Systemy wbudowane - wykład kolejny

Przemek Błaśkiewicz

9 maja 2019

- sterowanie silnikiem rakietowym;

Systemy czasu rzeczywistego

- sterowanie silnikiem rakietowym;
- system kontroli ABS;

Systemy czasu rzeczywistego

- sterowanie silnikiem rakietowym;
- system kontroli ABS;
- kontrola rdzenia reaktora;

Systemy czasu rzeczywistego

- sterowanie silnikiem rakietowym;
- system kontroli ABS;
- kontrola rdzenia reaktora;
- robot spawalniczy;

Systemy czasu rzeczywistego

- sterowanie silnikiem rakietowym;
- system kontroli ABS;
- kontrola rdzenia reaktora;
- robot spawalniczy;
- system podtrzymywania życia pacjenta;

Systemy czasu rzeczywistego

- sterowanie silnikiem rakietowym;
- system kontroli ABS;
- kontrola rdzenia reaktora;
- robot spawalniczy;
- system podtrzymywania życia pacjenta;
- przenośny odtwarzacz muzyki/telefon/aparat;

Systemy czasu rzeczywistego

- sterowanie silnikiem rakietowym;
- system kontroli ABS;
- kontrola rdzenia reaktora;
- robot spawalniczy;
- system podtrzymywania życia pacjenta;
- przenośny odtwarzacz muzyki/telefon/aparat;
- ...

Definicja

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*).

Gotowość

- Program (system) jest zawsze gotowy na napływające dane

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);
  - spełnienia dodatkowych założeń dot. np. kolejności.

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);
  - spełnienia dodatkowych założeń dot. np. kolejności.
- Współdziała ze środowiskiem, reagując na jego niezależne zmiany.

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);
  - spełnienia dodatkowych założeń dot. np. kolejności.
- Współdziała ze środowiskiem, reagując na jego niezależne zmiany.

Gotowość

- Program (system) jest zawsze gotowy na napływające dane
  - synchronicznie;
  - asynchronicznie.
- Proces/system przetwarza dane w sposób przewidywalny, w rozumieniu:
  - długości czasu przetwarzania;
  - na bieżąco (długości czasu oczekiwania na rozpoczęcie przetwarzania);
  - spełnienia dodatkowych założeń dot. np. kolejności.
- Współdziała ze środowiskiem, reagując na jego niezależne zmiany.

nieprzewidywalne bodźce $\rightarrow$ przewidywalny wynik

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.
- “Normalne” systemy operacyjne mogą być szybsze, bo:

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.
- “Normalne” systemy operacyjne mogą być szybsze, bo:
  - korzystają z pamięci cache;

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.
- “Normalne” systemy operacyjne mogą być szybsze, bo:
  - korzystają z pamięci cache;
  - mają wielopotokowe procesory;

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.
- “Normalne” systemy operacyjne mogą być szybsze, bo:
  - korzystają z pamięci cache;
  - mają wielopotokowe procesory;
  - mogą korzystać z akceleratorów (np. karty graficzne).

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.
- “Normalne” systemy operacyjne mogą być szybsze, bo:
  - korzystają z pamięci cache;
  - mają wielopotokowe procesory;
  - mogą korzystać z akceleratorów (np. karty graficzne).
- **ALE** Windows po szeregu aktualizacji i instalacji sterowników wyraźnie zwalnia...

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.
- “Normalne” systemy operacyjne mogą być szybsze, bo:
  - korzystają z pamięci cache;
  - mają wielopotokowe procesory;
  - mogą korzystać z akceleratorów (np. karty graficzne).
- **ALE** Windows po szeregu aktualizacji i instalacji sterowników wyraźnie zwalnia...
- **ALE** sposoby przyspieszania działają dobrze, ale wprowadzają zależność czasu wykonania działań od okoliczności ich wykonania.

Szybkość RT-OS

- Pojęcie “real-time” nie oznacza “szybki”.
- “Normalne” systemy operacyjne mogą być szybsze, bo:
  - korzystają z pamięci cache;
  - mają wielopotokowe procesory;
  - mogą korzystać z akceleratorów (np. karty graficzne).
- **ALE** Windows po szeregu aktualizacji i instalacji sterowników wyraźnie zwalnia...
- **ALE** sposoby przyspieszania działają dobrze, ale wprowadzają zależność czasu wykonania działań od okoliczności ich wykonania.
- RT-OS mają gwarantowany *pesymistyczny czas reakcji*.

