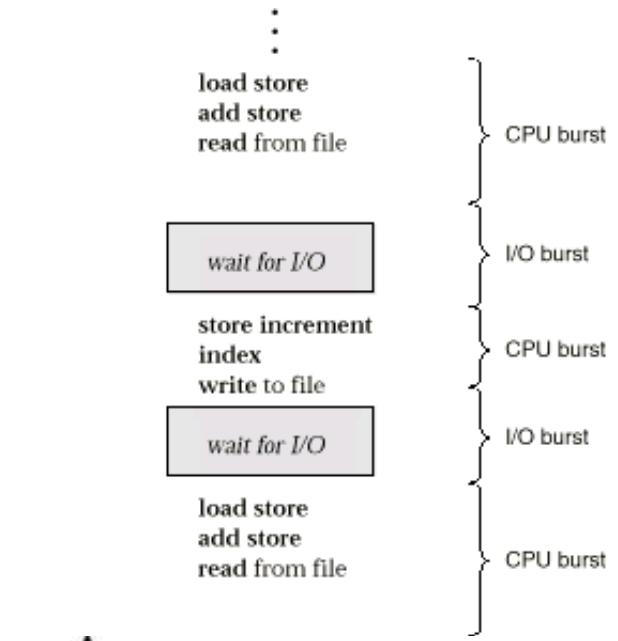
Planowanie przydziału procesora

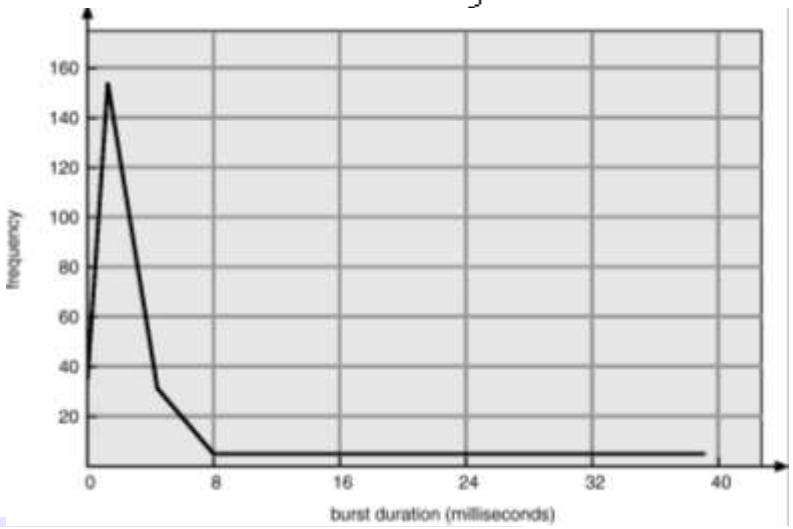
- Pojęcia podstawowe
- Kryteria planowania
- Algorytmy planowania
- Przykładowe rozwiązania

Data ostatniej modyfikacji: 24.11.2019

Pojęcia podstawowe

- Wieloprogramowanie zapewnia maksymalne wykorzystanie czasu procesora, przez stałe utrzymywanie w działaniu pewnej liczby procesów.
- Wykonanie procesu składa się z następujących po sobie cykli. W każdym cyklu jest **faza działania procesora** (*CPU burst*) i **faza oczekiwania –** zazwyczaj na urządzenie wejścia/wyjścia (*I/O burst*).
- Przy dobieraniu właściwego algorytmu planowania przydziału procesora duże znaczenie ma rozkład długości fazy procesora dla wykonywanych zadań.





Planista przydziału procesora

- **Planista przydziału procesora** (krótkoterminowy) wybiera spośród gotowych procesów w pamięci operacyjnej jeden i przydziela mu procesor.
- Decyzje o przydziale procesora mogą zapadać kiedy proces:
  1. przeszedł od stanu aktywności do stanu czekania
  2. przeszedł od stanu aktywności do stanu gotowości.
  3. przeszedł od stanu czekania do stanu gotowości
  4. kończy działanie.
- W sytuacjach 1 i 4 planowanie (szeregowanie) procesów jest **niewywłaszczeniowe** (*nonpreemptive*), w 2 i 3 - **wywłaszczeniowe** (*preemptive*).

