Systemy wbudowane - wykład 11

Przemek Błaśkiewicz

17 maja 2018

Notes

Systemy czasu rzeczywistego - powtórka

System czasu rzeczywistego

Real-time system to system, którego poprawność działania zależy nie tylko od poprawności rezultatów (obliczeń, decyzji, akcji), ale również od czasu, kiedy zostaną one wypracowane (*czas reakcji*).

Notes

RTOS - planowanie zadań - powtórka

Plan

Plan (*schedule*) dla zbioru zadań $(J_1, J_2, \ldots, J_n)$ to pewna funkcja: $\sigma : R^+ \to \{0, \ldots, n\}$, taka, że:

$$\forall t \in R^+, \exists t_1, t_2 \in R^+ : t \in [t_1, t_2), \forall t' \in [t_1, t_2) : \sigma(t) = \sigma(t').$$

Notes

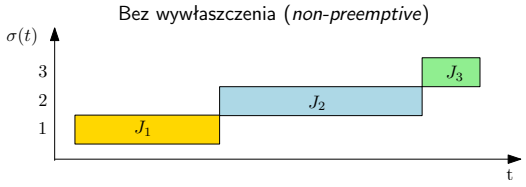
RTOS - planowanie zadań - powtórka

Plan

Plan (*schedule*) dla zbioru zadań $(J_1, J_2, \ldots, J_n)$ to pewna funkcja: $\sigma : R^+ \to \{0, \ldots, n\}$, taka, że:

$$\forall t \in R^+, \exists t_1, t_2 \in R^+ : t \in [t_1, t_2), \forall t' \in [t_1, t_2) : \sigma(t) = \sigma(t').$$

Bez wywłaszczenia (*non-preemptive*)



Notes

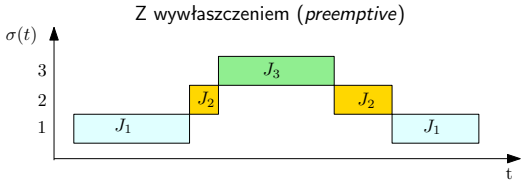
RTOS - planowanie zadań - powtórka

Plan

Plan (*schedule*) dla zbioru zadań $(J_1, J_2, \ldots, J_n)$ to pewna funkcja: $\sigma : R^+ \to \{0, \ldots, n\}$, taka, że:

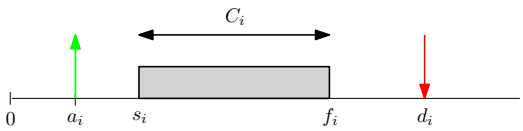
$$\forall t \in R^+, \exists t_1, t_2 \in R^+ : t \in [t_1, t_2), \forall t' \in [t_1, t_2) : \sigma(t) = \sigma(t').$$

Z wywłaszczeniem (*preemptive*)



Notes

Cechy charakterystyczne zadań - powtórka



Notes

Planiści - powtórka

- dla zadań aperiodycznych, synchronicznych – EDD

Notes

Planiści - powtórka

- dla zadań aperiodycznych, synchronicznych – EDD
- dla zadań aperiodycznych asynchronicznych – EDF

Notes

Planiści - powtórka

- dla zadań aperiodycznych, synchronicznych – EDD
- dla zadań aperiodycznych asynchronicznych – EDF
- j.w. + bez wywłaszczeń – EDF + "idle CPU"

Notes

EDF bez wywłaszczeń - powtórka

Notes

EDF bez wywłaszczeń - powtórka

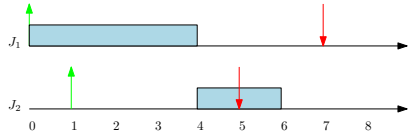
	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

Notes

EDF bez wywłaszczeń - powtórka

	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

- EDF bez wywłaszczeń:

