

Zakleszczenie

Typy żądań

Wyróżniamy dwa typy żądań, które mogą być wygenerowane przez każde zadanie P_j

❖ **żądanie przydziału dodatkowych zasobów** (ang. request for resource allocation),



gdzie

jest liczbą jednostek zasobu R_k żądanych dodatkowo przez P_j

❖ **żądanie zwolnienia zasobu** (ang. request for resource release),



gdzie

jest liczbą jednostek zasobu R_k zwalnianych przez P_j

Zakleszczenie

Rozważmy system składający się z n procesów (zadań) P_1, P_2, \dots, P_n współdzielący s zasobów **nieprzywłaszczalnych** tzn. zasobów, których zwolnienie może nastąpić jedynie z inicjatywy zadania dysponującego zasobem. Każdy zasób k składa się z m_k jednostek dla $k = 1, 2, \dots, s$. Jednostki zasobów tego samego typu są równoważne. Każda jednostka w każdej chwili może być przydzielona tylko do jednego zadania, czyli dostęp do nich jest wyłączny.

W każdej chwili zadanie P_j jest scharakteryzowane przez:

□ **wektor maksymalnych żądań** (ang. claims),

oznaczający maksymalne żądanie zasobowe zadania P_j w dowolnej chwili czasu

□ **wektor aktualnego przydziału** (ang. current allocations),

□ **wektor rang** zdefiniowany jako różnica między wektorami C i A ,



Zadanie przebywające w systemie

Łatwo wykazać:

Oczywiście żądanie przydziału dodatkowego zasobu może być spełnione tylko wówczas gdy:

Przez **zadanie przebywające w systemie** rozumiemy zadanie, któremu przydzielono co najmniej jedną jednostkę zasobu. Stan systemu jest zdefiniowany przez stan przydziału zasobu wszystkim zadaniom. Mówimy, że stan jest **realizowalny** jeżeli jest spełniona następująca zależność:



Zakleszczenie (2)



Zakładamy, że jeżeli żądania zadania przydziału zasobów są spełnione w skończonym czasie, to zadanie to zakończy się w skończonym czasie i zwolni wszystkie przydzielone mu zasoby. Na podstawie liczby zasobów w systemie oraz wektorów aktualnego przydziału można wyznaczyć wektor zasobów wolnych f , gdzie

gdzie

Stan bezpieczny

Stan systemu nazywamy **stanem bezpiecznym** (ang. safe) ze względu na zakleszczenie, jeżeli istnieje sekwencja wykonywania zadań przebywających w systemie oznaczona $\{P^1, P^2, \dots, P^n\}$ i nazywana **sekwencją bezpieczną**, spełniającą następującą zależność:



W przeciwnym razie, tzn. jeżeli sekwencja taka nie istnieje, stan jest nazywany **stanem niebezpiecznym**. Innymi słowy, stan jest bezpieczny jeżeli istnieje takie uporządkowanie wykonywania zadań, że wszystkie zadania przebywające w systemie zostaną zakończone. Powiemy, że **transycja** stanu systemu wynikająca z alokacji zasobów jest **bezpieczna**, jeżeli stan końcowy jest stanem bezpiecznym.

Zakleszczenie – definicja

Przez **zakleszczenie** (ang. deadlock) rozumiemy formalnie stan systemu, w którym spełniany jest następujący warunek:

gdzie Ω jest zbiorem indeksów (lub zbiorem zadań)

Mówimy, że system jest w **stanie zakleszczenia** (w systemie wystąpił **stan zakleszczenia**), jeżeli istnieje niepusty zbiór Ω zadań, które żądają przydziału dodatkowych zasobów nieprzywłaszczalnych będących aktualnie w dyspozycji innych zadań tego zbioru.

Innymi słowy, system jest w **stanie zakleszczenia**, jeżeli istnieje niepusty zbiór Ω zadań, których żądania przydziału dodatkowych zasobów nieprzywłaszczalnych nie mogą być spełnione nawet jeśli wszystkie zadania nie należące do Ω zwolnią wszystkie zajmowane zasoby.

Jeżeli $\Omega \neq \Phi$, to zbiór ten nazywamy **zbiorem zadań zakleszczonych**.

(c) Zakład Systemów Informatycznych

Slajd 8

Warunki konieczne wystąpienia zakleszczenia

Warunkami koniecznymi wystąpienia zakleszczenia są:

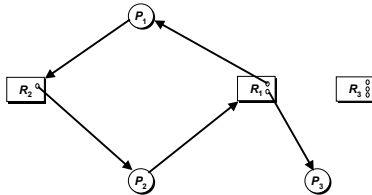
1. **Wzajemne wykluczenie** (ang. mutual exclusion condition),
W każdej chwili zasób może być przydzielony co najwyżej jednemu zadaniu.
2. **Zachowywanie zasobu** (ang. wait for condition),
Proces oczekujący na przydzielenie dodatkowych zasobów nie zwalnia zasobów będących aktualnie w jego dyspozycji.
3. **Nieprzywłaszczalność** (ang. non preemption condition),
Zasoby są nieprzywłaszczalne tzn. ich zwolnienie może być zainicjowane jedynie przez proces dysponujący w danej chwili zasobem.
4. **Istnienie cyklu oczekiwania** (ang. circular wait condition),
Występuje pewien cykl procesów z których każdy ubiega się o przydział dodatkowych zasobów będących w dyspozycji kolejnego procesu w cyklu.



