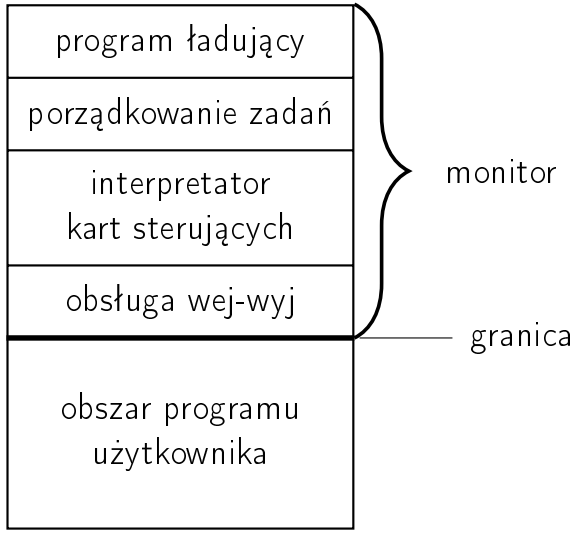
Jak system wykonuje zadania?

- Od monitora prostego po systemy wielodostępowe
- Wieloprogramowość
- Wielozadaniowość, systemy z podziałem czasu
- Wieloprocesorowość, systemy HPC
- Systemy rozproszone, klastrowe i chmurowe
- Systemy czasu rzeczywistego

Monitor prosty

- wczesne rozwiązania zarządzania zadaniami w systemach operacyjnych
- **przetwarzanie wsadowe** (*batch processing*) - zadania gromadzone w postaci wsadów są kolejno wykonywane, bez interakcji użytkownika
- **monitor rezydujący** – stale obecny w pamięci; automatycznie przekazuje sterowanie od zadania do zadania oraz monitoruje i zarządza określonymi zasobami lub zdarzeniami
- automatyczne porządkowanie zadań – przydzielanie czasu procesora poszczególnym zadaniom sekwencyjnie lub według prostych reguł
- każde zadanie musiało zakończyć się lub zrezygnować z procesora, zanim kolejne mogło zostać uruchomione
- w najprostszych monitorach brak mechanizmu przerwań sprzętowych - programista musiał jawnie wywoływać funkcje monitora, aby oddać sterowanie

Monitor rezydujący w pamięci



- **ochrona pamięci** - program nie może ingerować w obszar pamięci monitora
- **uprzywilejowane instrukcje** - pewne instrukcje mogą być wykonywane wyłącznie przez monitor (tryb jądra)

Wolne operacje wejścia-wyjścia

- Urządzenia wejścia-wyjścia są zwykle znacznie wolniejsze od procesora. W systemie wykonującym jedno zadanie naraz procesor może pozostawać bezczynny, gdy zadanie czeka na zakończenie operacji I/O.
- Celem jest oddzielenie tempa pracy procesora od tempa urządzeń, aby ograniczyć czas oczekiwania.
- Przykłady rozwiązań stosowanych we wczesnych systemach przetwarzania wsadowego:
  - **praca pośrednia (off-line, historycznie)** – obsługa wejścia i wyjścia przez odrębne urządzenia; dane przekazywano do komputera w blokach, np. na taśmie magnetycznej.
  - **spooling** – kolejka danych lub zadań przechowywana na dysku, np. zadań drukowania; podczas gdy urządzenie obsługuje kolejkę, procesor może wykonywać inne zadanie.

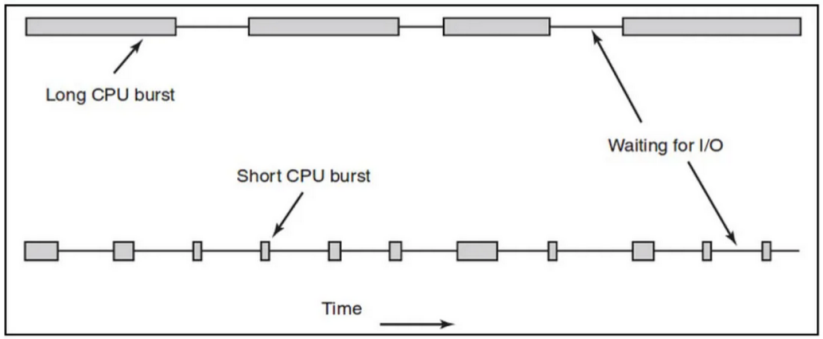
Buforowanie

Buforowanie usprawnia wykonywanie obliczeń i operacji wej-wyj w ramach zadania.

