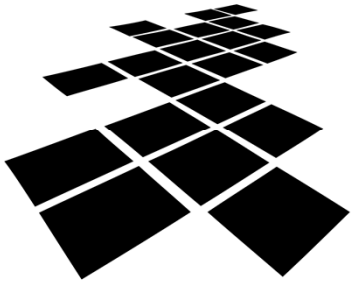
Systemy operacyjne

Planowanie przydziału procesora

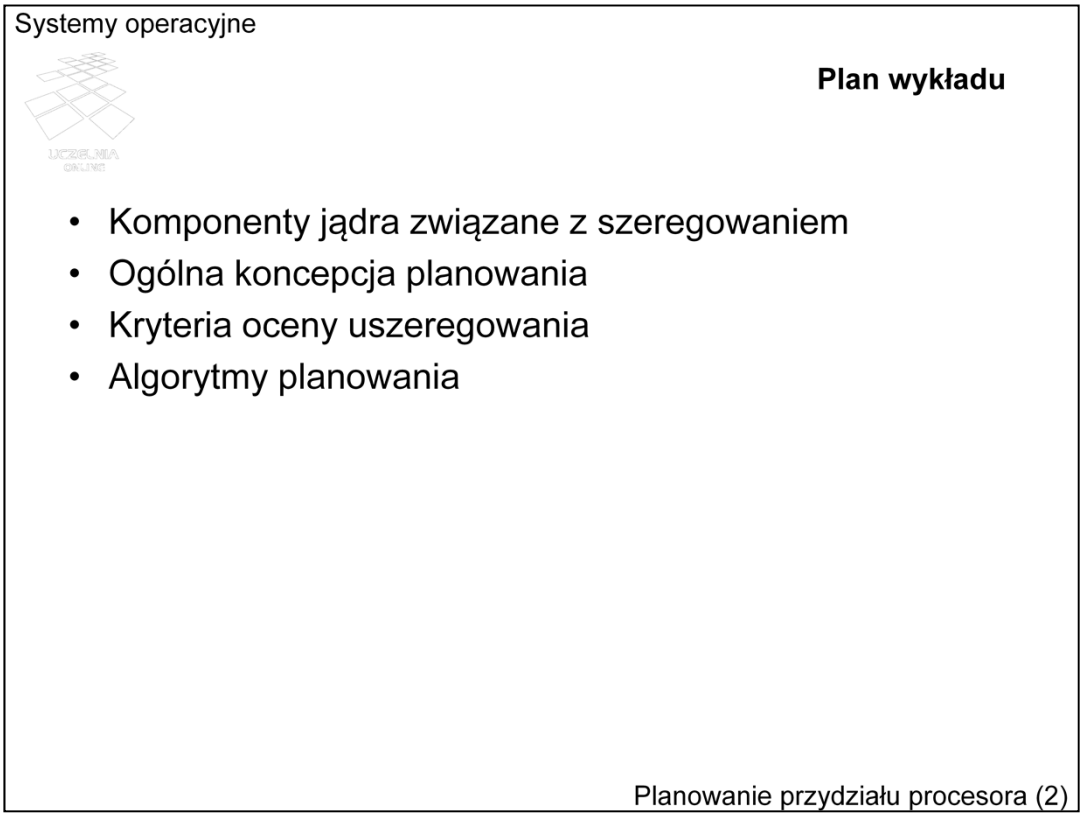
Wykład prowadzą:
Jerzy Brzeziński
Dariusz Wawrzyniak



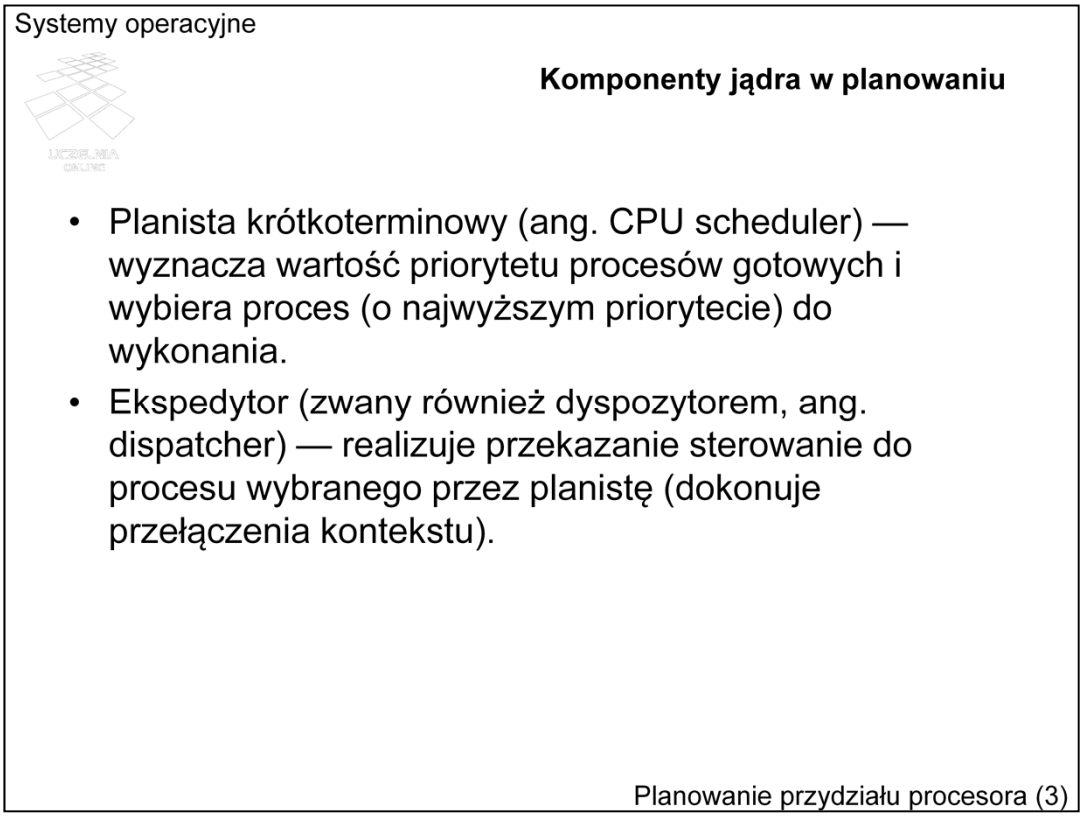
UCZELNIA
ONLINE

Celem wykładu jest przedstawienie zagadnień planowania przydziału procesora, czyli szeregowania procesów w dostępie do procesora. Planowanie takie sprowadza się do wyboru jednego z procesów (lub wątków) gotowych i przekazaniu mu procesora. Wobec różnych i często wzajemnie przeciwstawnych kryteriów optymalizacji oraz probabilistycznym charakterze niektórych przesłanek istnieje duża różnorodność podejść i algorytmów w tym zakresie. Jednym z celów jest więc pokazanie możliwych skutków podejmowanych decyzji planisty w kontekście różnych form przetwarzania i wynikających stąd oczekiwań użytkowników.

Omawiane podejścia ograniczone są do środowiska z jedną jednostką przetwarzającą i dotyczą procesów niezależnych. Kwestia zależności przewija się jedynie w problemie odwrócenia (inwersji) priorytetów, który został tylko zasygnalizowany, gdyż wybiega nieco poza planowanie przydziału samego procesora (dotyczy planowania dostępu do zasobów).



W nawiązaniu do poprzedniego wykładu krótko scharakteryzowane są komponenty jądra, istotne w realizacji szeregowania procesów. Następnie omawiana jest ogólna koncepcja szeregowania, umożliwiająca matematyczny opis podstawowych algorytmów. Przy jej omawianiu scharakteryzowane są parametry czasowe procesów, które są podstawą omawianych dalej kryteriów oceny algorytmów planowania. Ostatecznie omawiane są algorytmy planowania, wraz z niektórymi aspektami ich dostrajania, konsekwencjami stosowania oraz kwestiami implementacyjnymi.



W różnych systemach operacyjnych stosowana jest różna terminologia do określania komponentów jądra, istotnych w szeregowaniu zadań. Nie zawsze też komponenty te da się wyodrębnić strukturalnie. Funkcjonalnie jednak można oddzielić samo planowanie od realizacji decyzji, wynikających z tego planowania. Za planowanie, czyli utrzymywanie odpowiednich danych o procesach i ich powiązaniach, na podstawie których można wybrać następny proces do wykonania, odpowiedzialny jest planista krótkoterminowy (planista przydziału procesora, ang. scheduler). Zmiany wykonywanego proces, czyli przełączenia kontekstu, dokonuje ekspedytor.

Systemy operacyjne

Ogólna koncepcja planowania

- Tryb decyzji — określa okoliczności, w których oceniane i porównywane są priorytety procesów oraz dokonywany jest wybór procesu do wykonania.
- Funkcja priorytetu — funkcja wyznaczająca aktualny priorytet procesu na podstawie parametrów procesu i stanu systemu.
- Reguła arbitrażu — reguła rozstrzygania konfliktów w dostępie do procesora w przypadku procesów o tym samym priorytecie.

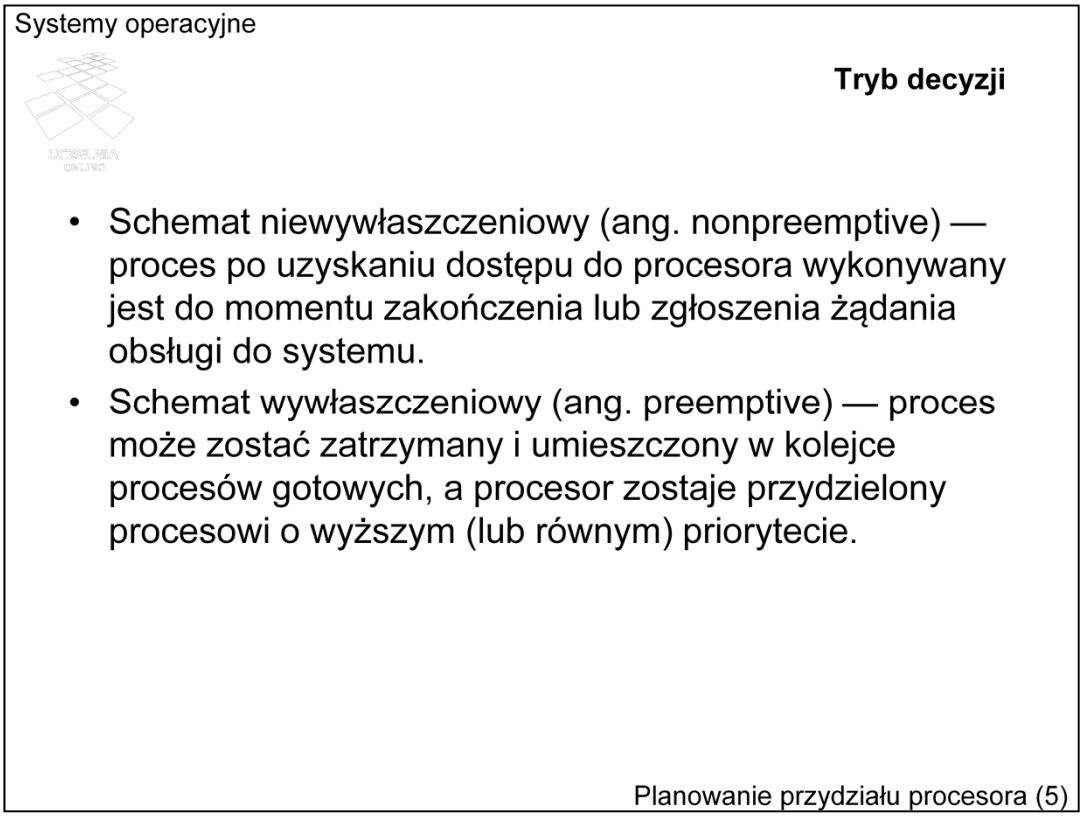
Planowanie przydziału procesora (4)

Planowanie opiera się na trzech elementach, z których dwa zasadnicze to tryb decyzji oraz funkcja priorytetu.