(c) Zakład Systemów Informatycznych

Slajd 11

Graf alokacji zasobów



Legenda:

- P_i proces P_i
- R_j zasób R_j posiadający 3 jednostki w systemie
- $P_i \rightarrow R_j$ proces P_i posiadający jednostkę zasobu R_j
- $P_i \leftarrow R_j$ proces P_i żądający jednostki zasobu R_j



(c) Zakład Systemów Informatycznych

Slajd 9

Przeciwdziałanie zakleszczeniom



- ✓ **Konstrukcje systemów immanentnie wolnych od zakleszczenia** (ang. construction of deadlock free systems)

Podejście to polega w ogólności na wyposażeniu systemu w taką liczbę zasobów, aby wszystkie możliwe żądania zasobowe były możliwe do zrealizowania. Przykładowo, uzyskuje się to, gdy liczba zasobów każdego rodzaju jest nie mniejsza od sumy wszystkich maksymalnych i możliwych jednocześnie żądań.

- ✓ **Detekcja zakleszczenia i otwieranie stanu wolnego od zakleszczenia** (ang. detection and recovery).

W podejściu detekcji i otwierania, stan systemu jest periodycznie sprawdzany i jeśli wykryty zostanie stan zakleszczenia, system podejmuje specjalne akcje w celu odwrócenia stanu wolnego do zakleszczenia.

- ✓ **Unikanie zakleszczenia** (ang. avoidance).

W podejściu tym zakłada się znajomość maksymalnych żądań zasobowych. Każda potencjalna transycja stanu jest sprawdzana i jeśli jej wykonanie prowadziłoby do stanu niebezpiecznego, to żądanie zasobowe nie jest w danej chwili realizowane.

- ✓ **Zapobieganie zakleszczeniu** (ang. prevention)

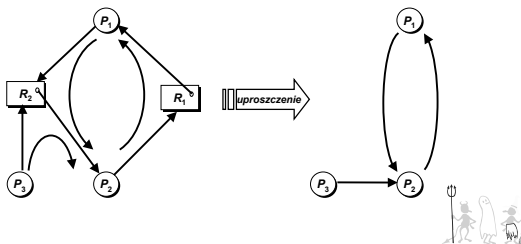
W ogólności podejście to polega na wyeliminowaniu możliwości zajścia jednego z warunków koniecznych zakleszczenia

(c) Zakład Systemów Informatycznych

Slajd 12

Grafy oczekiwania

Z grafu alokacji zasobów można uzyskać graf uproszczony przez usunięcie węzłów zasobowych i złączenie odpowiednich krawędzi. To uproszczenie wynika z obserwacji, że zasób może być jednoznacznie identyfikowany przez będącego właściciela. Ten uproszczony graf jest nazywany **grafem oczekiwania** (ang. wait-for-graph).



(c) Zakład Systemów Informatycznych

Slajd 10

Detekcja zakleszczenia

Algorytm Habermanna

1. Zainicjuj $D := \{1, 2, \dots, n\}$ i f;
2. Szukaj zadania o indeksie $j \in D$ takiego, że
 $\rho^s(P_j) \leq f$
3. Jeżeli zadanie takie nie istnieje, to zbiór zadań odpowiadający zbiorowi D jest zbiorem zadań zakleszczonych. Zakończ wykonywanie algorytmu.
4. W przeciwnym razie, podstaw:
 $D := D - \{j\}$; $f := f + A(P_j)$
5. Jeżeli $D = \emptyset$, to zakończ wykonywanie algorytmu. W przeciwnym razie przejdź do kroku 2.



(c) Zakład Systemów Informatycznych

Slajd 13

Odtwarzanie stanu

Algorytm Holt'a:

```

1. begin
2. initialize:  $I_k=1, k=1,2,\dots,s;$ 
                $C_i=s, i=1,2,\dots,n; C_0=n;$ 
3. LS:  $Y:=False;$ 
4. for  $k = 1$  step 1 until  $s$  do
5.   begin
6.     while  $E_{i,k,tk} \leq f_k \wedge I_k \leq n$  do
7.       begin
8.          $C_{E_{i,k,tk}} := C_{E_{i,k,tk}} - 1;$ 
9.          $I_k := I_k - 1;$ 
10.        if  $C_{E_{i,k,tk}} = 0$  then
11.          begin
12.             $C_0 := C_0 - 1;$ 
13.             $Y := True;$ 
14.            for  $i = 1$  step 1 until  $s$  do
15.               $f_i := f_i + A_i(P_{E_{i,k,tk}});$ 
16.            end;
17.          end;
18.        end;
19.      if  $Y = true$   $C_0 > 0$  then go to LS;
20.      if  $Y = true$  then answer "no"
21.      else answer "yes";
22.    end.

