Systemy operacyjne III

WYKŁAD 2

Jan Kazimirski

Procesy w systemie operacyjnym

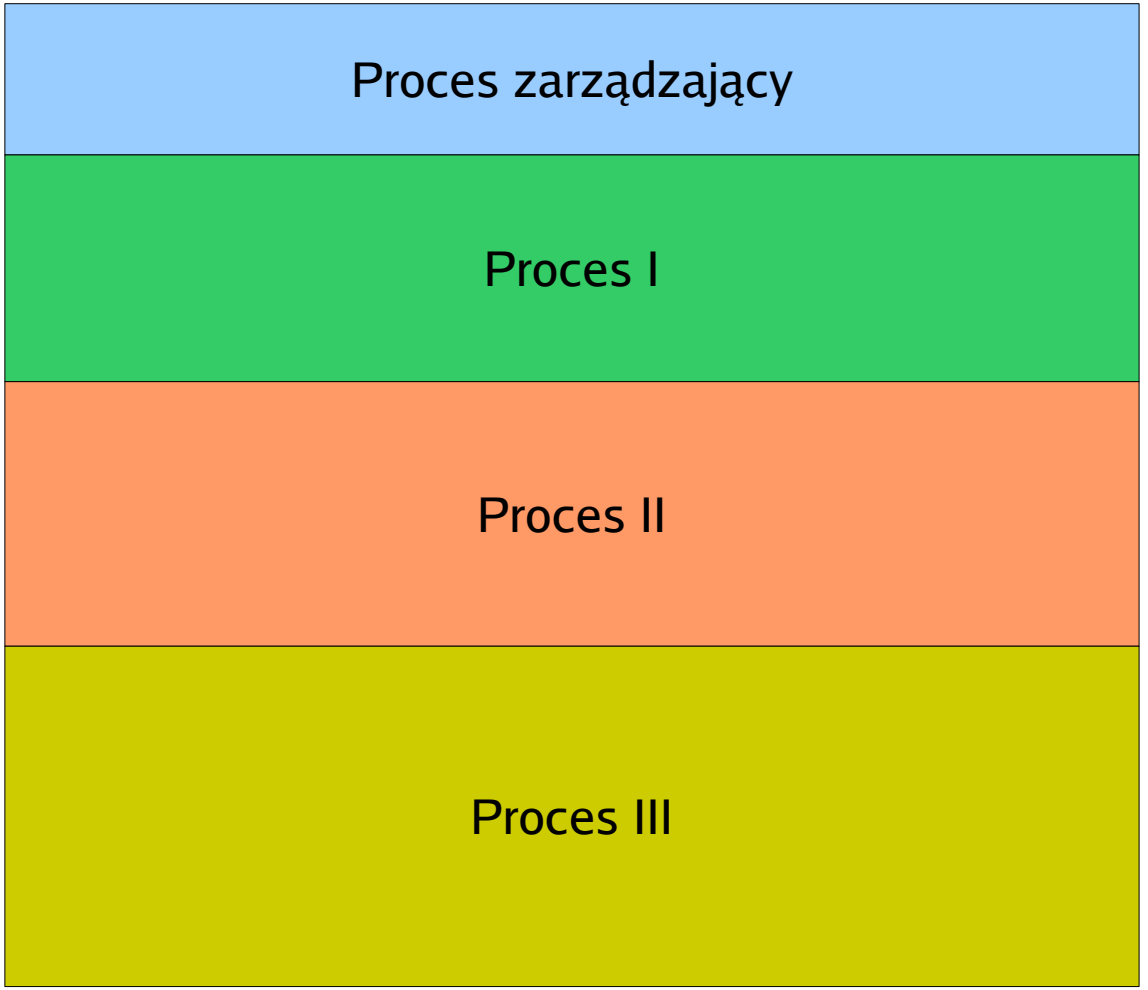
Proces

- Współczesne SO w większości są systemami wielozadaniowymi.
- W tym samym czasie SO obsługuje pewną liczbę zadań – procesów
- **Proces** = wykonywany program, razem ze swoimi danymi, stosem, wartościami rejestrów i licznika rozkazów.

