Składniki systemu operacyjnego

- zarządzanie procesami
  - *czym się zajmuje ten składnik s.o. ?*
  - tworzenie/usuwanie procesów
  - przełączanie procesora między procesami
    - mamy 1 procesor i wiele procesów
    - decydowanie na który gotowy do działania proces się przełączyć = "planowanie (przydziału) procesora"
  - rozwiązywanie problemów współbieżnie działających procesów
    - synchronizacja procesów (wstrzymywanie gdy to jest potrzebne)
    - komunikacja między procesami
- zarządzanie pamięcią operacyjną
  - proces potrzebuje obszaru w pamięci operacyjnej do przechowywania kodu programu i danych
  - przydzielanie/zwalnianie obszarów pamięci
  - zamiana adresów logicznych komórek pamięci na fizyczne
  - segmentacja, stronicowanie itp. (= sposoby zarządzania p.o.)

Składniki systemu operacyjnego

- zarządzanie plikami i katalogami
  - tworzy jednolity obraz plików i katalogów w pamięciach pomocniczych różnych typów
  - operacje na plikach i katalogach
  - zarządzanie pamięcią pomocniczą, czyli "implementacja systemu plików"
  - "planowanie" dostępu do dysku
- zarządzanie wejściem/wyjściem
- praca sieciowa
- system ochrony
  - mechanizmy zapewniające że dostęp do rozmaitych zasobów mają tylko osoby uprawnione
- interpreter poleceń
  - DOS/ Win: command.com, cmd.exe
  - Unix: sh, csh, ksh, bash

Zarządzanie procesami

Zarządzanie procesami

- *def* procesu = "uruchomiony program", któremu przydzielono pewne zasoby takie jak pamięć operacyjna (w której przechowuje się kod i dane), czas procesora, oraz inne ...

  proces – to coś dynamicznego
  program – to coś statycznego (plik z kodem)

- składniki procesu:
  - sekcja kodu
  - wartości rejestrów procesora
    (w tym licznik rozkazów; przykładowo 8086: IP)
  - sekcja stosu
  - sekcja danych

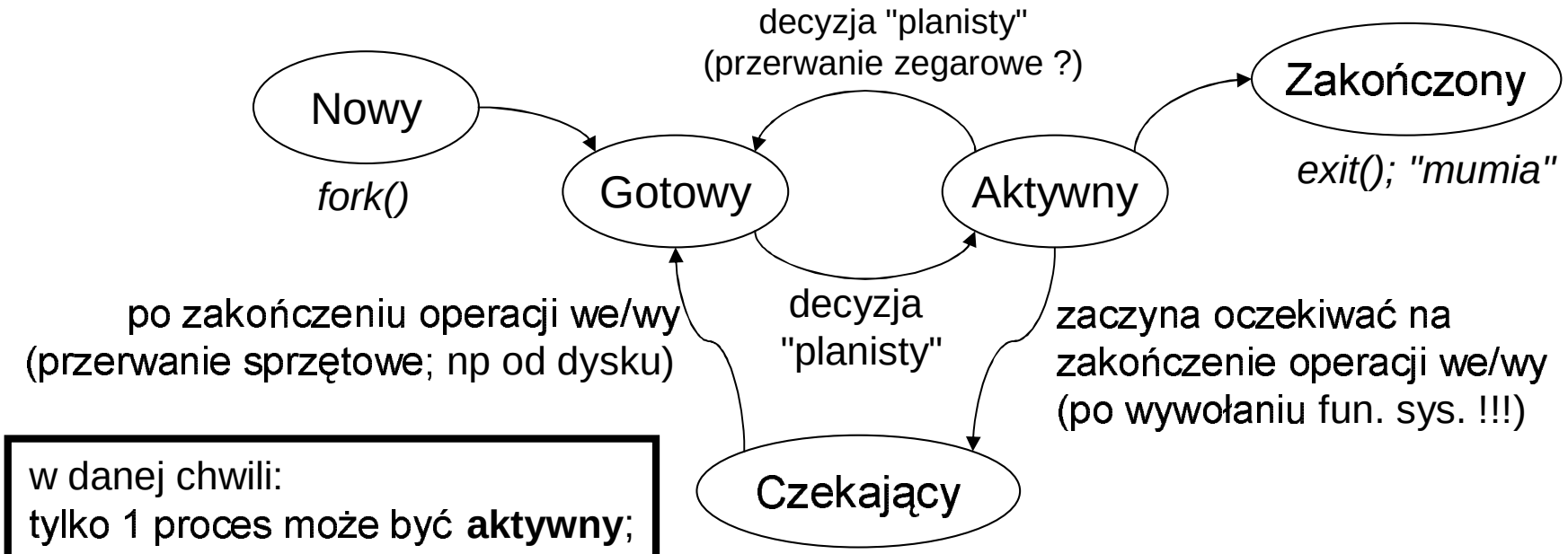
na stosie przechowuje się:
zmienne lokalne procedur (j. programowania)
adresy powrotne procedur

w sekcji danych przechowuje się:
zmienne globalne

... specjalne rozkazy procesora 8086 do operowania na stosie: push/pop, także call/ret i iret

Zarządzanie procesami; **stany procesu**

- proces może się znajdować w jednym ze *stanów*:
  Nowy, Zakończony,
  Gotowy (= gotowy do działania),
  Aktywny (= działający; jego kod jest wykonywany przez procesor),
  Czekający (= czeka na jakieś zdarzenie),
- w czasie działania proces zmienia swój stan
  wg następującego *"diagramu stanów procesu"*:



w danej chwili:
tylko 1 proces może być **aktywny**;
może być wiele procesów
czekających lub **gotowych**