- polega na tymczasowym przechowywaniu danych w pamięci (buforze) przed ich dalszym przetwarzaniem lub przesłaniem
- zmniejsza liczbę operacji I/O i przyspiesza dostęp do danych
- procesor może wykonywać inne zadania, gdy dane są ładowane do bufora, co pozwala lepiej wykorzystać jego zasoby i skrócić czas oczekiwania na dane
- przykłady: pamięć podręczna procesora, bufor dyskowy, buforowanie w systemach plików, buforowanie w sieciach komputerowych (DNS cache)

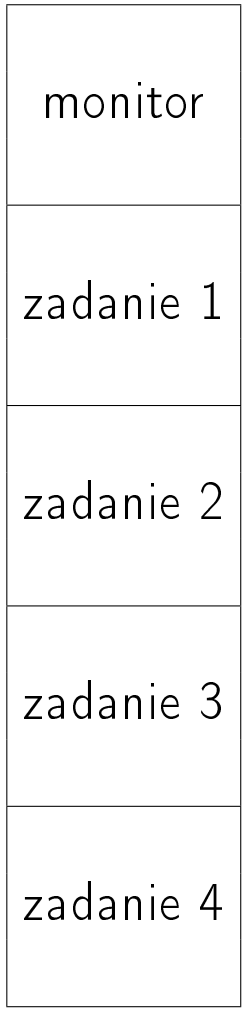
Zadania ograniczone przez CPU i I/O

- Zadania ograniczone operacjami wej-wyj (*I/O bound*)
  - szybkość przetwarzania zależy od szybkości urządzeń wej-wyj
  - buforowanie może zwiększyć przepustowość przez grupowanie operacji I/O
- Zadania ograniczone procesorem (*CPU bound*)
  - prędkość przetwarzania jest ograniczona szybkością CPU (bufor wejściowy jest zawsze pełny, a bufor wyjściowy – pusty)
  - buforowanie może zwiększyć wydajność przez gromadzenie często używanych danych i skrócenie czasu dostępu



Wieloprogramowość

- **wieloprogramowość** – w pamięci rezyduje więcej niż jeden program użytkownika, ale procesor wykonuje w danym momencie tylko jeden z nich
- po zablokowaniu zadania (np. podczas operacji I/O) system przydziela procesor innemu zadaniu aby maksymalizować wykorzystanie CPU
- **asynchroniczne operacje I/O** – procesor może kontynuować wykonywanie innych zadań, podczas gdy operacje wejścia-wyjścia są realizowane w tle



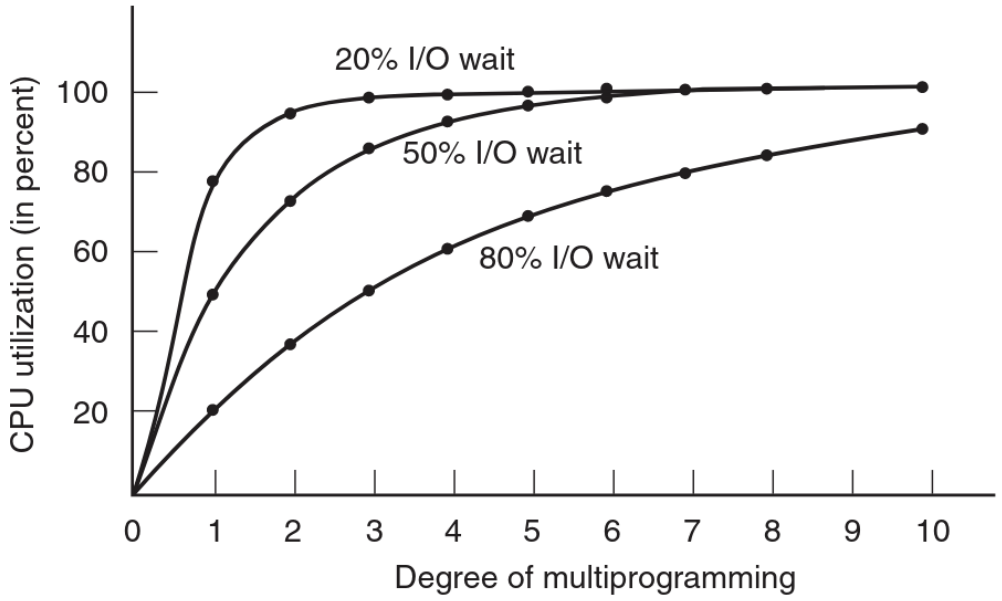
Realizacja wieloprogramowości wymaga systemu zapewniającego:

- ochronę pamięci
- planowanie przydziału procesora, zarządzanie kolejką zadań
- zarządzanie przydziałem pamięci dla zadań
- ochronę zasobów i synchronizację zadań

Wieloprogramowość: wykorzystanie CPU

Prosty model wykorzystania CPU:

- n procesów, z których każdy spędza część czasu w stanie I/O
- p - prawdopodobieństwo, że pojedynczy proces w danym przedziale czasowym nie korzysta z procesora (oczekuje na I/O)
- p^n – prawdopodobieństwo, że wszystkie n procesów będą w stanie I/O
- wykorzystanie CPU $= 1 - p^n$