Ekspedytor

- **Ekspedytor** (*dispatcher*) jest modułem jądra, który faktycznie przekazuje procesor do dyspozycji procesu wybranego przez planistę krótkoterminowego. Obowiązki ekspedytora:
  - przełączanie kontekstu
  - przełączanie do trybu użytkownika
  - skok do odpowiedniej komórki programu użytkownika dla wznowienia wykonywania programu.
- **Opóźnienie ekspedycji** (*dispatch latency*) – czas zużywany przez ekspedytora na wstrzymanie jednego i uaktywnienie innego procesu.

Kryteria planowania

Punkt widzenia systemu

- **Wykorzystanie czasu procesora** (typowo 40% do 90%)
- **Przepustowość** (*throughput*) – liczba zakończonych procesów w jednostce czasu
- **Sprawiedliwość** (*fairness*) – równe traktowanie procesów
- Respektowanie zewnętrznych priorytetów procesów
- **Równoważenie obciążenia** wykorzystywanych zasobów

Punkt widzenie użytkownika

- **Czas cyklu przetwarzania** (*turnaround time*) – czas potrzebny na wykonanie.
- **Czas oczekiwania** – suma okresów, w których proces czeka w kolejce procesów gotowych do wykonania
- **Czas odpowiedzi** (*response time*) – czas upływający pomiędzy przedłożeniem zamówienia, a pojawieniem się **pierwszej** reakcji systemu.
- **Przewidywalność** – realizacja zadania w określonym czasie bez względu na obciążenie systemu

Kryteria optymalizacji

- Max wykorzystanie procesora
- Max przepustowość
- Min czas cyklu przetwarzania
- Min czas oczekiwania
- Min czas odpowiedzi

Planowanie za pomocą zwykłej kolejki

Nazwy metody: **FCFS** (First-Come, First-Served), **FIFO** (First-In, First-Out)

- Przykład:

Proces	Dł. fazy procesora
P_1	24
P_2	3
P_3	3

- Porządek nadchodzenia: P_1 , P_2 , P_3
- **Diagram Gantta** dla FCFS:

- Czasy oczekiwania: P_1=0; P_2=24; P_3 =27
- **Średni czas oczekiwania**: (0+24+27)/3=**17**, **maks. czas oczekiwania**= **27**

Planowanie metodą FCFS (c.d.)

Inny porządek procesów: P_2 , P_3 , P_1 .

- **Diagram Gantta**:

- Czasy oczekiwania: P_1=6; P_2=0, P_3=3
- **Śr. czas oczekiwania**: (6+0+3)/3 = **3**, **maks. czas oczekiwania**= **6**
- Dużo lepiej niż poprzednio.
- **Efekt konwoju**: krótkie procesy czekają długo na zwolnienie procesora przez wielki proces.

Planowanie metodą „Shortest-Job-First” (SJF/SRTF)

- **Algorytm SJF** wiąże z procesem długość następnej fazy procesora - wybierając do wykonania proces o najkrótszej następnej fazie procesora

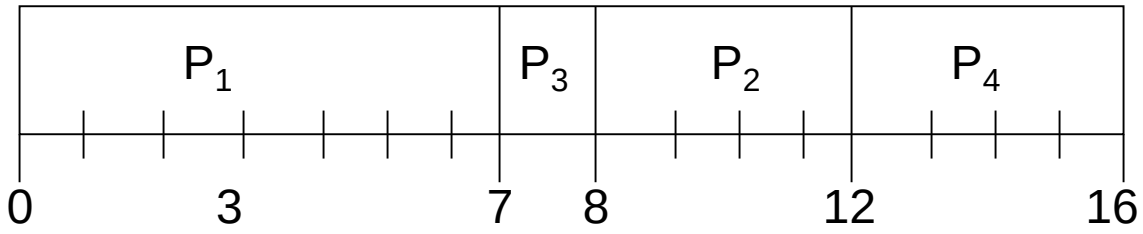
- Dwa typy algorytmu SJF:
  - **niewywłaszczający** (proces wykonujący się zawsze kończy swoją fazę procesora)
  - **wywłaszczający** (*Shortest-Remaining-Time-First* (**SRTF**)).

- Algorytm SJF jest **optymalny ze względu na średni czas oczekiwania** (dla danego zbioru procesów).

