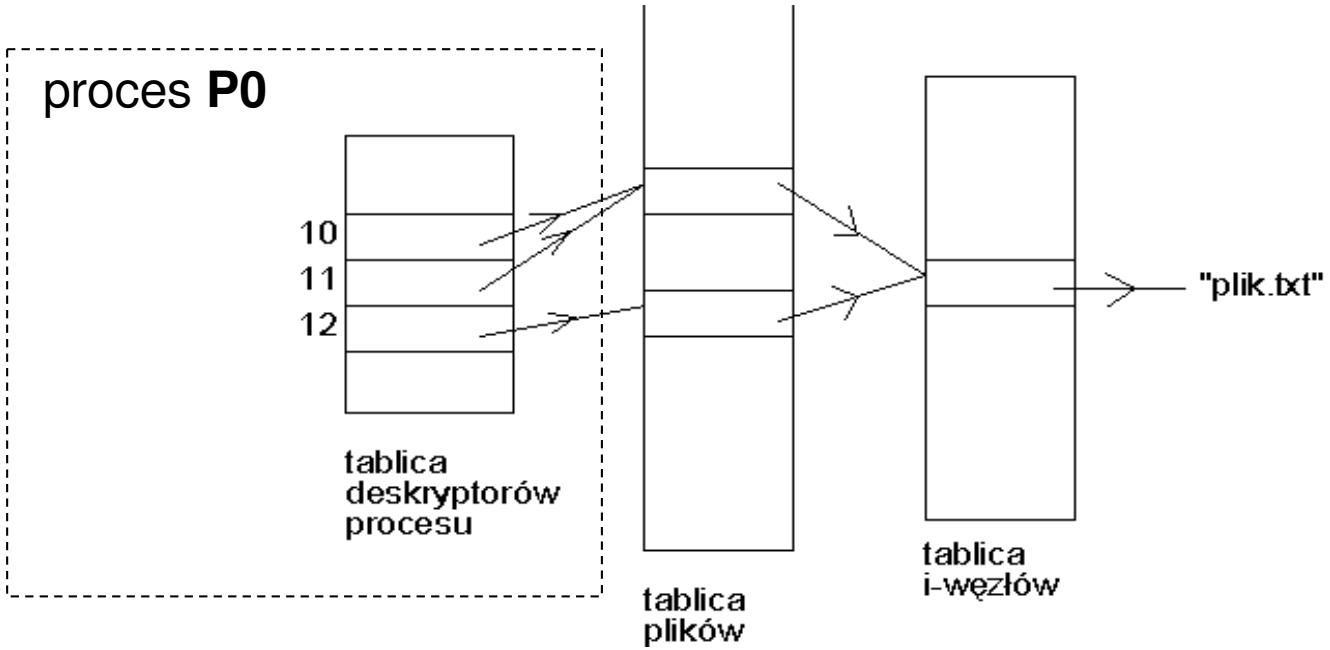


Rys. 10.1. Punkty wejścia do podprogramów obsługi urządzeń

System wejścia/wyjścia

podsystem we/wy w Unix-ie c.d.

- **deskryptory plików**
  - *tablica deskryptorów procesu* (deskryptory to indeksy elementów tej tablicy)
  - *tablica plików* (tu przechowuje się "bieżącą pozycję" !)
  - *tablica i-węzłów* (każdy element tej tablicy identyfikuje plik)
- Każdy proces ma własną *tablicę deskryptorów*.
- Na danej maszynie istnieje jedna *tablica plików* i jedna *tablica i-węzłów* (chodzi o i-węzły skopiowane do pamięci operacyjnej !!!)
- Na poniższym rysunku deskryptory 10,11,12 udostępniają ten sam plik "plik.txt". Jeśli teraz przesuniemy bieżącą pozycję poprzez desk 10 to ma to wpływ na bieżącą pozycje poprzez desk 11 lecz nie poprzez desk 12.