Funkcja priorytetu jest zbiorem wytycznych dla planisty. Zadaniem planisty jest po prostu realizacja funkcji priorytetu, ewentualnie rozwiązywanie problemów, wynikających z takiej samej wartości priorytetów dla więcej niż jednego procesu.

Wartość funkcji priorytetu jest jakąś liczbą, przy czym w niektórych systemach większa wartość tej liczby oznacza wyższy priorytet (np. w Windows), w innych wyższy priorytet to mniejsza wartość funkcji (np. w tradycyjnym systemie UNIX).

Tryb decyzji jest z kolei zbiorem wytycznych odnośnie uruchamiania ekspedytora. Szczegółowe wytyczne mogą obejmować również wartość funkcji priorytetu, gdyż jedną z okoliczności zmiany przydziału procesora jest wzrost lub spadek priorytetu procesów gotowych lub wykonywanego.



Tryb decyzji można sklasyfikować jako wywłaszczeniowy lub niewywłaszczeniowy.

W schemacie niewywłaszczeniowym procesor traktowany jest jako zasób niewywłaszczalny. Nie można go odebrać procesowi, ale proces może się go zrzec dobrowolnie (służy do tego np. funkcja `yield`, dostępna w niektórych systemach) lub się zakończyć. Rezygnacja z procesora jest też uboczną konsekwencją wejścia w stan oczekiwania (np. w wyniku zażądania operacji wejścia-wyjścia).

W schemacie wywłaszczeniowym, w ramach kontrolnego przekazania sterowania do jądra systemu operacyjnego może zostać podjęta decyzja o przełączeniu kontekstu pomimo, że wykonywany proces nie zażądał żadnej usługi, oznaczającej rezygnację z procesora. Przechodzi on wówczas do stanu gotowy, a rozpoczyna się wykonywanie innego procesu.

Systemy operacyjne

Podejmowanie decyzji o wywłaszczeniu

- Utworzenie i przyjęcie nowego procesu
- Obudzenie procesu w wyniku otrzymania komunikatu, sygnału gotowości urządzenia (przerwanie) lub sygnału wynikającego z synchronizacji
- Upłynięcie kwantu czasu odmierzanego przez czasomierz
- Wzrost priorytetu innego procesu w stanie gotowy powyżej priorytetu procesu wykonywanego — możliwe w systemie ze zmiennymi priorytetami

Planowanie przydziału procesora (6)

Typowe przypadki odebrania procesora wynikają z upływu kwantu czasu lub z pojawienia się procesu gotowego o wyższym priorytecie, niż dotychczas wykonywany. Pojawienie się takiego procesu może być następstwem:

- przyjęcia nowego procesu,
- zajścia zdarzenia, oczekiwanego przez proces (np. zakończenie operacji wejścia wyjścia, otrzymania sygnału synchronizacji),
- wzrostu priorytetu procesu gotowego, co może mieć miejsce wówczas, gdy priorytety procesów zmieniają się i przeliczane są częściej, niż to wynika z upływu kwantu czasu, czy opuszczenie procesora przez proces.

Systemy operacyjne

Funkcja priorytetu

- Argumentami funkcji priorytetu są wybrane składowe stanu procesu oraz stanu systemu.
- Priorytet procesu w danej chwili jest wartością wynikową funkcji priorytetu dla bieżących wartości parametrów stanu danego procesu i aktualnego stanu systemu.

Planowanie przydziału procesora (7)

Funkcja priorytetu uwzględnia przede wszystkim stan procesu. Może też uwzględniać stan systemu, np. stan zasobów pamięci. Ubocznym skutkiem wyznaczania wartości priorytetu jest wskazanie następnego procesu do wykonania, dlatego czasami używa się określenia *funkcja wyboru*.

Systemy operacyjne

Argumenty funkcji priorytetu (1)

- Czas oczekiwania — czas spędzony w kolejce procesów gotowych (czas spędzony w stanie gotowości)
- Czas obsługi — czas, przez który proces był wykonywany (wykorzystywał procesor) od momentu przyjęcia do systemu
- Rzeczywisty czas przebywania w systemie — czas spędzony w systemie od momentu przyjęcia (czas obsługi + czas oczekiwania + czas realizacji żądań zasobowych)
- Czasowa linia krytyczna — czas, po którym wartość wyników spada (nawet do zera, np. przy przewidywaniu pogody)

Planowanie przydziału procesora (8)

Najprostsze algorytmy planowania (szeregowania) wynikają z konstrukcji funkcji priorytetu w oparciu o parametry czasowe procesu, takie jak czas oczekiwania, czas obsługi, czy czas przebywania w systemie.

Określenie *czas oczekiwania* może być mylące i kojarzyć się ze stanem oczekiwania. W rzeczywistości jest to czas spędzony w stanie *gotowości*, a więc czas **oczekiwania na procesor**. Czas oczekiwania nie obejmuje więc czasu spędzonego przez proces w oczekiwaniu na przydział zasobów, związanych z realizacją operacji wejścia-wyjścia, czy synchronizacją. Na rzeczywisty czas spędzony w systemie składa się czas obsługi (przez procesor), czas oczekiwania (na procesor) i czas realizacji żądań zasobowych, podczas którego proces znajduje się w stanie oczekiwania.

Systemy operacyjne

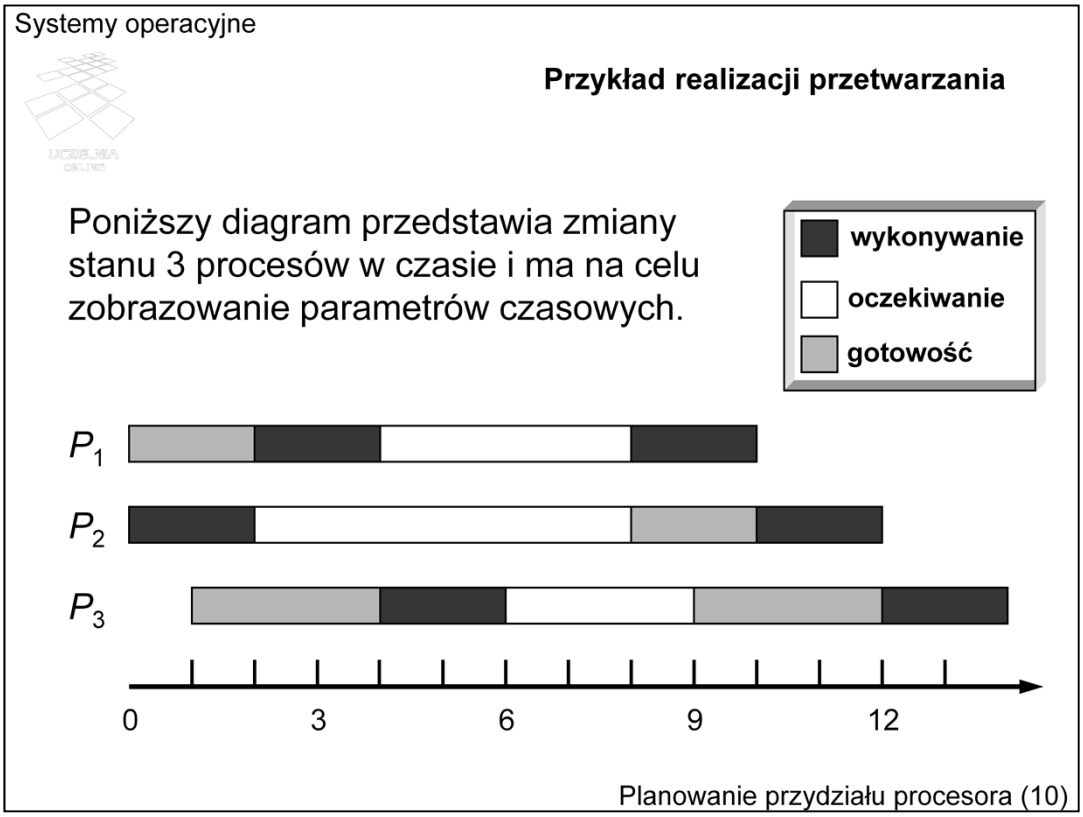
Argumenty funkcji priorytetu (2)

- Priorytet zewnętrzny — składowa priorytetu, która pozwala wyróżnić procesy ze względu na klasy użytkowników lub rodzaj wykonywanych zadań
- Wymagania odnośnie wielkości przestrzeni adresowej pamięci
- Obciążenie systemu — liczba procesów przebywających w systemie i ubiegających się (potencjalnie) o przydział procesora lub innych zasobów, zajętość pamięci

Planowanie przydziału procesora (9)

Istotnym parametrem procesu, stanowiącym argument funkcji priorytetu, jest priorytet zewnętrzny. Nadając różne priorytety, można pewne procesy uprzywilejować, a inne, mało istotne, degradować. Jądro systemu dostarcza mechanizm uruchamiania procesów, ale nie zna ich specyfiki i roli, stąd priorytet taki musi być ustalony przez użytkownika lub administratora poza jądrem systemu.

W priorytecie procesu można też uwzględnić bilans żądań procesu i możliwości systemu w bieżącym jego stanie. Taka regulacja priorytetu ma na celu wczesne przeciwdziałanie nadmiernemu obciążeniu systemu w czasie niedostępności zasobów w bezpiecznej ilości lub szybkie zwalnianie zasobów, potrzebnych innym, wysokopriorytetowym procesom (problem inwersji priorytetu).



W przedstawionym przykładzie w chwili 0 w systemie istnieją 2 procesy. Proces P_2 otrzymuje procesor i jest wykonywany przez 2 jednostki czasu, po czym wybierany jest proces P_1, a po kolejnych 2 jednostkach czasu proces P_3. W chwili 6 wszystkie procesy są w stanie oczekiwania, procesor jest więc bezczynny (wykonuje nieskończoną pętlę zwaną procesem/wątkiem bezczynności). W chwili 8 przydzielane są zasoby, oczekiwane przez proces P_1 i P_2 (np. kończą się operacje wejścia-wyjścia, docierają sygnały synchronizacji itp.). Proces P_1 otrzymuje procesor, a proces P_2 przechodzi w stan gotowości. W międzyczasie w stan gotowości po zakończonym oczekiwaniu wchodzi proces P_3. Każdy z procesów do zakończenie potrzebuje jeszcze 2 jednostek czasu procesora (czasu obsługi). Proces P_1 kończy się zatem w chwili 10, proces P_2 w chwili 12, a proces P_3 w chwili 14.

Parametry czasowe procesów, wynikające z tego przetwarzania są następujące:

- czas obsługi: 4 jednostki w przypadku każdego procesu,
- czas cyklu przetwarzania: proces P_1 — 10 jednostek czasu, proces P_2 — 12 jednostek czasu, proces P_3 — 13 jednostek czasu,
- czas oczekiwania: proces P_1 — 2 jednostki czasu, proces P_2 — 2 jednostki czasu, proces P_3 — 6 jednostek czasu.