Niewywłaszczające planowanie SJF

Proces	Przybył.	Dł. fazy proc.
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **SJF** (niewywłaszczający)

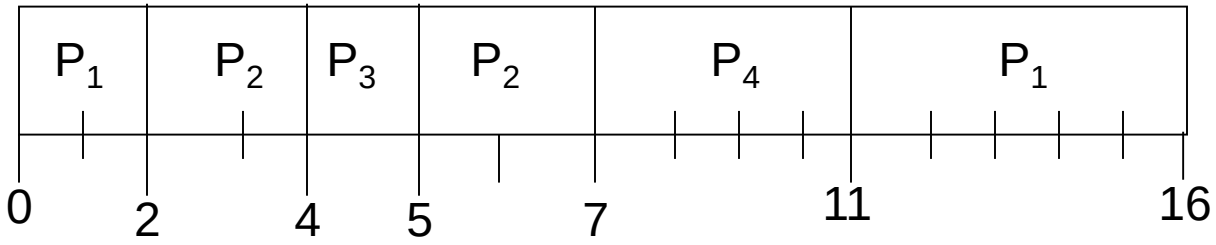


- **Śr. czas oczekiwania**=(0+6+3+7)/4 = **4**, **maks. czas oczekiwania**= 7

Wywłaszczające planowanie SJF: SRTF

Proces	Przybył	Dł. fazy proc.
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF wywłaszczający, czyli **SRTF**



- **Śr. czas oczekiwania**=(9+1+0+2)/4= **3**, **maks. czas oczekiwania**= **9**

Estymacja długości następnej fazy procesora

- Nie można *bezbłędnie przewidzieć*, ale można **estymować** długość *następnej* fazy.

- Estymacja zakłada, że długość następnej fazy będzie podobna do długości faz poprzednich. Dla średniej wykładniczej (*exponential averaging*):