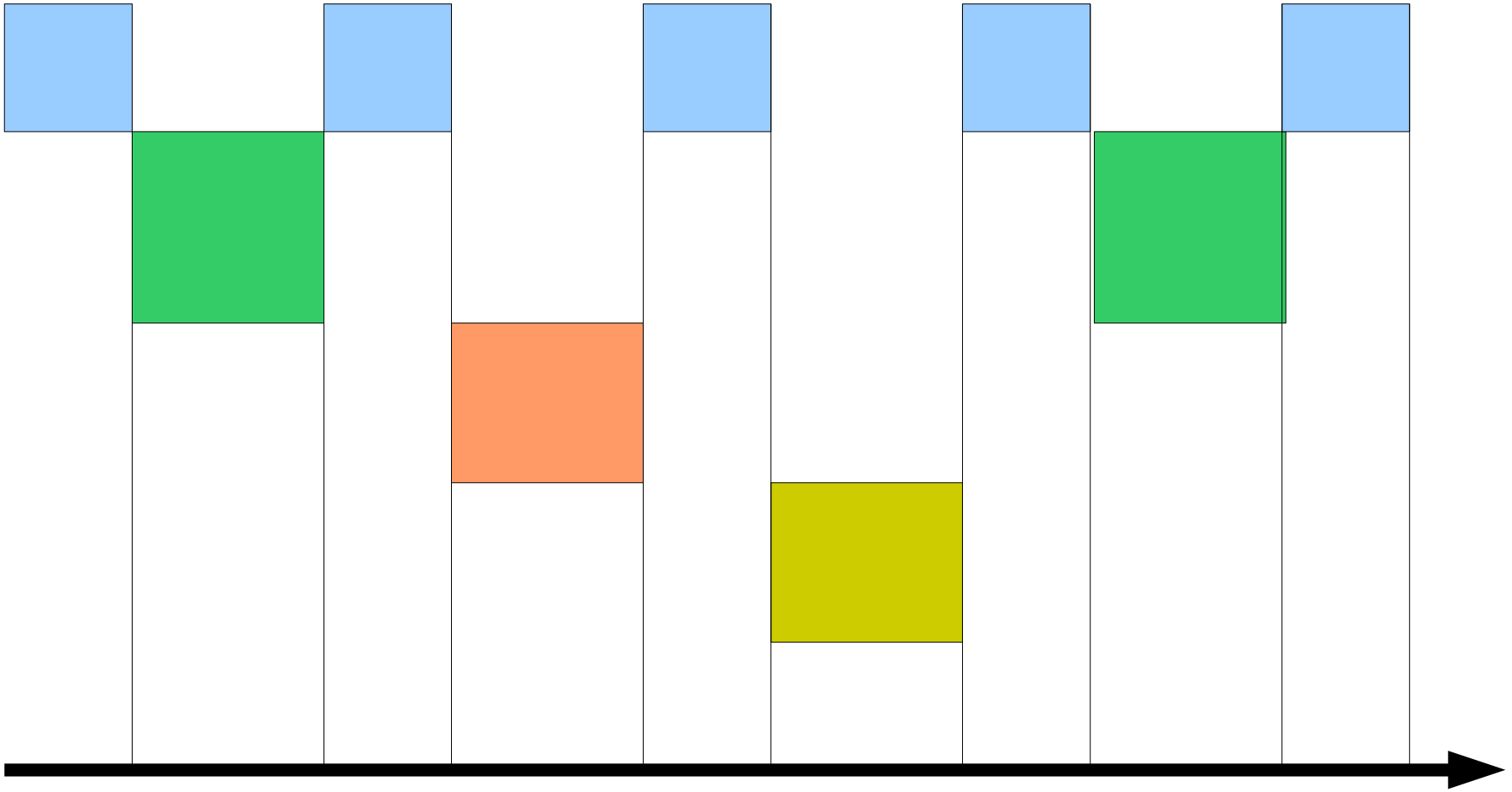
Składniki procesu (obraz procesu)

- Kod programu
- Dane programu
- Stos
- Dodatkowe informacje związane z wykonywaniem procesu.