“PC” vs. SW vs. RTOS

- Systemy PC są oceniane pod względem prędkości działania, przepustowości, wszechstronności *w średnim przypadku*.

“PC” vs. SW vs. RTOS

- Systemy PC są oceniane pod względem prędkości działania, przepustowości, wszechstronności *w średnim przypadku*.
- Dlatego są one zmiennicze pod względem sprzętu i oprogramowania, które jest na nim uruchomione co daje różnorodność konfiguracji, kontrolowanych przez nie procesów oraz sposobu, w jaki się pojawiają w systemie.

“PC” vs. SW vs. RTOS

- Systemy PC są oceniane pod względem prędkości działania, przepustowości, wszechstronności *w średnim przypadku*.
- Dlatego są one zmiennicze pod względem sprzętu i oprogramowania, które jest na nim uruchomione co daje różnorodność konfiguracji, kontrolowanych przez nie procesów oraz sposobu, w jaki się pojawiają w systemie.
- System wbudowany jest często zamknięty pod względem sprzętu i oprogramowania, jak i środowiska, w którym pracuje.

“PC” vs. SW vs. RTOS

- Systemy PC są oceniane pod względem prędkości działania, przepustowości, wszechstronności *w średnim przypadku*.
- Dlatego są one zmiennicze pod względem sprzętu i oprogramowania, które jest na nim uruchomione co daje różnorodność konfiguracji, kontrolowanych przez nie procesów oraz sposobu, w jaki się pojawiają w systemie.
- System wbudowany jest często zamknięty pod względem sprzętu i oprogramowania, jak i środowiska, w którym pracuje.
- Można więc przedstawić lepsze mechanizmy planowania wykonania zadań, bo:

“PC” vs. SW vs. RTOS

- Systemy PC są oceniane pod względem prędkości działania, przepustowości, wszechstronności *w średnim przypadku*.
- Dlatego są one zmiennicze pod względem sprzętu i oprogramowania, które jest na nim uruchomione co daje różnorodność konfiguracji, kontrolowanych przez nie procesów oraz sposobu, w jaki się pojawiają w systemie.
- System wbudowany jest często zamknięty pod względem sprzętu i oprogramowania, jak i środowiska, w którym pracuje.
- Można więc przedstawić lepsze mechanizmy planowania wykonania zadań, bo:
  - znamy charakter zadań (okresowość, wymagania obliczeniowe);

“PC” vs. SW vs. RTOS

- Systemy PC są oceniane pod względem prędkości działania, przepustowości, wszechstronności *w średnim przypadku*.
- Dlatego są one zmiennicze pod względem sprzętu i oprogramowania, które jest na nim uruchomione co daje różnorodność konfiguracji, kontrolowanych przez nie procesów oraz sposobu, w jaki się pojawiają w systemie.
- System wbudowany jest często zamknięty pod względem sprzętu i oprogramowania, jak i środowiska, w którym pracuje.
- Można więc przedstawić lepsze mechanizmy planowania wykonania zadań, bo:
  - znamy charakter zadań (okresowość, wymagania obliczeniowe);
  - wiemy jak szybko trzeba je wykonać;

“PC” vs. SW vs. RTOS

- Systemy PC są oceniane pod względem prędkości działania, przepustowości, wszechstronności *w średnim przypadku*.
- Dlatego są one zmiennicze pod względem sprzętu i oprogramowania, które jest na nim uruchomione co daje różnorodność konfiguracji, kontrolowanych przez nie procesów oraz sposobu, w jaki się pojawiają w systemie.
- System wbudowany jest często zamknięty pod względem sprzętu i oprogramowania, jak i środowiska, w którym pracuje.
- Można więc przedstawić lepsze mechanizmy planowania wykonania zadań, bo:
  - znamy charakter zadań (okresowość, wymagania obliczeniowe);
  - wiemy jak szybko trzeba je wykonać;
  - inaczej SW nie ma sensu (?).

