1. Szeregowanie w systemach czasu rzeczywistego

1.1 Definicje

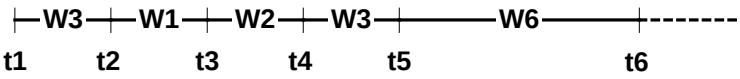
Zadanie - proces lub wątek

Gotowych do wykonania zadań jest zwykle dużo więcej niż mogących je wykonać procesorów. Zatem nieustannie należy rozstrzygać który z gotowych zadań ma otrzymać procesor.

Rozstrzyganie które z zadań gotowych ma być teraz wykonywany, realizowane jest przez procedurę szeregującą.

Procedura szeregująca (*ang. scheduler, dispatcher*)

Procedura szeregująca – część systemu operacyjnego, wykonująca funkcję wybierania ze zbioru zadań gotowych, zadania które ma być teraz wykonywany.



Rys. 0-1 Działanie funkcji szeregującej

Kryteria optymalizacji procedury szeregującej:
- czas reakcji na zdarzenie,
- przepustowość,
- wydajność,
- wykorzystanie zasobów

☞ Szeregowanie w systemach czasu rzeczywistego powinno zapewniać gwarantowany czas reakcji na zdarzenie, terminowość i przewidywalność.

Decyzje szeregujące podejmowane są na bieżąco na skutek określonych zdarzeń i uwzględniają atrybuty wszystkich gotowych do wykonania zadań .

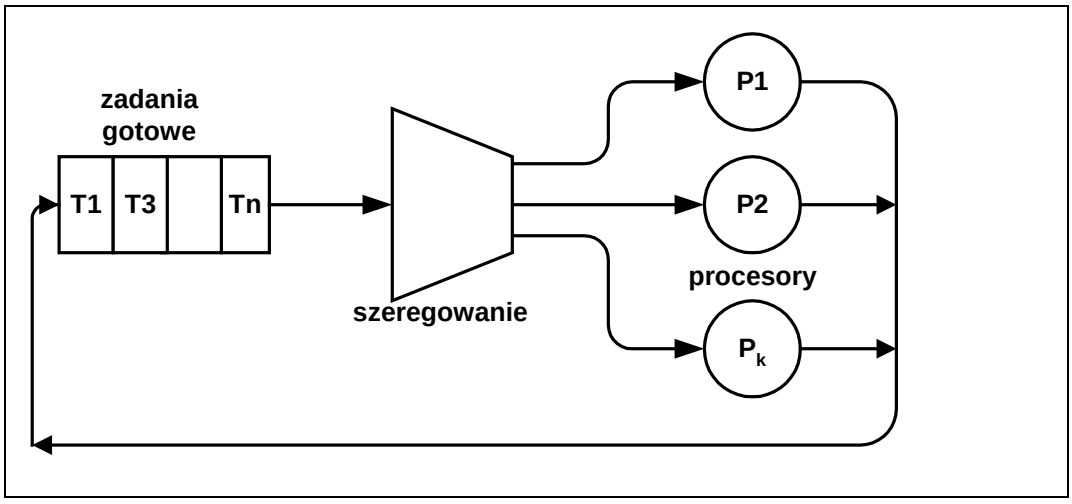
Atrybuty: priorytetzadania , stanzadania, strategia szeregowania, dotychczasowe zużycie zasobów.

Ze względu na sposób podejmowania decyzji wyróżniamy dwa rodzaje szeregowania:

- Szeregowanie statyczne (ang. *pre-run-time scheduling)*
- Szeregowanie dynamiczne (ang. *run-time scheduling)*

Szeregowanie statyczne – plan przydziału zadań do zasobów sporządzany jest z góry. Warunkiem stosowalności metody jest aprioryczna znajomość zadań wraz z ich charakterystykami czasowymi

Szeregowanie dynamiczne – plan przydziału zadań do zasobów sporządzany jest na bieżąco. Podstawą działania szeregowania dynamicznego są priorytety. Metoda może być stosowana, gdy charakterystyki czasowe zadań nie są z góry znane.