$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\, \tau_n$$

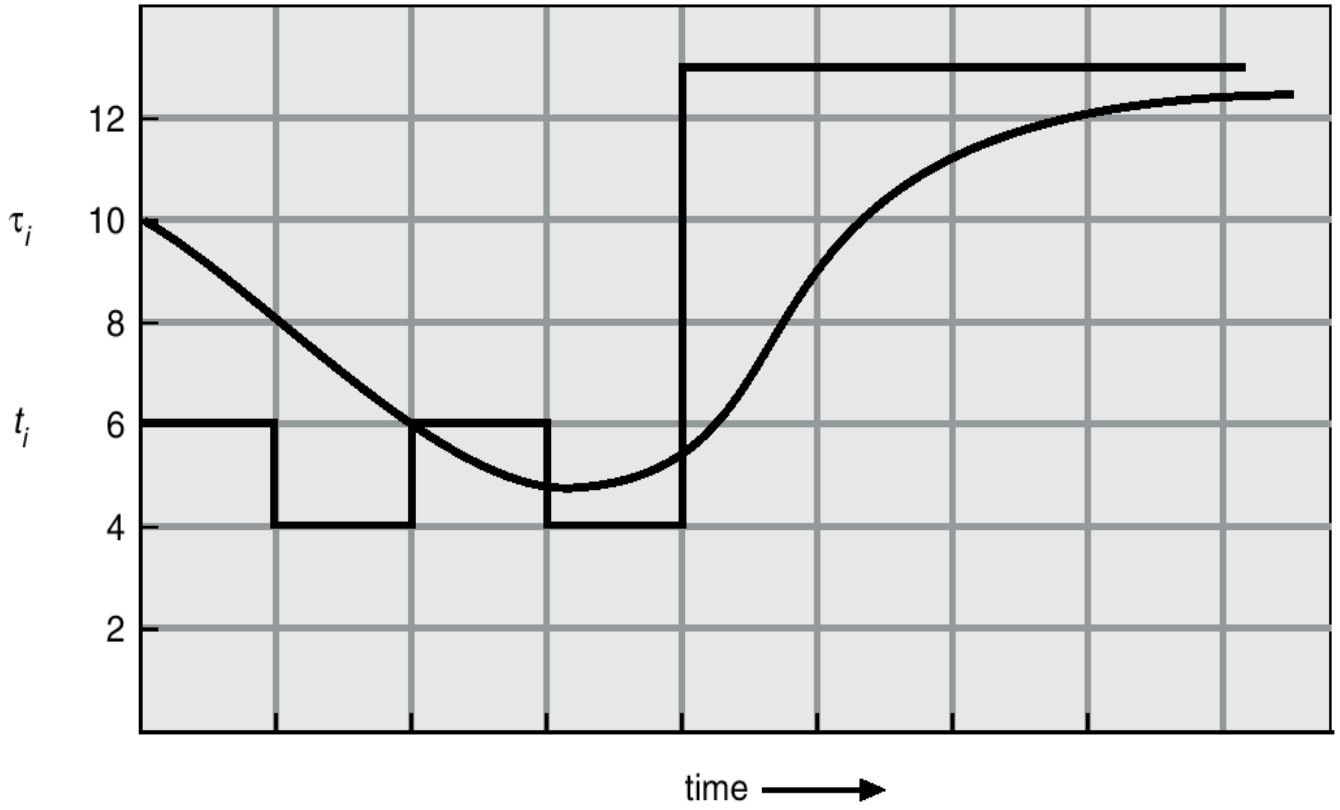
gdzie

t_n = rzeczywista długość n-tej fazy procesora

τ_{n+1} = przewidywana długość następnej fazy procesora

α = współczynnik z przedziału (0, 1)

Estymacja długości następnej fazy procesora - c.d.



CPU burst (t_i)	6	4	6	4	13	13	13	. . .
"guess" (τ_i)	10	8	6	6	5	9	11	12	. . .

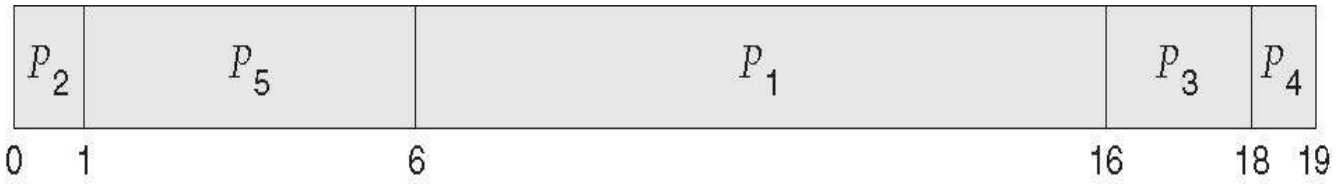
Planowanie priorytetowe

- Każdy proces ma przypisany **priorytet** (liczba całkowita).
- Procesor jest przydzielany procesowi o **najwyższym priorytecie** (najmniejsza liczba = najwyższy priorytet) z wywłaszczaniem lub bez wywłaszczania.
- SJF jest planowaniem priorytetowym (priorytet=estymowany czas następnej fazy procesora).
- Problem: **głodzenie** (nieskończone blokowanie) - niektóre procesy mogą nigdy się nie wykonać.
- Rozwiązanie: **postarzanie** (oczekiwanie zwiększa priorytet procesu) .

Ilustracja planowania priorytetowego

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- DiagramGantta:



- **Śr. czas oczekiwania= 8.2, maks. czas oczekiwania= 18**

Planowanie rotacyjne (RR)

- Każdy proces ma dostęp do procesora przez pewien **kwant czasu *q*** (rzędu 10-100ms), po czym (jeśli wcześniej nie zacznie sam oczekiwać) jest wywłaszczany i wstawiany na koniec kolejki procesów oczekujących.

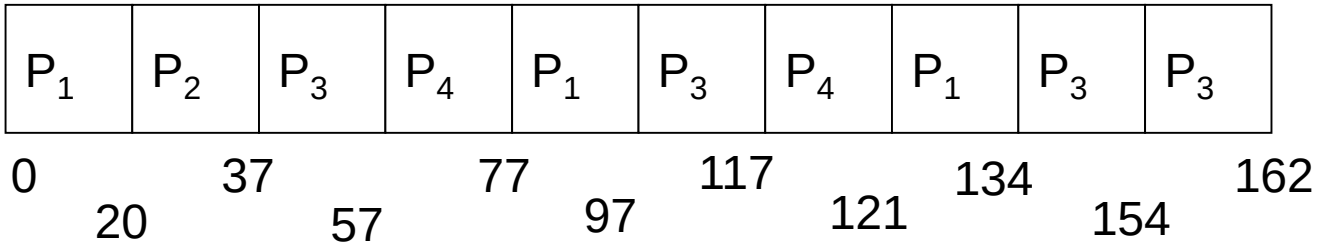
- Dla ***n*** procesów gotowych każdy proces otrzymuje średnio ***1/n*** czasu w odcinkach ***q*** jednostek czasu. Żaden proces nie czeka dłużej niż ***(n-1)q*** jednostek czasu.