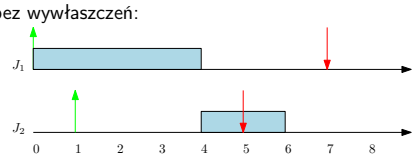


Notes

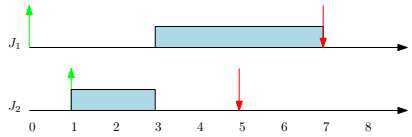
EDF bez wywłaszczeń - powtórka

	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

- EDF bez wywłaszczeń:



- optymalnie:



Notes

EDF bez wywłaszczeń - powtórka

Jeffay i in. 1991

Przy braku zezwolenia na okresy bezczynności przy gotowym do wykonania zadaniu, EDF jest optymalny również dla modelu bez wywłaszczeń.

Notes

Planiści - powtórka

- dla zadań aperiodycznych, synchronicznych – EDD
- dla zadań aperiodycznych asynchronicznych – EDF
- j.w. + bez wywłaszczeń – EDF + "idle CPU"

Notes

Planiści - powtórka

- dla zadań aperiodycznych, synchronicznych – EDD
- dla zadań aperiodycznych asynchronicznych – EDF
- j.w. + bez wywłaszczeń – EDF + "idle CPU"
- dla zadań aperdiodycznych, asynchronicznych, bez wywłaszczenia, "idle time" – algorytm Bratley'a

Notes

Algorytm Bartley'a – powtórka

- Bazuje na *odpowiedniej* permutacji zestawu zadań $J_1, \ldots, J_n$.
- **Branch:** wybierz do listy kolejne zadanie (z pozostałych)
- **Bound:**
  1. znaleziono realizowalny plan $\rightarrow$ bieżąca lista jest realizowalnym planem;
  2. jeśli istnieje zadanie, którego termin minął, a nie zostało jeszcze zaplanowane $\rightarrow$ bieżąca ścieżka to nierealizowalny plan (cofnij do ostatniego dokonanego wyboru).

Notes

Algorytm Bartley'a – powtórka

- Bazuje na *odpowiedniej* permutacji zestawu zadań $J_1, \ldots, J_n$.
- **Branch:** wybierz do listy kolejne zadanie (z pozostałych)
- **Bound:**
  1. znaleziono realizowalny plan $\rightarrow$ bieżąca lista jest realizowalnym planem;
  2. jeśli istnieje zadanie, którego termin minął, a nie zostało jeszcze zaplanowane $\rightarrow$ bieżąca ścieżka to nierealizowalny plan (cofnij do ostatniego dokonanego wyboru).
- Złożoność wykładnicza – tylko jako algorytm offline;

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...
- gdy wszystkie zadania są dostępne od razu – LDF

Notes

LDF – latest deadline first

1. utrzymuj zadania do planowania w grafie G, do wykonania na liście P;

Notes

LDF – latest deadline first

1. utrzymuj zadania do planowania w grafie G, do wykonania na liście P;
2. spośród wszystkich liści G, wybierz zadanie J o najpóźniejszym terminie wykonania (deadline);

Notes

LDF – latest deadline first

1. utrzymuj zadania do planowania w grafie G, do wykonania na liście P;
2. spośród wszystkich liści G, wybierz zadanie J o najpóźniejszym terminie wykonania (deadline);
3. wstaw J na początek P i usuń je z G

Notes

LDF – latest deadline first

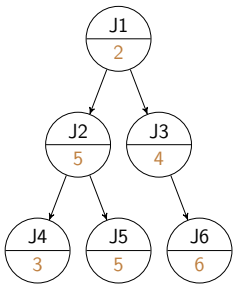
1. utrzymuj zadania do planowania w grafie G, do wykonania na liście P;
2. spośród wszystkich liści G, wybierz zadanie J o najpóźniejszym terminie wykonania (deadline);
3. wstaw J na początek P i usuń je z G
4. wykonuj ponownie od 2.