```

(c) Zakład Systemów Informatycznych

Slajd 14

Spośród zadań zakleszczonych wybierz zadanie (zadania), którego usunięcie spowoduje osiągnięcie stanu wolnego od zakleszczenia najmniejszym kosztem.



Wady i zalety podejścia unikania



- ❑ Duży narzut czasowy wynikający z konieczności wykonywania algorytmu unikania przy każdym żądaniu przydziału dodatkowego zasobu i przy każdym żądaniu zwolnienia zasobu.
- ❑ Mało realistyczne założenie o znajomości maksymalnych żądań zasobów.
- ❑ Założenie, że liczba zasobów w systemie nie może maleć



- ❑ Potencjalnie wyższy stopień wykorzystania zasobów niż w podejściu zapobiegania.



(c) Zakład Systemów Informatycznych

Slajd 17

Wady i zalety podejścia detekcji do odtwarzania stanu



- ❑ Narzut wynikający z opóźnionego wykrycia stanu zakleszczenia
- ❑ Narzut czasowy algorytmu detekcji i odtwarzania stanu
- ❑ Utrata efektów dotychczasowego przetwarzania odrzuconego zadania.



- ❑ Brak ograniczeń na współbieżność wykonywania zadań
- ❑ Wysoki stopień wykorzystania zasobów
- ❑ Podejście unikania



(c) Zakład Systemów Informatycznych

Slajd 15

Podejście zapobiegania

Rozwiązania wykluczające możliwość wystąpienia cyklu żądań.

Algorytm wstępnego przydziału

1. Przydziel w chwili początkowej wszystkie wymagane do realizacji zadania zasoby lub nie przydzielaj żadnego z nich.

Algorytm przydziału zasobów uporządkowanych

1. Uporządkuj jednoznacznie zbiór zasobów.
2. Narzuć zadaniom ograniczenie na żądania przydziału zasobów, polegające na możliwości żądania zasobów tylko zgodnie z uporządkowaniem zasobów



Przykładowo, proces może żądać kolejno zasobów 1, 2, 3, 6, ... , natomiast nie może żądać zasobu 3 a później 2. Jeśli więc z kontekstu programu wynika kolejność żądań inna niż narzucony porządek, to proces musi zażądać wstępnej alokacji zasobów, generując na przykład żądanie przydział zasobów 2 i 3.

(c) Zakład Systemów Informatycznych

Slajd 18

Algorytm podejścia unikania



1. Za każdym razem, gdy wystąpi żądanie przydziału dodatkowego zasobu, sprawdź bezpieczeństwo tranzycji stanu odpowiadającej realizacji tego żądania. Jeśli tranzycja ta jest bezpieczna, to przydziel żądany zasób i kontynuuj wykonywanie zadania. W przeciwnym razie zawieś wykonywanie zadania.
2. Za każdym razem, gdy wystąpi żądanie zwolnienia zasobu, zrealizuj to żądanie i przejrzyj zbiór zadań zawieszonych w celu znalezienia zadania, którego tranzycja z nowego stanu odpowiadałaby tranzycji bezpiecznej. Jeśli takie zadanie istnieje, zrealizuj jego żądanie przydziału zasobów

(c) Zakład Systemów Informatycznych

Slajd 16

Algorytmy Wait-Die i Wound-Wait

Algorytm Wait-Die

Rozwiązanie negujące zachowywanie zasobów (ang. wait for condition)

1. Uporządkuj jednoznacznie zbiór zadań według etykiet czasowych.
2. Jeżeli zadanie P_1 , będące w konflikcie z zadaniem P_2 , jest starsze (ma mniejszą etykietę czasową), to P_1 czeka (wait) na zwolnienie zasobu przez P_2 . W przeciwnym razie zadanie P_1 jest w całości odrzucane (abort) i zwalnia wszystkie posiadane zasoby.

Algorytm Wound-Wait

Rozwiązanie dopuszczające przywłaszczalność

1. Uporządkuj jednoznacznie zbiór zadań według etykiet czasowych .
2. Jeżeli zadanie P_1 , będące w konflikcie z zadaniem P_2 , jest starsze (ma mniejszą etykietę czasową), to zadanie P_2 odrzucane (abort) i zwalnia wszystkie posiadane zasoby. W przeciwnym razie P_1 czeka (wait) na zwolnienie zasobu przez P_2 .



(c) Zakład Systemów Informatycznych

Slajd 19

Wady i zalety podejścia zapobiegania



❑ Ograniczony stopień wykorzystania zasobów.



❑ Prostota i mały narzut czasowy.

