Procesy

Proces - program w czasie wykonania; wykonanie musi przebiegać sekwencyjnie

W skład procesu wchodzi:

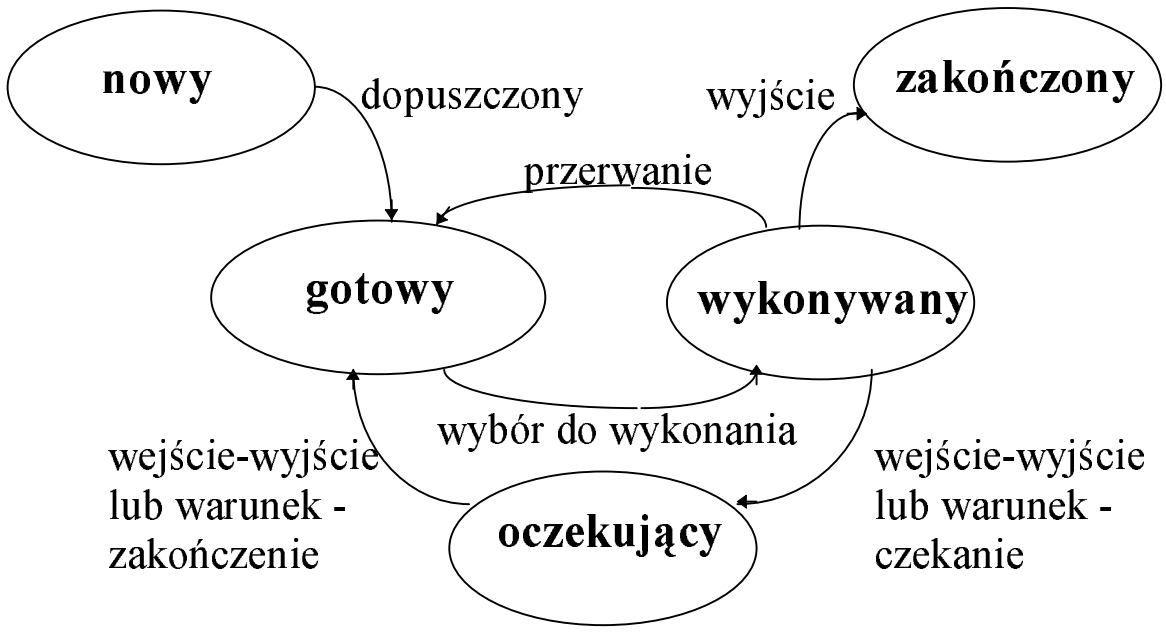
- program
- licznik rozkazów
- stos
- sekcja danych

Procesy wykonują się *współbieżnie* (niekoniecznie *równolegle*)

Proces podczas wykonania zmienia ***stan***:

- **nowy**: proces został utworzony
- **wykonywany**: są wykonywane instrukcje procesu
- **oczekujący**: proces czeka na zajście jakiegoś zdarzenia
- **gotowy**: proces czeka na przydzielenie procesora
- **zakończony**: proces zakończył wykonanie

Diagram stanów procesu:



Blok kontrolny procesu (PCB) - zawiera informacje związane z procesem:

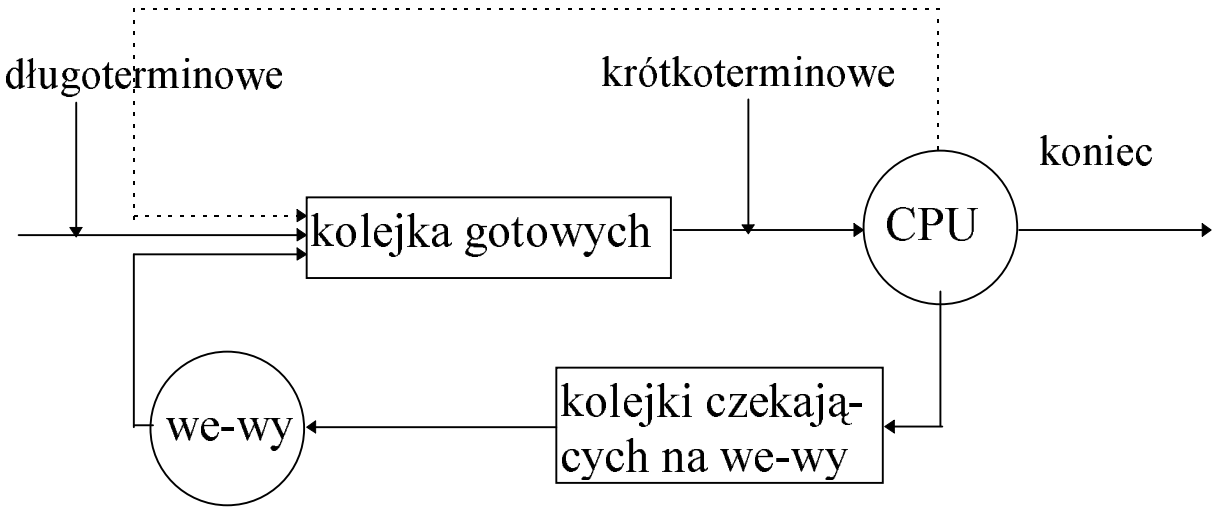
- stan procesu
- licznik rozkazów
- rejestry procesora
- informacje o szeregowaniu przez procesor
- informacje o pamięci zajętej przez proces
- informacje rozliczeniowe
- informacje o stanie wejścia-wyjścia

Kolejki szeregowania procesów

- kolejka zadań - zbiór wszystkich procesów w systemie
- kolejka gotowych - zbiór wszystkich procesów umieszczonych w pamięci głównej, gotowych i czekających na wykonanie
- kolejki do urządzeń - zbiór procesów czekających na określone urządzenie wejścia-wyjścia

Szeregowanie procesów

- *szeregowanie długoterminowe* - wybór procesów, które przejdą do kolejki gotowych; wykonywane rzadko (sekundy, minuty); może być wolne; określa poziom wieloprogramowości
- *szeregowanie krótkoterminowe* - wybór następnego procesu do wykonania i przydział CPU; wykonywane często (milisekundy); musi być szybkie



Czasami dochodzi *szeregowanie średnioterminowe*: **wymiana** (*swapping*) czyli czasowe usuwanie zadania w całości z pamięci głównej do pomocniczej

Procesy można określić jako:

- *zorientowane na wejście-wyjście* - spędzają więcej czasu wykonując wejście-wyjście niż obliczenia; wiele krótkich odcinków czasu zapotrzebowania na CPU
- *zorientowane na obliczenia* - spędzają więcej czasu wykonując obliczenia; kilka bardzo długich odcinków czasu zapotrzebowania na CPU

Przełączanie kontekstu:

- Przełączanie procesora do innego procesu wymaga przechowania stanu starego procesu i załadowanie przechowanego stanu nowego procesu
- Czas przełączenia kontekstu jest narzutem na działanie systemu
- Czas przełączenia kontekstu zależy od wsparcia ze strony sprzętu (rząd wielkości: 1 - 100 mikrosekund)

Tworzenie procesów

- Proces macierzysty tworzy procesy potomne, które z kolei tworzą inne procesy; powstaje w ten sposób *drzewo procesów*
- *Współdzielenie zasobów*: proces potomny i macierzysty współdzielą wszystkie zasoby; potomkowie dzielą część zasobów przodka; przodek i potomkowie nie dzielą żadnych zasobów
- *Wykonanie*: współbieżnie lub przodek czeka na zakończenie potomków
- *Przestrzeń adresowa*: potomek jest kopią przodka lub potomek ładuje nowy program
- Unix: **fork**, **exec**