Rygorystyczne (hard)

- gwarantują wypełnienie zadań krytycznych na czas;

Podział RTOS

Rygorystyczne (hard)

- gwarantują wypełnienie zadań krytycznych na czas;
- ograniczają opóźnienie wszystkich zadań w systemie;

Podział RTOS

Rygorystyczne (hard)

- gwarantują wypełnienie zadań krytycznych na czas;
- ograniczają opóźnienie wszystkich zadań w systemie;
- dane przechowywane w szybkiej pamięci lub pamięci ROM;

Podział RTOS

Rygorystyczne (hard)

- gwarantują wypełnienie zadań krytycznych na czas;
- ograniczają opóźnienie wszystkich zadań w systemie;
- dane przechowywane w szybkiej pamięci lub pamięci ROM;
- brak wirtualizacji (bo wprowadza niedeterministyczne opóźnienia);

Podział RTOS

Rygorystyczne (hard)

- gwarantują wypełnienie zadań krytycznych na czas;
- ograniczają opóźnienie wszystkich zadań w systemie;
- dane przechowywane w szybkiej pamięci lub pamięci ROM;
- brak wirtualizacji (bo wprowadza niedeterministyczne opóźnienia);
- *NIE* współpracują z systemami z podziałem czasu!

Podział RTOS

Łagodne (soft)

- zadania krytyczne otrzymują i utrzymują pierwszeństwo przed innymi;

Łagodne (soft)

- zadania krytyczne otrzymują i utrzymują pierwszeństwo przed innymi;
- opóźnienia są ograniczone - zadania mają skończony czas oczekiwania na wykonanie;

Łagodne (soft)

- zadania krytyczne otrzymują i utrzymują pierwszeństwo przed innymi;
- opóźnienia są ograniczone - zadania mają skończony czas oczekiwania na wykonanie;

Podział RTOS

Łagodne (soft)

- zadania krytyczne otrzymują i utrzymują pierwszeństwo przed innymi;
- opóźnienia są ograniczone - zadania mają skończony czas oczekiwania na wykonanie;

Mocne (firm)

- pośrednie między rygorystycznymi a łagodnymi

Podział RTOS

Łagodne (soft)

- zadania krytyczne otrzymują i utrzymują pierwszeństwo przed innymi;
- opóźnienia są ograniczone - zadania mają skończony czas oczekiwania na wykonanie;

Mocne (firm)

- pośrednie między rygorystycznymi a łagodnymi
- niewykonanie zadania $\rightarrow$ wyniki nieprzydatne, ale OK

Plan

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

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

Plan

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

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

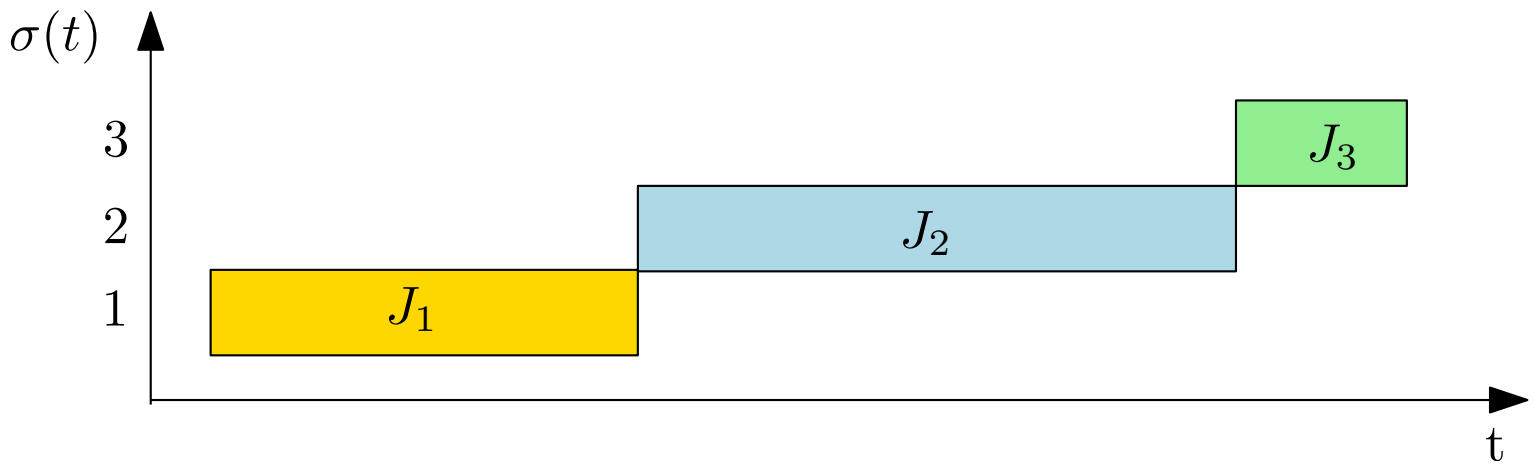
Czyli, jeśli $\sigma(t) = j$ to wykonywane jest zadanie J_j, jeśli $\sigma(t) = 0$, to nie wykonywane jest żadne zadanie.