Notes

LDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6

$P = ()$

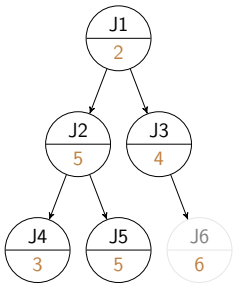


Notes

LDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6

$P = (J_6)$

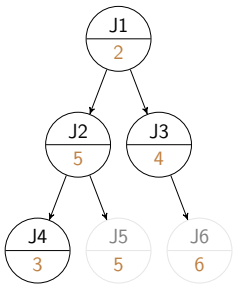


Notes

LDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6

$P = (J_5, J_6)$

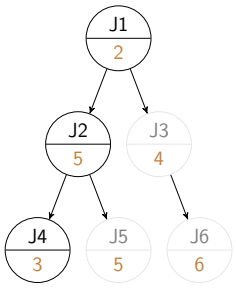


Notes

LDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6

$P = (J_3, J_5, J_6)$

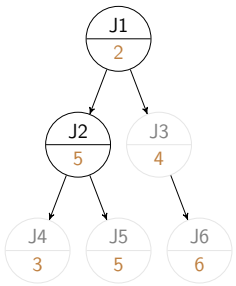


Notes

LDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6

$P = (J_4, J_3, J_5, J_6)$

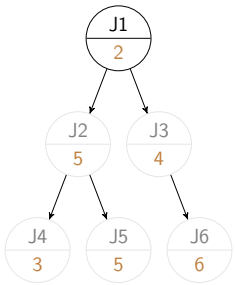


Notes

LDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6

$P = (J_2, J_4, J_3, J_5, J_6)$

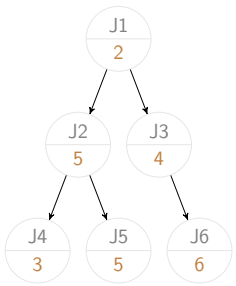


Notes

LDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	0	0	0	0	0	0
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6

$P = (J_1, J_2, J_4, J_3, J_5, J_6)$



Notes

LDF – optymalność

Notes

LDF – optymalność

Lawler '73

LDF jest optymalny ze względu na minimalizację maksymalnego opóźnienia.

Notes

LDF – optymalność

Lawler '73

LDF jest optymalny ze względu na minimalizację maksymalnego opóźnienia.

- dane są zadania $(J_1, \ldots, J_n)$
- niech J_l nie ma zadań "poniżej" siebie w drzewie zależności
- niech J_l ma najpóźniejszy termin wykonania

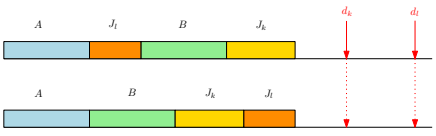
Notes

LDF – optymalność

Lawler '73

LDF jest optymalny ze względu na minimalizację maksymalnego opóźnienia.

- dane są zadania $(J_1, \ldots, J_n)$
- niech J_l nie ma zadań "poniżej" siebie w drzewie zależności
- niech J_l ma najpóźniejszy termin wykonania



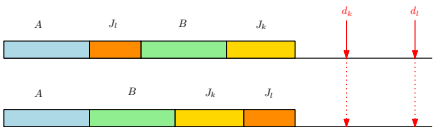
Notes

LDF – optymalność

Lawler '73

LDF jest optymalny ze względu na minimalizację maksymalnego opóźnienia.

- dane są zadania $(J_1, \ldots, J_n)$
- niech J_l nie ma zadań "poniżej" siebie w drzewie zależności
- niech J_l ma najpóźniejszy termin wykonania



1. przesuń J_l na koniec kolejki wykonania (bo $d_l > d_k$);

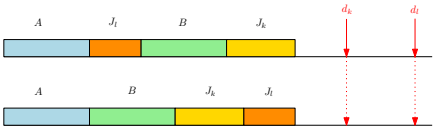
Notes

LDF – optymalność

Lawler '73

LDF jest optymalny ze względu na minimalizację maksymalnego opóźnienia.