System wejścia/wyjścia

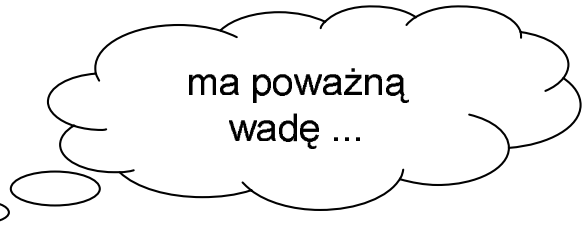
jak zwiększyć wydajność we/wy ?

- planowanie dostępu do dysku
  (zamówienia do dysku nie są obsługiwane na zasadzie FCFS = "pierwszy przyszedł pierwszy obsłużony" lecz w bardziej wyrafinowany sposób ...)
- buforowanie
  (aby ograniczyć liczbę operacji we/wy; *zasada*: piszemy do bufora, "prawdziwy" zapis następuje dopiero po przepełnieniu się tego bufora; stosowane np. przez język C - funkcja printf() itp)
- pamięć podręczna
  (pamięć podręczna dysku, zwana też "podręczną pamięcią buforową"; przechowuje się bloki dyskowe w pamięci operacyjnej ...)
- spool-ing

System wejścia/wyjścia

we/wy z perspektywy programisty (Unix)

- rozważmy taki problem:
  mamy 6 deskryptorów x_1, x_2, x_3, y_1, y_2, y_3;
  chcemy przepisywać dane między parami deskryptorów x_i -> y_i
  jak to zrobić ???
  [gdy czekamy na dane w x_1, w międzyczasie mogą się pojawić dane w x_2 lub x_3 ...]
- możliwe rozwiązania:
  - 3 wątki obsługujące pary deskryptorów



  - przełączenie desk. x_i w tzw **tryb nieblokujący**, przy pomocy
    fcntl(x_i, ... O_NONBLOCK); // dla i=1,2,3
    a następnie próby odczytu danych z x_1, x_2, x_3 w pętli;
    gdy nie ma nic do przeczytania fun. sys. zwraca błąd
    read(x_i, ...) = -1
    errno = EAGAIN
  - użycie fun. sys. **select()** pozwalającej czekać na możliwość czytania na *zbiorze* deskryptorów:
    select(Read, ...)

zmienna zawierająca zbiór desk.

Opisy konkretnych systemów operacyjnych

System operacyjny Unix

System operacyjny Unix;
historia

- pierwsze wersje Unix-a
  - 1969, Ken Thompson, Dennis Ritchie
    firma: AT&T, Bell Laboratories
    maszyna: PDP-7
  - silny wpływ s.o. MULTICS
  - nowe koncepcje:
    - interpreter poleceń użytkownika (shell) jest "zwykłym" programem (może zostać wymieniony na inny, jest wiele interpreterów: sh, ksh, csh)
    - komendy wewnętrzne i zewnętrzne (osobne programy) – z pkt widzenia użytkownika właściwie nie ma między nimi różnicy ...
    - Unix został zaprogramowany prawie w całości w języku C
      (język C został właśnie do tego celu wymyślony w Bell Laboratories !!!)
      powód:
      - łatwiej się programuje w C niż w assemblerze
      - łatwość przenoszenia Unix-a na inne maszyny ...

System operacyjny Unix;
historia

- ważniejsze wersje Unix-a
  - "**Unix System V**" ("V" czytaj jako "pięć")
    firma: AT&T - *to jest prawdziwy Unix !*
    "**SVR2**" = "Unix System V Release 2" (wydanie 2)
    ← opisany szczegółowo w książce Bacha "Budowa s.o. Unix"
    "**SVR3**" zawiera m.in.:
    - strumienie we/wy STREAMS (pliki specjalne są realizowane przez takie strumienie; można do str we/wy wstawiać moduły przetwarzające dane; w przypadku str we/wy obsługującego terminal, moduł przetwarzający może w specjalny sposób interpretować niektóre znaki np. generowane przez klawisze strzałek na klawiaturze ...)
    - RFS = Remote File Sharing; jest to sieciowy system plików, podobny do NFS (=Network File System) firmy SUN !