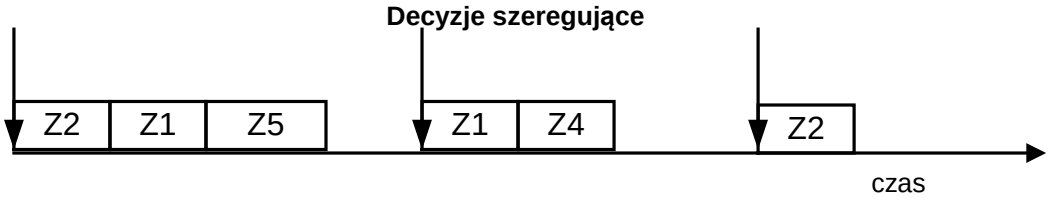
Zasady przydziału zadań do procesorów:

- Do każdego z procesorów w danym momencie przydzielony jest, co najwyżej jedno zadanie.
- Każde zadanie w danym momencie przydzielone jest do dokładnie jednego procesora.

Podział ze względu na przyczynę podejmowania decyzji szeregujących:

- Szeregowanie wymuszane czasem (*ang. clock driven scheduling*)
- Szeregowanie wymuszane zdarzeniami(*ang. event driven scheduling*)

Szeregowanie wymuszane czasem
Decyzje szeregujące podejmowane są w specyficznych często ustalonych z góry momentach czasu.



Gdy parametry zadań są znane apriori plan szeregowania można ułożyć w postaci tabeli – szeregowanie cykliczne.

Szeregowanie wymuszane zdarzeniami
Decyzje szeregujące podejmowane są gdy, zachodzą określone zdarzenia:

- Zadanie staje się gotowe
- Procesor jest samoistnie zwalniany przez bieżące zadanie wywołanie blokujące)
- Zmienia się priorytet jakiegoś zadania

Priorytet zadania(*ang. task priority*)

Priorytet zadania jest miarą pilności danego zadania względem innych zadań wykonywanych na tym samym komputerze.

Procedura szeregująca może być uaktywniona gdy

1. Wystąpiło przerwanie sprzętowe.
2. Wystąpiło przerwanie wewnętrzne (wyjątek).
3. Proces bieżący wykonał wywołanie systemowe.

Szeregowanie jest powoływane także gdy zmienia się stan zadania bieżącego.

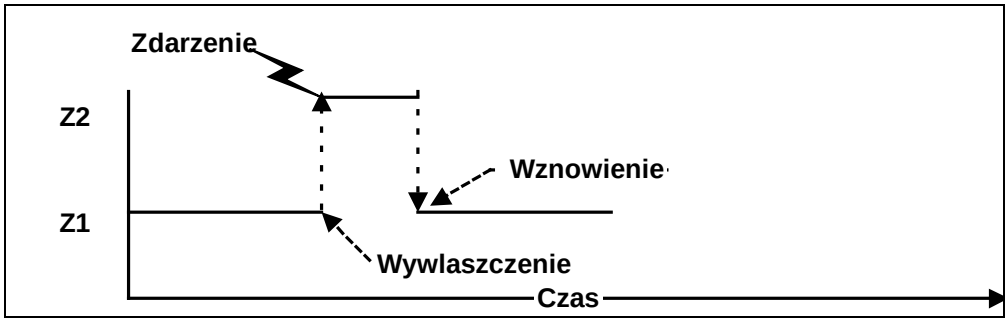
1. Zadanie bieżące blokuje się.
2. Zadanie bieżące jest wywłaszczane.
3. Zadanie bieżące samoistnie zwalnia procesor.

Podział ze względu na wywłaszczalność

- Szeregowanie wywłaszczające
- Szeregowanie kooperacyjne

Wywłaszczanie

Gdy wykonywane zadanie jest przez system zawieszone bo pojawiło się zadanie ważniejsze mówimy że zostało wywłaszczone (ang. *preempted*).



Rysunek 0-1 Zadanie Z2 wywłaszcza zadanie Z1

Szeregowanie wywłaszczające (*ang. preemptive scheduling*)
Zadanie bieżące może, w nie dającym się przewidzieć momencie czasu, utracić procesor na którym wykonywany będzie inne, pilniejsze zadanie.

Szeregowanie wywłaszczające (ang. *preemptive scheduling*)
Decyzje szeregujące podejmowane są gdy:

- Zadanie staje się gotowe
- Procesor jest zwalniany przez bieżące zadanie
- Zmienia się priorytet jakiegoś zadania

Szeregowanie kooperacyjne (*ang. coperative scheduling*)
Szeregowanie kooperacyjne to taki sposób organizacji pracy zadań takie że zadanie bieżące przełączany jest na inne tylko poprzez wykonanie określonych funkcji systemowych.