- dane są zadania $(J_1, \ldots, J_n)$
- niech J_l nie ma zadań "poniżej" siebie w drzewie zależności
- niech J_l ma najpóźniejszy termin wykonania



1. przesuń J_l na koniec kolejki wykonania (bo $d_l > d_k$);
2. pokaż, że kolejność wykonania jest zachowana

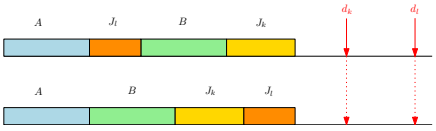
Notes

LDF – optymalność

Lawler '73

LDF jest optymalny ze względu na minimalizację maksymalnego opóźnienia.

- dane są zadania $(J_1, \ldots, J_n)$
- niech J_l nie ma zadań "poniżej" siebie w drzewie zależności
- niech J_l ma najpóźniejszy termin wykonania



1. przesuń J_l na koniec kolejki wykonania (bo $d_l > d_k$);
2. pokaż, że kolejność wykonania jest zachowana
3. pokaż, że maksymalne opóźnienie nie zwiększa się

Notes

LDF – właściwości

- działa jedynie jako algorytm off-line

Notes

LDF – właściwości

- działa jedynie jako algorytm off-line
- złożoność:

Notes

LDF – właściwości

- działa jedynie jako algorytm off-line
- złożoność:
  - szukanie liści w grafie G – $\mathcal{O}(|E|)$

Notes

LDF – właściwości

- działa jedynie jako algorytm off-line
- złożoność:
  - szukanie liści w grafie G – $\mathcal{O}(|E|)$
  - utrzymywanie posortowanej (wg. d_i) listy liści – $\mathcal{O}(\log n)$

Notes

LDF – właściwości

- działa jedynie jako algorytm off-line
- złożoność:
  - szukanie liści w grafie G – $\mathcal{O}(|E|)$
  - utrzymywanie posortowanej (wg. d_i) listy liści – $\mathcal{O}(\log n)$
  - ogółem: $\mathcal{O}(n * \max(|E|, \log n))$

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...
  - wszystkie zadania są dostępne od razu – LDF

Notes

Planowanie zadań z wymaganą kolejnością wykonania

- Planowanie bez wywłaszczenia, z asynchronicznymi pojawieniami się zadań, czasami wykonania i wymaganą kolejnością jest **NP-trudne**.
- Dla pewnych rozluźnień problemu można znaleźć optymalne rozwiązania...
  - wszystkie zadania są dostępne od razu – LDF
  - dla schematów wywłaszczających – modyfikowany EDF

Notes

Zmodyfikowany EDF – przykład

Aby zaadaptować przypadek, gdy zadania pojawiają się w systemie asynchronicznie ($\exists i, j : a_i \neq a_j$):

Notes

Zmodyfikowany EDF – przykład

Aby zaadaptować przypadek, gdy zadania pojawiają się w systemie asynchronicznie ($\exists i, j : a_i \neq a_j$):

1. zaktualizuj czasy nadejścia a_i:

Notes

Zmodyfikowany EDF – przykład

Aby zaadaptować przypadek, gdy zadania pojawiają się w systemie asynchronicznie ($\exists i, j : a_i \neq a_j$):

1. zaktualizuj czasy nadejścia a_i:
   - każde zadanie w "korzeniu": $a_i^* = a_i$

Notes

Zmodyfikowany EDF – przykład

Aby zaadaptować przypadek, gdy zadania pojawiają się w systemie asynchronicznie ($\exists i, j : a_i \neq a_j$):

1. zaktualizuj czasy nadejścia a_i:
   - każde zadanie w "korzeniu": $a_i^* = a_i$
   - dla każdego zadania, którego czas nadejścia "rodzica" J_r został zaktualizowany: $a_i^* = \max\{a_i, a_r^* + C_r\}$

Notes

Zmodyfikowany EDF – przykład

Aby zaadaptować przypadek, gdy zadania pojawiają się w systemie asynchronicznie ($\exists i, j : a_i \neq a_j$):