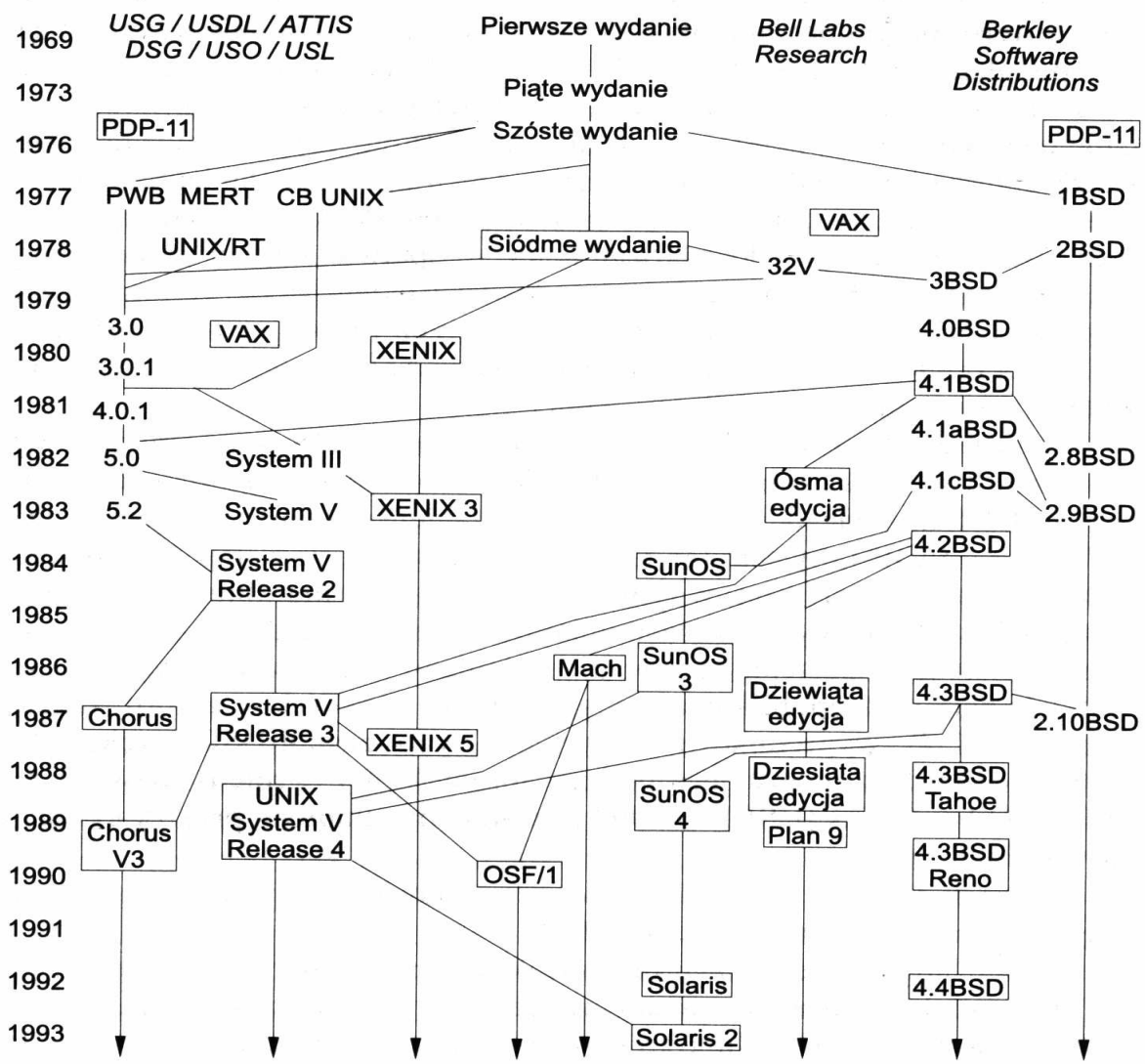
    "**SVR4**" to obecnie obowiązująca wersja;
    - dodaje cechy 4.3BSD, SunOS
    - zgodny ze standardem POSIX