Procesy w systemie

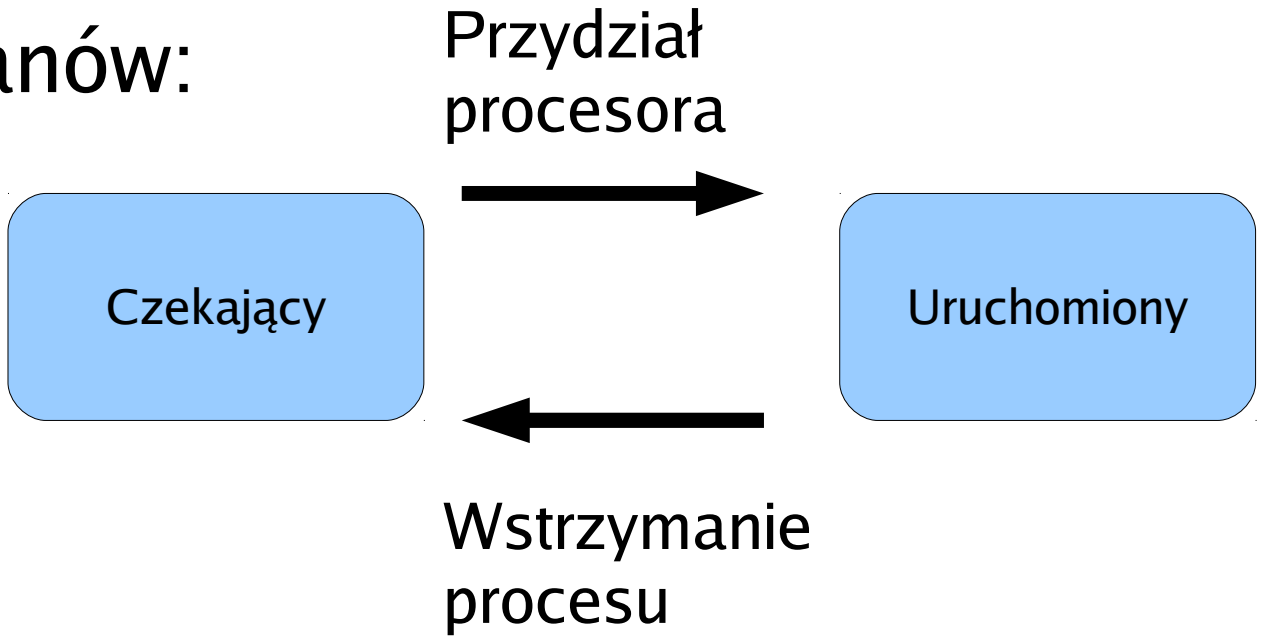


Przełączanie procesów

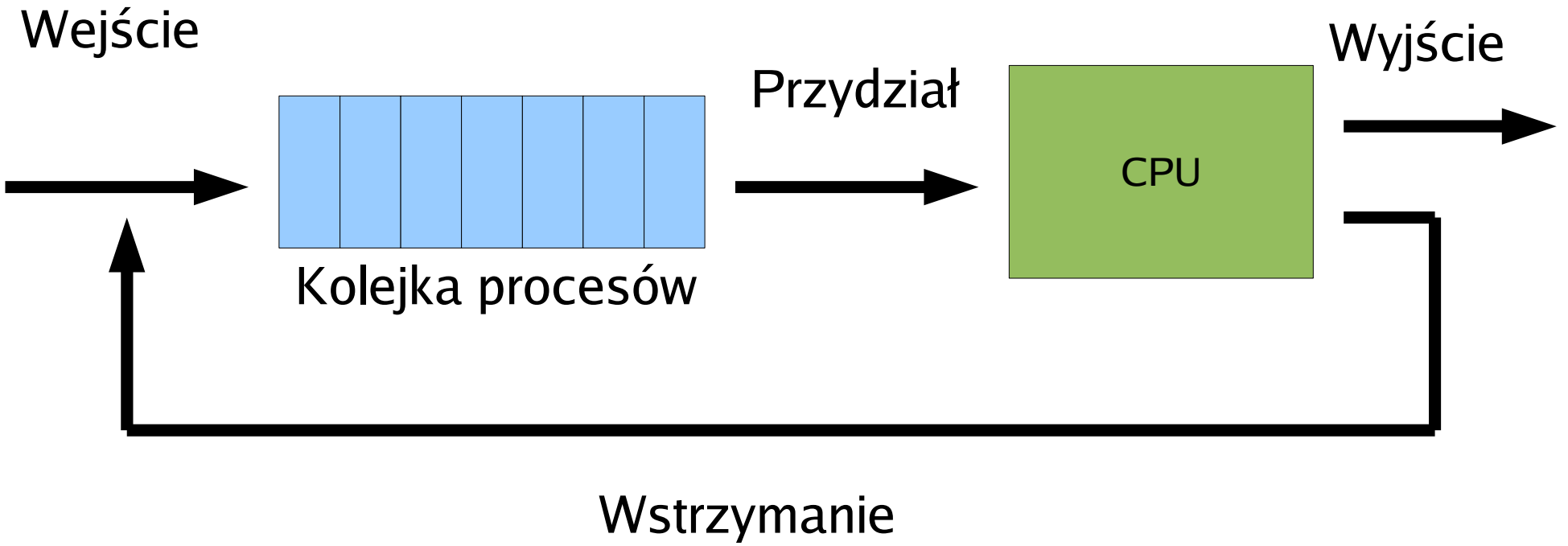


Dwustanowy model procesu

- Proces w dwóch stanach – uruchomiony, oczekujący.
- Diagram stanów:



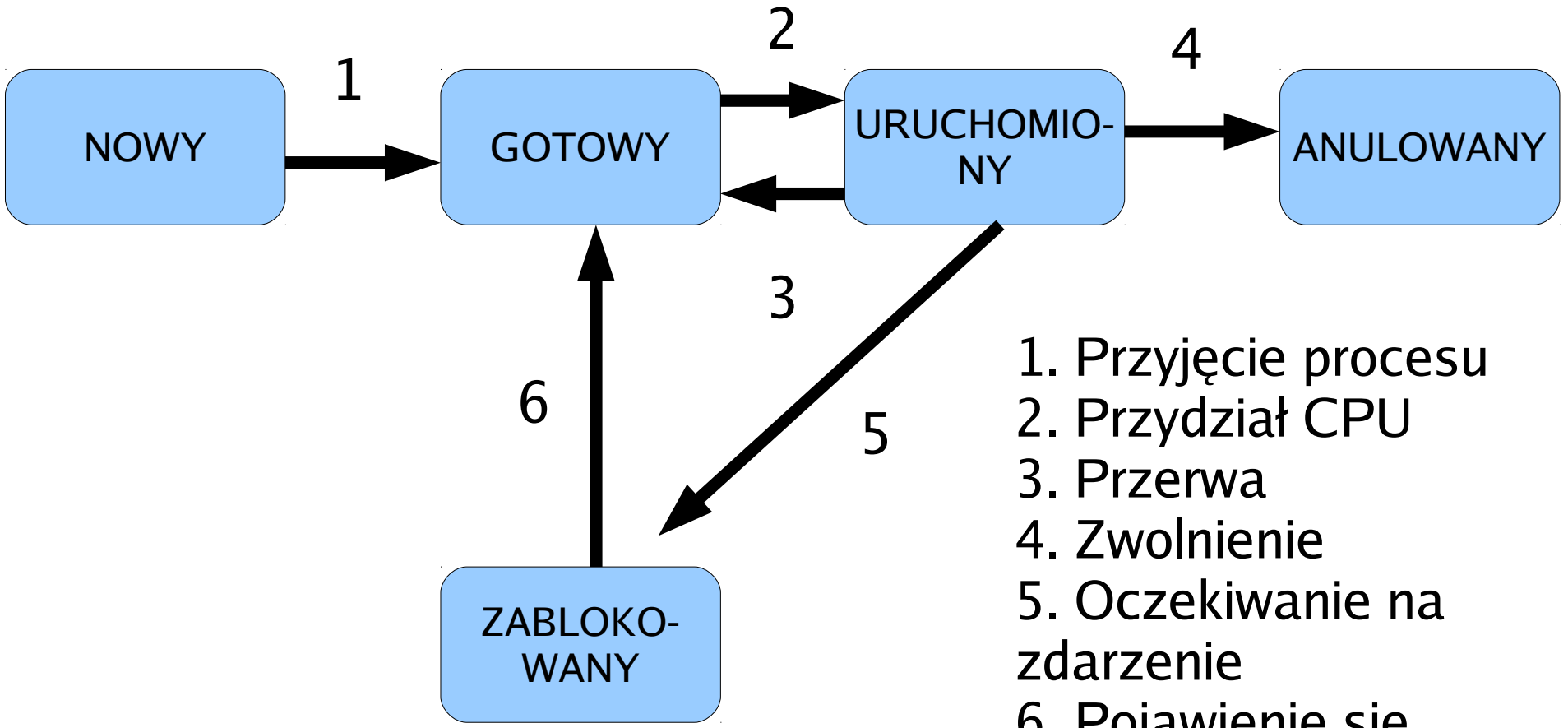
Model 2-stanowy c.d.