1. zaktualizuj czasy nadejścia a_i:
   - każde zadanie w "korzeniu": $a_i^* = a_i$
   - dla każdego zadania, którego czas nadejścia "rodzica" J_r został zaktualizowany: $a_i^* = \max\{a_i, a_r^* + C_r\}$
2. zaktualizuj terminy wykonania:

Notes

Zmodyfikowany EDF – przykład

Aby zaadaptować przypadek, gdy zadania pojawiają się w systemie asynchronicznie ($\exists i, j : a_i \neq a_j$):

1. zaktualizuj czasy nadejścia a_i:
   - każde zadanie w "korzeniu": $a_i^* = a_i$
   - dla każdego zadania, którego czas nadejścia "rodzica" J_r został zaktualizowany: $a_i^* = \max\{a_i, a_r^* + C_r\}$
2. zaktualizuj terminy wykonania:
   - dla każdego zadania w "liściu": $d_i^* = d_i$

Notes

Zmodyfikowany EDF – przykład

Aby zaadaptować przypadek, gdy zadania pojawiają się w systemie asynchronicznie ($\exists i, j : a_i \neq a_j$):

1. zaktualizuj czasy nadejścia a_i:
   - każde zadanie w "korzeniu": $a_i^* = a_i$
   - dla każdego zadania, którego czas nadejścia "rodzica" J_r został zaktualizowany: $a_i^* = \max\{a_i, a_r^* + C_r\}$
2. zaktualizuj terminy wykonania:
   - dla każdego zadania w "liściu": $d_i^* = d_i$
   - dla każdego zadania J_i, którego "potomkowie" J_p zostali zaktualizowani: $d_i^* = \min\{d_i, d_p^* - C_p\}$

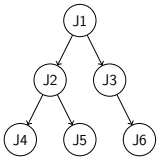
Notes

Zmodyfikowany EDF – przykład

Notes

Zmodyfikowany EDF – przykład

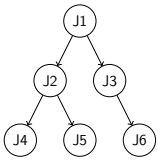
	J_1	J_2	J_3	J_4	J_5	J_6
a_i	1	2	3	3	3	4
a_i^*						
C_i	1	2	1	1	1	3
d_i	2	5	4	6	5	7
d_i^*						



Notes

Zmodyfikowany EDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	1	2	3	3	3	4
a_i^*	1	2	3	4	4	4
C_i	1	2	1	1	1	3
d_i	2	5	4	6	5	7
d_i^*						

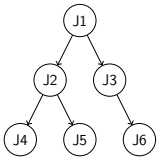


- $a_1^* = a_1 = 1$
- $a_2^* = \max\{a_2, a_1^* + C_1\} = 2$
- $a_3^* = \max\{a_3, a_1^* + C_1\} = 3$
- $a_4^* = \max\{a_4, a_2^* + C_2\} = 4$
- $a_5^* = \max\{a_5, a_2^* + C_2\} = 4$
- $a_6^* = \max\{a_6, a_3^* + C_3\} = 4$

Notes

Zmodyfikowany EDF – przykład

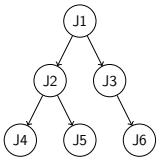
	J_1	J_2	J_3	J_4	J_5	J_6
a_i	1	2	3	3	3	4
a_i^*	1	2	3	4	4	4
C_i	1	2	1	1	1	3
d_i	2	5	4	6	5	7
d_i^*						



Notes

Zmodyfikowany EDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	1	2	3	3	3	4
a_i^*	1	2	3	4	4	4
C_i	1	2	1	1	1	3
d_i	2	5	4	6	5	7
d_i^*	0	4	4	6	5	7

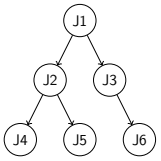


- $d_6^* = d_6 = 7$
- $d_5^* = d_5 = 5$
- $d_4^* = d_4 = 6$
- $d_3^* = \min\{d_3, d_6^* - C_6\} = 4$
- $d_2^* = \min\{d_2, d_5^* - C_5, d_4^* - C_4\} = 4$
- $d_1^* = \min\{d_1, d_2^* - C_2, d_3^* - C_3\} = 0$

Notes

Zmodyfikowany EDF – przykład

	J_1	J_2	J_3	J_4	J_5	J_6
a_i	1	2	3	3	3	4
a_i^*	1	2	3	4	4	4
C_i	1	2	1	1	1	3
d_i	2	5	4	6	5	7
d_i^*	0	4	4	6	5	7



Jak zmodyfikować zadania, by plan był wykonalny?