System operacyjny Unix;
historia

- ważniejsze wersje Unix-a (czy też systemów "unixo-podobnych")
  - "**(Unix?) BSD"** = "Berkeley Software Distribution"
    University of California, Berkeley
    "**3BSD**" – 1978, zawierał pamięć wirtualną (stronicowanie na żądanie)
    "**4BSD**" – zawiera fun. sys. do komunikacji przez siec z protokołami TCP/IP
    tzw "gniazda BSD"
    "**4.3BSD**", 1986
    "**4.4BSD**", 1993 – ostatnie wydanie BSD
- prace standaryzacyjne:
  - *cel*: aby łatwo było przenosić oprogramowanie między systemami
  - **IEEE** = Institute for Electrical and Electronics Engineers
  - **ISO** = International Standards Organisation
  - IEEE + ISO stworzyły standard **POSIX** = Portable Operating System Interface for Computer Environment (głównie zbiór fun. sys. !)
  - **X/Open** – konsorcjum producentów sprzętu i oprogramowania; stworzyło **XPG3** i **XPG4** = X/Open Portability Guide

System operacyjny Unix; ***historia***

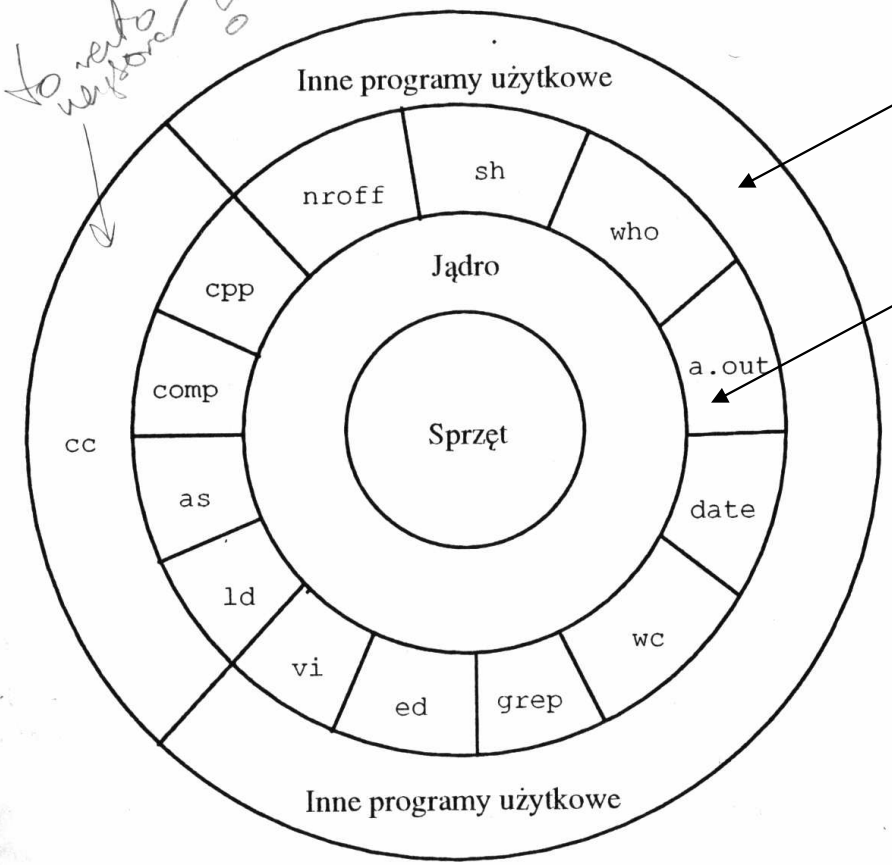
- dziedziczenie cech przez systemy unikso-podobne ...



Rys. 21.1 Historia wersji systemu UNIX (zaczerpnięty z: Leffler S., Karels M., McKusick M., Quarterman J.: *The Desing and Implementation of the 4.3 BSD UNIX Operating System*, (fig 1.1, p.5). ©1989 Addison Wesley Publishing Company Inc., Reprinted by permission of Addison Wesley Longman)

System operacyjny Unix;
budowa

- tradycyjny "diagram kołowy"
  - zewnętrzne warstwy odwołują się do wewnętrznych
  - dotyczy wszystkich wersji Unix-a