- Wydajność
  - ***q*** duże $\Rightarrow$ FIFO
  - ***q*** małe $\Rightarrow$ problem: ***q*** musi być dostatecznie duże, aby narzut na przełączanie nie dominował.

Planowanie rotacyjne z kwantem q=20

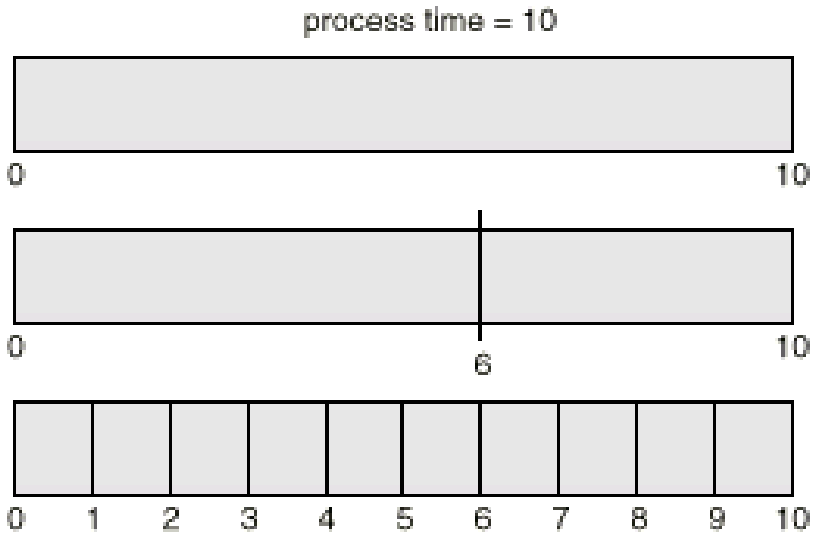
Proces	Dł. fazy proc.
P_1	53
P_2	17
P_3	68
P_4	24

- DiagramGantta:



- Typowo - **dłuższy czas cyklu przetwarzania** niż dla SJF, ale **krótszy czas odpowiedzi**.