Plan

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

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

Czyli, jeśli $\sigma(t) = j$ to wykonywane jest zadanie J_j, jeśli $\sigma(t) = 0$, to nie wykonywane jest żadne zadanie.

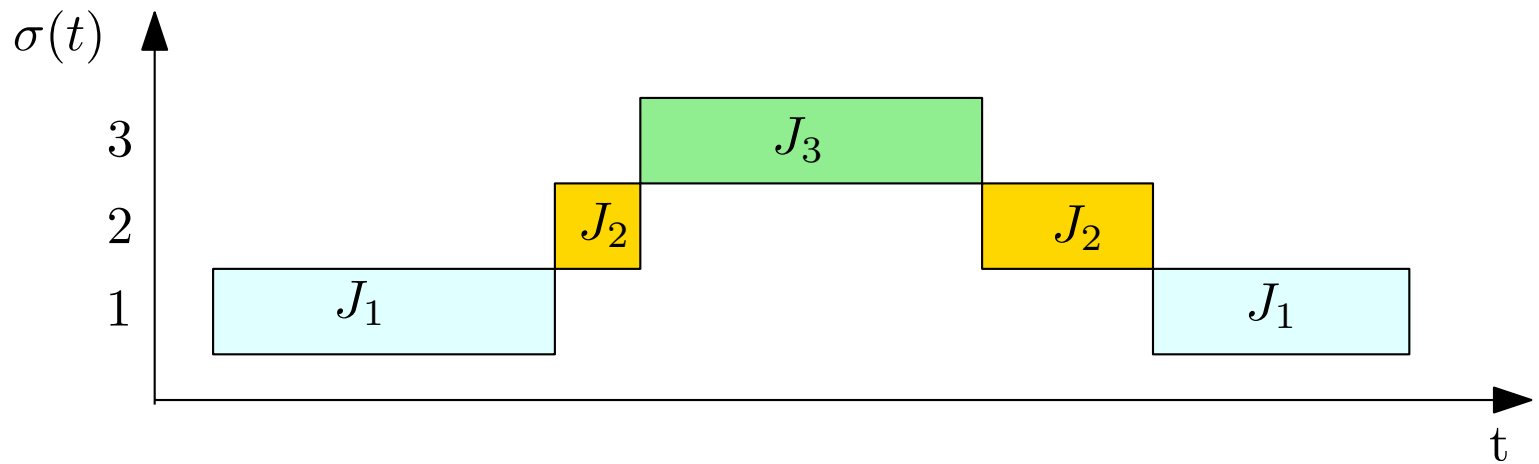


Plan

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

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

Czyli, jeśli $\sigma(t) = j$ to wykonywane jest zadanie J_j, jeśli $\sigma(t) = 0$, to nie wykonywane jest żadne zadanie.