Rys. 1.1. Architektura systemów unixowych

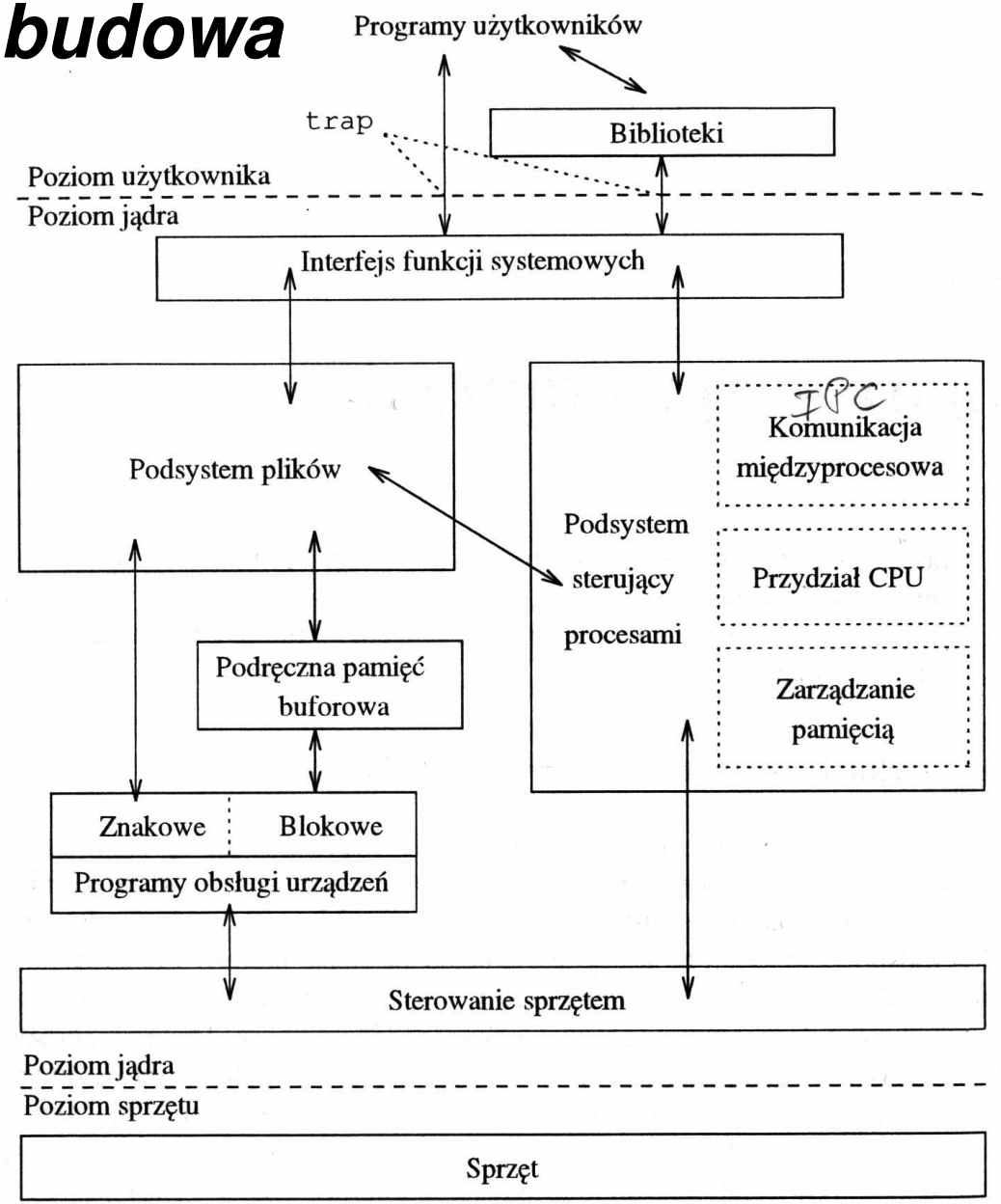
programy użytkowników
(często uruchamiają inne programy)

programy używające tylko fun. sys.