Kończenie procesów

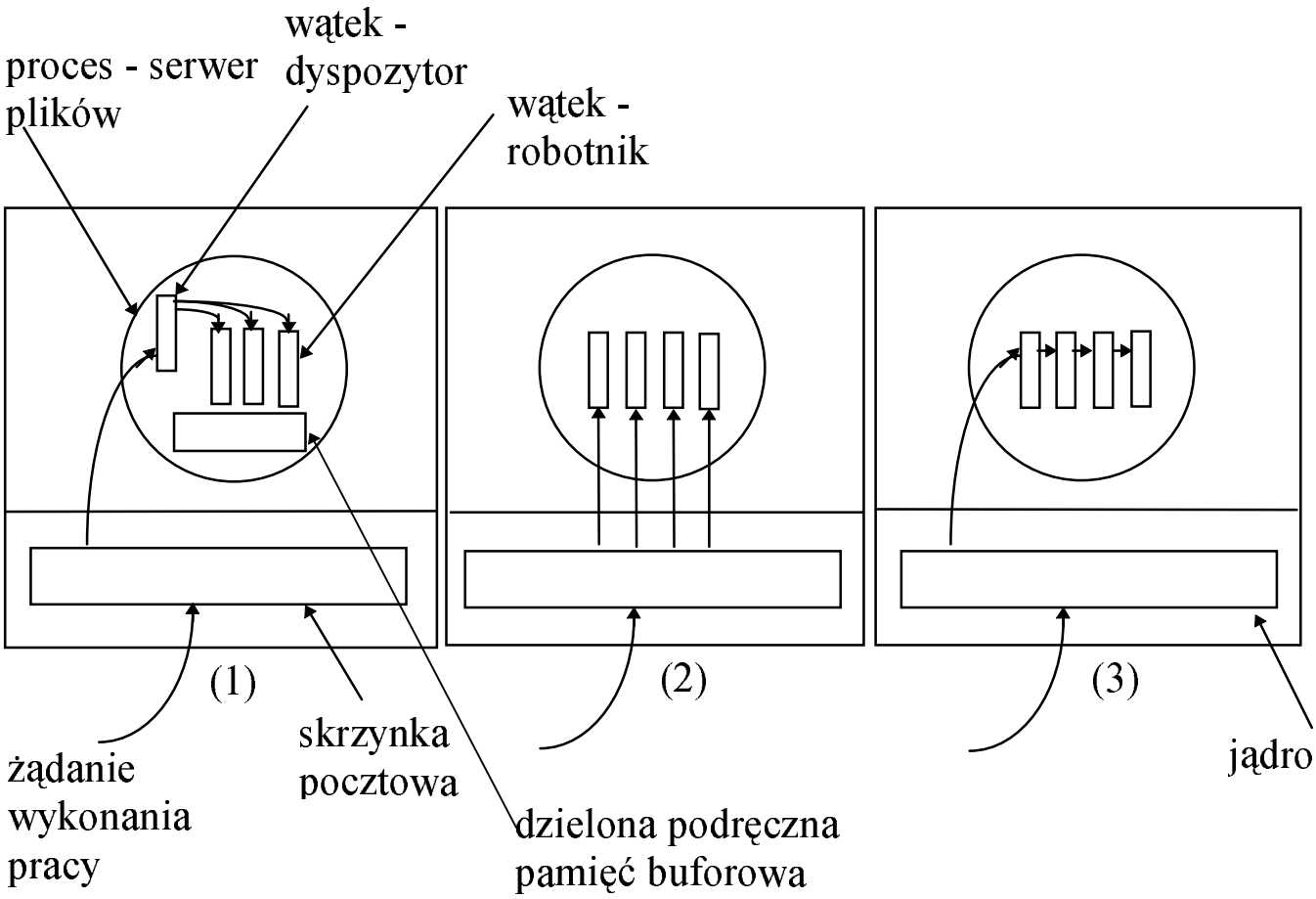
- Proces wykonuje ostatnią instrukcję i prosi system operacyjny o usunięcie (**exit**); przekazanie danych z procesu potomnego do macierzystego (**wait**); zwolnienie przydzielonych zasobów
- Przodek może przerwać wykonanie potomka: potomek przekroczył przydzielone zasoby; zlecone potomkowi zadanie nie jest już potrzebne; przodek kończy działanie