Model 2-stanowy – implementacja i ograniczenia

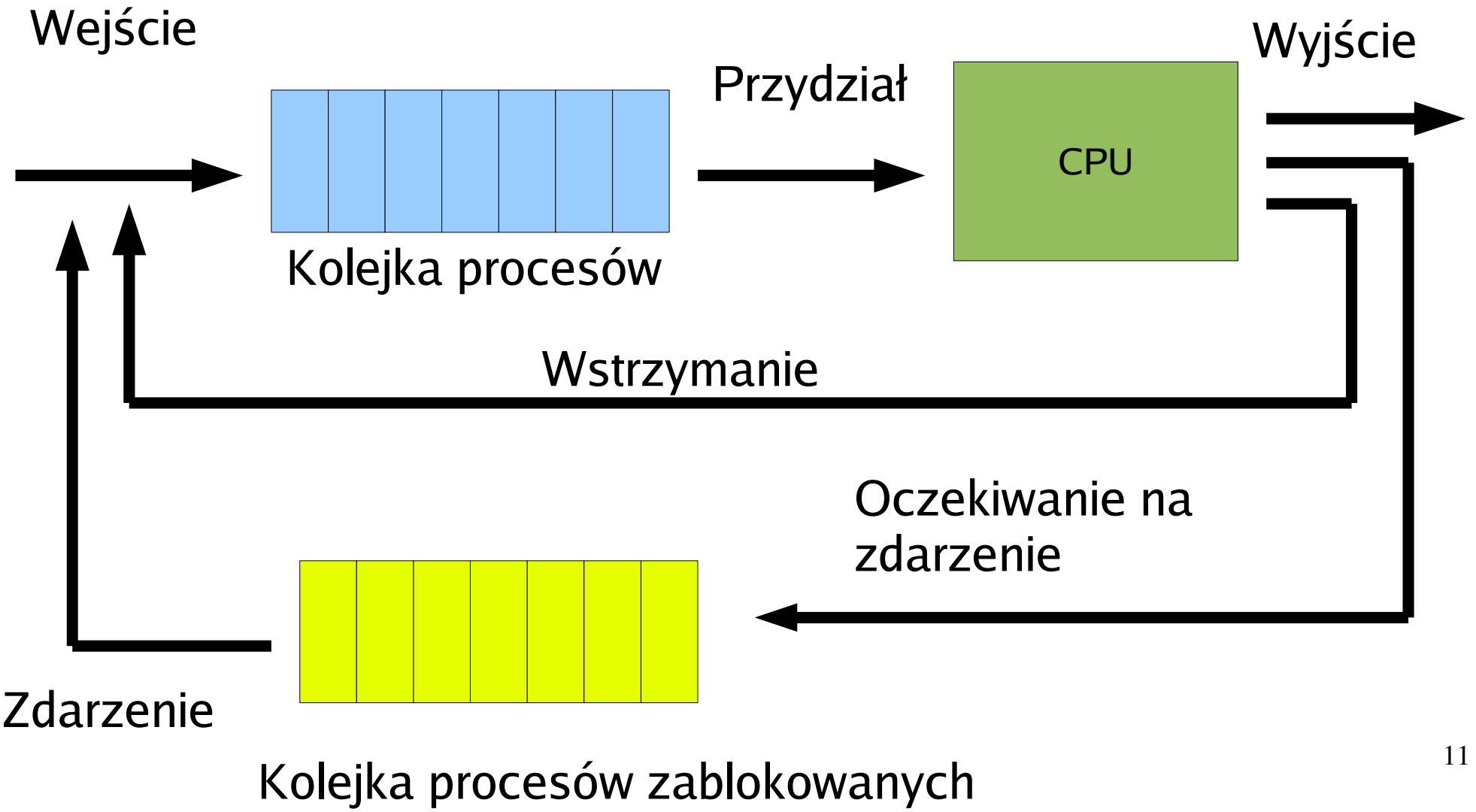
- Model 2-stanowy umożliwia prostą implementację – Przydział cykliczny, kolejka procesów na zasadzie FIFO
- Problem: nie zawsze proces czekający jest gotowy do wykonania.
  - Oczekujący na we/wy
  - Oczekujący na stronę pamięci (stronicowanie)
  - Oczekuje na sygnał od innego procesu

Model 5-stanowy



1. Przyjęcie procesu
2. Przydział CPU
3. Przerwa
4. Zwolnienie
5. Oczekiwanie na zdarzenie
6. Pojawienie się zdarzenia

Realizacja modelu 5-stanowego



Rozwinięcia

- Wprowadzenie priorytetu procesu – osobne kolejki gotowych procesów o różnych priorytetach
- Różne zdarzenia – osobne kolejki procesów oczekujących na poszczególne zdarzenia
- Mechanizm wymiany – przenoszenie zablokowanych procesów do pamięci pomocniczej

Mechanizm wymiany

- 2 nowe stany:
  - zablokowany zawieszony
  - gotowy zawieszony
- Przejścia
  - zablokowany → zablokowany zawieszony
  - zablokowany zawieszony → gotowy zawieszony
  - gotowy zawieszony → gotowy

Zarządzanie procesami

- W celu efektywnego zarządzania procesami jądro SO przechowuje w pamięci **tablicę procesów.**
- Każdy proces reprezentowany jest przez **deskryptor procesu** (**blok kontrolny procesu – PCB**).
- **Deskryptor procesu** zawiera informacje o stanie procesu oraz dane potrzebne przy przełączaniu procesów

Deskryptor procesu

Zarządzanie procesem

Rejestry
Licznik programu
Wskaźnik stosu
Stan procesu
Priorytet
Parametry szeregowania
ID procesu
Grupa procesu
ID rodzica
Sygnały
Czas rozpoczęcia procesu
Wykorzystany czas CPU
Czas CPU potomków

Zarządzanie pamięcią

Wskaźnik segmentu kodu
Wskaźnik segmentu danych
Wskaźnik segmentu stosu

Zarządzanie systemem plików

Katalog główny
Katalog roboczy
Deskryptory otwartych plików
ID użytkownika
ID grupy

Hierarchia procesów

- Niektóre systemy operacyjne (MS Windows) nie stosują hierarchii procesów.
- W systemach UNIX procesy tworzą hierarchię
  - Każdy proces ma „rodzica” (proces nadrzędny)
  - Rodzic może mieć wielu „potomków” (procesów podrzędnych)

Tablica procesów

- Tablica procesów może być przechowywana w pamięci w postaci statycznej lub dynamicznej
- Przykład implementacji – cykliczna lista dwukierunkowa. Operacje:
  - Rezerwacja nowego obszaru na deskryptor procesu
  - Zwolnienie określonego obszaru deskryptora procesu
  - Przeglądanie tablicy (prev, next)

Tablica procesów c.d.