Przy okazji można też stwierdzić, że średnie wykorzystanie procesora w czasie tego przetwarzania wynosi 12/14 = 86% (w przybliżeniu).

Systemy operacyjne

Reguła arbitrażu

- Losowo — możliwe w przypadku, gdy liczba procesów o tym samym priorytecie jest niewielka
- Cyklicznie — cykliczny przydział procesora kolejnym procesom
- Chronologicznie — w kolejności przyjmowania procesów do systemu
- FIFO — w kolejności uzyskiwania gotowości (wejścia do kolejki procesów gotowych)

Planowanie przydziału procesora (11)

Arbitraż losowy przy małej zmienności priorytetów mógłby prowadzić do głodzenia procesów.

Arbitraż cykliczny jest trudny w realizacji przy zmiennych priorytetach. Można go z powodzeniem realizować przy stałych priorytetach przy odpowiednim wsparciu ze strony struktur danych.

Arbitraż chronologiczny wydaję się być najbardziej sprawiedliwym, ale wymaga utrzymania odpowiednich atrybutów procesów lub użycia pewnych struktur danych do powiązania procesów w celu ustalenia kolejności przyjmowania ich do systemu.

Systemy operacyjne

Kryteria oceny uszeregowania (1)

- Efektywność z punktu widzenia systemu
  - wykorzystanie procesora (processor utilization) — procent czasu, przez który procesor jest zajęty pracą
  - przepustowość (throughput) — liczba procesów kończonych w jednostce czasu
- Inne aspekty z punktu widzenia systemu
  - sprawiedliwość (fairness) — równe traktowanie procesów
  - respektowanie zewnętrznych priorytetów procesów
  - równoważenie obciążenia (wykorzystania) zasobów

Planowanie przydziału procesora (12)

W ocenie jakości planowania można przyjąć różne kryteria. Kryteria te bardzo często są ze sobą w sprzeczności w tym sensie, że poprawa uszeregowania z punktu widzenie jednego kryterium powoduje pogorszenie z punktu widzenia innego kryterium (trade-off pomiędzy kryteriami).

Część kryteriów ma charakter ilościowy — można je zmierzyć i obiektywnie ocenić. Przykładem jest wykorzystanie procesora lub przepustowość. W innych przypadkach ocena ilościowa jest trudna, a ewentualne miary nie jednoznaczne i nie zawsze obiektywne. Przykładem może być sprawiedliwość, co zostało zasygnalizowane przy omawianiu zagadnień synchronizacji procesów.

Systemy operacyjne

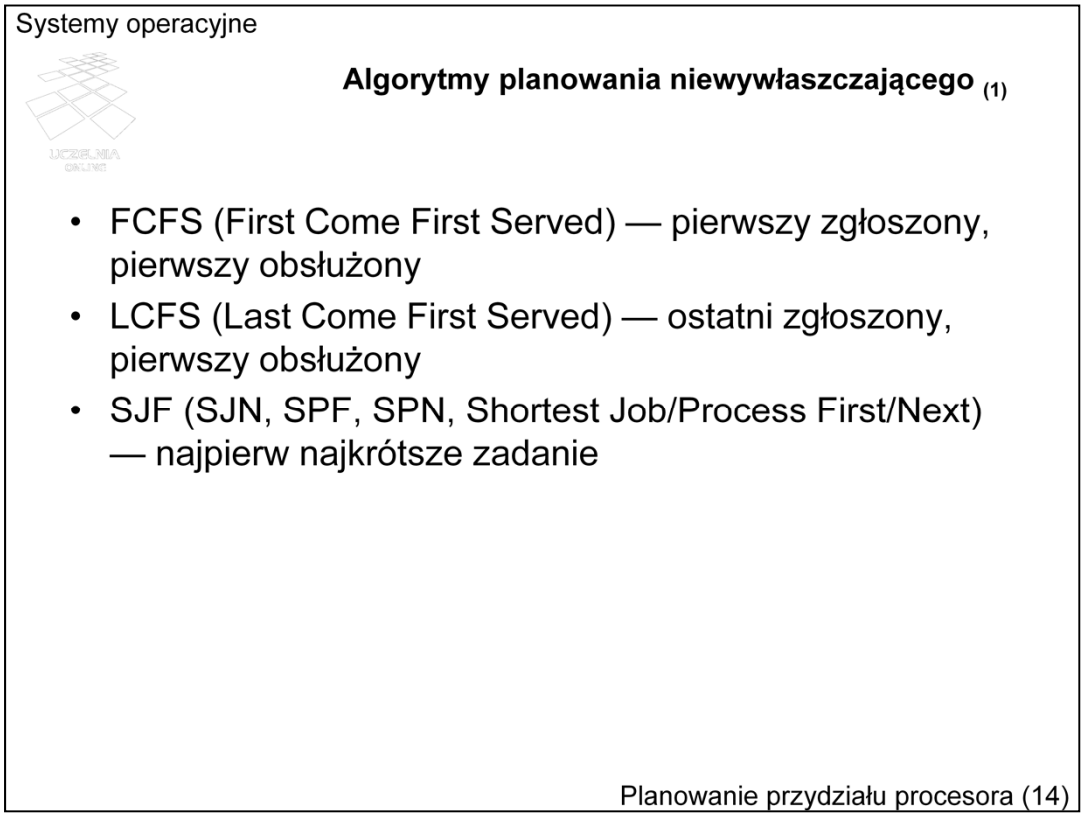
Kryteria oceny uszeregowania (2)

- Efektywność z punktu widzenia użytkownika
  - czas cyklu przetwarzania (turnaround time) — czas pomiędzy przedłożeniem zadania, a zakończeniem jego wykonywania (rzeczywisty czas przebywania w systemie w momencie zakończenie procesu),
  - czas odpowiedzi (reakcji, response time) — czas pomiędzy przedłożeniem żądania, a rozpoczęciem przekazywania odpowiedzi.
  - czas opóźnienia — czas od linii krytycznej do momentu zakończenia wykonywania
- Inne aspekty z punktu widzenia użytkownika
  - przewidywalność — realizacja przetwarzania w zbliżonym czasie niezależnie od obciążenia systemu.

Planowanie przydziału procesora (13)

Sprzeczność występuje najczęściej pomiędzy kryteriami, istotnymi z punktu widzenie interesów użytkowników, a kryteriami oceny całego systemu. Dla użytkownika istotna jest minimalizacja czasu odpowiedzi, czasu cyklu przetwarzania lub czasu oczekiwania. W przypadku ustalenia czasowej linii krytycznej istotna jest również minimalizacja opóźnienia, zakładając że jest dopuszczalne (np. w systemach łagodnego czasu rzeczywistego). Dla systemu istotna jest natomiast maksymalizacja wykorzystania zasobów (np. procesora) lub przepustowość. Obciążenie procesora przez dużą liczbę zadań może jednak wpływać na zwiększenie czasu cyklu przetwarzania, czasu oczekiwania, czasu odpowiedzi oraz opóźnienia.

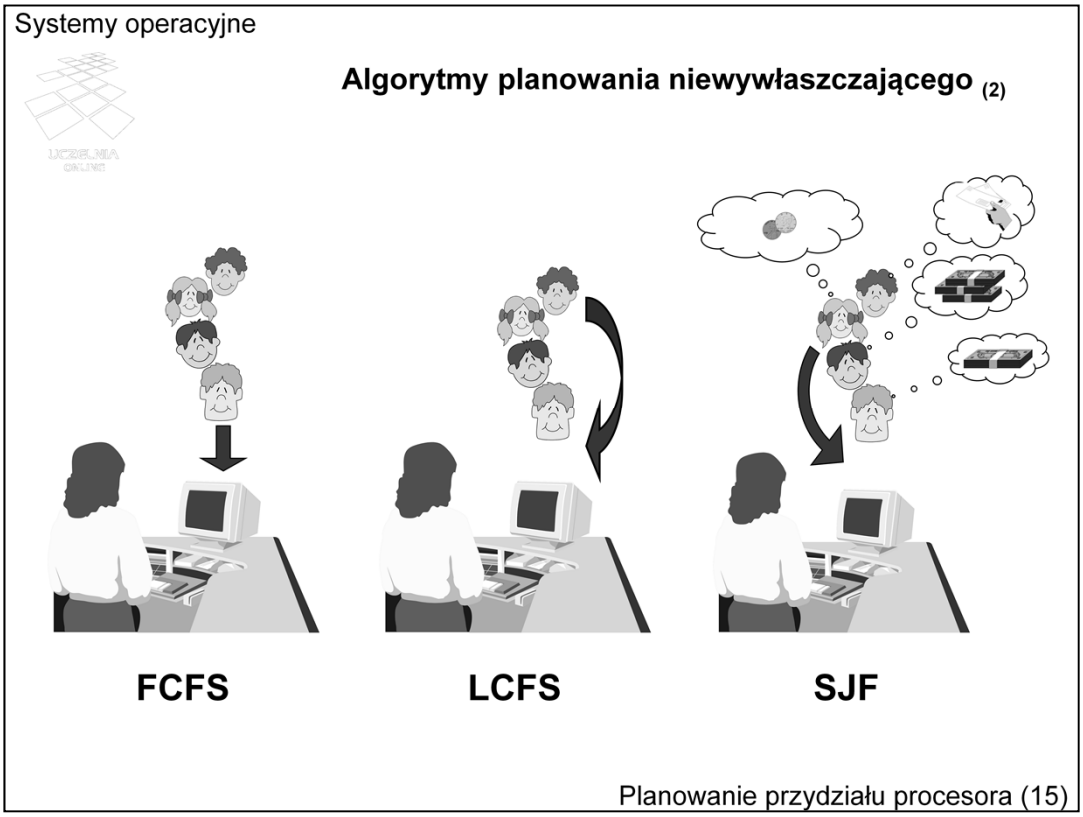
Interes użytkownika powinien być przedkładany w systemach interakcyjnych. Istotnym parametrem w tych systemach jest czas odpowiedzi. Odpowiedź przekazywana jest za pośrednictwem jakiegoś urządzenia wejścia-wyjścia, a więc z punktu widzenia planowania przydziału procesora istotny jest czas przetwarzania do momentu zażądania odpowiedniej operacji wejścia-wyjścia. Obok minimalizacji tego czasu, ważna jest też jego przewidywalność. Do oceny przewidywalności można użyć jakieś statystycznej miary rozrzutu (np. wariancji), ale wykorzystanie takiej miary w optymalizacji uszeregowania jest trudne. W systemach wsadowych dąży się z kolei przede wszystkim do optymalizacji wykorzystania zasobów. Próbę optymalizacji w tego typu systemach można podjąć, gdyż znany jest najczęściej zbiór zadań do zrealizowania. Nawet jeśli spontanicznie pojawia się nowe zadanie, nie ma potrzeby podejmowani natychmiastowej obsługi.