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$
2. Jeśli zadania $J_i \prec J_j$ planowane są przez EDF, to:

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$
2. Jeśli zadania $J_i \prec J_j$ planowane są przez EDF, to:
   - $a_i^* < a_j^*$: J_i pojawiło się wcześniej niż J_j;

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$
2. Jeśli zadania $J_i \prec J_j$ planowane są przez EDF, to:
   - $a_i^* < a_j^*$: J_i pojawiło się wcześniej niż J_j;
   - $d_i^* < d_j^*$: J_i ma wcześniejszy termin wykonania;

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$
2. Jeśli zadania $J_i \prec J_j$ planowane są przez EDF, to:
   - $a_i^* < a_j^*$: J_i pojawiło się wcześniej niż J_j;
   - $d_i^* < d_j^*$: J_i ma wcześniejszy termin wykonania;

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$
2. Jeśli zadania $J_i \prec J_j$ planowane są przez EDF, to:
   - $a_i^* < a_j^*$: J_i pojawiło się wcześniej niż J_j;
   - $d_i^* < d_j^*$: J_i ma wcześniejszy termin wykonania;

   zatem:

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$
2. Jeśli zadania $J_i \prec J_j$ planowane są przez EDF, to:
   - $a_i^* < a_j^*$: J_i pojawiło się wcześniej niż J_j;
   - $d_i^* < d_j^*$: J_i ma wcześniejszy termin wykonania;

   zatem:
   - J_j nie może się rozpocząć przed J_i, oraz J_j nie wywłaszczy J_i;

Notes

Zmodyfikowany EDF – przechodniość planowalności

Przechodniość planowalności

Zbiór zadań jest planowalny z zachowaniem wymogów następstwa wykonania zadań wtedy i tylko wtedy, gdy zbiór zadań o zmodyfikowanych parametrach (a_i, d_i) jest planowalny przez EDF.

1. Dla każdego planu σ zachowującego zależności musi być:
   - $s_i \geq \max\{a_i, a_r^* + C_r\}$
   - $f_i \leq \min\{d_i, d_p^* - C_p\}$
2. Jeśli zadania $J_i \prec J_j$ planowane są przez EDF, to:
   - $a_i^* < a_j^*$: J_i pojawiło się wcześniej niż J_j;
   - $d_i^* < d_j^*$: J_i ma wcześniejszy termin wykonania;

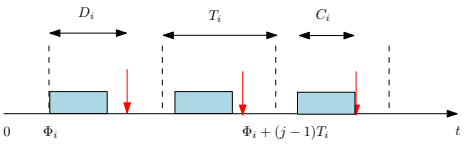
   zatem:
   - J_j nie może się rozpocząć przed J_i, oraz J_j nie wywłaszczy J_i;
   - $a_i^* \geq a_i$ oraz $d_i^* \leq d_i$

Notes

Planowanie zadań okresowych

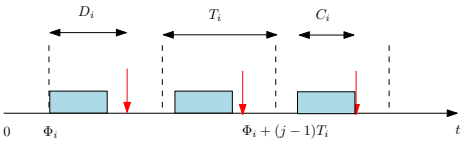
Notes

Planowanie zadań okresowych



Notes

Planowanie zadań okresowych

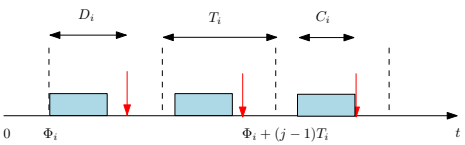


Każde zadanie okresowe t_i określone jest przez:

- fazę Φ_i – pierwszy moment wystąpienia;

Notes

Planowanie zadań okresowych

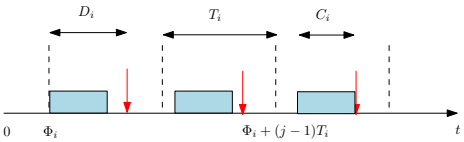


Każde zadanie okresowe t_i określone jest przez:

- fazę Φ_i – pierwszy moment wystąpienia;
- okres T_i – czas jaki upływa między dwoma kolejnymi wystąpieniami;

Notes

Planowanie zadań okresowych

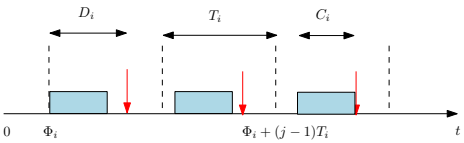


Każde zadanie okresowe t_i określone jest przez:

- fazę Φ_i – pierwszy moment wystąpienia;
- okres T_i – czas jaki upływa między dwoma kolejnymi wystąpieniami;
- względny termin wykonania (deadline) D_i – okres czasu przeznaczony na wykonanie zadania od momentu jego wystąpienia;

Notes

Planowanie zadań okresowych

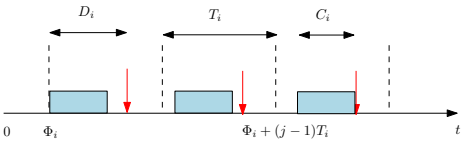


Każde zadanie okresowe t_i określone jest przez:

- fazę Φ_i – pierwszy moment wystąpienia;
- okres T_i – czas jaki upływa między dwoma kolejnymi wystąpieniami;
- względny termin wykonania (deadline) D_i – okres czasu przeznaczony na wykonanie zadania od momentu jego wystąpienia;
- czas wykonania C_i – czas trwania obliczeń zadania.

Notes

Planowanie zadań okresowych

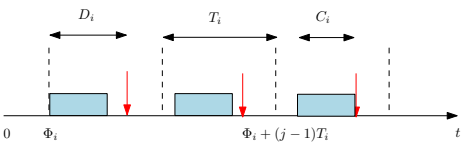


Każde zadanie okresowe t_i określone jest przez:

- fazę Φ_i – pierwszy moment wystąpienia;
- okres T_i – czas jaki upływa między dwoma kolejnymi wystąpieniami;
- względny termin wykonania (deadline) D_i – okres czasu przeznaczony na wykonanie zadania od momentu jego wystąpienia;
- czas wykonania C_i – czas trwania obliczeń zadania.

Notes

Planowanie zadań okresowych



Każde zadanie okresowe t_i określone jest przez:

- fazę Φ_i – pierwszy moment wystąpienia;
- okres T_i – czas jaki upływa między dwoma kolejnymi wystąpieniami;
- względny termin wykonania (deadline) D_i – okres czasu przeznaczony na wykonanie zadania od momentu jego wystąpienia;
- czas wykonania C_i – czas trwania obliczeń zadania.

Plan realizowalny składa się z czasów startu $s_{i,j}$ i ukończenia $f_{i,j}$ dla każdego zadania i w każdej jego instancji (pojawieniu się) j.

Notes

Planista RM - rate-monotonic

(Liu, Layland, 1973):

- każde zadanie otrzymuje priorytet $1/T_i$ (proporcjonalny do częstotliwości pojawiania się);

Notes

Planista RM - rate-monotonic

(Liu, Layland, 1973):

- każde zadanie otrzymuje priorytet $1/T_i$ (proporcjonalny do częstotliwości pojawiania się);
- wykonywane jest gotowe zadanie z aktualnie najwyższym priorytetem;

Notes

Planista RM - rate-monotonic

(Liu, Layland, 1973):