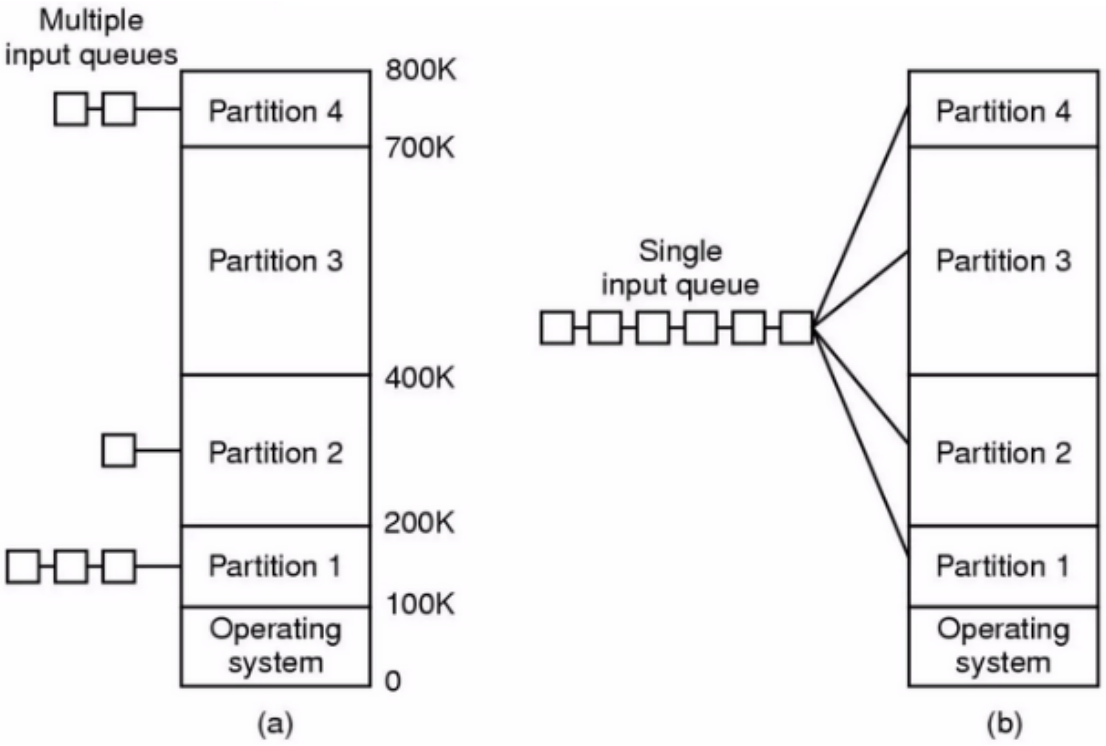


Wieloprogramowość: wykorzystanie pamięci

- podział pamięci:
  - partycje o **stałym** rozmiarze (jednakowej lub różnej wielkości)
  - podział **dynamiczny** – obszary pamięci dopasowane do rozmiaru zadania
- kolejki zadań oczekujących na przydział pamięci
  - jedna wspólna kolejka dla wszystkich partycji
  - osobne kolejki dla każdej partycji
- algorytm wyboru zadania z kolejki, np. FIFO
- problem fragmentacji pamięci:
  - **wewnętrzna** - niewykorzystana pamięć wewnątrz przydzielonego bloku, gdy blok pamięci większy od rozmiaru zadania
  - **zewnętrzna** - niewykorzystane obszary pomiędzy przydzielonymi blokami, wolna pamięć jest rozproszona w wielu małych, nieciągłych obszarach

Wieloprogramowość: przykład wykorzystania pamięci

- przykładowa sytuacja, w której zadania oczekują w kolejkach na przydział pamięci o stałym podziale
- pojedyncza kolejka vs. osobne kolejki dla każdej partycji



Wieloprogramowość ze stałym podziałem pamięci

a.) Osobne kolejki zadań do każdej partycji

- zadanie trafia do kolejki przypisanej najmniejszej pasującej partycji
- duże partycje mogą pozostawać puste, nawet gdy w innych kolejkach czekają zadania, które mogłyby się w nich zmieścić
- małe zadania mogą czekać

b.) Wspólna kolejka do wszystkich segmentów:

- zadanie z początku kolejki trafia do pierwszej wolnej partycji
- może powodować marnowanie pamięci na małe zadania
- potrzebny jest algorytm wyszukiwania zadań pasujących do wolnej partycji; może to jednak dyskryminować małe zadania