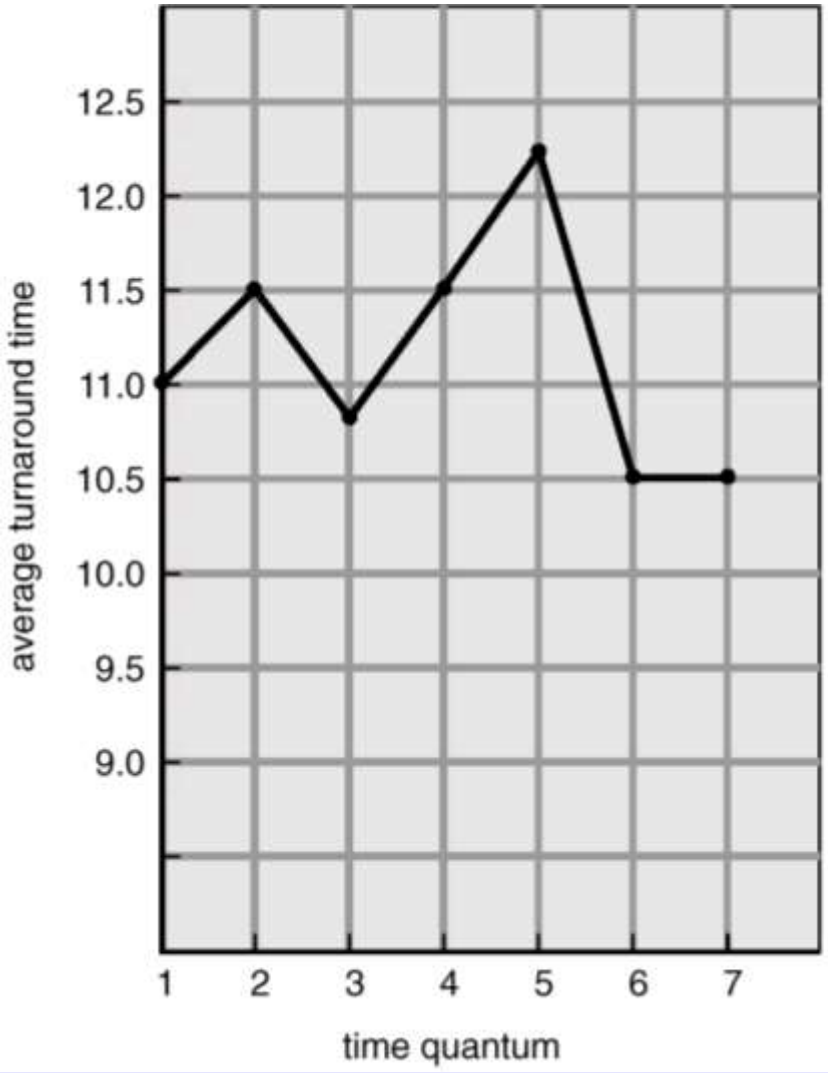
Skutki krótkiego kwantu czasu



quantum	context switches
12	0
6	1
1	9

- Z punktu widzenia przetwarzania użytkowego przełączanie kontekstu jest marnotrawstwem czasu procesora → nie może być zbyt częste
- Decyzja planisty musi zapaść w możliwie krótkim czasie → struktury danych muszą być tak zaprojektowane, żeby ułatwić dokonanie szybkiego wyboru procesu do wykonania.

Średni czas przetwarzania a długość kwantu czasu



process	time
P_1	6
P_2	3
P_3	1
P_4	7

80% faz procesora powinno być krótszych od kwantu **q**

Wielopoziomowe kolejki

Osobną klasę algorytmów planowania opracowano dla sytuacji, w których procesy można zaklasyfikować do grup o wyraźnie różnych cechach.

Przykładowo:

- Kolejka procesów gotowych jest podzielona na dwie oddzielne:
  - **kolejka procesów pierwszoplanowych** (*interakcyjnych*, foreground)
  - **kolejka procesów drugoplanowych** (*wsadowych*, *background*)

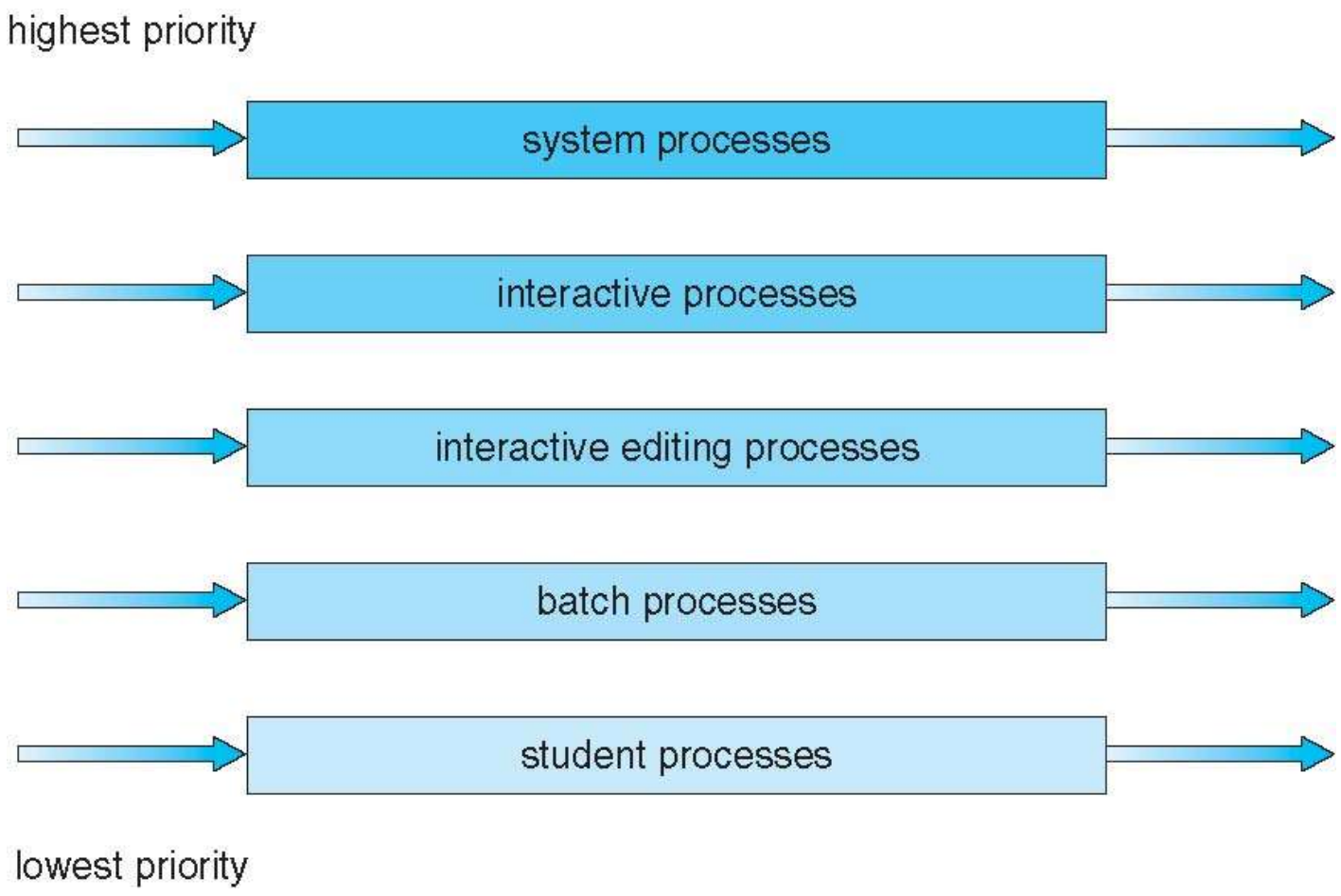
- Każda kolejka ma **swój algorytm planowania**, np.
  - dla **procesów pierwszoplanowych** (*foreground*) – **RR**
  - dla **procesów drugoplanowych** (*background*) – **FCFS**