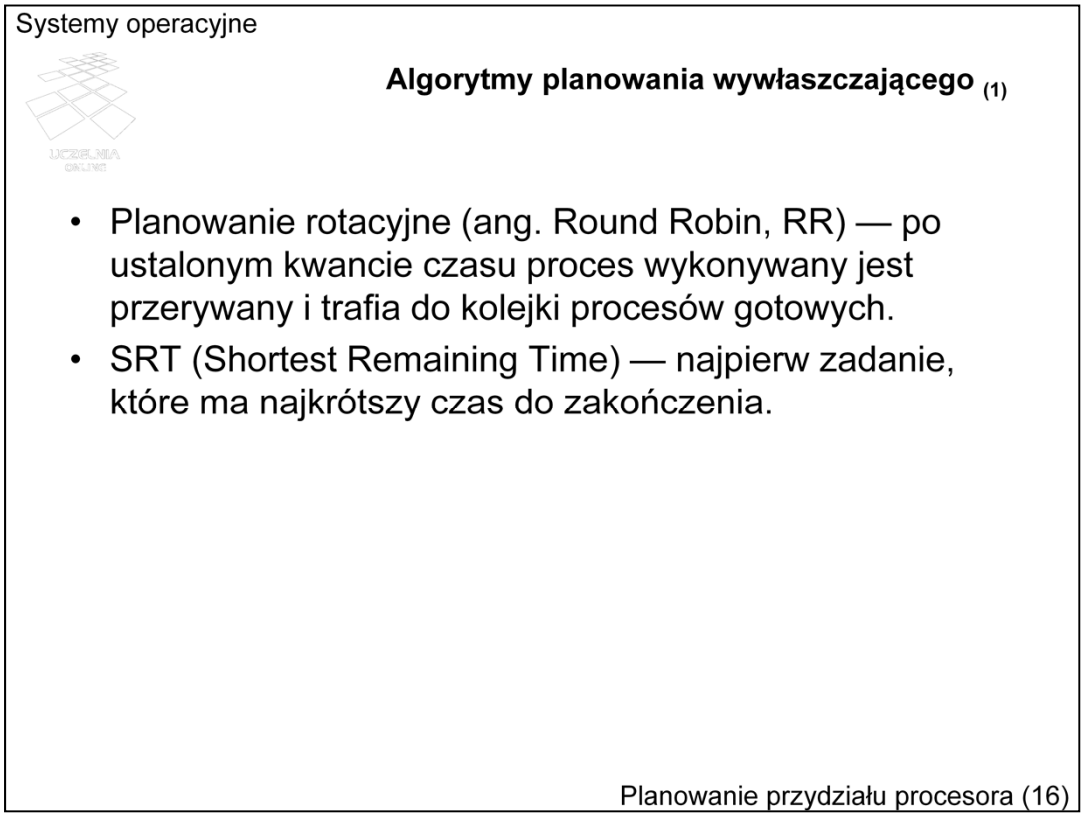
FCFS jest naturalnym algorytmem w systemach obsługi masowej, takich jak kasy sklepowe, kasy biletowe, banki, urzędy itp. Procesy otrzymują procesor w kolejności, w jakiej zgłosiły się do systemu. Specyficzną cechą w systemach komputerowych, nie zawsze mającą odpowiednik w systemach masowej obsługi, jest możliwość oddania procesora innemu procesowi na czas oczekiwania na przydział dodatkowego zasobu, np. wykonania operacji wejścia-wyjścia.

Algorytm LCFS obsługuje procesy w kolejności odwrotnej do kolejności zgłoszeń. Algorytm nie wywłaszcza procesów, więc nowo przychodzący proces jest pierwszy w kolejce i czeka na zwolnienie procesora przez bieżąco wykonywany proces. Algorytm ten wymieniany jest dla porządku, nie ma natomiast praktycznego zastosowania we współczesnych koncepcjach planowania przydziału procesora.

Algorytm SJF preferuje procesy, które mają najmniejsze wymagania odnośnie czasu procesora, potrzebnego na realizację przetwarzania. W kontekście systemów komputerowych preferencje te należałoby raczej określić jako *najpierw zadanie z najkrótszą następną fazą procesora*, gdyż po odwołaniu do jądra w celu przydziału dodatkowych zasobów nastąpi zwolnienie procesora. Algorytm ten ma sens również w systemach masowej obsługi (często w kolejce do kasy w sklepie przepuszczamy kogoś, kto ma niewiele produktów w koszyku). Problemem praktycznej stosowalności jest określenie przyszłego zapotrzebowania na procesor.

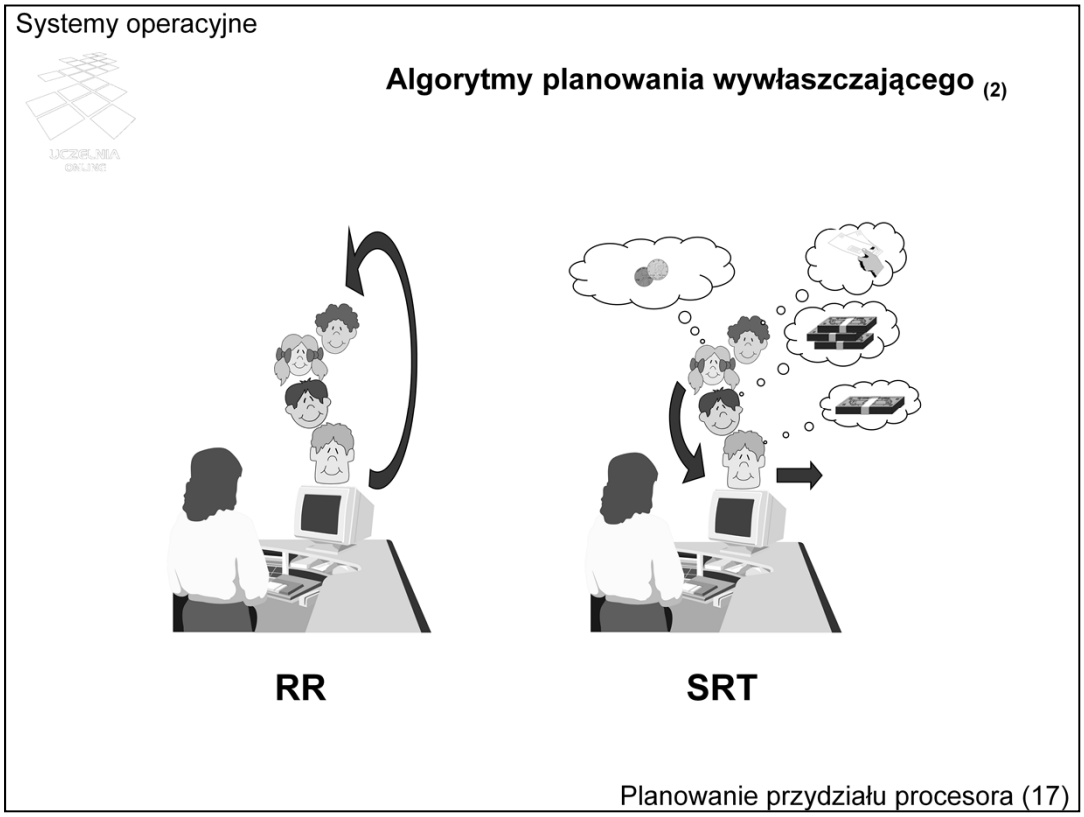


Na slajdzie zobrazowano działanie podstawowych algorytmów planowania niewywłaszczającego. Zakładając, że procesy kolejkowane są zgodnie z kolejnością zgłoszeń, w algorytmie FCFS wybierany jest proces z czoła kolejki, w algorytmie LCFS wybierany jest proces z ogona (końca) kolejki, a w algorytmie SJF kolejkę należy przejrzeć w celu znalezienia procesu, który najmniej zaabsorbuje procesor.



Typowym algorytmem planowania wywłaszczającego jest algorytm rotacyjny. Każdy proces wykonywany jest co najwyżej przez pewien okres czasu, po czym następuje przełączenie kontekstu na inny proces. Po jakimś czasie nastąpi wznowienie procesu przerwanego. Proces może przed upływem kwantu czasu zgłosić żądanie zasobowe, zrezygnować dobrowolnie z procesora lub zakończyć się, co skutkuje przydzieleniem nowego kwantu dla następnego procesu. W planowaniu rotacyjnym wszystkie procesy mają ten sam priorytet. Zasadniczym kosztem stosowania algorytmu rotacyjnego jest zużycie czasu procesora na przełączanie kontekstu. Z punktu widzenia przetwarzania użytkowego czas ten jest marnowany. Taki algorytm trudno byłoby wdrożyć w systemach masowej obsługi, ale w czasach kryzysu, gdy każdy towar podlegał reglamentacji, właściwie coś takiego funkcjonowało...

SRT jest wywłaszczającą wersją algorytmu SJF. Zakładając, że znany jest czas następnej fazy procesora dla każdego procesu, sprawdza się, czy jakiś proces gotowy ma mniejsze wymagania odnośnie czasu procesora, niż proces aktualnie wykonywany. Jeśli tak, to podejmowana jest decyzja o wywłaszczeniu.



Na slajdzie po lewej stronie zobrazowano działanie algorytmu RR, w którym proces po wykorzystaniu przysługującego mu kwantu czasu przechodzi na koniec kolejki procesów gotowych i czeka na kolejny przydział procesora.

W algorytmie SRT (po prawej) proces, który ma mniejsze potrzeby odnośnie czasu procesora wywłaszcza proces obsługiwany. Ponieważ procesy obsługiwane są wg. zapotrzebowania na czas procesora, polityka porządkowania w zbiorze procesów gotowych nie ma większego znaczenia, chyba że kolejność uwzględnia to zapotrzebowanie.

Systemy operacyjne

Podstawowe algorytmy planowania a funkcja priorytetu

- Podstawowe algorytmy planowania można uzyskać przez odpowiednią definicję funkcji priorytetu.
- Parametrami funkcji priorytetu dla podstawowych algorytmów planowania są następujące atrybuty czasowe procesów:
  - *a* — bieżący (dotychczasowy) czas obsługi
  - *r* — rzeczywisty czas w systemie
  - *t* — całkowity wymagany czas obsługi (czas obsługi do momentu zakończenia)

Planowanie przydziału procesora (18)

Systemy operacyjne

Własności algorytmów planowania

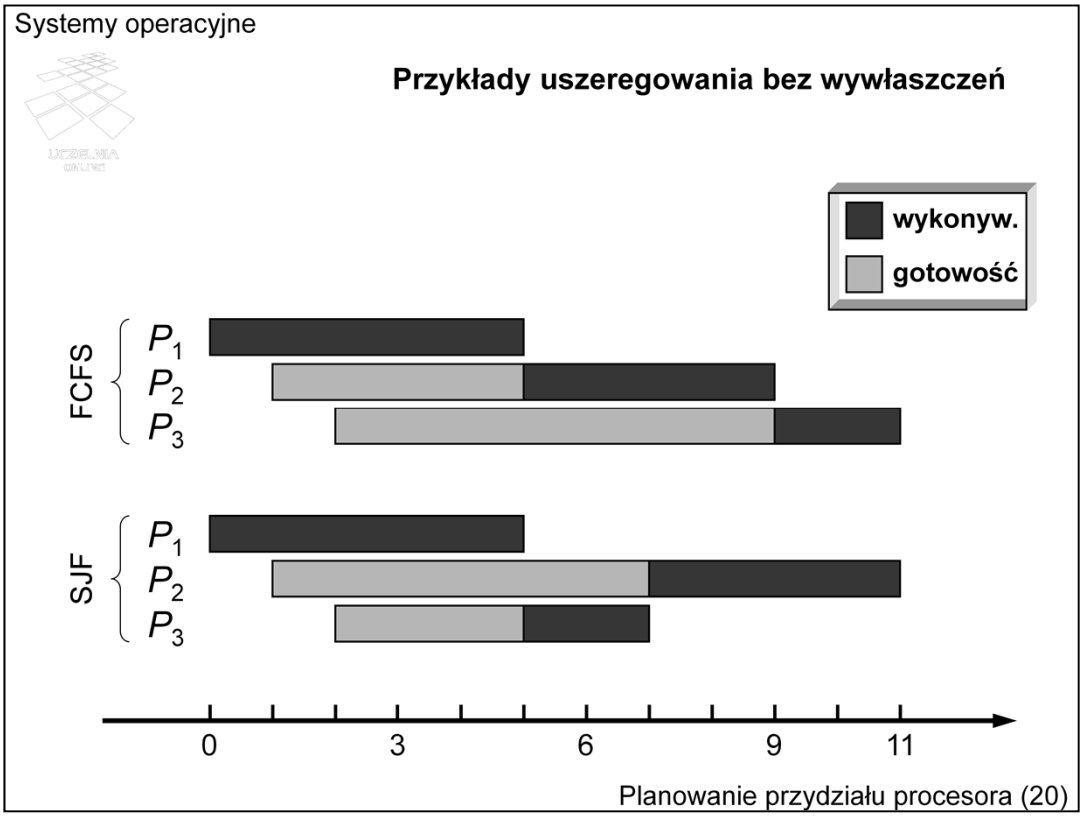
algorytm	priorytet	tryb decyzji	arbitraż
FCFS	*r*	niewywłaszczeniowy	losowy
LCFS	*-r*	niewywłaszczeniowy	losowy
SJF	*-t*	niewywłaszczeniowy	losowy lub chronologiczny
SRT	*a - t*	wywłaszczeniowy	losowy lub chronologiczny
RR	stały	wywłaszczeniowy	cykliczny