- W celu zwiększenia efektywności jądro przechowuje procesy gotowe do wykonania na osobnej liście.
- Procesy czekające na zdarzenia grupowane są w klasy i umieszczane w osobnych listach.
- Dodatkowo stosuje się inne struktury danych np. tablice mieszające (hash table) do szybkiego znajdowania procesu za pomocą jego PID.

Tworzenie procesu

- **Tworzenie procesu** – dodawanie nowego procesu do puli przetwarzanych procesów
- Przyczyny:
  - Nowe zadanie wsadowe
  - Logowanie użytkownika
  - System tworzy proces (np. realizacja usługi)
  - Istniejący proces tworzy proces potomny

Tworzenie procesu

- Nadanie numeru identyfikacyjnego
- Alokacja przestrzeni na obraz procesu
- Utworzenie PCB procesu
- Zainicjowanie danych przechowywanych w PCB procesu.
- Umieszczenie na odpowiedniej liście procesów.

Tworzenie procesu c.d.

- Klasycznie:
  - stworzenie nowego obrazu procesu w pamięci
  - wczytanie pliku wykonywalnego
  - rozpoczęcie wykonywania procesu
- UNIX – nietypowe podejście
  - stworzenie dokładnej kopii procesu-rodzica (funkcja fork)
  - wczytanie do przestrzeni adresowej procesu nowego pliku wykonywalnego (funkcja exec)

Tworzenie procesu c.d.

- Kopiowanie procesu (fork)
  - Podejście tradycyjne – fizyczne kopiowanie obszaru pamięci.
  - Technika „Copy on Write” - oba procesy korzystają z tych samych stron pamięci do czasu pierwszej próby zapisu
  - Stosowanie tzw. „lekkich procesów” (LWP)

Zakończenie procesu

- Zakończenie procesu – usunięcie procesu z puli działających procesów.
- Przyczyny:
  - Normalne zakończenie
  - Brak pamięci
  - Naruszenie ochrony pamięci, zasobów, przekroczenie limitów
  - Błąd wykonania (nielegalny rozkaz, dane)
  - Błąd wejścia/wyjścia
  - Zakończenie procesu macierzystego
  - Interwencja operatora

Procesy ZOMBIE

- UNIX-owe procesy mogą odpytać jądro o stan swoich procesów potomnych (np. funkcja wait).
- Aby na to pozwolić deskryptor zakończonego procesu musi być przechowywany aż proces-rodzic odbierze informację o zakończeniu potomka.
- Martwy proces ma status **TASK_ZOMBIE**

Przełączanie procesów

- Powody przełączania procesów
  - Przerwanie zegara – proces wykorzystał swój przedział czasu
  - Przerwanie we/wy – nastąpiło zdarzenie które odblokowuje proces o wyższym priorytecie
  - Błąd braku strony – proces potrzebuje strony której aktualnie nie ma w pamięci.

Etapy przełączania procesów

- Zapisanie kontekstu aktualnego procesu
- Aktualizacja PCB procesu
- Przeniesienie procesu do odpowiedniej kolejki
- Wybór innego procesu
- Aktualizacja PCB nowego procesu
- Przywrócenie kontekstu nowego procesu do stanu sprzed jego zatrzymania.

Szeregowanie procesów

- Określeniem kiedy przełączyć procesy i ile czasu przydzielać każdemu z nich zajmuje się moduł szeregowania procesów.
- Szeregowanie:
  - planowanie długoterminowe
  - planowanie średnioterminowe
  - planowanie krótkoterminowe

Planowanie długoterminowe

- Realizowane w trakcie tworzenia nowego procesu.
- Decyzja o liczbie procesów w systemie.
- Najbardziej efektywny dobór uruchamianych procesów (wydajne wykorzystanie zasobów).

Planowanie średnioterminowe

- Związane z procesem wymiany między pamięcią główną a urządzeniem wymiany (dysk).
- Decyzja, które procesy (lub ich fragmenty). usunąć z pamięci do urządzenia wymiany, a które pozostawić w pamięci głównej.
- Procesy, które będą wkrótce wykonywane, powinny być w pamięci główne, pozostałe powinny zwolnić pamięć dla procesów uruchamianych.

Planowanie krótkoterminowe

- Związane z udostępnianiem procesom czasu procesora.
- Zwykle oparte na systemie priorytetów
- Powinno reagować na różne zdarzenia w systemie np. przerwania zegarowe, przerwania z urządzeń wejścia i wyjścia, wywołania funkcji systemowych i sygnały.