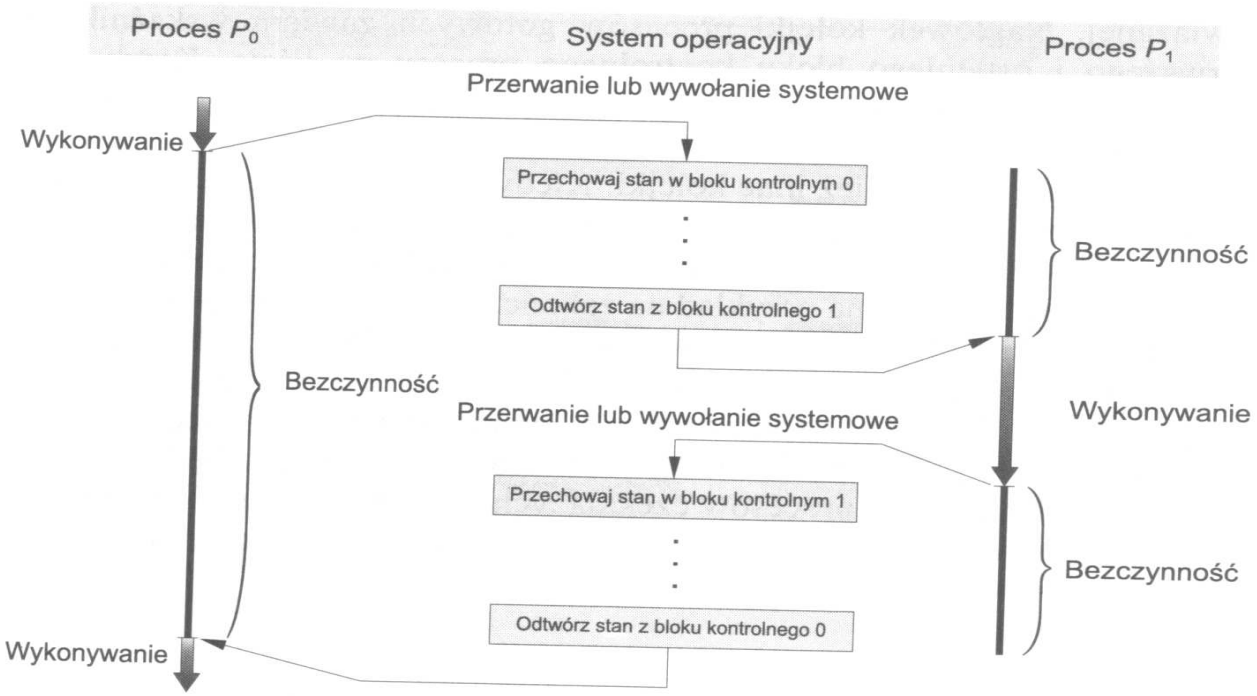
... obok strzałek wyjaśnienie przyczyny zmiany stanu

Zarządzanie procesami; **Blok Kontrolny Procesu (BKP)**

- (ang. PCB = Process Control Block)
- każdy proces jest reprezentowany przez swój BKP
- BKP zawiera:
  - stan procesu (patrz *diagram stanów procesu*)
  - licznik rozkazów i wartości pozostałych rejestrów procesora *(licznik rozkazów wskazuje na bieżący rozkaz kodu !)*
  - informacje dla "planowania przydziału procesora"
  - informacje o "zarządzaniu pamięcią operacyjną":
    - wartość rejestrów baza/limit (patrz: ochrona pamięci)
    - tablice stron i/lub tablice segmentów (omówimy to później ...)
  - informacje do rozliczeń
  - informacje o stanie urządzeń we/wy
    - wykaz otwartych plików (Unix: *tablica deskryptorów procesu*)
    - informacje o urządzeniach we/wy przydzielonych procesowi (np. przewijak taśmy magnetycznej)

Zarządzanie procesami; **przełączanie procesora**

- przełączenie się procesora z procesu P0 na proces P1 wymaga:
  - zapamiętania stanu P0 w BKP0
  - odtworzenia stanu P1 z BKP1
- chodzi tu głównie o zapamiętywanie/ odtwarzanie wartości rejestrów procesora (w tym także rej związanych z zarządzaniem pamięcią !); operacja ta jest także nazywana *przełączeniem kontekstu*;
- przełączanie kontekstu marnuje nieco czasu procesora ...