Planowanie przydziału procesora (19)

Interpretacja priorytetu procesu jest taka, że większa wartość oznacza wyższy priorytet. W implementacjach mechanizmów planowania w systemach operacyjnych czasami jest odwrotnie (np. w systemie UNIX lub Linux).

W przypadku wywłaszczeniowego trybu decyzji moment zmiany kontekstu zależy ogólnie od priorytetów. W algorytmie SRT priorytetem jest czas, pozostały do zakończenia. Momentem, w którym analiza priorytetu ma sens, jest przyjęcie procesu do systemu lub zakończenie procesu. W algorytmie RR, gdzie priorytet jest stały (w najprostszym przypadku równy dla wszystkich), momentem podejmowania decyzji jest upływ kwantu czasu.

Oczywiście niezależnie od algorytmu i trybu decyzji zmiana kontekstu następuje w przypadku zakończenia procesu lub wejścia procesu w stan oczekiwania. W tych przypadkach stosowane są takie same reguły wyboru następnego procesu do wykonania.



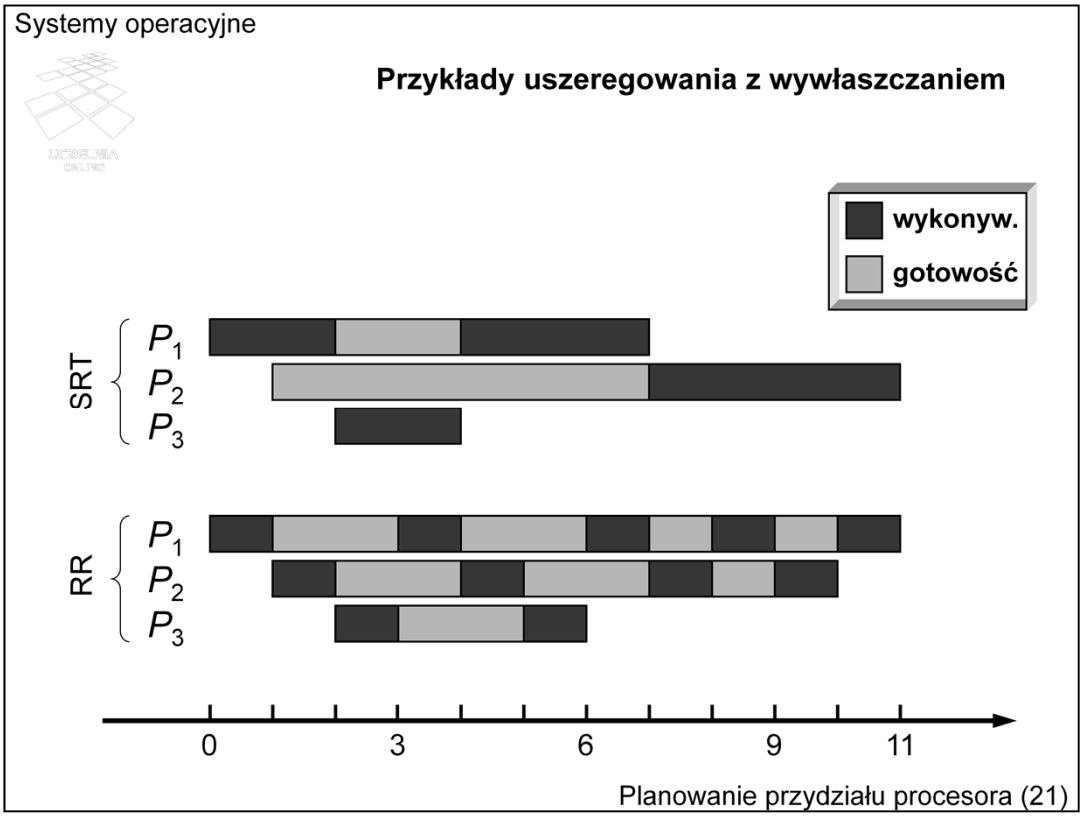
Na diagramie zaprezentowano uszeregowanie 3 procesów P_1, P_2 i P_3, które zgłaszają się do systemu w odstępie 1 jednostki czasu (przyjmijmy sekundę, jako jednostkę). Proces P_1 potrzebuje 5 jednostek czasu procesora na obsługę, P_2 potrzebuje 4 jednostek, a P_3 — 2 jednostek. Dla uproszczenia założono, że procesy nie generują żądań zasobowych i tym samym nie wchodzą w stan oczekiwania.

W pierwszym uszeregowaniu zgodnym z algorytmem FCFS procesy wykonują się w kolejności P_1, P_2, P_3. Czasy oczekiwania tych procesów wynoszą odpowiednio 0, 4, 7. Średni czas oczekiwania wynosi wynosi 11/3≈3,67. Czasy cyklu przetwarzania wynoszą odpowiednio 5, 8, 9, co daje średni czas cyklu przetwarzania 22/3≈7,33.

Drugie uszeregowanie jest dla algorytmu SJF, chociaż **zupełnie przypadkowo** takie samo uszeregowanie byłoby dla algorytmu LCFS. Dokonując podobnej analizy uzyskujemy średni czas oczekiwania (0+6+3)/3 = 3, a średni czas cyklu przetwarzania (5+10+5)/3≈6,67.

Z indywidualnej perspektywy każdego procesu należałoby określić relacje pomiędzy czasem oczekiwania i czasem obsługi. Przyjmując stosunek czasu obsługi do sumy czasu oczekiwania i czasu obsługi jako miarę efektywności z punktu widzenia procesu otrzymujemy:

- uszeregowanie FCFS: P_1 — 100%, P_2 — 50%, P_3 — 22%,
- uszeregowanie SJF: P_1 — 100%, P_2 — 40%, P_3 — 40%.



W algorytmie SRT w chwili uzyskania gotowości przez proces P_2 oba procesy (P_1 i P_2) mają ten sam priorytet. Optymalizując liczbę przełączeń kontekstu wykonywany jest w dalszym ciągu proces P_1, co jest również zgodne z arbitrażem chronologicznym.

Kontynuując analizę dla algorytmu SRT otrzymujemy średni czas oczekiwania (2+6+0)/3≈2,67, a średni czas cyklu przetwarzania — (7+10+2)/3≈6,33. Stosunek czasu obsługi do czasu cyklu przetwarzania wynosi: 71%, 40%, 100% odpowiednio dla procesów P_1, P_2 i P_3.

Analogiczne wyliczenia dla algorytmu RR przy kwancie 1 jednostki czasu dają następujące wyniki: średni czas oczekiwania (6+5+2)≈4,33, a średni czas cyklu przetwarzania (11+9+4)/3≈8 oraz współczynnik efektywności dla procesów P_1, P_2 i P_3 odpowiednio 45%, 44%, 50%. W formie ćwiczenia proponuje się przeprowadzanie podobnej analizy przy kwancie czasu 2 jednostki. W analizie pominięto koszt przełączania kontekstu, które zależnie od wielkości kwantu może być częstym zjawiskiem w algorytmie RR.

Przedstawione przykłady są ilustracja niektórych własności podstawowych algorytmów planowania. Algorytm FCFS gwarantuje efektywność przetwarzania całego systemu, gdyż nie wymaga częstego przełączania kontekstu, ale jest niesprawiedliwy dla procesów. Algorytmy SJF/SRT minimalizują średni czas oczekiwania (SJF dla zbioru procesów, znanego z góry) i poprawiają przepustowość, ale dyskryminują (w skrajnym przypadku głodzą) procesy wymagające dużego czasu obsługi. Na sprawiedliwe traktowanie procesów zorientowany jest algorytm RR, ale wymaga kompromisu pomiędzy długością kwantu czasu, a kosztem działania.

Systemy operacyjne

Algorytmy SJF/SRT — estymacja czasu obsługi

- Średnia arytmetyczna

$$S_{n+1} = \frac{1}{n} \cdot \sum_{i=1}^{n} T_i \quad \Leftrightarrow \quad S_{n+1} = \frac{T_n + (n-1) \cdot S_n}{n}$$

- Średnia wykładnicza

$$S_{n+1} = \alpha \cdot T_n + (1-\alpha) \cdot S_n$$

$$\Updownarrow$$

$$S_{n+1} = \sum_{i=0}^{n-1} (1-\alpha)^i \cdot \alpha \cdot T_{n-i} =$$

$$= \alpha T_n + (1-\alpha)\alpha T_{n-1} + \ldots + (1-\alpha)^i \alpha T_{n-i} + \ldots + (1-\alpha)^{n-1} \alpha T_1$$