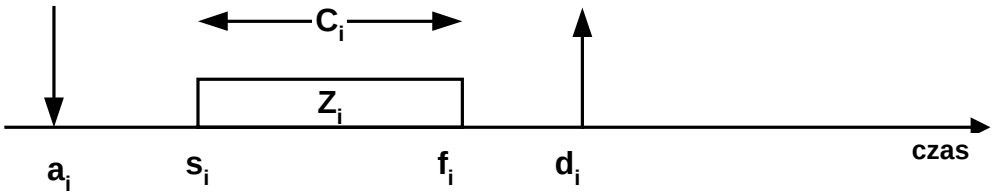
1.2 Charakterystyka zadań

W systemach czasu rzeczywistego wyróżniamy następujące typy zadań:

1. Zadania asynchroniczne (*ang. asynchronous*) – aktywowane przerwaniami.
2. Zadania synchroniczne (*ang. synchronous*) – aktywowane układami odmierzania czasu.
3. Zadania drugoplanowe (*ang. background*) – wykonywane w miarę wolnego czasu procesora.

1.2.1 Procesy i wątki asynchroniczne

Zadanie (asynchroniczne i synchroniczne) Z_i charakteryzuje się następującymi ograniczeniami czasowymi:



Rysunek 0-2 Charakterystyki czasowe zadania

- Czas napłynięcia zadania (ang. *arrival time*) – a_i czas w którym pojawi się zadanie Z_i
- Czas wykonania zadania (ang. *computation time*) - c_i maksymalny czas potrzebny do wykonania zadania w sytuacji gdy wykonuje się na procesorze jako jedyne i ma ono wszystkie potrzebne zasoby.
- Ostateczny termin zakończenia (ang. *deadline time*) - d_i
- Czas rozpoczęcia przetwarzania (ang. *start time*) - s_i
- Czas zakończenia przetwarzania (ang. *finishing time*) - f_i
- Luźny czas zadania (ang. *laxity time*) - $X_i = d_i - a_i - e_i$
- Czas spóźnienia zadania (ang. *lateness time*) - $L_i = f_i - d_i$
- Ograniczenia kolejności (ang. *Precedence Constraints),* specyfikuje że dane zadanie powinno poprzedzać inne.

Elementarną funkcją systemu czasu rzeczywistego jest reagowanie na zdarzenia reprezentowane przez przerwania. Reakcja na zdarzenia może być wykonana na trzy sposoby:

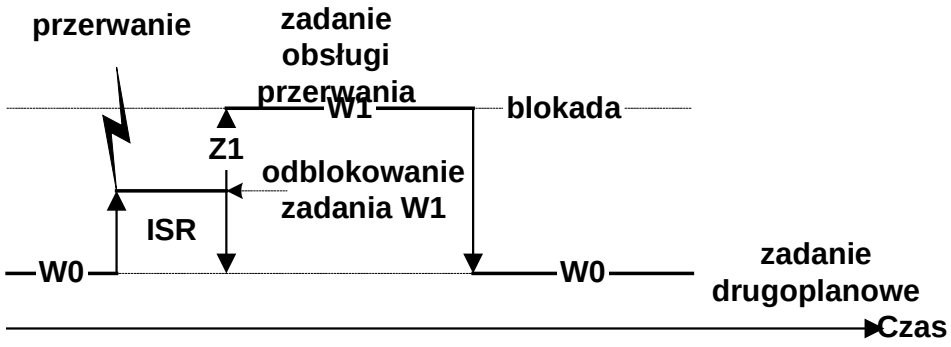
Trzy strategie reagowania na zdarzenie:

1. Wszystkie czynności wykonywane są przez procedurę obsługi przerwania.
2. Wewnątrz procedury obsługi przerwania wykonane będą najważniejsze czynności a resztę pracy wykona odblokowany specjalnie wątek lub proces.
3. Wewnątrz procedurę obsługi przerwania nie są wykonywane żadne czynności a jedyną jego funkcją jest odblokowanie pewnego wątku lub procesu.

W ramach ISR można wykonać tylko niewiele prostych czynności. Powody tego są następujące:

1. Czynności wykonywane w ramach ISR nie podlegają szeregowaniu.
2. W czasie wykonywania ISR przerwania pozostają domyślnie zablokowane.
3. Konstrukcja większości systemów operacyjnych nie dopuszcza aby w ramach ISR wykonywane były wywołania systemowe.

Zasada obsługi przerwań:
W kodzie procedury obsługi przerwania wykonać należy tylko niezbędne czynności. Następnie, gdy jest taka potrzeba, należy powiadomić pewien wątek lub proces o wystąpieniu przerwania. Wątek ten lub proces wykona resztę pracy.