Obsługą procesów zajmuje się **jądro systemu operacyjnego**

Wątki (lekkie procesy)

- *Wątek* jest podstawową jednostką wykorzystania CPU; posiada własny licznik rozkazów, zbiór rejestrów, stos, stan, wątki potomne
- Wątek współdzieli z innymi wątkami tego samego zadania: przestrzeń adresową, zmienne globalne, zasoby systemowe
- Tradycyjny proces (*ciężki*) jest równoważny zadaniu z jednym wątkiem
  Jeśli zadanie składa się z wielu wątków, to w czasie gdy jeden wątek jest zablokowany, może się wykonywać inny wątek tego zadania; współpraca wielu wątków w jednym zadaniu pozwala zwiększyć przepustowość i poprawić wydajność
- Implementacja wątków:
  - na poziomie użytkownika (projekt Andrew z CMU); dobra wydajność, potrzebny system czasu wykonania wspomagający wątki
  - wspierana przez jądro systemu (Mach, OS/2); gorsza wydajność, ale implementacja nie wymaga „sztuczek”
  - podejście hybrydowe (Solaris 2)
- Tworzenie wątków i przełączanie procesora między wątkami tego samego procesu są dużo tańsze niż tworzenie procesów i przełączanie kontekstu między procesami

Wątki umożliwiają połączenie *równoległości* z *sekwencyjnym wykonaniem* i *blokującymi funkcjami systemowymi*:



(1) model dyspozytor-pracownik
(2) model zespołu
(3) model potoku

Przydział procesora

Podstawowe pojęcia

- Wieloprogramowość umożliwia zwiększenie wykorzystania CPU i urządzeń zewnętrznych
- Wykonanie procesu przebiega *cyklicznie*: fazy *zapotrzebowania na CPU* przeplatają się z fazami *zapotrzebowania na wejście-wyjście*
- Rozkład faz zapotrzebowania na CPU powinien mieć wpływ na dobór algorytmu szeregowania procesów

Działanie procesu szeregującego (krótkoterminowo):

- Wybiera spośród procesów gotowych i umieszczonych w pamięci jeden i przydziela mu procesor
- Decyzję podejmuje się gdy proces:
    1. przechodzi ze stanu **wykonywany** do stanu **oczekujący**
    2. przechodzi ze stanu **wykonywany** do stanu **gotowy**
    3. przechodzi ze stanu **oczekujący** do stanu **gotowy**
    4. kończy się

  Szeregowanie bez wywłaszczania: 1 i 4
  Szeregowanie z wywłaszczaniem: pozostałe

Dyspozytor (*dispatcher*)

Przekazuje procesor procesowi wybranemu przez proces szeregujący; obowiązki:

- przełączanie kontekstu
- przełączenie do trybu użytkownika
- wykonanie skoku do odpowiedniego adresu w programie użytkownika w celu wznowienia wykonania programu

Opóźnienie związane z działaniem dyspozytora - czas potrzebny dyspozytorowi na zatrzymanie jednego procesu i uruchomienie innego

Kryteria szeregowania

- *Wykorzystanie CPU* - procent czasu zajętości CPU (max)
- *Przepustowość* - liczba procesów, które zakończyły wykonanie w jednostce czasu (max)
- *Czas obrotu* - czas potrzebny na wykonanie procesu (min)
- *Czas oczekiwania* - czas spędzony przez proces w kolejce procesów gotowych (min)
- *Czas reakcji* - czas liczony od chwili dostarczenia żądania do chwili uzyskania odpowiedzi (w systemie z podziałem czasu) (min)