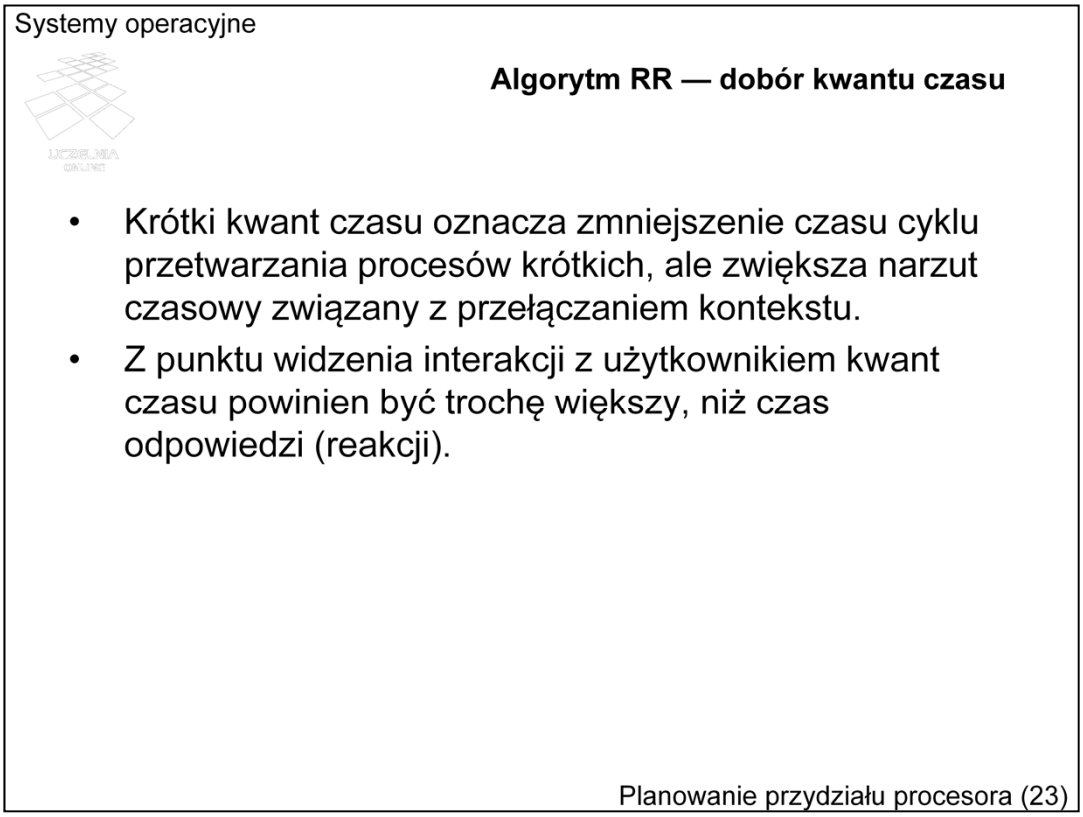
Planowanie przydziału procesora (22)

W przedstawionych przykładach uszeregowania nie rozważano przypadku wejścia procesu w stan oczekiwania w wyniku zażądania operacji wej.wyj. lub innych żądań zasobowych. Z punktu widzenia algorytmu SJF lub SRT oznacza to, że całkowity czas obsługi musi być znany w momencie zgłoszenia procesu do systemu. Tak sytuacja jest potencjalnie możliwa w systemach wsadowych, gdzie użytkownik specyfikuje ten czas. Jego zaniżenie może skutkować odrzuceniem zadania w trakcie przetwarzania po przekroczeniu zadeklarowanej wielkości, więc nie należy spodziewać się nadużyć w tym zakresie.

Z drugiej strony sytuacja, w której proces wymaga przez większość czasu wyłącznie obsługi ze strony procesora jest nieprawdopodobna w realnym przetwarzaniu. Dlatego w algorytmach SJF lub SRT uwzględnia się czas następnej (przyszłej) fazy procesora. W ogólnym przypadku czas taki trudno oczywiście wydedukować z atrybutów procesu, można jedynie próbować estymować go na podstawie czasu obsługi wcześniejszych faz.

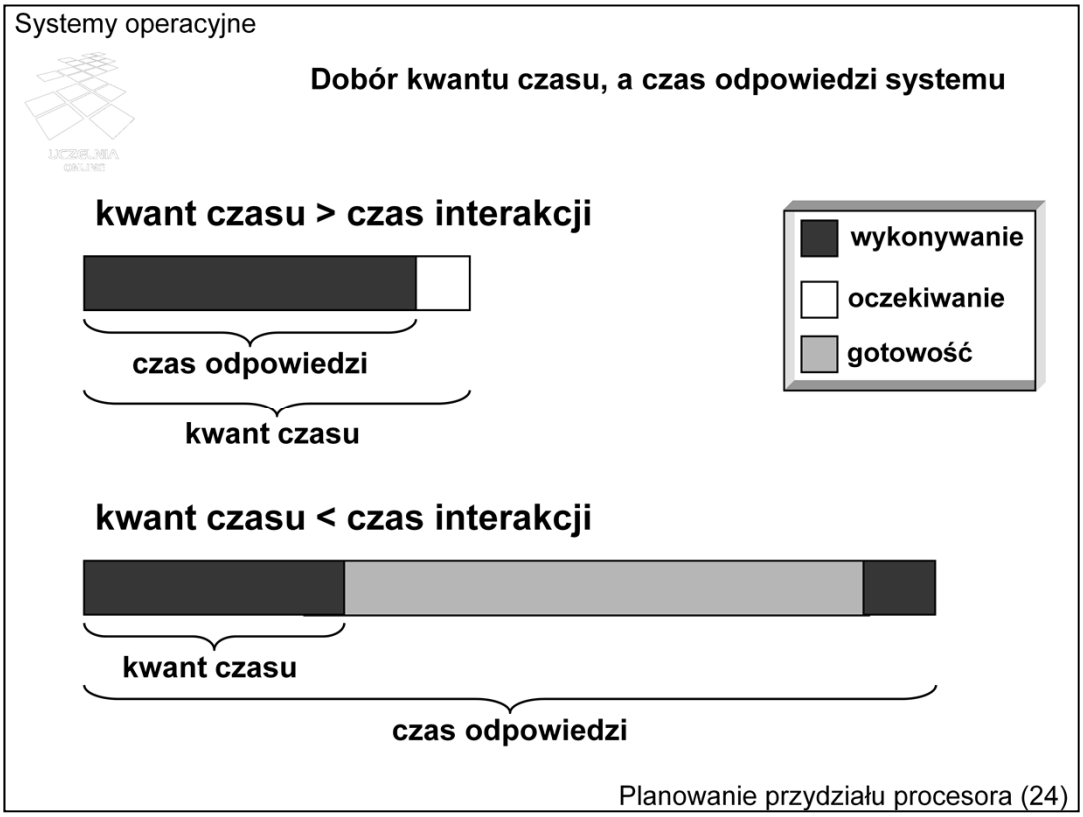
Jedno podejście polega na wyznaczeniu średniego czasu obsługi poprzednich faz procesu. Dla pierwszego okresu można przyjąć średnią po wszystkich procesach w systemie. Dotychczasową średnią można utrzymywać jako jeden z atrybutów procesu i wykorzystać do szybszego wyliczenia średniej, uwzględniającej kolejny okres, zgodnie z równoważnym wzorem.

Charakterystyka procesu może się jednak zmieniać i ważniejsze mogą się okazać przesłanki z ostatniego okresu, niż z bardziej odległej przeszłości. Można więc zastosować średnią wykładniczą, w której współczynnik α ($0 \le \alpha \le 1$) określa poziom istotności ostatniej fazy. Jak wynika z przedstawionego rozwinięcia, im wyższa potęga przy wartości $1-\alpha$, tym mniejsza istotność czasu z tego okresu. Tego typu podejścia określa się jako *postarzanie*.



Pomimo wsparcia na poziomie maszynowym, przełączanie kontekstu jest operacją wymagającą pewnej ilości czasu procesora na wykonanie odpowiednich instrukcji, związanych z zachowaniem kontekstu procesu przerywanego i odtworzeniem kontekstu procesu wznawianego. Sam czas przełączania kontekstu nie jest jedynym kosztem tej operacji, jest nim również zwiększony czas dostępu do pamięci po przełączaniu kontekstu, wynikający z braku odpowiednich danych w pamięci podręcznej. Zbyt częste przełączanie kontekstu może więc spowodować spadek znaczenia pamięci podręcznej i tym samym spadek efektywności przetwarzania.

Algorytm RR jest właściwy dla systemów interaktywnych, ale zbyt częste przełączanie kontekstu ma również bezpośredni wpływ na najistotniejszy parametr czasowy w tych systemach — czas odpowiedzi (reakcji).



Rozważmy przykład, w którym proces interaktywny został wznowiony w wyniku operacji wejścia-wyjścia, związanej z żądaniem użytkownika. Załóżmy, że proces ten otrzymuje właśnie kwant czasu procesora. Jeśli czas procesora, potrzebny na uzyskanie odpowiedzi, mieści się w jednym kwancie, odpowiedź będzie przekazana tak szybko, jak to tylko było możliwe. Jeśli jednak kwant czasu jest zbyt krótki na uzyskanie odpowiedzi, należy poczekać, aż wszystkie inne procesy gotowe wykorzystają swój kwant. Czas uzyskania odpowiedzi przez użytkownika może się więc wydłużyć wielokrotnie.

Systemy operacyjne

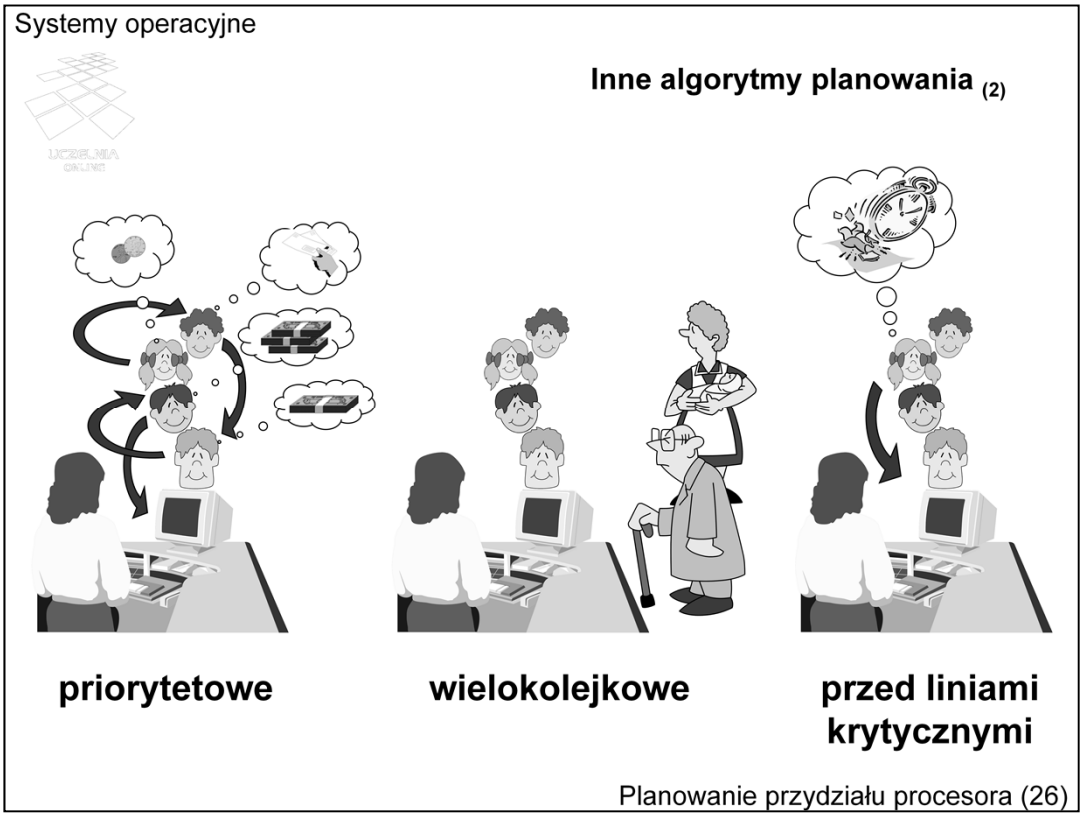
Inne algorytmy planowania (1)

- Planowanie priorytetowe — oparte na priorytecie zewnętrznym
- Planowanie wielokolejkowe — w systemie jest wiele kolejek procesów gotowych i każda z kolejek może być inaczej obsługiwana.
- Planowanie przed liniami krytycznymi — zakończenie zadania przed czasową linią krytyczną lub możliwie krótko po tej linii

Planowanie przydziału procesora (25)