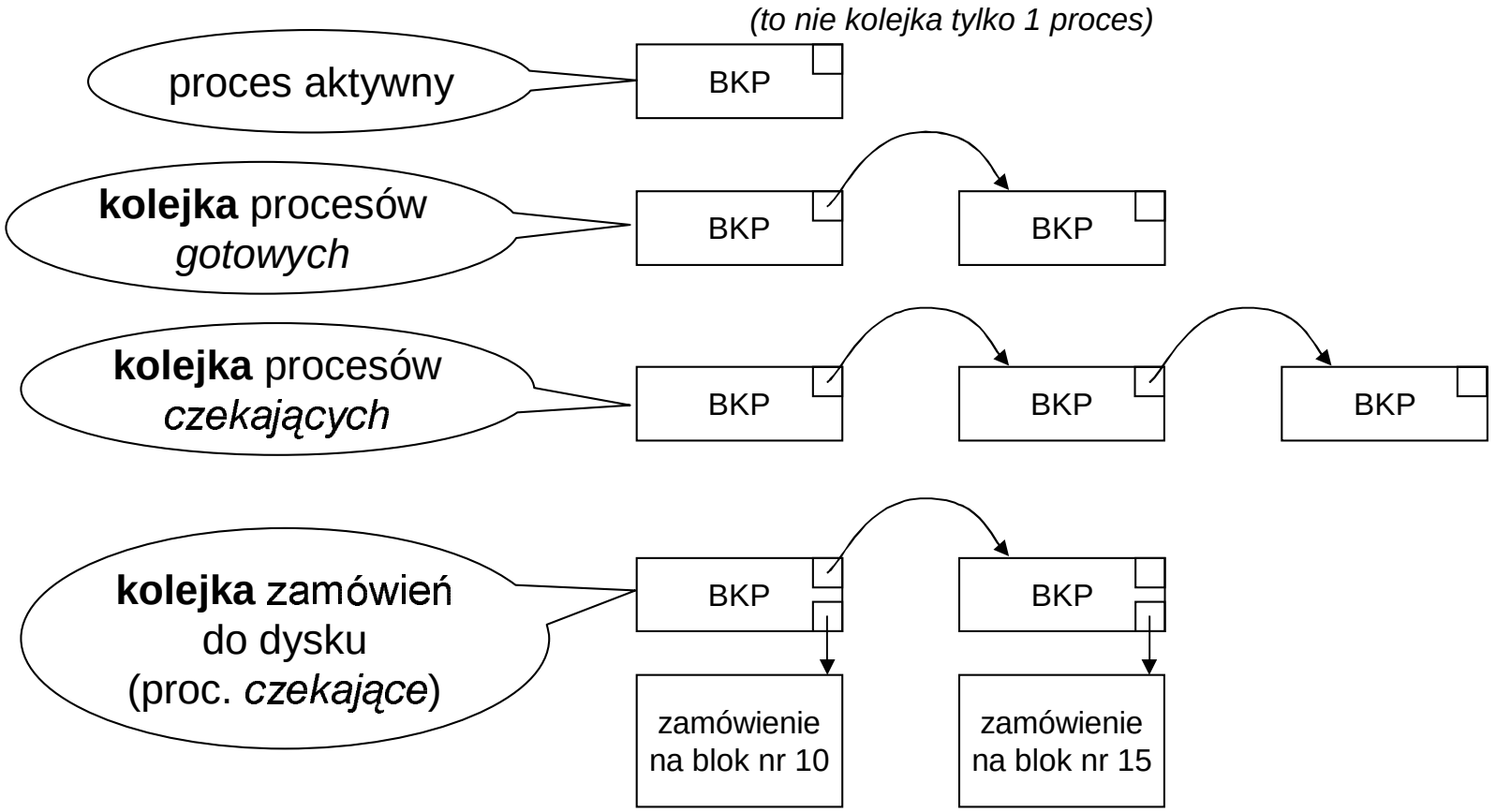


Zarządzanie procesami; **planowanie (przydziału) procesora**

- mamy wiele procesów **gotowych** do działania ...
- *planista (przydziału) procesora* to procedura, która:
  - wybierania spośród procesów gotowych jeden
  - przydziela mu procesor
    (jego stan zmienia się: **gotowy -> aktywny**)
  - poprzedni aktywny staje się gotowy lub czekający (zgodnie z diagramem)
  - planista jest uruchamiany "przez przerwanie":
    - wywołanie fun. sys. (przerwanie programowe)
    - przerwanie sprzętowe od dysku
    - przerwanie zegarowe
- jak to wszystko jest zaimplementowane ?
  Odp: **kolejki procesów**
  (*Uwaga*: to niekoniecznie są kolejki FIFO ...)

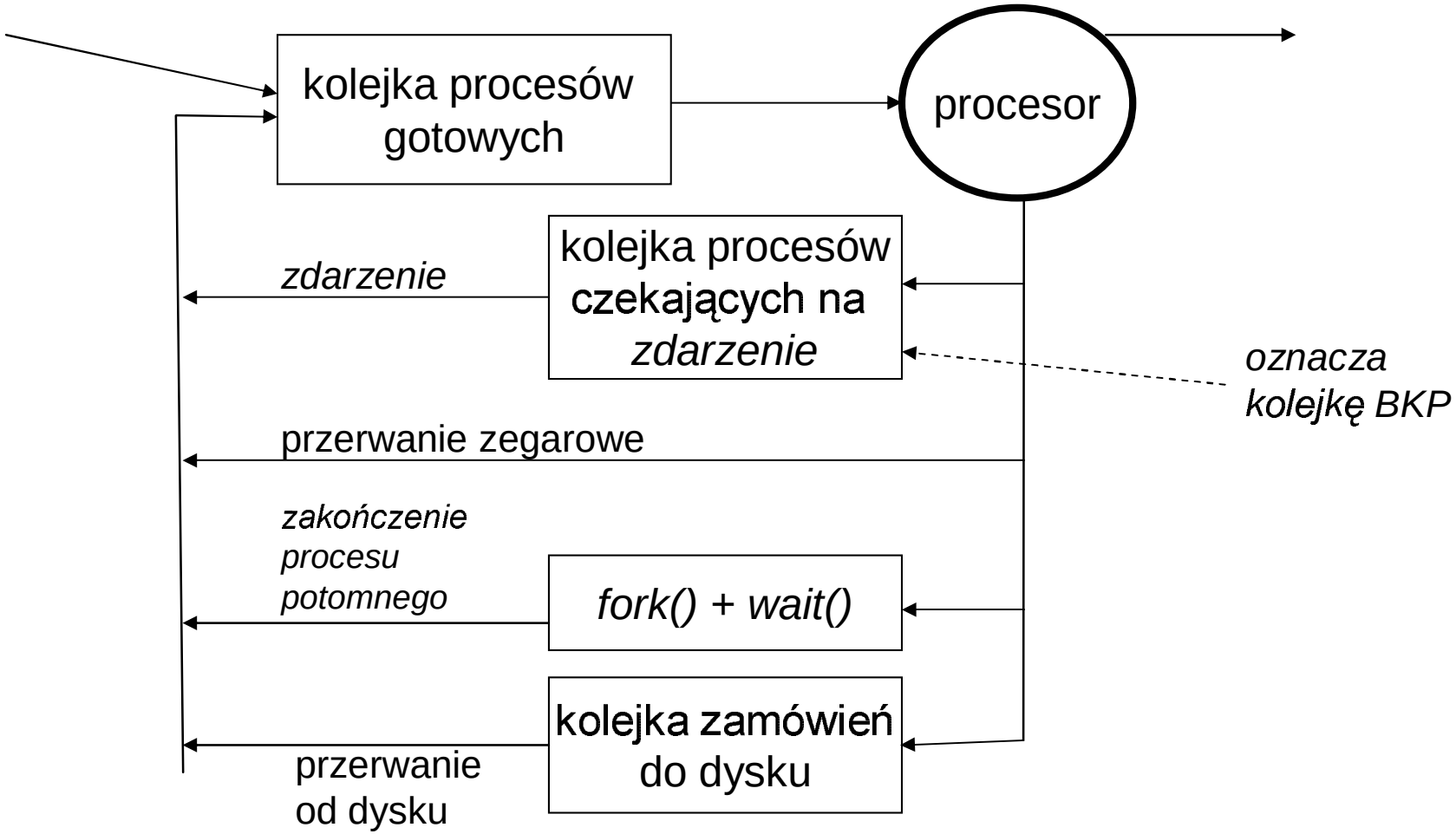
Zarządzanie procesami; **kolejki procesów**

- BKP "przeskakują" między kolejkami procesów
- przeskakiwanie powoduje planista, uaktywniany przez przerwania
- wszystkie kolejki podlegają *planowaniu* (który G->A ?, który Cz->G ? itp)
- ... wizja s.o. jako zbioru BKP przeskakujących między kolejkami w odpowiedzi na przerwania/zdarzenia (s.o. jest sterowany przerwaniami/zdarzeniami)