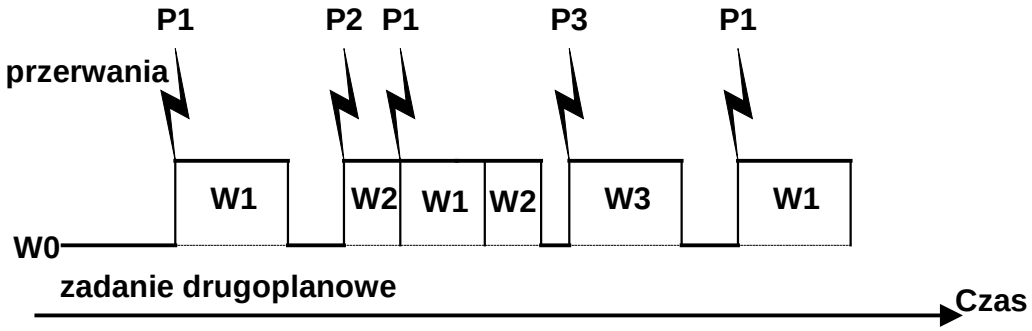
Rys. 2 Procedura obsługi przerwania odblokowuje zadanie

```
do {
    czekaj_na_zdarzenie(Z1);
    obsłuż_zdarzenie;
} while(1);
```

Przykład 1 Kod działania wątku obsługującego zdarzenie Z1

Procesy i wątki asynchroniczne (*ang. asynchronous processes, threads*)

Procesy i wątki asynchroniczne po utworzeniu pozostają w stanie zablokowania a wznawiane są poprzez przerwania reprezentujące zachodzące w systemie zdarzenia. Po obsłużeniu zdarzenia ponownie przechodzą one w stan zablokowania.



Rys. 3 Zadania asynchroniczne

1.3 Procesy i wątki synchroniczne

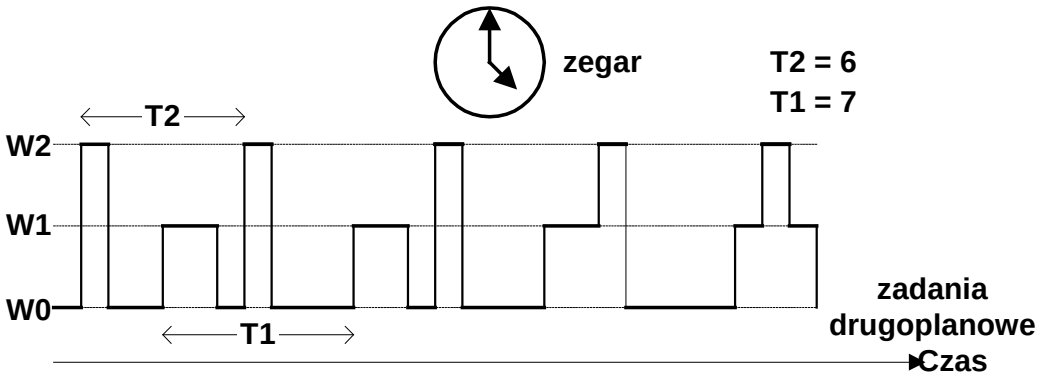
W systemach czasu rzeczywistego wiele zdarzeń synchronizowanych jest upływem czasu.

Przykład: regulator cyfrowy.

- Mierzy cyklicznie wielkość regulowaną,
- oblicza sygnał błędu
- według oblicza wielkość sterującą która podawana jest na układ wykonawczy.

Procesy i wątki synchroniczne (*ang. synchronous processes, threads*)

Procesy i wątki synchroniczne po utworzeniu pozostają w stanie zablokowania a wznawiane są poprzez układy odmierzania czasu. Po wykonaniu zaplanowanych czynności ponownie przechodzą w stan zablokowania lub kończą się.



Rys. 4 Zadania synchroniczne

Ze względu na regularność wyróżnia się dwa rodzaje zadań synchronicznych: periodyczne i aperiodyczne.

- Zadania periodyczne uruchamiane są w stałych odstępach czasowych.
- Zadania aperiodyczne uruchamiane są w nieregularnych odstępach czasowych. Czas kolejnej aktywacji obliczany jest dla nich w trakcie aktywacji bieżącej.

Nie da się wyznaczyć harmonogramu czasu aktywacji zadań aperiodycznych, można jednak wyznaczyć czas kolejnej aktywacji na podstawie informacji znanych w czasie bieżącego uruchomienia.