Cechy charakterystyczne zadań

Właściwości a priori:

Cechy charakterystyczne zadań

Właściwości a priori:

pojawienie się w systemie – a_i;

Cechy charakterystyczne zadań

Właściwości a priori:

pojawienie się w systemie – a_i;

czas obliczenia – C_i - wymagany czas (max) na wykonanie obliczenia;

Cechy charakterystyczne zadań

Właściwości a priori:

pojawienie się w systemie – a_i;

czas obliczenia – C_i - wymagany czas (max) na wykonanie obliczenia;

czas zakończenia (deadline) – d_i - maksymalny czas, przed upływem którego zadanie musi zostać przetworzone;

Cechy charakterystyczne zadań

Właściwości a priori:

pojawienie się w systemie – a_i;

czas obliczenia – C_i - wymagany czas (max) na wykonanie obliczenia;

czas zakończenia (deadline) – d_i - maksymalny czas, przed upływem którego zadanie musi zostać przetworzone;

czas luzu (slack time) – $X_i = d_i - a_i - C_i$, maksymalny czas opóźnienia wykonania zadania po jego nadejściu, nadal umożliwiający jego wykonanie;

Cechy charakterystyczne zadań

Właściwości a priori:

pojawienie się w systemie – a_i;

czas obliczenia – C_i - wymagany czas (max) na wykonanie obliczenia;

czas zakończenia (deadline) – d_i - maksymalny czas, przed upływem którego zadanie musi zostać przetworzone;

czas luzu (slack time) – $X_i = d_i - a_i - C_i$, maksymalny czas opóźnienia wykonania zadania po jego nadejściu, nadal umożliwiający jego wykonanie;

Cechy charakterystyczne zadań

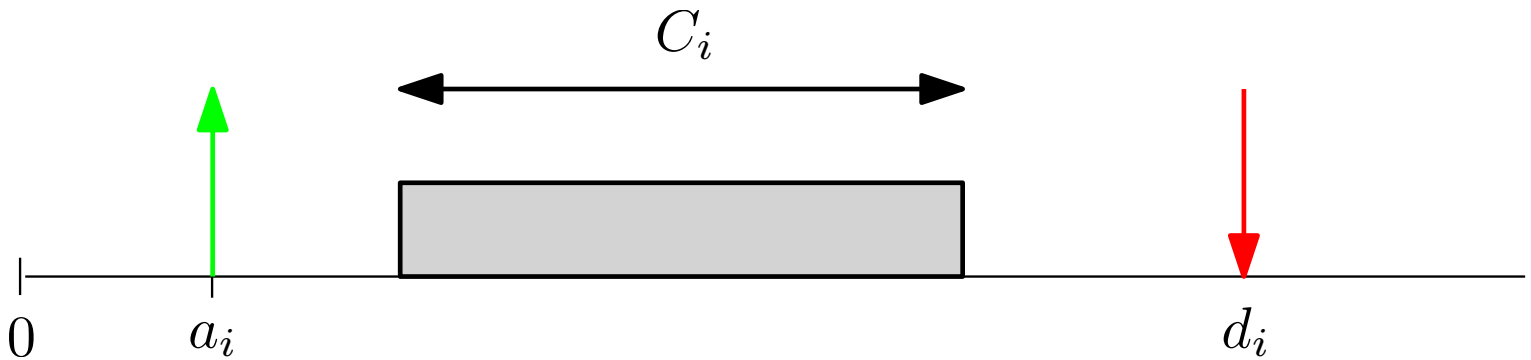
Właściwości a priori:

pojawienie się w systemie – a_i;

czas obliczenia – C_i - wymagany czas (max) na wykonanie obliczenia;

czas zakończenia (deadline) – d_i - maksymalny czas, przed upływem którego zadanie musi zostać przetworzone;

czas luzu (slack time) – $X_i = d_i - a_i - C_i$, maksymalny czas opóźnienia wykonania zadania po jego nadejściu, nadal umożliwiający jego wykonanie;



Cechy charakterystyczne zadań

Właściwości w planowaniu:

Cechy charakterystyczne zadań

Właściwości w planowaniu:

start wykonania – s_i;

Cechy charakterystyczne zadań

Właściwości w planowaniu:

start wykonania – s_i;

koniec wykonania – f_i;

Cechy charakterystyczne zadań

Właściwości w planowaniu:

start wykonania – s_i;

koniec wykonania – f_i;

opóźnienie - (lateness) – $L_i = f_i - d_i$;

Cechy charakterystyczne zadań

Właściwości w planowaniu:

start wykonania – s_i;

koniec wykonania – f_i;

opóźnienie - (lateness) – $L_i = f_i - d_i$;

przekroczenie czasu - (exceeding time) – $E_i = \max(0, L_i)$;