Zarządzanie procesami; **diagramy kolejkowe**

- przemieszczanie BKP między kolejkami pokazują tzw *diagramy kolejkowe*:



Zarządzanie procesami; **planowanie**

- ogólnie ***planista*** to procedura wybierająca jeden proces z kolejki (np. z kolejki procesów gotowych lub jakiejś innej kolejki)
- "planuje się" w określonym celu, np:
  - żeby zmniejszyć średni czas przebywania procesów w kolejce procesów gotowych
  - żeby zminimalizować czas cyklu przetwarzania procesu, tj czas jaki upływa między chwilą nadejścia procesu a chwilą jego zakończenia
- rodzaje planistów:
  - **planista krótko-terminowy**
    - jest to planista (przydziału) procesora
    - najczęściej uruchamiany ze wszystkich planistów (przez przerwania pochodzące z różnych źródeł ...)
  - **planista średnio-terminowy**
    - występuje w systemach z podziałem czasu
    - gdy procesy nie mieszczą się w pamięci operacyjnej planista średnio-term. podejmuje decyzje żeby niektóre z nich (tymczasowo) zapisać na dysku
    - jest to tzw *wymiana* procesów (ang. swapping)

Zarządzanie procesami; **planowanie c.d.**

- **planista średnio-terminowy c.d.**
  - *uwaga*: wymiana dotyczy "całych" procesów (a nie ich części jak to ma miejsce w tzw *pamięci wirtualnej*)
  - można rozbudować diagram kolejkowy uwzględniając planistę śred-terminowego:

Zarządzanie procesami; **planowanie c.d.**

- **planista długo-terminowy**
  - występuje w systemach wsadowych/wieloprogramowych
  - jest to *planista zadań;* decyduje które zadanie z puli zadań na dysku załadować do pamięci operacyjnej
  - uaktywniany gdy któreś zadanie się zakończy (i zwolni się miejsce w pamięci)
  - powinien ładować do pamięci operac. "najbardziej korzystną" mieszaninę zadań, tj zadania wykonujące głownie obliczenia ORAZ zadania wykonujące głównie operacje we/wy; chodzi o nakładanie się obliczeń i operacji we/wy

Zarządzanie procesami; **planowanie (przydziału) procesora**

- ... czyli planista krótkoterminowy
- rodzaje planowania procesora:
  - wywłaszczeniowe (wymaga przerwań zegarowych)
  - nie-wywłaszczeniowe (przełączenie procesora występuje wyłącznie wtedy gdy nasz proces się kończy lub wywołał fun. sys.)
- procedura realizująca przełączenie procesora nazywa się *ekspedytor* (ang. dispatcher)
- przełączenie procesora oznacza:
  - przełączenie kontekstu (zapisanie/odtworzenie rejestrów procesora)
  - przełączenie się do "trybu użytkownika"
  - skok do odpowiedniego rozkazu (?)
- kryteria planowania procesora:
  (czyli czym planista się kieruje wybierając proces z kolejki)
  - czas oczekiwania procesów w kolejce procesów gotowych (*precyzyjniej*: średnia arytmetyczna czasów przebywania w kolejce procesów gotowych wszystkich działających procesów)

Zarządzanie procesami; **planowanie procesora c.d.**

faza procesora kończy się po wywołaniu fun. sys. !

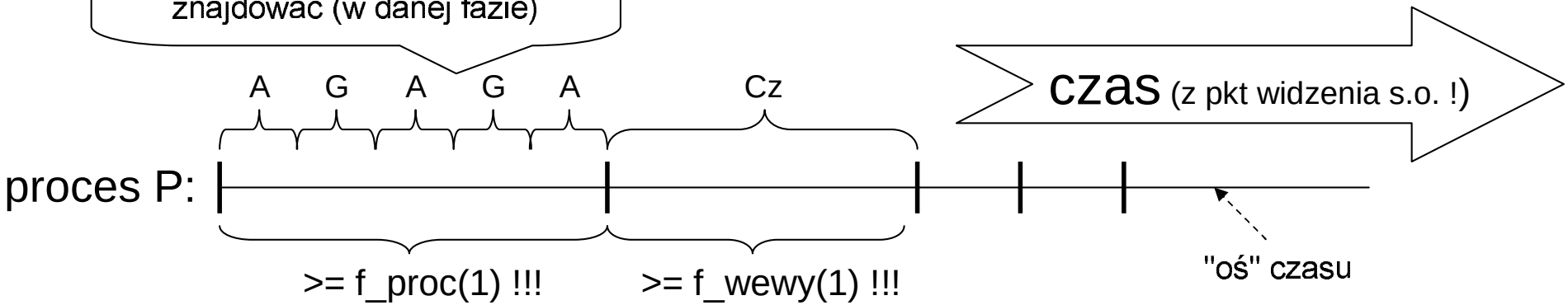
- fazy działania procesu:
  - ***faza procesora*** – okres gdy procesor wykonuje kodu procesu
  - ***faza we/wy*** – okres gdy proces czeka na zakończenie operacji we/wy
  - fazy występują na przemian (począwszy od fazy procesora)
- działanie procesu można opisać przez ciąg liczb:
  f_proc(1), f_wewy(1), f_proc(2), f_wewy(2), ...

czas trwania fazy procesora (faza nr 1)

czas trwania fazy we/wy (faza nr 1)

czas (z pkt widzenia procesu !)

stan w jakim proces może się znajdować (w danej fazie)



Zarządzanie procesami; **algorytm SJF**

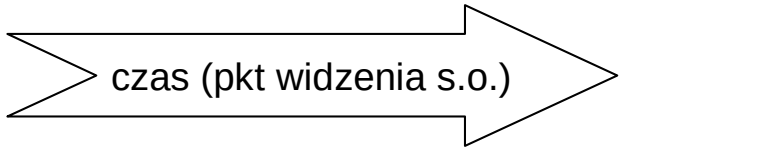
- **SJF** = Shortest Job First
- planista SJF preferuje procesy z krótką (przyszłą) fazą procesora -wydaje się, że wtedy ***średni czas gotowości*** będzie najmniejszy ...
- *założenie upraszczające*: rozpatrujemy jedną fazę procesora każdego procesu
- (?) skąd się biorą "nowe" procesy w kolejce procesów gotowych ? Odp: rozpatrujemy jedną fazę procesora, ale niekoniecznie pierwszą !; tak więc czas pojawienia się "nowych" procesów (zmiana stanu: Cz->G) jest nieprzewidywalny ...
- (?) gdzie się stosuje SJF ? Odp: raczej systemy wsadowe/ wieloprog. (gdyż w SJF ten sam proces może być długo aktywny)
- SJF: wysoki *priorytet procesu* ⇔ krótka (przyszła) faza procesora; można inaczej definiować priorytety procesu ... np. aktywność procesu może obniżać jego priorytet (wtedy trzeba częściej wywoływać planistę)