- każde zadanie otrzymuje priorytet $1/T_i$ (proporcjonalny do częstotliwości pojawiania się);
- wykonywane jest gotowe zadanie z aktualnie najwyższym priorytetem;
- system z wywłaszczeniami;

Notes

RM – przykład (1)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	6	12
C_i	2	1	4
D_i	4	6	12

Notes

RM – przykład (1)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	6	12
C_i	2	1	4
D_i	4	6	12

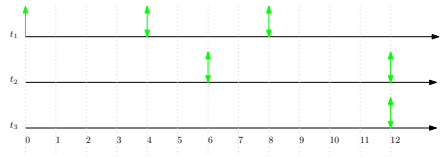
Policzmy: $C_1/T_1 + C_2/T_2 + C_3/T_3 = 2/4 + 1/6 + 4/12 = 1$.

Notes

RM – przykład (1)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	6	12
C_i	2	1	4
D_i	4	6	12

Policzmy: $C_1/T_1 + C_2/T_2 + C_3/T_3 = 2/4 + 1/6 + 4/12 = 1$.

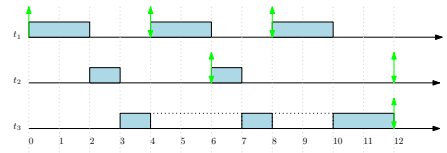


Notes

RM – przykład (1)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	6	12
C_i	2	1	4
D_i	4	6	12

Policzmy: $C_1/T_1 + C_2/T_2 + C_3/T_3 = 2/4 + 1/6 + 4/12 = 1$.



Notes

RM – przykład (2)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	5	10
C_i	2	2	1
D_i	4	5	10

Notes

RM – przykład (2)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	5	10
C_i	2	2	1
D_i	4	5	10

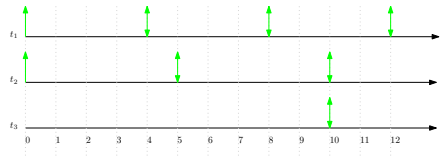
Policzmy: $C_1/T_1 + C_2/T_2 + C_3/T_3 = 2/4 + 2/5 + 1/10 = 1$.

Notes

RM – przykład (2)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	5	10
C_i	2	2	1
D_i	4	5	10

Policzmy: $C_1/T_1 + C_2/T_2 + C_3/T_3 = 2/4 + 2/5 + 1/10 = 1$.

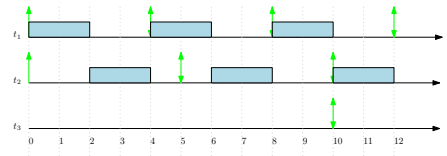


Notes

RM – przykład (2)

	t_1	t_2	t_3
Φ_i	0	0	0
T_i	4	5	10
C_i	2	2	1
D_i	4	5	10

Policzmy: $C_1/T_1 + C_2/T_2 + C_3/T_3 = 2/4 + 2/5 + 1/10 = 1$.



Notes

Stopień wykorzystania procesora - warunek planowalności

Stopień wykorzystania procesora

Dla danego zbioru n zadań okresowych, stopień wykorzystania procesora to:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}.$$

- warunek konieczny dla stworzenia realizowalnego planu: $U \leq 1$

Notes

Stopień wykorzystania procesora - warunek planowalności

Stopień wykorzystania procesora

Dla danego zbioru n zadań okresowych, stopień wykorzystania procesora to:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}.$$

- warunek konieczny dla stworzenia realizowalnego planu: $U \leq 1$
- dla algorytmu RM w pesymistycznym wypadku, dla $U \geq 2 \cdot (2^{1/n} - 1)$ brak gwarancji poprawnego rozwiązania.

Notes

Rzeczy do zapamiętania

- różnice między RTOS a "PC-OS";
- właściwości czasowe zadań w systemie (w tym okresowych);
- rodzaje planistów, jakie warunki biorą pod uwagę;
- algorytm EDD, EDF, LDF, Bratley'a, RM

Notes

Notes

Notes