// Zadania periodyczne	// Zadanie aperiodyczne

zaprogramuj czasomierz
na cykl T i zdarzenie Z;
do {
czekaj_na_zdarzenie(Z);
wykonaj_czynności;
} while(1);
	do {
	wykonaj_czynności;
	i=i+1;
	oblicz Ti;
	czekaj(Ti;)
	} while(1);

Przykład 2 Pseudokod zadań periodycznych i aperiodycznych

1.3.1 Zadania drugoplanowe

W systemach oprócz funkcji kluczowych występują także funkcje co do których nie jest wymagane spełnienie wymagań czasu rzeczywistego.

Zadania drugoplanowe (*ang. background tasks*)

Zadania drugoplanowe realizują zadania co do których nie jest wymagane spełnienie ograniczeń czasowych.

Typowe zadania drugoplanowe to obsługa procesów wyświetlania, drukowania czy wykonywanie procedur diagnostycznych.

Przedłużenie się czasu ich realizacji nie spowoduje istotnego zaburzenia pracy systemu. Wykonują się w miarę wolnego czasu procesora.

	Asynchroniczne	Synchroniczne	Drugoplanowe
Aktywacja	Zdarzenia	Czas	Procedura szeregująca
Wymagany punkt zablokowania	Tak	Tak	Nie
Priorytet	Wysoki	Wysoki, średni	Niski
Funkcja	Obsługa zdarzeń	Czynności periodyczne i aperiodyczne	Czynności pomocnicze

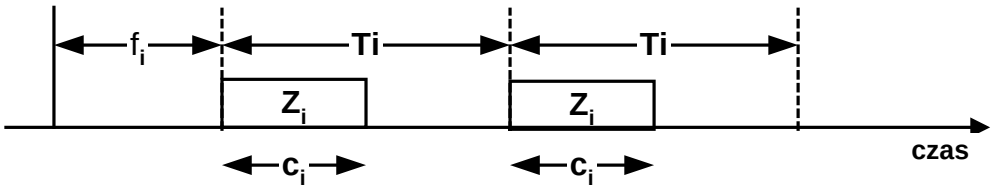
Tabela 1 Porównanie zadań asynchronicznych, synchronicznych i drugoplanowych

1.4 Algorytmy szeregowania zadań cyklicznych

1.4.1 Zadania cykliczne

W systemach czasu rzeczywistego często występują zadania cykliczne. Zadania cykliczne charakteryzują się:

- Okresem (ang. period) – T_i czas co który się pojawiają
- Fazą (ang. *phase*) – f_i czas pierwszego pojawienia się
- Czasem wykonania - c_i



Rysunek 0-3 Zadanie cykliczne

Własności zadań cyklicznych

- Wszystkie zadania są periodyczne ze znanym okresem. Przedział T_i pomiędzy kolejnymi wystąpieniami zadania nazywamy okresem.
- Wszystkie instancje zadania mają niezmienny czas wykonania c_i
- Względny ostateczny termin zakończenia zadania d_i jest dla wszystkich instancji zadania taki sam równy ich okresowi (instancja zadania musi się zakończyć przed pojawieniem się następnej).
- Zadania są od siebie niezależne

Współczynnik wykorzystania procesora: $$U = \sum_{i=1}^{n} \frac{c_i}{T_i}$$

Jeżeli współczynnik wykorzystania procesora jest większy od 1 to dany zbiór zadań nie może być szeregowany przez żaden algorytm.

1.4.2 Wykonywanie cykliczne

Jednym z pierwszych algorytmów szeregowania zadań było wykonywanie cykliczne CE (ang. *Cyclic Executive*)

Procedura szeregująca wywoływana jest w cyklu małym (ang. *minor cycle*).

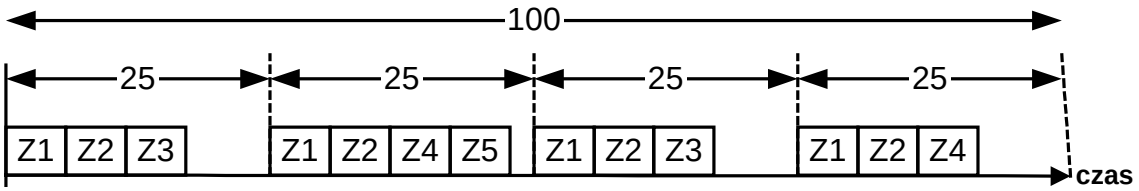
Cyklem głównym (ang. *major cycle*) jest najmniejszy wspólna wielokrotność z cykli poszczególnych zadań (p_1, p_2, ..., p_n).