Wielopoziomowe kolejki (c.d.)

Dla dwóch kolejek procesów oczekujących na czas procesora trzeba zdecydować o **rozdziale czasu procesora pomiędzy kolejki**. Warianty rozdziału:

- **Ustalony porządek** - najpierw wykonanie wszystkich procesów pierwszoplanowych (możliwe głodzenie procesów drugoplanowych).
- **Przydział kwantu czasu dla kolejki**, n.p.
  80% dla pierwszego planu (RR), 20% dla drugiego planu (FCFS)

Wielopoziomowe kolejki (c.d.)



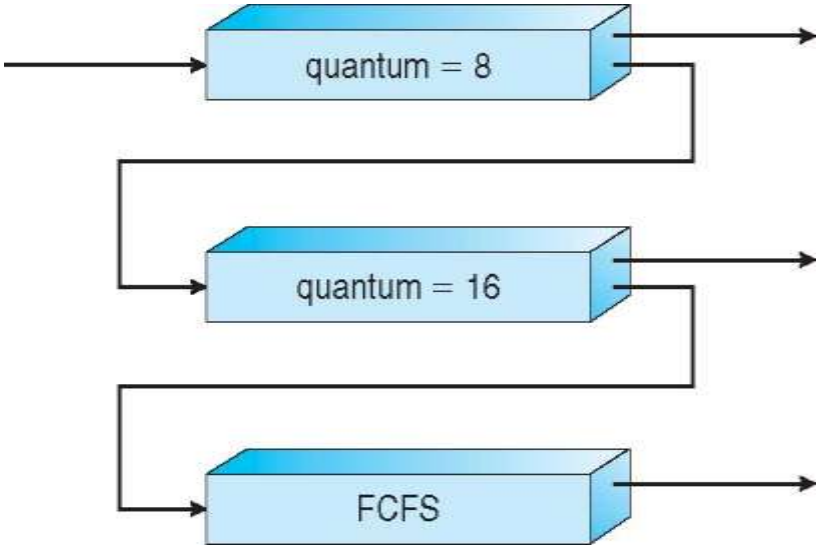
Wielopoziomowe kolejki ze sprzężeniem zwrotnym