Wywłaszczanie

- **Szeregowanie bez wywłaszczania** – po uzyskaniu dostępu do CPU proces wykonuje się dopóki nie zgłosi żądania I/O lub nie wywoła jakiejś funkcji systemowej.
- **Szeregowanie z wywłaszczaniem** – system operacyjny może w dowolnym momencie zabrać dostęp do CPU. Proces przenoszony jest do kolejki procesów gotowych i czeka na kolejny przydział procesora.

Kryteria ustalania priorytetu

- **Czas oczekiwania** – czas spędzony w kolejce procesów gotowych.
- **Czas obsługi** – dotychczasowy czas wykorzystany przez proces.
- **Czas przebywania w systemie** – suma czasu oczekiwania, obsługi, oraz oczekiwania na podsystem wejścia i wyjścia.
- **Obciążenie systemu** – liczba procesów w systemie, dostępne zasoby.
- **Priorytet zewnętrzny** – parametr nadany przez użytkownika.

Algorytmy szeregowania – kryteria oceny

- Z punktu widzenia systemu operacyjnego
  - wykorzystanie CPU.
  - przepustowość (liczba zakończonych procesów na jednostkę czasu).
- Z punktu widzenia użytkownika
  - czas cyklu przetwarzania (czas od uruchomienia do zakończenia procesu).
  - czas reakcji (czas między zgłoszeniem żądania a odpowiedzią).

Przykłady algorytmów szeregowania

- FCFS – First Come First Served
- LCFS – Last Come First Served
- RR - Round Robin
- SPF – Shortest Process First
- SRT – Shortest Remaining Time
- Highest Response Ratio Next
- Feedback

Algorytm FCFS

- Proces gotowy dołącza na koniec kolejki procesów gotowych.
- Gdy dany proces przerywa działanie, pobierany jest kolejny proces z kolejki.
- Stosowany przy planowaniu bez wywłaszczania.
- Lepsze zachowanie przy długich procesach wymagających CPU.

Algorytm RR

- Szeregowanie z wywłaszczaniem.
- Wywłaszczanie inicjowane jest sygnałem zegarowym o określonej częstości.
- Problem z określeniem długości kwantu czasu.
- Procesy wykorzystujące CPU są bardziej faworyzowane niż procesy związane z obsługą I/O.

Algorytm SPF

- Szeregowanie bez wywłaszczania.
- Preferowanie krótkich procesów, duża przepustowość.
- Problem z estymacją czasu potrzebnego do zakończenia procesu.
- Niebezpieczeństwo zagłodzenia długich procesów.

Wybór algorytmu szeregowania

- Brak idealnego algorytmu szeregowania.
- Efektywność zależy od specyfiki wykorzystania.
- Przykład – aktualna dyskusja związana z modułem szeregowania Linuxa:
  - Algorytm BFS – lepszy dla procesów interaktywnych, korzystających z I/O (desktop).
  - Algorytm CFS – lepszy dla przetwarzania wsadowego obciążającego CPU (serwer).

Tryb wykonywania

- Tryb użytkownika
  - Mniej uprzywilejowany
  - Niedostępne niektóre rozkazy
  - Niedostępne niektóre obszary pamięci i dane systemowe
- Tryb jądra
  - Tryb uprzywilejowany
  - Dostęp do wewnętrznych struktur danych jądra
- Przejście do trybu jądra następuje **tylko** poprzez wywołanie jednej ze ściśle zdefiniowanych funkcji systemowych (syscall).

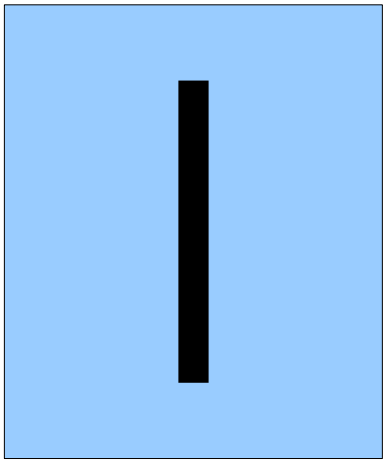
Organizacja jądra SO

- Jądro działa poza strukturą procesów
  - jądro działa poza kontekstem procesów użytkowników
  - jądro ma własną pamięć i stos
- Kod jądra działa w kontekście procesów
  - wywołania systemowe realizowane są w ramach procesu
  - procesy przechowują swoje i systemowe struktury danych
- Jądro oparte na procesach
  - jądro SO składa się z kolekcji procesów
  - wygodna implementacja
  - wydajne w systemach wieloprocesorowych