Wieloprogramowość z dynamicznym podziałem pamięci

- obszary tworzone dynamicznie dopasowane do rozmiaru zadania
- brak fragmentacji wewnętrznej - efektywne wykorzystanie pamięci RAM
- konieczność wykonywania upakowań (defragmentacji) w celu uniknięcia fragmentacji zewnętrznej (niewykorzystane obszary pomiędzy zaalokowanymi blokami), co zwiększa użycie CPU

Rozwiązania stosowane we współczesnych systemach operacyjnych:

- pamięć wirtualna
- segmentacja o stałym rozmiarze
- mechanizm stronicowania

Wielozadaniowość

- praca wsadowa a praca interakcyjna
- **wielozadaniowość** (*multitasking*) - wykonywanie wielu procesów w określonym przedziale czasu. Użytkownik ma wrażenie, że programy działają jednocześnie:
  - **współbieżnie** na jednym procesorze przez szybkie przełączanie zadań (podział czasu)
  - **równolegle** na wielu procesorach
- **wielodostępowość** - wielu użytkowników korzysta z systemu jednocześnie, np. za pośrednictwem terminali podłączonych do jednego komputera

Wielozadaniowość w systemach z podziałem czasu

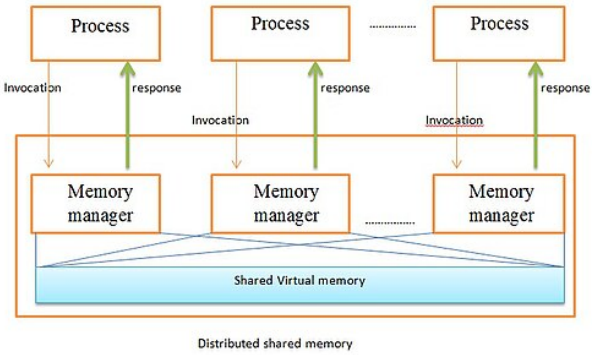
- **System z podziałem czasu** – system wielozadaniowy, w którym procesor jest przydzielany procesom na krótkie kwanty czasu.
- Po upływie kwantu przerwanie zegarowe przekazuje sterowanie jądru, które może wywłaszczyć bieżący proces i przydzielić procesor innemu gotowemu procesowi.
- Celem jest **interaktywność** – każdy użytkownik ma wrażenie ciągłej pracy systemu.

Systemy z podziałem czasu są skomplikowane, wymagają m.in.:

- realizacji mechanizmów działań współbieżnych
- zarządzania pamięcią i ochrony pamięci
- planowania przydziału CPU
- administrowania pamięcią dyskową, obsługi systemów plików
- obsługi systemu przerwań

Systemy HPC (*High Performance Computing*)

- **wieloprocesowość** (*multiprocessing*) – systemy wyposażone w więcej niż jeden procesor, umożliwiające równoczesne przetwarzanie zadań
- **SMP** (*Symmetric MultiProcessing*)
  - współdzielona pamięć, I/O
  - wspólny system operacyjny
  - jednostki CPU są równoważne
- **DSM** (*Distributed Shared Memory*)
  - rozdzielone fizycznie pamięci są adresowane wspólną przestrzenią adresową
  - tańszy niż SMP i dobrze się skaluje
  - pozwala na obsługę dużych baz danych
  - wolniejsza komunikacja między węzłami niż SMP



Systemy HPC (*High Performance Computing*)

- **MPP** (*Massively Parallel Processor*)
  - setki/tysiące procesorów, rozproszona pamięć
  - każdy CPU posiada własny system operacyjny i własną pamięć
  - podsystemy komunikują się łączami o dużej szybkości
- **klastry obliczeniowe**
  - zbiór węzłów pracujących wspólnie widocznych jako jedna jednostka obliczeniowa
  - połączenie przez sieć lokalną lub szybkie łącza
- **przetwarzanie sieciowe (grid)**
  - zbiór geograficznie rozproszonych zasobów komputerowych

Rygorystyczne systemy czasu rzeczywistego

- sterowniki urządzeń o ściśle określonym zastosowaniu: nadzorowanie procesów produkcyjnych, eksperymentów naukowych, kierowanie sygnalizacją świetlną, autopilot, itp.
- działanie podlega ostrym rygorom czasowym, nawet kosztem efektywnego wykorzystania systemu

Łagodne systemy czasu rzeczywistego

- działanie podlega złagodzonym rygorom czasowym
- niespełnienie wymagań czasowych nie jest niebezpieczne, ale powoduje pogorszenie jakości usług świadczonych przez system
- zastosowania: techniki multimedialne, tworzenie wirtualnej rzeczywistości, urządzenia zdolne do samodzielnej eksploracji

Podsumowanie i pytania sprawdzające

- Czym różni się wieloprogramowość od wielozadaniowości?
- Jakie są główne cechy systemów czasu rzeczywistego?
- Co to znaczy, że zadanie jest ograniczone przez procesor (*CPU bound*)?