- W poprzednio omawianych algorytmach planowania wielopoziomowego przydział procesu do jednej z kolejek jest statyczny.
- W planowaniu wielopoziomowym ze sprzężeniem zwrotnym proces może się przemieszczać pomiędzy kolejkami - stosownie od aktualnej długości faz procesora. Procesy ograniczone przez wyjście-wyjście mają tendencję do przebywania w kolejkach o dużych priorytetach. Z kolei długie przebywanie w kolejce o niskim priorytecie może przemieścić proces do kolejki o wyższym priorytecie (postarzanie z przemieszczaniem zapobiega głodzeniu).
- Parametry planisty wielopoziomowych kolejek ze sprzężeniem zwrotnym:
  - liczba kolejek
  - algorytm planowania dla każdej kolejki
  - metoda awansowania procesu do kolejki o wyższym priorytecie
  - metoda dymisjonowania procesu do kolejki o niższym priorytecie
  - metoda wyznaczania kolejki, do której trafia proces potrzebujący obsługi

Wielopoziomowe kolejki ze sprzężeniem zwrotnym

Przykład

- Są trzy kolejki procesów gotowych:
  - Q_0 – kwant czasu8 ms
  - Q_1 – kwant czasu 16 ms
  - Q_2 – FCFS



- Planowanie
  - Nowe zadanie wchodzi do Q_0 (na koniec kolejki). Jeśli zadanie nie skończy się w ciągu przydzielonych 8 ms przechodzi na koniec kolejki Q_1.
  - Po opróżnieniu Q_0 procesy w Q_1 otrzymują kwant czasu 16 ms. Jeśli zadanie z Q_1 nadal się nie kończy – jest wywłaszczane i przechodzi na koniec kolejki Q_2.
  - Procesy w Q_2 są wykonywane w porządku FCFS i tylko wtedy, gdy kolejki Q_0 i Q_1 są puste.

Planowanie w MS Windows

- Ekspedytor korzysta z 32 poziomów priorytetów . Klasy priorytetów
  - czasu rzeczywistego 16 do 32.
  - zmienna (*variable class*) - priorytety od 0 do 15.
- Z każdym priorytetem planowania ekspedytor wiąże kolejkę i przebiega zbiór kolejek (od 32 do 0) aż znajdzie wątek gotowy (specjalny *wątek postojowy* jest zawsze gotowy)
- Wątek wybrany przez dyspozytora będzie działać aż do: wywłaszczenia przez wątek o wyższym priorytecie, wyczerpania kwantu czasu lub użycia w nim blokującego wywołania systemowego.
- Jeśli kwant czasu się wyczerpie, wątek zostaje przerwany i, jeśli należał do klasy zmiennej, to mu się zmniejsza priorytet (nigdy poniżej wartości podstawowej)
- Przejście wątku ze stanu oczekiwania do gotowości powoduje wzrost priorytetu (większy dla operacji interaktywnych, dla aktywnego okna: 3x).
- Dla klasy NORMAL_PRIORITY_CLASS rozróżniany jest proces pierwszoplanowy, aktualnie wybrany na ekranie (ma zwiększony kwant czasu)
- W systemie Windows 7 dodano **user-mode scheduling (UMS)**:
  - Programy użytkownika mogą zarządzać wątkami niezależnie od jądra
  - Dla dużej liczby wątków uzyskuje się większą efektywność.

POSIX - planowanie przydziału procesora

- Przydział czasu procesora procesowi lub wątkowi zależy od **klasy szeregowania** (*scheduling policy*) oraz **priorytetu** tego procesu/wątku Definicja polityki określa minimalny zakres priorytetów. Zakresy priorytetów różnych klas mogą nie być rozłączne. Cztery klasy szeregowania są zdefiniowane przez standard POSIX (inne mogą być realizowane przez implementacje):
  - Dla procesów/wątków czasu rzeczywistego (RT) : SCHED_RR i SCHED_FIFO, procesy sporadycznego serwera: SCHED_SPORADIC
  - Domyślna : SCHED_OTHER

- Planista w systemie Linux – komponent jądra, określające któremu, z **gotowych do wykonania wątków** (*runnable thread)*:, zostanie przydzielony procesor. Planowanie jest **wywłaszczające**.
- Klasy szeregowania w systemie Linux:
  - Dla zwykłych wątków (normal scheduling policies): SCHED_OTHER, SCHED_BATCH, SCHED_IDLE
  - Dla wątków czasu rzeczywistego (real-time scheduling policies): SCHED_FIFO, SCHED_RR, SCHED_DEADLINE (in place of SCHED_SPORADIC)

Więcej szczegółów: w następnym semestrze