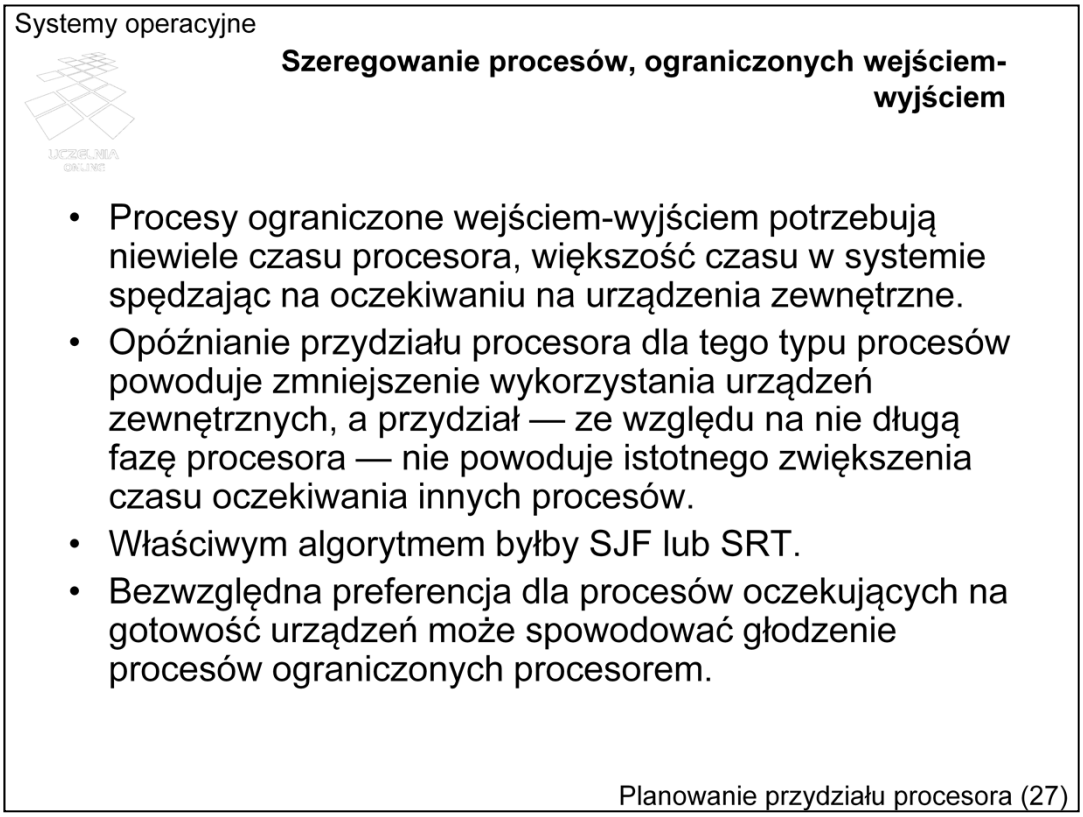
Każdy rodzaj planowania można opisać funkcją priorytetu, chociaż w realizacji strategii szeregowania nie zawsze funkcja ta jest odrębnym fragmentem kodu, realizującym matematyczną definicję. Ze względu na wymaganą szybkość działania planisty jest to wręcz niewskazane. W tym sensie każde planowanie można by nazwać priorytetowym. Planowanie priorytetowe jest tu zatem rozumiane jako oparcie strategii szeregowania na arbitralnie przyjętym priorytecie, nie wynikającym z parametrów czasowych ani innych obiektywnych parametrów procesów. Jest to wiec priorytet zewnętrzny względem procesu. Priorytet taki może być nadany przez użytkownika, nadzorcę (administratora) lub może wynikać z konfiguracji, czy też wewnętrznych uwarunkowań systemu (np. w przypadku procesów systemowych). W praktyce stosowane są często rozwiązania hybrydowe, w których priorytet zewnętrzny jest jedną ze składowych. Planowanie priorytetowe może być wywłaszczające lub niewywłaszczające.

Planowanie wielokolejkowe, zwane również planowaniem wielopoziomowych kolejek, polega na zarządzaniu wieloma kolejkami, które mogą być w różny sposób obsługiwane, tzn. przy użyciu różnych strategii, czy przypisaniu różnych priorytetów kolejkom lub przydzielaniu różnych kwantów czasu. Koncepcje wykorzystania wielu kolejek zostaną przedstawione w dalszej części.



Planowanie przed liniami krytycznymi związane jest najczęściej z systemami czasu rzeczywistego. Optymalizacja uszeregowania zależy od charakteru linii krytycznej i ewentualnych kosztów jej przekroczenia oraz wymaganych czasów obsługi. Ze znajomością czasów obsługi wiążą się podobne problemy, jak opisane przy algorytmach SJF i SRT. Szeregowanie w opisywanym tutaj zakresie sprowadza się do wyboru procesu gotowego, natomiast zasadniczym problemem w systemach czasu rzeczywistego jest odpowiednie zarządzanie zasobami, gwarantujące jak najszybsze osiąganie stanu gotowości przez procesy.

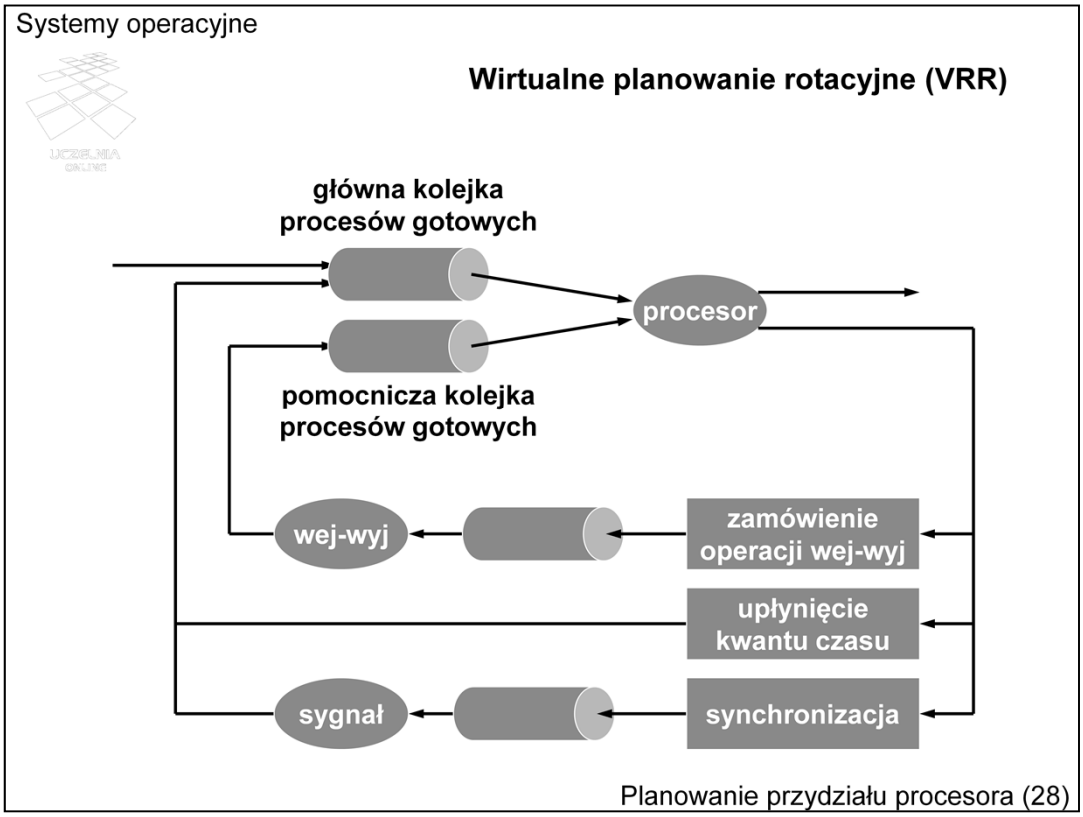
Problemem związanym z zarządzaniem zasobami, który można rozwiązać poprzez odpowiednie planowanie przydziału procesora, jest tzw. problem inwersji priorytetów. Problem powstaje wówczas, gdy proces o wysokim priorytecie jest w stanie oczekiwania na zasób, przetrzymywany przez inny proces. Proces przetrzymujący krytyczny zasób jest wprawdzie gotowy, ale ma niski priorytet, w związku z czym jest pomijany przez planistę. Wysoki priorytet procesu oczekującego na zasób nie ma znaczenia wobec niskiego priorytetu procesu przetrzymującego. Rozwiązaniem problemu jest nadanie równie wysokiego (lub wyższego) priorytetu procesowi gotowemu, żeby mógł on uzyskać procesor, wykonać kolejny fragment przetwarzania, w wyniku którego zwolni krytyczny zasób i tym samym umożliwi dalsze przetwarzanie procesu wysokopriorytetowego.



W szeregowaniu procesów ograniczonych wejściem-wyjściem chodzi głównie o to, żeby jak najszybciej zgłaszać żądania do urządzeń zewnętrznych, które są stosunkowo powolne. Jeśli proces będzie przetrzymywany w stanie gotowości, to z braku dostępu do procesora nie będzie mógł zgłosić żądania obsługi, co z kolei może powodować przestój urządzania. Po otrzymaniu czasu procesora natomiast, bardzo szybko wygeneruje on żądanie, po czym i tak zwolni procesor, wchodząc w stanu oczekiwania na powolne urządzenie.

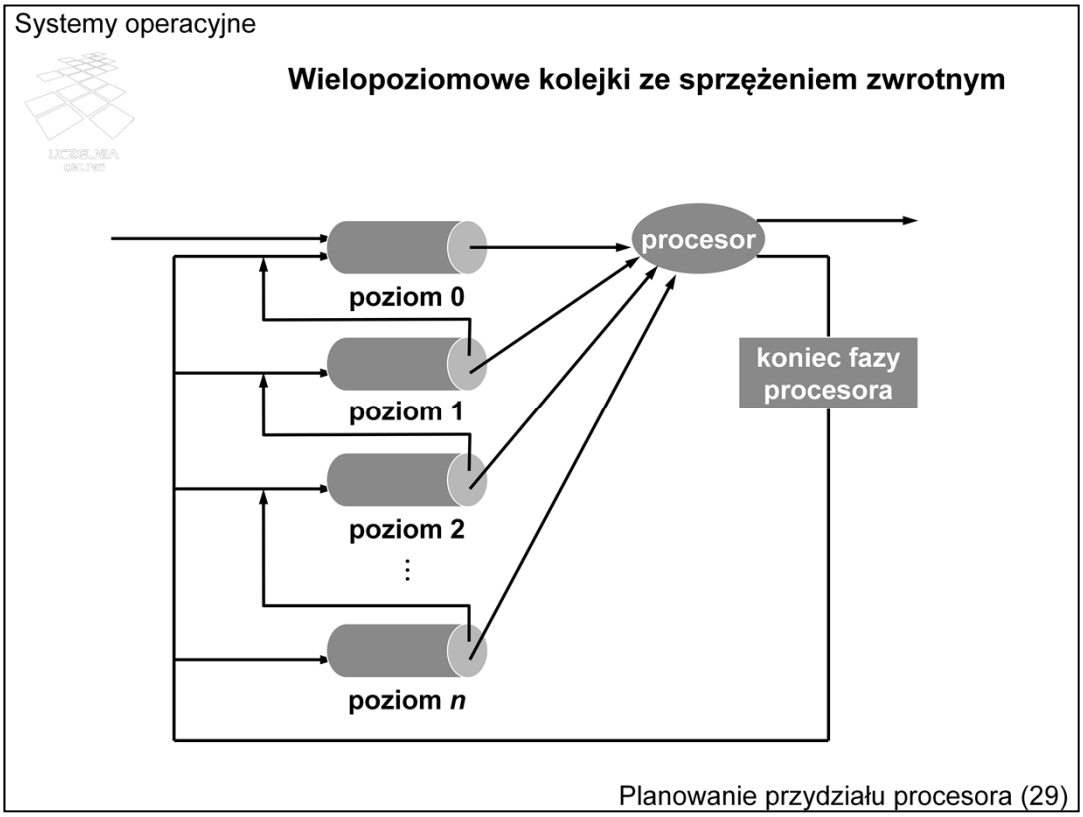
Właściwym algorytmem szeregowania byłby tu SJF lub SRT, który promuje procesy z krótką fazą procesora. Konieczność estymowania czasu obsługi jest jednak dość kosztowna, a identyfikacja procesów ograniczonych wejściem-wyjściem możliwa jest również na podstawie innych przesłanek. W najprostszym przypadku każdy proces przechodzący ze stanu oczekiwania do stan gotowości można potraktować jako proces ograniczony wejściem-wyjściem. Takie podejścia stwarzają jednak ryzyko głodzenie procesów ograniczonych procesorem.