Plan zaszeregowania sporządzany jest dla cyklu głównego. W cyklu głównym wszystkie zadania przyporządkowane do procesora powinny zostać wykonane tak, aby czasy zakończenia zadań nie zostały przekroczone.

Zadanie	Okres	Czas obliczeń
Z1	25	10
Z2	25	7
Z3	50	5
Z4	50	4
Z5	100	5

Przykład 3 Szeregowanie cykliczne, duży cykl 100, mały cykl 25

```
ustaw przerwania na 25 jednostek
do {
  czekaj_na_przerwanie;
  wykonaj(Z1); wykonaj(Z2); wykonaj(Z3);
  czekaj_na_przerwanie;
  wykonaj(Z1); wykonaj(Z2); wykonaj(Z4); wykonaj(Z5);
  czekaj_na_przerwanie;
  wykonaj(Z1); wykonaj(Z2); wykonaj(Z3);
  czekaj_na_przerwanie;
  wykonaj(Z1); wykonaj(Z2); wykonaj(Z4);
} while(1)
```



Rys. 5 Szeregowanie cykliczne zadań Z1, Z2, Z3, Z4, Z5

Zalety szeregowania cyklicznego:

- Determinizm
- Brak wywłaszczeń

Wady szeregowania cyklicznego:

- Opracowanie tabeli wykonywania cyklicznego jest w ogólności trudne (problem NP trudny).
- Trudna aktualizacja
- Problem z włączeniem zadań o długim okresie
- Niemożliwe włączenie zadań asynchronicznych

1.4.3 Szeregowanie RM (Rate Monotonic)

Szeregowanie stosowane tylko do zadań cyklicznych

Zasady szeregowania RM:

- Priorytety przydzielane są zadaniom w zależności od okresu wznawiania.
- Zadania najczęściej wznawiane otrzymują najwyższe priorytety a kolejne priorytety niższe.
- Priorytety przydziela się na stałe przed rozpoczęciem przetwarzania.
- Dopuszcza się możliwość wywłaszczenia bieżącego zadania, gdy zadanie o wyższym priorytecie staje się gotowe.

Wyniki teoretyczne:

- Wykazano że algorytm RM jest optymalny spośród wszystkich algorytmów ze stałym przydziałem priorytetów.

Wykazano że zbiór n zadań jest szeregowany według algorytmu RM gdy współczynnik wykorzystania procesora jest mniejszy od $U_{gr} = n\left(2^{\frac{1}{n}} - 1\right)$. Dla $n \to \infty$ $U_{gr} = \ln 2 \approx 0.69$

- Mogą jednak istnieć plany dla wyższego współczynnika wykorzystania procesora.

Z	e_i	T_i	$Ui = e_i / T_i$
Z1	1	4	0.25
Z2	2	5	0.4
Z3	5	20	0.25

Przykład 4 Szeregowanie RM

1.4.4 Algorytm szeregowania EDF (Earliest Deadline First)

Zasady szeregowania RM:

- Priorytety przydzielane są zadaniom dynamicznie w zależności od wartości wymaganego czasu zakończenia obliczeń.
- Zadanie które musi najszybciej zakończyć obliczenia otrzymuje najwyższy priorytet.
- Jako że absolutny czas zakończenia zadania może ulec zmianie to priorytety zadań także mogą ulegać zmianie.
- Dopuszcza się możliwość wywłaszczenia bieżącego zadania gdy zadanie o wyższym priorytecie staje się gotowe.

Wyniki teoretyczne:

- Wykazano że algorytm EDF jest optymalny spośród wszystkich algorytmów z dynamicznym przydziałem priorytetów.
- Wykazano że zbiór n zadań jest szeregowany według algorytmu EDF gdy współczynnik wykorzystania procesora jest mniejszy od $U_{gr} \leq 1$.

Porównanie:

- Algorytmy ze stałymi priorytetami są łatwiejsze w implementacji
- W przypadku przeciążenia zachowanie się algorytmów ze stałymi priorytetami jest bardziej przewidywalne – nie wykonają się zadania o niższym priorytecie.
- W przypadku przeciążenia zachowanie się algorytmu EDF jest mniej przewidywalne – mogą się nie wykonać zadania o wysokim priorytecie.
- W przypadku algorytmów ze stałymi priorytetami możliwe jest poprawne szeregowanie mimo przekroczonego teoretycznego współczynnika wykorzystania procesora.