Cechy charakterystyczne zadań

Właściwości w planowaniu:

start wykonania – s_i;

koniec wykonania – f_i;

opóźnienie - (lateness) – $L_i = f_i - d_i$;

przekroczenie czasu - (exceeding time) – $E_i = \max(0, L_i)$;

Cechy charakterystyczne zadań

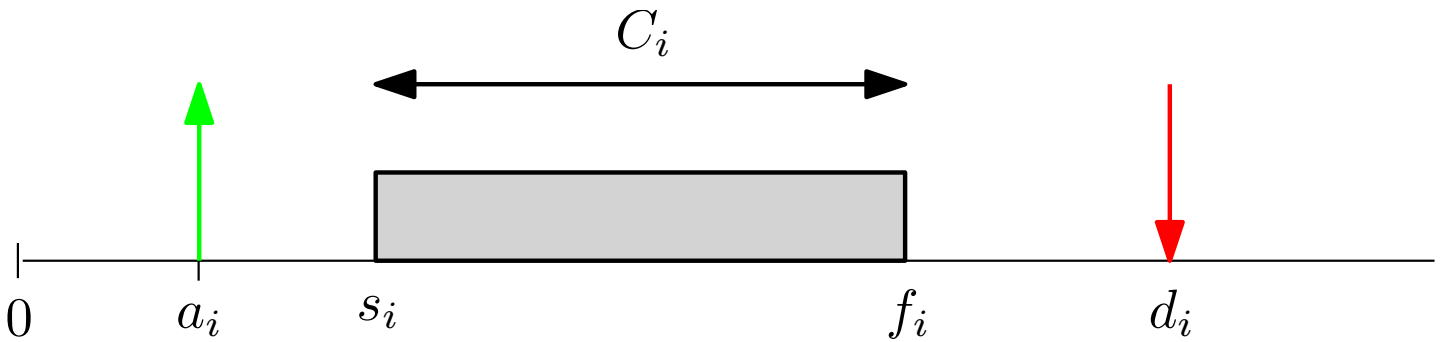
Właściwości w planowaniu:

start wykonania – s_i;

koniec wykonania – f_i;

opóźnienie - (lateness) – $L_i = f_i - d_i$;

przekroczenie czasu - (exceeding time) – $E_i = \max(0, L_i)$;



Ograniczenia na zadania – ograniczenie kolejności

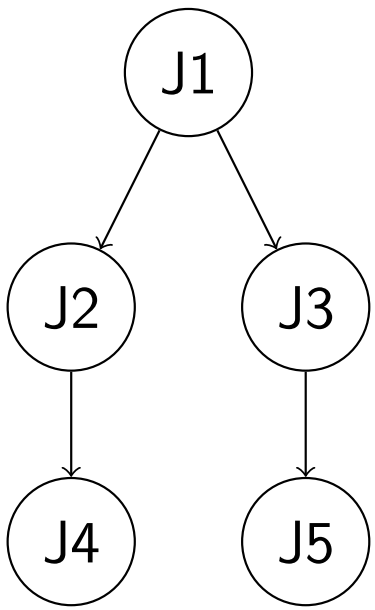
Pewne zadania muszą być wykonane przed innymi, np.

$$J_1 \prec J_2, J_1 \prec J_3, J_2 \prec J_4, J_3 \prec J_5$$

Ograniczenia na zadania – ograniczenie kolejności

Pewne zadania muszą być wykonane przed innymi, np.

$$J_1 \prec J_2, J_1 \prec J_3, J_2 \prec J_4, J_3 \prec J_5$$



Algorytmy planowania (1)

Realizowalność planu

(*feasibility*) – plan jest realizowalny, jeśli jego wykonanie powoduje że wszystkie zadania kończą się nie później niż w swoim terminie wykonania.

Algorytmy planowania (1)

Realizowalność planu

(*feasibility*) – plan jest realizowalny, jeśli jego wykonanie powoduje że wszystkie zadania kończą się nie później niż w swoim terminie wykonania.

Zestaw zadań jest *planowalny* jeśli istnieje co najmniej jeden realizowalny plan ich wykonania.