System operacyjny Unix;
budowa

budowa Unix-a SVR2

- wg książki Bacha
- główne podsystemy:
  plików,
  procesów,
  we/wy

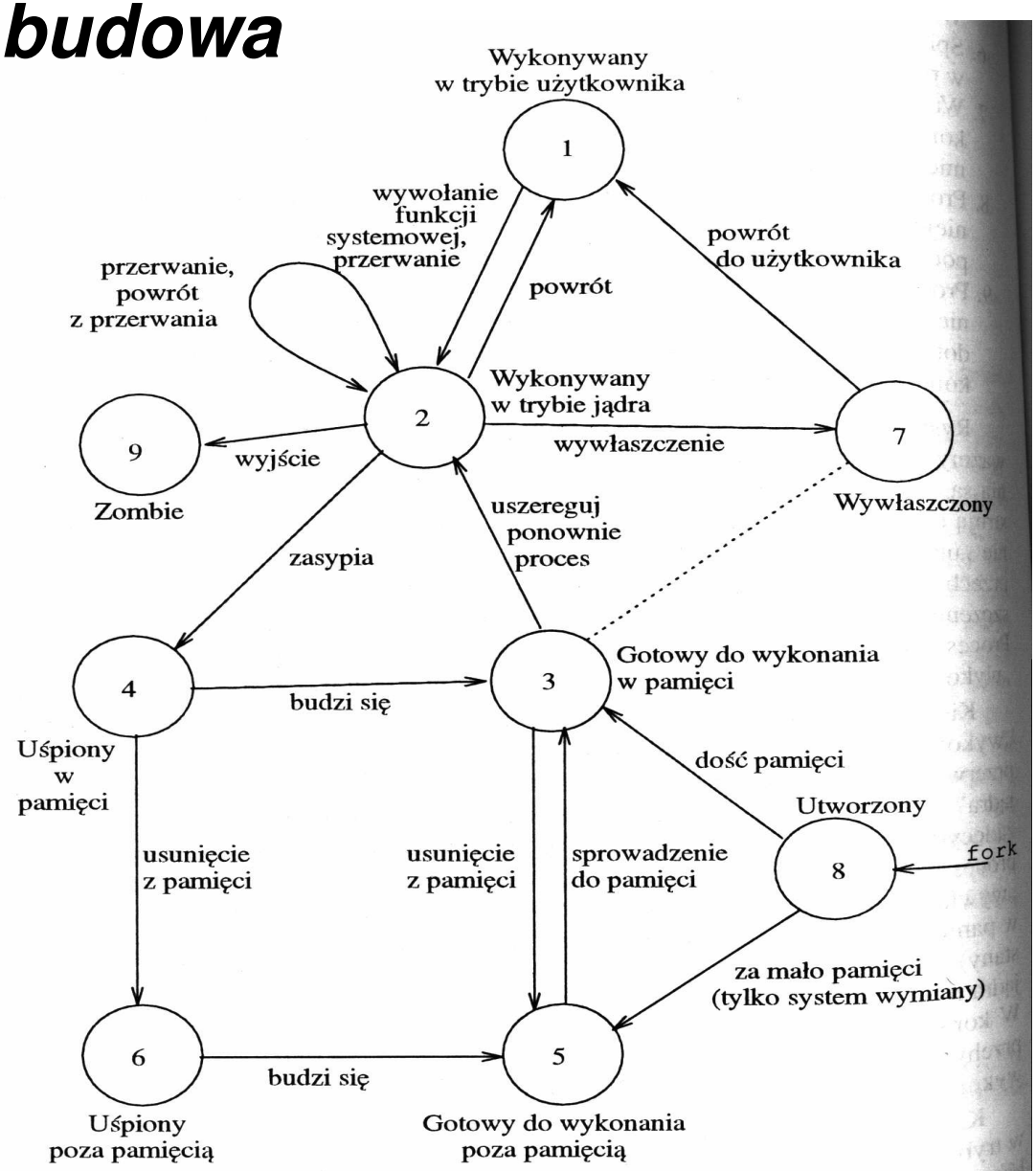


Rys. 2.1. Diagram blokowy jądra systemu

System operacyjny Unix;
budowa

budowa Unix-a SVR2

- wg książki Bacha
- diagram stanów procesu



Rys. 6.1. Diagram przejść międzystanowych procesu