Algorytm FCFS nie uwzględnia oczywiście potrzeb procesów ograniczonych wejściemwyjściem, co może prowadzić do niezrównoważenia obciążenia. Algorytm RR, który jest sprawiedliwy dla procesów ograniczonych procesorem, gdyż daje preferencje dla zadań krótkich, nie jest jednak sprawiedliwy dla procesów ograniczonych wejściemwyjściem. Procesy te muszą rywalizować o kolejny kwantu czasu na równy zasadach z procesami ograniczonymi procesorem, chociaż w większości sytuacji nie wykorzystują tego kwantu do końca.

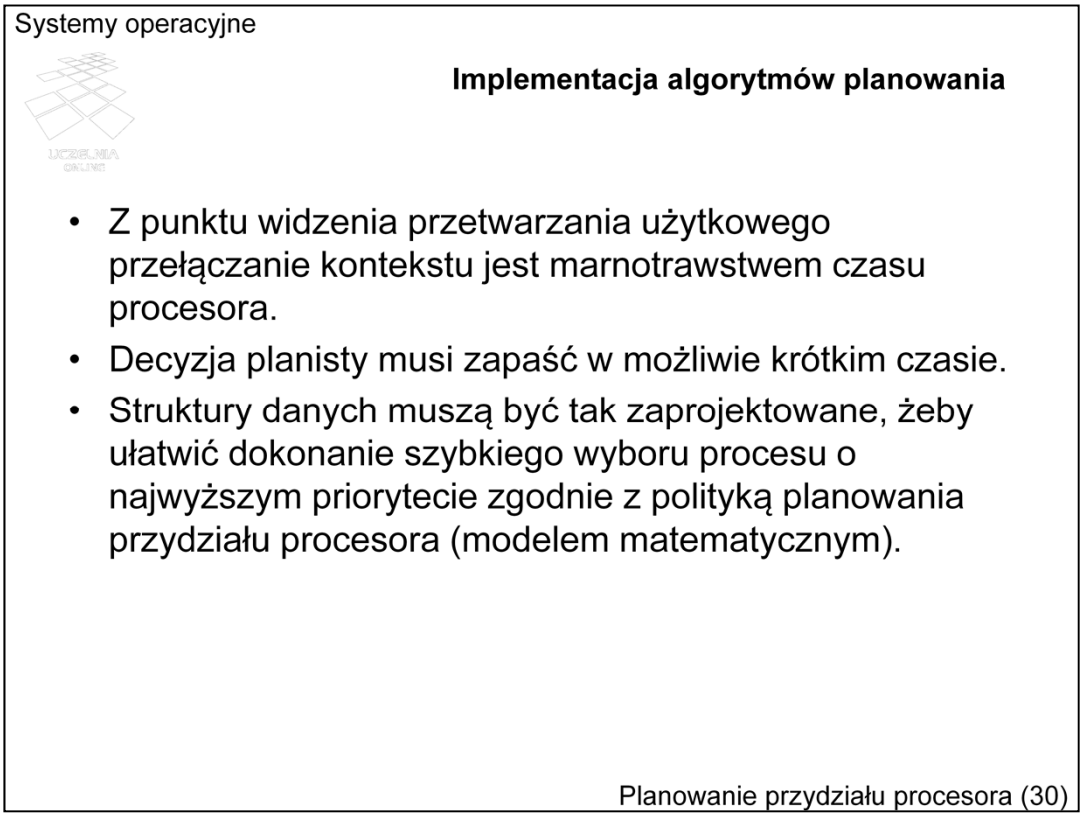


Rozwiązanie problemu procesów ograniczonych wejściemwyjściem można uzyskać przez zwiększenie preferencji dla procesów, które wchodzą w stan gotowości po zakończeniu oczekiwania na urządzenie zewnętrzne. Można w tym celu zastosować planowanie dwukolejkowe z obsługą obu kolejek zgodnie z algorytmem rotacyjnym. Jedna z tych kolejek — pomocnicza — przeznaczona jest na procesy gotowe po zakończeniu operacji wejściawyjścia, a druga — główna — na procesy, które wykorzystały kwant czasu lub z innych powodów oddały procesor. Kolejkę pomocniczą obsługiwać należy oczywiście w pierwszej kolejności. Taka bezwzględna preferencja dla jednej grupy procesów mogłaby spowodować głodzenie drugiej grupy. W tym przypadku procesy z kolejki głównej mogłyby nigdy nie dostać procesora. Procesy z kolejki pomocniczej otrzymują jednak do dyspozycji tylko tę część kwantu czasu, której nie wykorzystały w wyniku zażądania operacji wejściawyjścia. Jeśli zatem proces po wykorzystaniu połowy kwantu czasu zażądał operacji wejściawyjścia, po zakończeniu tej operacji trafi do kolejki pomocniczej z drugą połową kwantu czasu do dyspozycji. W ten sposób każdy proces, nawet jeśli wielokrotnie zażąda operacji wejściawyjścia w ramach jednego kwantu, ostatecznie wykorzysta swój kwant czasu i trafi na koniec kolejki głównej.

Zaprezentowane podejście jest przykładem wykorzystania kolejki dwupoziomowej ze sprzężeniem zwrotnym. Podobny efekt można uzyskać różnicując względne priorytety procesów, co jednak najczęściej i tak implementowane jest za pomocą kolejek wielopoziomowych ze sprzężeniem zwrotnym.



Wielopoziomowa kolejka ze sprzężeniem zwrotnym polega na tym, że tak jak w systemie wielokolejkowym, obsługiwanych jest wiele kolejek, ale procesy nie są na stałe związane z konkretną kolejką, tylko mogą się przemieszczać pomiędzy nimi. Typowy scenariusz zastosowania wielopoziomowych kolejek polega na wybieraniu procesów z najwyższej niepustej kolejki, wykonywaniu przez określony czas (zależny od kwantu i zdarzeń w systemie), a po uzyskaniu stanu gotowości umieszczaniu w któreś z kolejek, w szczególności tej samej. Każda kolejka może być inaczej obsługiwana, np. z innym kwantem czasu. W celu uniknięcia głodzenie procesów na niższych poziomach, mogą one być przenoszone po pewnym czasie na wyższy poziom.



Implementacja funkcji priorytetu zgodnie z matematycznym modelem wymaga odpowiednio częstego przeliczania priorytetu procesów. Takie rozwiązanie nawet w przypadku prostych obliczeniowo funkcji wymaga czasu procesora. Istotne jest zatem zastosowanie odpowiednich struktur danych w implementacji kolejki procesów gotowych, żeby wybór przyspieszyć. Z drugiej strony struktury te nie mogą być zbyt kosztowne w utrzymaniu, gdyż niweczy to zysk czasowy, wynikający z możliwości dokonania szybkiego wyboru. Przykładem może być utrzymywanie kolejki procesów gotowych, posortowanej wg. priorytetów. Wybór procesu do wykonania jest natychmiastowy — jest to proces z czoła kolejki. Wstawienie procesu do kolejki jest jednak operacją czasochłonną, chociaż algorytmicznie niezbyt złożoną.

Systemy operacyjne

Implementacja algorytmu FCFS

- Struktura danych dla kolejki procesów gotowych → kolejka FIFO
- Umieszczenie procesu w kolejce procesów gotowych → dopisanie procesu na końcu kolejki FIFO
- Wybór procesu do wykonania → pobranie procesu z czoła kolejki FIFO
- Czy taki algorytm realizuje dokładnie założenia modelu matematycznego?

Planowanie przydziału procesora (31)

Implementację algorytmu FCFS można oprzeć o kolejkę FIFO. Aktualizacja takiej kolejki jest prostą operacją, nie wymagającą czasochłonnych obliczeń. Kolejka dostarcza też natychmiastowo informację o kolejnym procesie do przydziału procesora.

Taka implementacja nie jest jednak dokładną realizacją modelu matematycznego, przedstawionego wcześniej, albo model matematyczny jest nieadekwatny do przedstawionej implementacji. Po wejściu w stan oczekiwania, a następnie ponownym uzyskaniu gotowości proces umieszczany jest na końcu kolejki. Jest więc traktowany tak, jak gdyby dopiero został przyjęty do systemu, podczas gdy model matematyczny definiuje funkcję priorytetu jako czas przebywania w systemie od momentu przyjęcia.

Systemy operacyjne

Kolejki priorytetowe

- Kolejka priorytetowa jest wielopoziomową kolejką ze sprzężeniem zwrotnym, w której każdy poziom odpowiada pewnej wartości priorytetu lub pewnemu zakresowi wartości.
- Umieszczenie procesu w kolejce priorytetowej sprowadza się do wyznaczenia pozycji odpowiedniej dla priorytetu procesu, a następnie umieszczeniu procesu na końcu kolejki na tej pozycji.
- Wybór procesu o najwyższym priorytecie sprowadza się do zlokalizowania pierwszej niepustej w kolejności malejących priorytetów i wybrania pierwszego procesu z tej kolejki.

Planowanie przydziału procesora (32)

Kolejka priorytetowa jest powszechnie używaną strukturą w implementacji kolejki procesów gotowych w przypadku planowania uwzględniającego priorytety procesów. Jest ona łatwa w aktualizacji, gdyż priorytet jest najczęściej indeksem kolejki odpowiedniego poziomu. Niekiedy liczba poziomów priorytetu jest większa niż liczba poziomów kolejek, wówczas każda kolejka odpowiada pewnemu zakresowi priorytetów. Odwzorowanie priorytetu na poziom kolejki wymaga wówczas prostej operacji arytmetycznej np. dzielenia lub przesunięcia bitów w prawo.

Umieszczanie procesu w kolejce po jej zlokalizowaniu jest operacją wymagającą odpowiedniego powiązania deskryptorów procesów, co sprowadza się do operacji podstawienia kilku wskaźników.

Zlokalizowanie procesu gotowego o najwyższym priorytecie wymaga wyszukania odpowiedniej kolejki. Przyspieszenie tej operacji możliwe jest z wykorzystaniem wektora bitowego, w którym każdy bit odpowiada jednej kolejce, a jego wartość (0 lub 1) wskazuje, czy kolejka jest pusty, czy nie. Przy odpowiednim wsparciu w rozkazach procesora ustalenie indeksu z pierwszą kolejką niepustą sprowadza się do wykonania jednego lub kilku rozkazów maszynowych, zależnie od długości wektora.