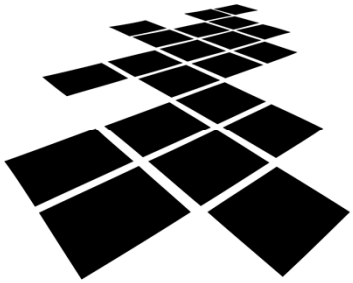
Systemy operacyjne

Przeciwdziałanie zakleszczeniu

Wykład prowadzą:
Jerzy Brzeziński
Dariusz Wawrzyniak



UCZELNIA
ONLINE

Zasadniczo można wyróżnić dwa rodzaje podejść do rozwiązania problemu zakleszczenia. Jedno polega na niedopuszczeniu do powstania zakleszczenia, drugie dopuszcza zakleszczenie, ale jego istotą jest wykrywanie (detekcja, ang. deadlock detection) i usuwanie tego stanu. Niedopuszczenie do zakleszczenia sprowadza się do zapobiegania (ang. deadlock prevention) lub unikania (ang. deadlock avoidance). Zapobieganie jest metodą dość zachowawczą, polegającą na przeciwdziałaniu zajściu jednego z warunków koniecznych wystąpienia zakleszczenia. Unikanie jest z kolei metodą pośrednią pomiędzy zapobieganiem a detekcją. Jej istotą jest przewidywanie przyszłych zdarzeń w systemie i sprawdzanie, czy w osiągalnych stanach występuje zakleszczenie. Stosuje się przy tym takie same metody, jak w przypadku wykrywania. Jeśli stan zakleszczenia jest osiągalny, to mamy do czynienie ze stanem niebezpiecznym (stanem zagrożenia), którego należy unikać, realizując żądania procesów. Przewidywanie przyszłych zdarzeń wymaga jednak pewnych przesłanek. Najczęściej muszą być znane maksymalne potencjalne potrzeby zasobowe współpracujących procesów.

Systemy operacyjne

Plan wykładu

- Wykrywanie zakleszczenia
- Usuwanie zakleszczenia
- Unikanie zakleszczeń
- Zapobieganie zakleszczeniom

Przeciwdziałanie zakleszczeniu (2)

Algorytmy wykrywania zakleszczenia, omawiane w tym module, opierają się na dwóch reprezentacjach stanu systemu, macierzowej i grafowej. Reprezentacja w postaci grafu przydziału zasobów została już omówiona w poprzednim module. Tutaj zostanie jedynie omówiona reprezentacja uproszczona w postaci grafu oczekiwania. Zostanie też omówiona reprezentacja macierzowa, oparta na wektorach i macierzach, zorientowana przede wszystkim na aspekty ilościowe w dostępności procesów do zasobów.

Systemy operacyjne

UCZELNIA ONLINE

Podejścia do zakleszczenia w przypadku zasobów odzyskiwalnych

- Zignorowanie problemu — zakleszczenie traktowane jest jako awaria systemu.
- Zapobieganie zakleszczeniom — przeciwdziałanie powstaniu któregoś z warunków koniecznych.
- Unikanie zakleszczeń — utrzymywanie rezerwy wolnych zasobów, umożliwiających bezpieczne zakończenie procesów.
- Wykrywanie i likwidowanie zakleszczeń — dopuszczenie do zakleszczenia, ale wykrywanie i usuwanie takich stanów przez odzyskanie zasobów, niezbędnych do zakończenia zadań przez (niektóre) procesy.

Przeciwdziałanie zakleszczeniu (3)

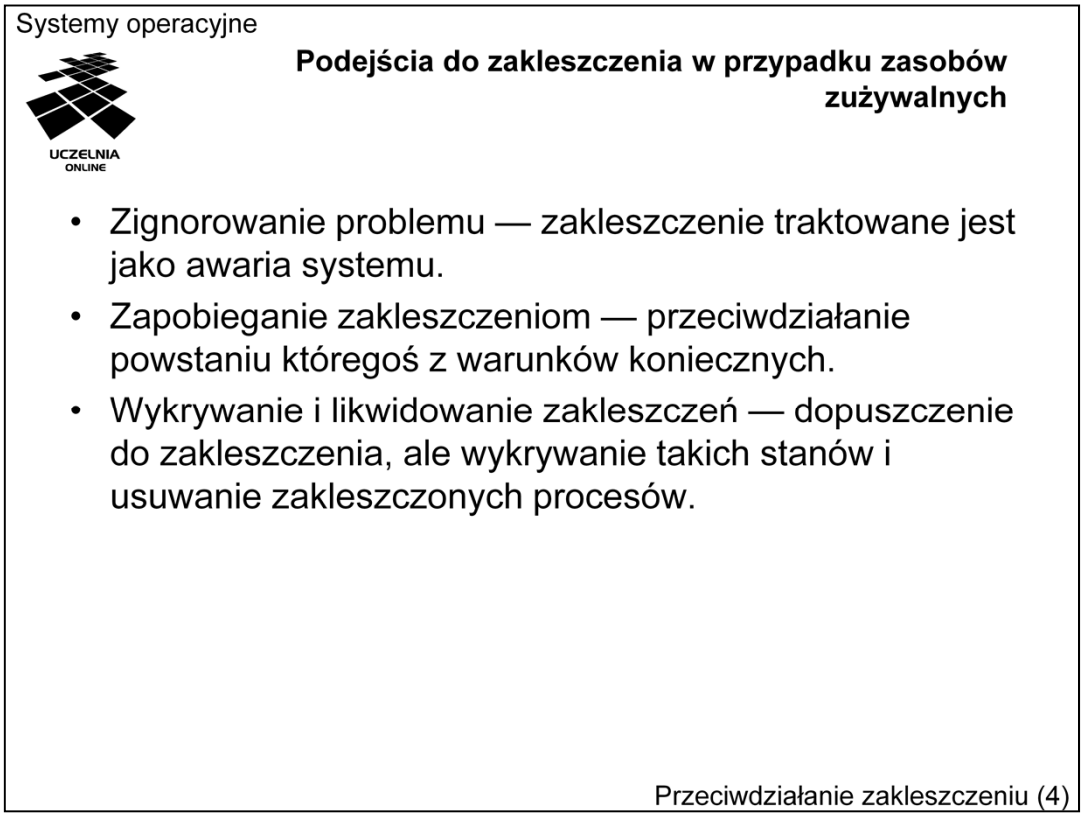
Zapobieganie zakleszczeniom polega na narzuceniu restrykcji na generowanie zamówień lub na ich realizację. Restrykcje te mają na celu zagwarantowanie, że nie będzie spełniony któryś z warunków koniecznych zakleszczenia. Może to jednak prowadzić do słabego wykorzystania zasobów i ograniczenia przepustowości systemu.