Algorytmy (krótkoterminowego) szeregowania procesów

1. Kolejka prosta (FIFO)

Procesy są wykonywane w kolejności przybywania
Brak wywłaszczania

Przykład:
Procesy i ich zapotrzebowanie na CPU: P1 (24), P2 (3), P3 (3)

Jeśli procesy przybyły w kolejności P1, P2, P3:
- czas oczekiwania: P1 = 0, P2 = 24, P3 = 27
- średni czas oczekiwania: (0 + 24 + 27)/3 = 17

Jeśli procesy przybyły w kolejności P2, P3, P1:
- czas oczekiwania: P1 = 6, P2 = 0, P3 = 3
- średni czas oczekiwania: (6 + 0 + 3)/3 = 3

Efekt konwoju: krótkie procesy wstrzymywane przez długie

Zalety: sprawiedliwy, niski narzut systemowy
Wady: długi średni czas oczekiwania i wariancja czasu oczekiwania, nieakceptowalny w systemach z podziałem czasu

2. „Najkrótsze zadanie najpierw" (SJF)

Procesor przydziela się temu procesowi, który ma najkrótszą najbliższą fazę zapotrzebowania na CPU (gdy równe, to FCFS)
Z wywłaszczaniem (SRTF) lub bez

Przykład:
Procesy, ich czas przybycia i zapotrzebowanie na CPU: P1 w 0 (7), P2 w 2 (4), P3 w 4 (1), P4 w 5 (4)

Bez wywłaszczania - kolejność obsługi: P1, P3, P2, P4
- czas oczekiwania: P1 = 0, P2 = 6, P3 = 3, P4 = 7
- średni czas oczekiwania: (0 + 6 + 3 + 7)/4 = 4

Z wywłaszczaniem - kolejność obsługi: P1, P2, P3, P2, P4, P1
- czas oczekiwania: P1 = 9, P2 = 1, P3 = 0, P4 = 2
- średni czas oczekiwania: (9 + 1 + 0 + 2)/4 = 3

Zalety: optymalny
Wady: wymaga określenia długości przyszłej fazy CPU (można próbować oszacować np. jako średnią wykładniczą poprzednich faz: $\tau_{n+1} = \alpha \cdot t_n + (1-\alpha) \cdot \tau_n$)

3. Szeregowanie priorytetowe

Z każdym procesem jest związany *priorytet* (liczba całkowita)
CPU przydziela się procesowi z najwyższym priorytetem, FCFS gdy priorytety równe (zwykle mała liczba ≡ wysoki priorytet)
Z wywłaszczaniem lub bez wywłaszczania
SJN jest strategią szeregowania priorytetowego (priorytet ≡ przewidywana długość następnej fazy CPU)

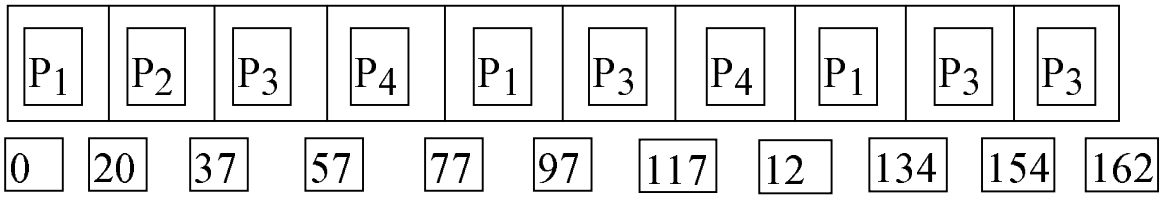
Problem: zagłodzenie
Rozwiązanie: wzrost priorytetu z upływem czasu (proces się „starzeje")

4. Szeregowanie karuzelowe (Round Robin)

Każdy proces otrzymuje kwant czasu CPU, zwykle 10 - 100 milisekund. Po upływie kwantu proces zostaje wywłaszczony i idzie na koniec kolejki procesów gotowych