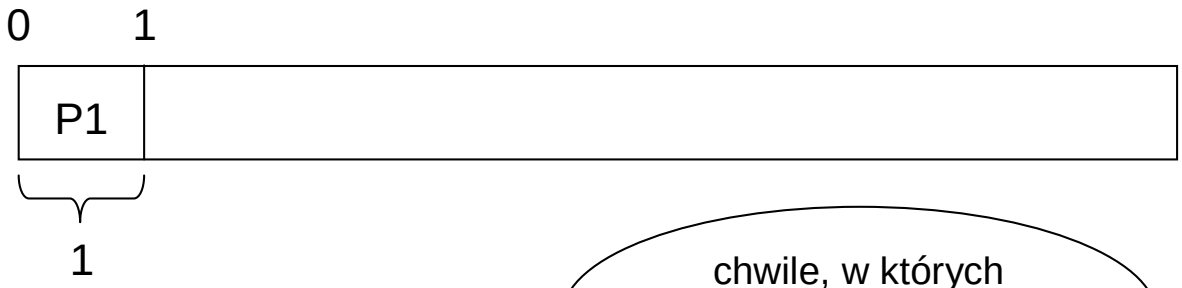
symulacja SJF/ z wywłaszczaniem

czas trwania fazy procesora

proces	czas przybycia procesu	f_proc(1)
P1	0 *ms*	8 *ms*
P2	1	5
P3	2	9
P4	3	2

"czas przybycia"
do kolejki procesów gotowych



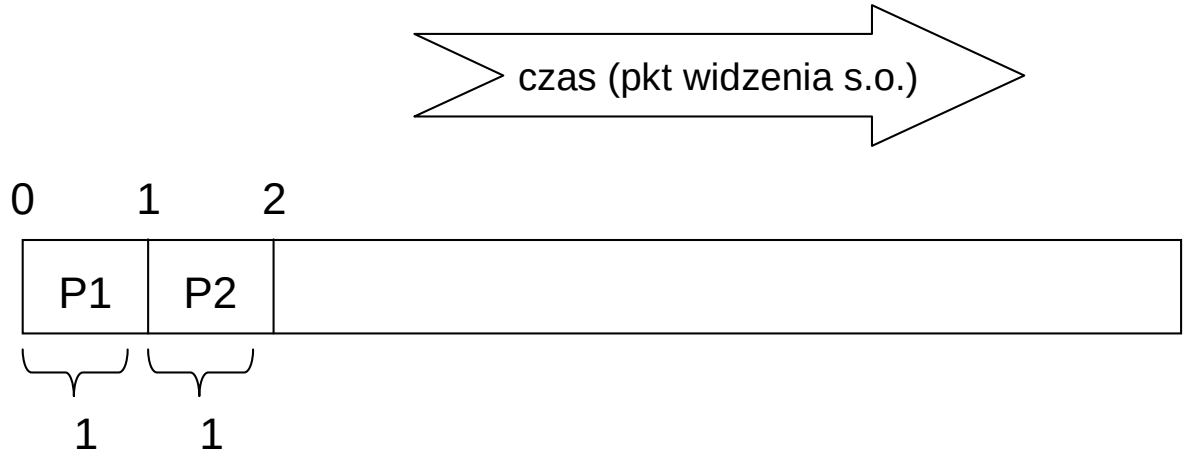


chwile, w których procesor ma szansę się przełączyć !!!

Proces	0				
P1	8				
P2					
P3					
P4					

symulacja SJF/ z wywłaszczaniem

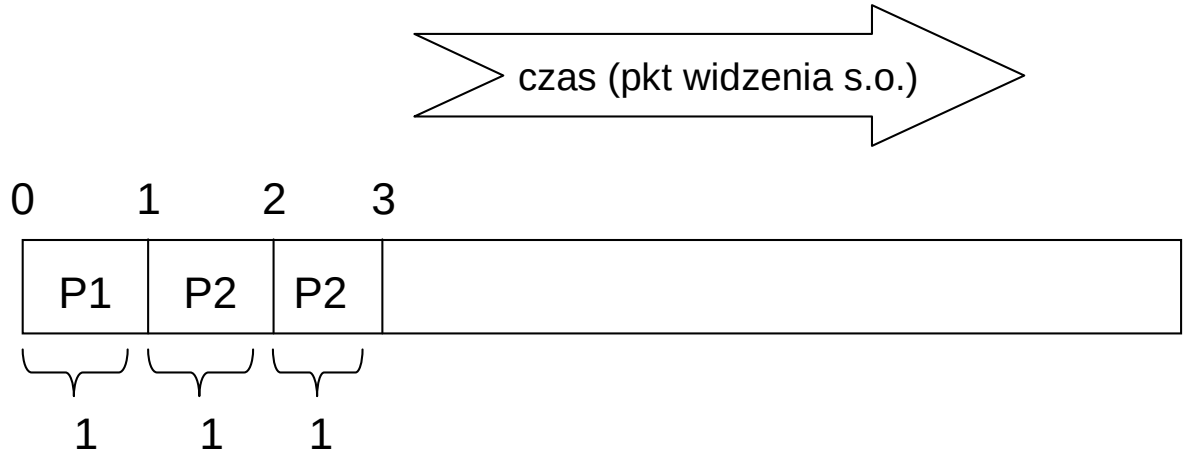
proces	czas przybycia procesu	f_proc(1)
P1	0 *ms*	8 *ms*
P2	1	5
P3	2	9
P4	3	2