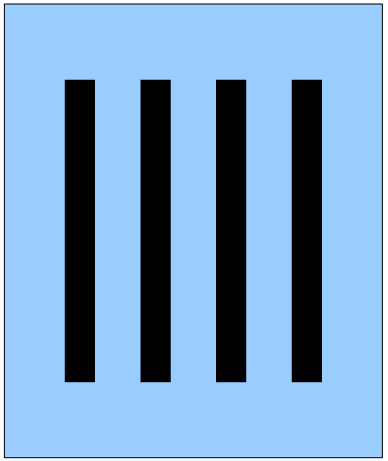
Procesy i wątki

- Proces charakteryzuje się dwiema właściwościami:
  - Prawa własności do zasobów
  - Szeregowanie / wykonywanie
- Właściwości te można traktować oddzielnie wprowadzając **wątki**

Realizacja wielowątkowości

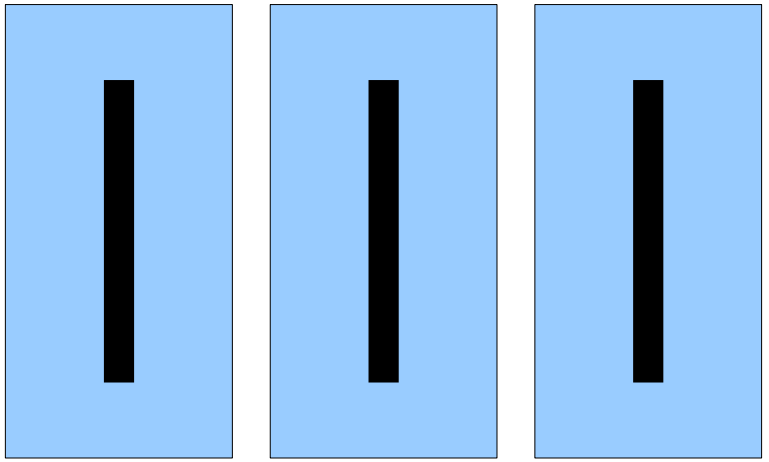


Jeden proces – jeden wątek

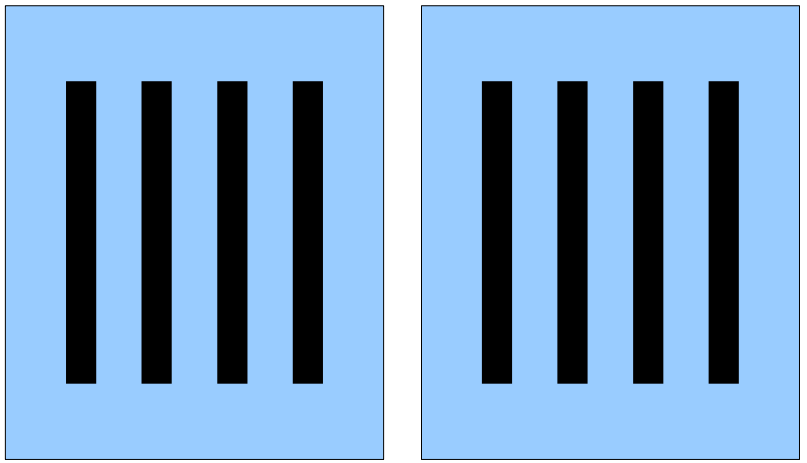


Jeden proces – wiele wątków

Realizacja wielowątkowości c.d.



Wiele procesów po 1 wątku



Wiele procesów – wiele wątków

Elementy procesu i wątku

- Proces
  - Wirtualna przestrzeń adresowa – obraz procesu
  - Chroniony dostęp do procesora, zasobów, innych procesów
- Wątek
  - Kontekst wątku
  - Lokalne dane i lokalny stos wykonawczy
  - Dostęp do pamięci i zasobów procesu (współdzielony z innymi wątkami)

Elementy procesu i wątku

ELEMENTY PROCESU	ELEMENTY WĄTKU
Przestrzeń adresowa Zmienne globalne Otwarte pliki Procesy potomne Uchwyty obsługi sygnałów Informacje rozliczeniowe	Licznik programu Rejestry Stos Stan

Korzyści stosowania wątków

- Szybkie tworzenie wątku
- Szybkie przerwanie i wznowienie wątku
- Szybkie przełączanie wątku
- Wydajna komunikacja między wątkami – poprzez pamięć wspólną

Wady stosowania wątków

- Brak ochrony zasobów – wszystkie wątki je współdzielą
- Konieczność **synchronizacji wątków** w celu ochrony danych

Wątki vs. procesy

Czas potrzebny na utworzenie wątku użytkownika, procesu lekkiego i procesu na komputerze SPARCstation 2

Mechanizm	Czas tworzenia (mikrosekundy)
Wątek użytkownika	52
Proces lekki	350
Proces	1700

Implementacje wątków

- W przestrzeni jądra (Lightweight Processes – LWP)
  - jądro jest świadome istnienia wątków i bierze je pod uwagę przy szeregowaniu procesów
- W przestrzeni użytkownika – np. biblioteka pthreads.
  - realizowane całkowicie w ramach kontekstu użytkownika – jądro traktuje je jako jeden proces.