Jeśli w kolejce procesów gotowych jest n procesów, a q to wielkość kwantu, to każdy proces dostaje $1/n$ czasu procesora w odcinkach długości co najwyżej q. Żaden proces nie czeka na następny kwant dłużej niż $(n\text{-}1)\cdot q$ jednostek czasu

Przykład:
Procesy i ich zapotrzebowanie na CPU: P1 (53), P2 (17), P3 (68), P4 (24) (kwant = 20)



- czas oczekiwania: P1 = 81, P2 = 20, P3 = 94, P4 = 97
- średni czas oczekiwania: (81 + 20 + 94 + 97)/4 = 73

Wydajność:
- q duże ⇒ FIFO
- q małe ⇒ q musi być duże w porównaniu z czasem przełączenia kontekstu, wpp narzut systemowy jest zbyt wysoki
- *reguła*: 80% faz CPU powinno być krótszych niż kwant czasu

PS (*processor sharing,* czyli *dzielenie procesora*): gdy kwant bardzo mały, to wydaje się, że każdy z procesów ma własny procesor działający z $1/n$ prędkości rzeczywistego procesora

Cechy: zwykle wyższy średni czas obrotu niż dla SRTF, lecz lepszy czas reakcji

5. Kolejki wielopoziomowe

Kolejka procesów gotowych jest podzielona na odrębne kolejki.
Przykład:
- procesy *piewszoplanowe* (interakcyjne)
- procesy *drugoplanowe* (wsadowe)

Każda kolejka ma własny algorytm szeregowania
Przykład:
- procesy *piewszoplanowe* - strategia karuzelowa
- procesy *drugoplanowe* - kolejka prosta

Szeregowanie pomiędzy kolejkami:
- *Stały priorytet*, np. obsługuj najpierw wszystkie procesy pierwszoplanowe, potem drugoplanowe; możliwość zagłodzenia
- *Kwant czasu* - każda kolejka otrzymuje ustaloną część czasu CPU do podziału pomiędzy swoje pocesy, np. 80% dla pierwszoplanowych z RR, 20% dla drugoplanowych z FCFS

6. Kolejki wielopoziomowe ze sprzężeniem zwrotnym

Procesy mogą się przemieszczać pomiędzy kolejkami

Parametry metody:
- liczba kolejek
- algorytm szeregowania dla każdej kolejki
- metoda przemieszczania procesów do „wyższych" kolejek
- metoda przemieszczania procesów do „niższych" kolejek
- metoda wybierania kolejki dla nowego procesu

Przykład: proces zużywający dużo czasu procesora (np. cały kwant w RR) przechodzi do kolejki o niższym priorytecie, ale dłuższym kwancie. Na najniższym poziomie: FCFS. Procesy interakcyjne i zorientowane na wejście-wyjście będą pozostawać w kolejkach o wyższym priorytecie, a procesy zorientowane obliczeniowo - w kolejkach o niższym priorytecie.

Zalety: bardziej elastyczna niż zwykłe kolejki wielopoziomowe, umożliwia implementację „starzenia się" procesów

Wady: dużo parametrów wymagających dostrajania, wysoki koszt implementacji

7. Szeregowanie w systemach wieloprocesorowych

Systemy heterogeniczne: każdy procesor ma własną kolejkę i własny algorytm szeregowania
Systemy homogeniczne:

- *dzielenie obciążenia* (osobna kolejka dla każdego procesora)
- wspólna kolejka
  - każdy procesor sam wybiera proces do wykonania
  - jeden procesor przydziela procesy do procesorów (*wieloprzetwarzanie asymetryczne*)

8. Szeregowanie w systemach czasu rzeczywistego

- Systemy czasu rzeczywistego ze *„sztywnymi" wymaganiami* - zadania krytyczne **muszą** się zakończyć w zadanym czasie
- Systemy czasu rzeczywistego z *„łagodnymi" wymaganiami* - zadania krytyczne są obsługiwane z wyższym priorytetem niż pozostałe