Proces	0	1			
P1	8	7			
P2		5			
P3					
P4					

symulacja SJF/ z wywłaszczaniem

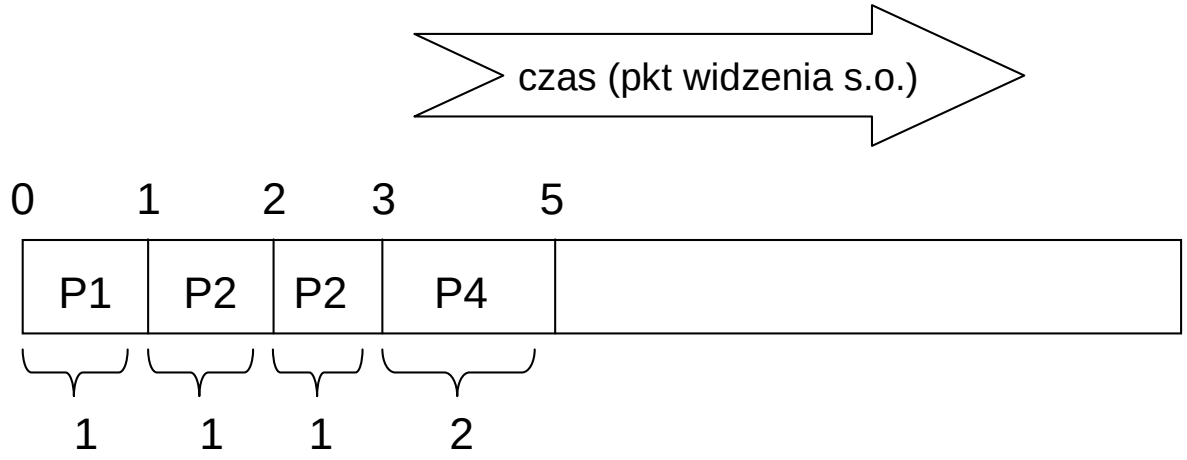
proces	czas przybycia procesu	f_proc(1)
P1	0 *ms*	8 *ms*
P2	1	5
P3	2	9
P4	3	2



Proces	0	1	2		
P1	8	7	7		
P2		5	4		
P3			9		
P4					

symulacja SJF/ z wywłaszczaniem

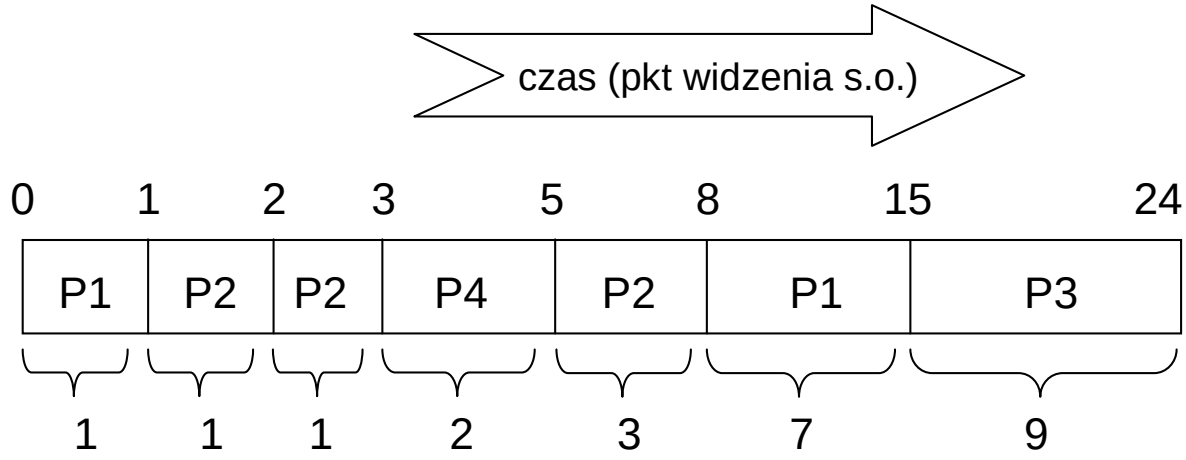
proces	czas przybycia procesu	f_proc(1)
P1	0 *ms*	8 *ms*
P2	1	5
P3	2	9
P4	3	2



Proces	0	1	2	3	
P1	8	*7*	*7*	*7*	
P2		5	4	3	
P3			9	9	
P4				2	

symulacja SJF/ z wywłaszczaniem

proces	czas przybycia procesu	f_proc(1)
P1	0 *ms*	8 *ms*
P2	1	5
P3	2	9
P4	3	2
	Suma=	24



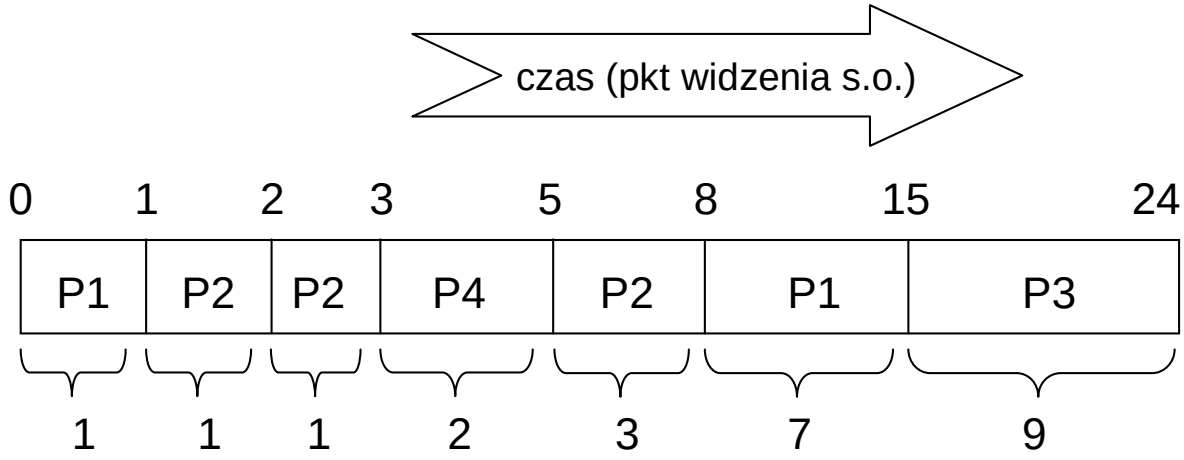
Proces	0	1	2	3	5
P1	8	7	7	7	7
P2		5	4	3	3
P3			9	9	9
P4				2	0