Algorytmy planowania (2)

- wywłaszczające i niewywłaszczające – zadania mogą być przerwane przez wykonanie innych zadań (lub nie, odpowiednio);

Algorytmy planowania (2)

- wywłaszczające i niewywłaszczające – zadania mogą być przerwane przez wykonanie innych zadań (lub nie, odpowiednio);
- statyczne i dynamiczne – decyzja planowania podjęta w momencie kompilacji (tworzenia sytemu), lub w trakcie działania, “na bieżąco”;

Algorytmy planowania (2)

- wywłaszczające i niewywłaszczające – zadania mogą być przerwane przez wykonanie innych zadań (lub nie, odpowiednio);
- statyczne i dynamiczne – decyzja planowania podjęta w momencie kompilacji (tworzenia sytemu), lub w trakcie działania, “na bieżąco”;
- jednoprocesorowe, wieloprocesorowe;

Algorytmy planowania (2)

- wywłaszczające i niewywłaszczające – zadania mogą być przerwane przez wykonanie innych zadań (lub nie, odpowiednio);
- statyczne i dynamiczne – decyzja planowania podjęta w momencie kompilacji (tworzenia sytemu), lub w trakcie działania, “na bieżąco”;
- jednoprocesorowe, wieloprocesorowe;
- optymalne, heurystyczne;

Algorytmy planowania (2)

- wywłaszczające i niewywłaszczające – zadania mogą być przerwane przez wykonanie innych zadań (lub nie, odpowiednio);
- statyczne i dynamiczne – decyzja planowania podjęta w momencie kompilacji (tworzenia sytemu), lub w trakcie działania, "na bieżąco";
- jednoprocesorowe, wieloprocesorowe;
- optymalne, heurystyczne;
- dla zadań okresowych (periodycznych) i asynchronicznych;

Lemat o wywłaszczaniu

Jeśli czasy nadejścia zadań są synchroniczne, to mechanizm wywłaszczenia nie polepsza maksymalnego opóźnienia. Czyli jeśli istnieje algorytm wywłaszczeniowy o opóźnieniu L_{max} to istnieje również algorytm niewywłaszczeniowy o tym samym opóźnieniu dla tego samego zestawu zadań.

Lemat o wywłaszczaniu

Jeśli czasy nadejścia zadań są synchroniczne, to mechanizm wywłaszczenia nie polepsza maksymalnego opóźnienia. Czyli jeśli istnieje algorytm wywłaszczeniowy o opóźnieniu L_{max} to istnieje również algorytm niewywłaszczeniowy o tym samym opóźnieniu dla tego samego zestawu zadań.

Dowód (szkic).

1. Załóżmy że istnieje plan wywłaszczający z opóźnieniem L_{max}.

Lemat o wywłaszczaniu

Jeśli czasy nadejścia zadań są synchroniczne, to mechanizm wywłaszczenia nie polepsza maksymalnego opóźnienia. Czyli jeśli istnieje algorytm wywłaszczeniowy o opóźnieniu L_{max} to istnieje również algorytm niewywłaszczeniowy o tym samym opóźnieniu dla tego samego zestawu zadań.

Dowód (szkic).

1. Załóżmy że istnieje plan wywłaszczający z opóźnieniem L_{max}.
2. Jeśli istnieje zadanie wywłaszczone (np. J_w) w czasie t i powraca w czasie t', to przesuń całość zadania J_w sprzed t na tuż przed t' (uniknięcie wywłaszczenia).

Lemat o wywłaszczaniu

Jeśli czasy nadejścia zadań są synchroniczne, to mechanizm wywłaszczenia nie polepsza maksymalnego opóźnienia. Czyli jeśli istnieje algorytm wywłaszczeniowy o opóźnieniu L_{max} to istnieje również algorytm niewywłaszczeniowy o tym samym opóźnieniu dla tego samego zestawu zadań.

Dowód (szkic).

1. Załóżmy że istnieje plan wywłaszczający z opóźnieniem L_{max}.
2. Jeśli istnieje zadanie wywłaszczone (np. J_w) w czasie t i powraca w czasie t', to przesuń całość zadania J_w sprzed t na tuż przed t' (uniknięcie wywłaszczenia).
3. Nie pogorszy to opóźnienia J_w, co najwyżej zmniejszy opóźnienie innych zadań.