Systemy czasu rzeczywistego

Mariusz Żynel
mariusz@math.uwb.edu.pl
http://math.uwb.edu.pl/~mariusz/

Uniwersytet w Białymstoku

2024/2025

Model referencyjny systemów czasu rzeczywistego

.

A reference model of real-time systems

Prace i zadania

Praca : jednostka działania, która jest zaplanowana i wykonywana przez system (*job*), np.: obliczenie transformaty Fouriera dla danych z sensora, przesłanie pakietu danych po sieci komputerowej, odczytanie pliku z dysku twardego

Zadanie : zestaw powiązanych ze sobą prac, które wspólnie razem zapewniają pewną funkcjonalność systemu (*task*), np.: utrzymywanie stałej wysokości samolotu

Procesory i zasoby

- Praca jest wykonywana na *procesorze* i może wymagać pewnych *zasobów*

Procesor : aktywny komponent, na którym prace są wykonywane (*processor*)

- Na przykład:
  - CPU – wykonuje instrukcje maszynowe
  - łącze danych – przesyła dane z jednego miejsca do drugiego
  - dysk/nośnik – odczytuje i zapisuje pliki
  - baza danych – przetwarza kwerendy
- Procesor posiada atrybut *prędkość*, który oznacza tempo postępu pracy w kierunku jej ukończenia, określany jako np. ilość instrukcji na sekundę, przepustowość sieci
- Dwa procesory są tego samego typu, jeżeli są funkcjonalnie identyczne i można używać ich zamiennie

Zasób : pasywny komponent, wymagany do wykonania pracy (*resource*)

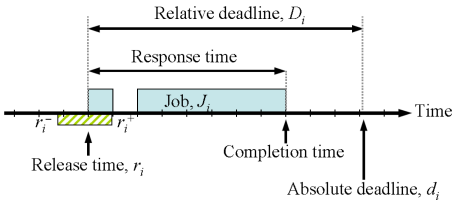
- Na przykład: pamięć, semafory, blokady bazy danych
- Zasoby mają różne typy i rozmiary, ale nie mają atrybutu prędkości
- Zasoby są zazwyczaj wielokrotnego użytku i nie są konsumowane w trakcie używania

Wykorzystanie zasobów

- Jeśli system zawiera ρ typów zasobów, oznacza to, że:
  - istnieje ρ różnych typów zasobów wielokrotnego użytku
  - istnieje przynajmniej jedna jednostka (*unit*) każdego typu zasobu
  - jedna jednostka zasobu może być używana najwyżej przez jedną pracę na raz (dostęp wykluczający się wzajemnie, *mutually exclusive access*)
  - praca musi uzyskać jednostkę wymaganego zasobu, użyć jej, a następnie ją zwolnić
- Zasób jest *obfity* (*plentiful*), jeśli nie zdarza się aby jakakolwiek praca była blokowana z powodu niedostępności tego zasobu, innymi słowy, zasób ten jest zawsze dostępny
  - obfity nie oznacza nieskończony
  - jeśli wiemy, że dany zasób nie wpływa na czas wykonania pracy to możemy go pominąć

Ograniczenia czasowe

Czas uwolnienia : moment, w kórym praca staje się gotowa do wykonania (*release time*)
Czas ostateczny : moment, w którym wykonanie pracy musi zostać ukończone (*deadline*)
Czas wykonania : sumaryczny czas spędzony przez pracę na procesorze (*execution time*)
Czas reakcji : czas od momentu uwolnienia pracy do chwili zakończenia (*response time*)
Względny czas ostateczny : maksymalny dopuszczalny czas reakcji (*relative deadline*)
Bezwzględny czas ostateczny : czas uwolnienia + względny czas ostateczny (*absolute deadline*)

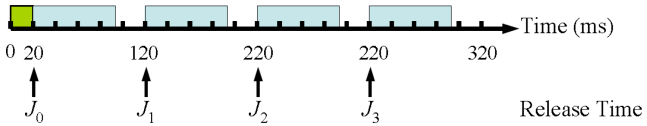


Time constraints

Ograniczenia czasowe - przykład cz. 1

- System do monitorowania i sterowania piecem grzewczym
- System potrzebuje 20ms, aby zainicjować się po załączeniu
- Po inicjalizacji, co 100ms, system:
  - sampluje i odczytuje dane z czujnika temperatury
  - przetwarza odczyty temperatury obliczając regułę sterowania
  - określa prawidłowe natężenia przepływu paliwa, powietrza i chłodziwa
- Fakt, że reguła sterowania jest obliczana okresowo można wyrazić w terminach czasów uwolnienia prac $J_0, J_1, \ldots, J_k, \ldots$ obliczających tę regułę:

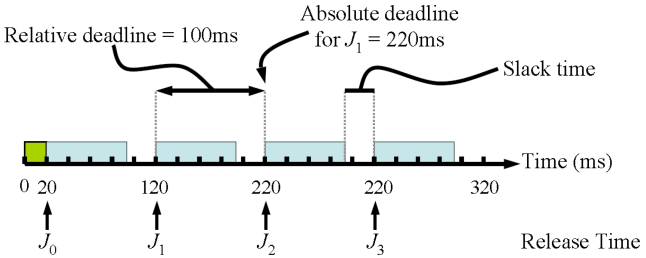
$$r_k = 20 + 100k \quad \text{(ms)}$$



Ograniczenia czasowe - przykład cz. 2

- Każda praca musi zostać ukończona przed uwolnieniem następnej pracy:
  - względny czas ostateczny pracy J_k wynosi: $D_k = 100$ ms
  - bezwzględny czas ostateczny pracy J_k wynosi $d_k = 20 + 100(k + 1)$ ms
- Zwykle wymaga się, aby każde obliczenie reguły sterowania zakończyło się wcześniej, to znaczy względny czas ostateczny powinien być krótszy niż czas między kolejnymi pracami, co daje pewien zapas czasu (*slack time*) dla innych zadań

$$\text{slack time} = \text{relative deadline - execution time}$$



```
> FIN < ACK < FIN > ACK

init 5
```