2
1
3

żaden nowy proces
się już nie pojawi ...

symulacja SJF/ z wywłaszczaniem

proces	czas przybycia procesu	f_proc(1)
P1	0 *ms*	8 *ms*
P2	1	5
P3	2	9
P4	3	2
	Suma=	24



średni czas oczekiwania
w kolejce procesów gotowych
(czyli *średni czas gotowości*):

$$= \frac{\overbrace{(8-1)}^{P1} + \overbrace{(5-3)}^{P2} + \overbrace{(15-2)}^{P3} + \overbrace{(3-3)}^{P4}}{4} = 5{,}5ms$$

dla każdego procesu
policz jak długo był w
stanie "gotowości"

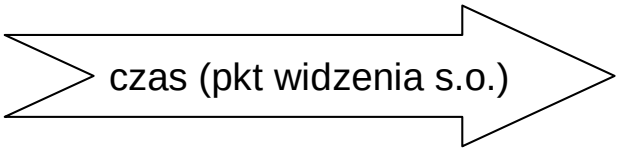
Zarządzanie procesami; **algorytm RR**

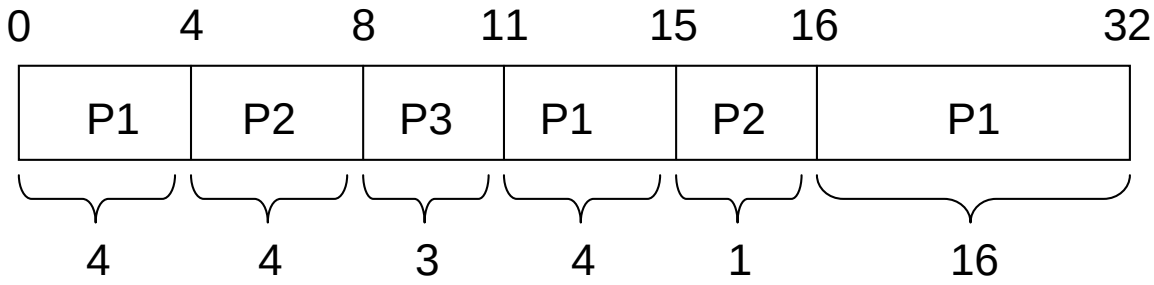
- RR = Round Robin = "planowanie rotacyjne"
- działa jak FCSC + przerwanie zegarowe wymuszające cykliczne przełączenie procesora (zgłaszane co "kwant" czasu)
- RR jest używany w s.o. z podziałem czasu (np. w Unix-ie) (*dlaczego ???*)

proces	f_proc(1)
P1	24 *ms*
P2	5
P3	3

wszystkie procesy pojawiły się w chwili 0 (w podanej kolejności)

kwant czasu = 4 *ms*





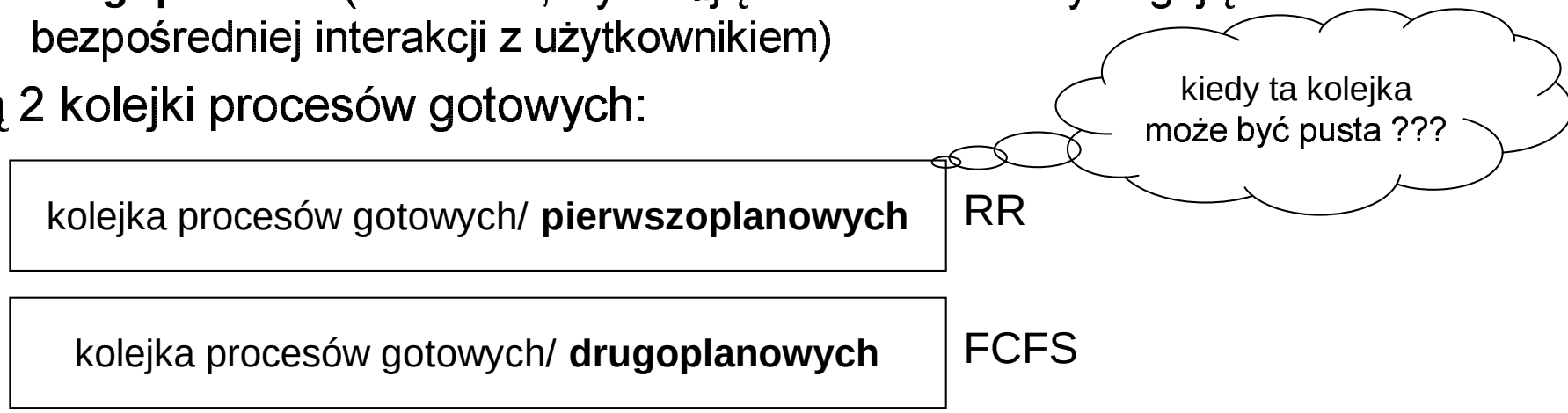
średni czas oczekiwania w kolejce procesów gotowych:

$$= \frac{([0-0]+[11-4]+[16-15])+([4-0]+[15-8])+(8-0)}{3} = 9ms$$

P1 P2 P3

Zarządzanie procesami; **algorytm wielopoziomowy**

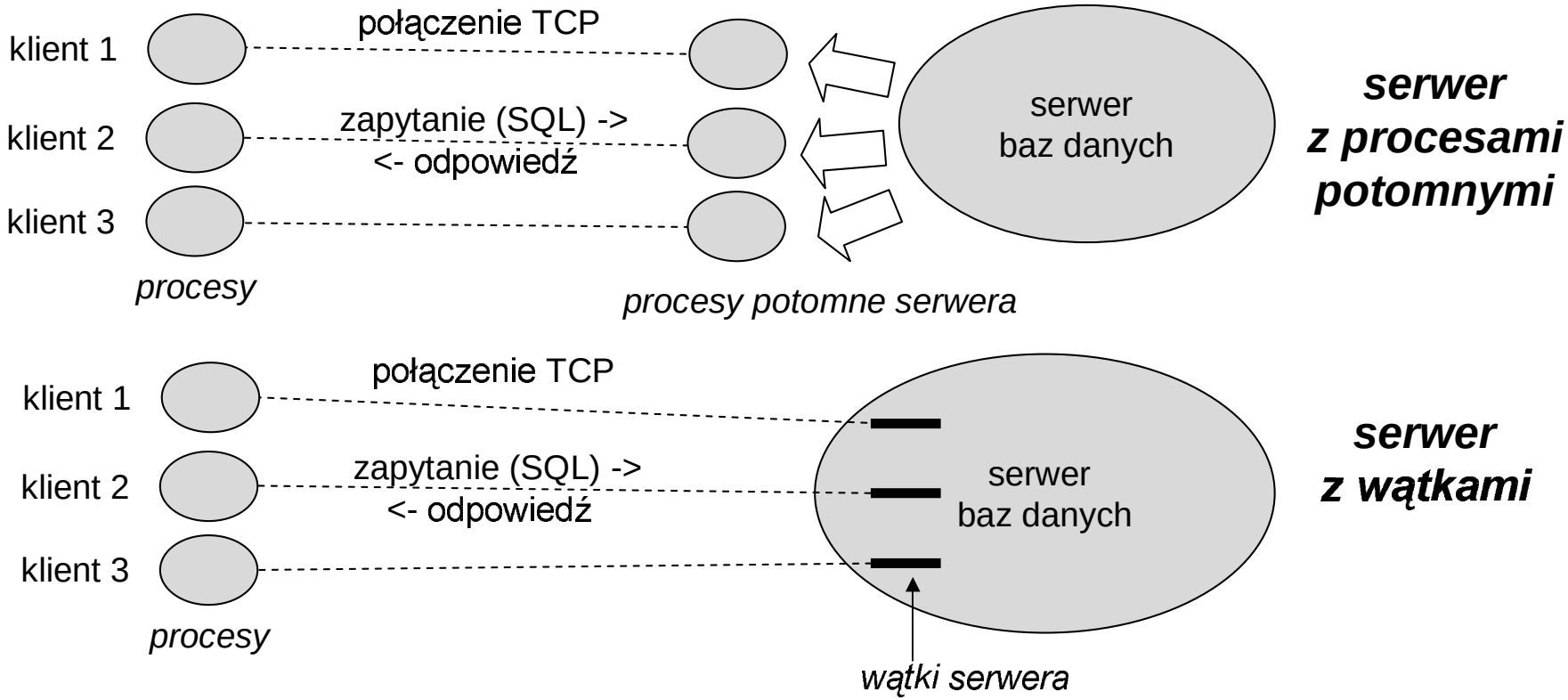
- zakłada się że są 2 rodzaje procesów
  - **pierwszoplanowe** (interakcyjne; interakcja z użytkownikiem, np. edytor)
  - **drugoplanowe** (wsadowe; wykonują obliczenia nie wymagające bezpośredniej interakcji z użytkownikiem)
- są 2 kolejki procesów gotowych:



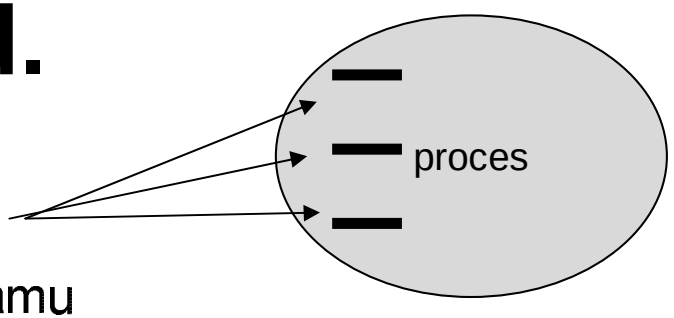
- zasada: procesy drugoplanowe otrzymują procesor tylko gdy kolejka procesów pierwszoplanowych jest pusta (np. wszystkie są "czekające")
- każda kolejka jest obsługiwana innym algorytmem planowania
- gdy proces pierwszoplanowy pojawi się w swojej kolejce (czyli stanie się gotowy) to proces drugoplanowy zostanie wywłaszczony !

Zarządzanie procesami; **wątki**

- ... czasami jest wygodne aby kod procesu był wykonywany równocześnie (współbieżnie) kilka razy
  - do tego służą *wątki*
- *przykład*: serwer baz danych obsługujący wielu klientów równocześnie



Zarządzanie procesami; **wątki c.d.**



- cechy wątków:
  - jeden proces może posiadać kilka wątków
  - każdy wątek wykonuje kod swojego programu (różne wątki mogą być w różnych "miejscach" tego kodu)
  - większość zasobów jest przydzielana procesom a nie wątkom, dotyczy to:
    - obszaru pamięci operacyjnej
    - otwartych plików
  - (wniosek z powyższego) wątki danego procesu mają **tą samą prz. adr. pamięci** i te same otwarte pliki
- co wątki danego procesu mają "wspólne", a co "własne":
  - wspólne:
    - sekcja danych (zmienne globalne)
    - sekcja kodu
    - inne zasoby przydzielone procesowi (m.in. otwarte pliki)
  - własne:
    - wartości rejestrów procesora, przechowywane w **BKW = Blok Kontrolny Wątku**
    - sekcja stosu (zmienne lokalne procedur + adr. powrotne) [*dlaczego ?*]

Zarządzanie procesami; **wątki c.d.**

- zalety wątków:
  - wątki są mniej "kosztowne" dla systemu niż procesy
    - przełączenie procesora "między wątkami" jest znacznie szybsze niż "między procesami" [*dlaczego ? np. nie trzeba modyfikować rej. baza/limit*]
    - BKW (=Blok Kontrolny Wątku) jest znacznie mniejszy niż BKP ! (m.in. nie zawiera informacji o zarządzaniupamiecią)
    - tworzenie nowych wątków także jest szybsze niż tworzenie nowych procesów
  - komunikacja między wątkami jest łatwiejsza niż między procesami (mogą się komunikować przez wspólną prz. adr., przez zmienne globalne)
  - brak ochrony pamięci między wątkami (to chyba wada ...)
- biblioteki wątków:
  - linux/unix: pthread
  - WinTN: wątki wbudowane w jądro
- sposoby implementacji wątków:
  - *wątki poziomu użytkownika* – s.o. nieświadomy istnienia wątków (np. niektóre wersje biblioteki pthread)
  - *wątki poziomu jądra* – wątki wbudowane w jądro s.o. (np. WinNT)