Unikanie zakleszczeń polega na takiej realizacji zamówień, żeby zawsze zagwarantować odpowiednią liczbę wolnych zasobów, niezbędnych dla zakończenia zadań. Strategia unikania zakleszczeń wymaga znajomości przyszłego zbioru zamówień procesów na zasoby. Innymi słowy, przed przydzieleniem pierwszego zasobu procesowi system musi wiedzieć, jakie zasoby będą żądane przez proces w dalszej kolejności, aż do momentu ich całkowitego zwolnienia. Oczywiście zamówienia mogą być realizowalne tylko w zakresie wcześniejszych deklaracji. Przekroczenie zadeklarowanych ograniczeń musi spowodować odrzucenie zamówienia.

Wykrywanie zakleszczeń wymaga okresowego uruchomienia odpowiedniego algorytmu. Jest to dodatkowe obciążenie dla systemu, gdyż w wielu przypadkach efektem będzie stwierdzenie braku zakleszczenia.

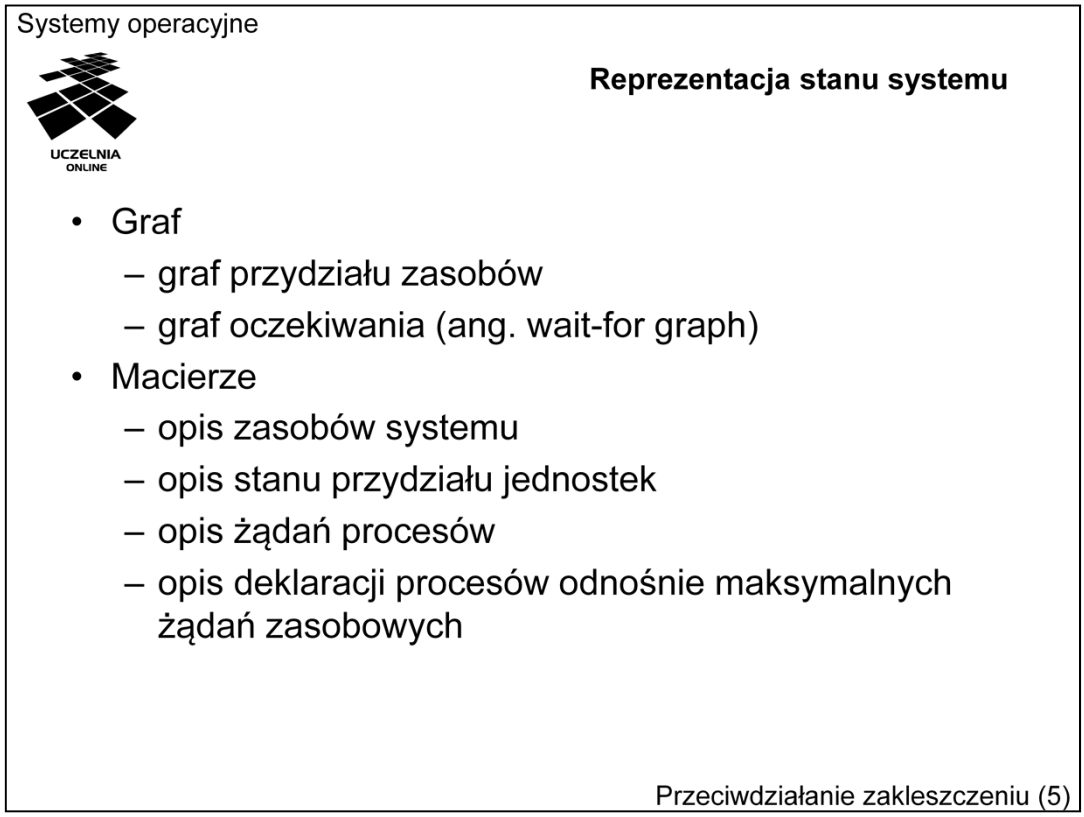
Likwidowanie zakleszczeń polega na zmuszeniu jakiegoś procesu (lub kilku procesów) do ustąpienia w rywalizacji o zasoby. Może to oznaczać:

- usunięcie procesu i zwolnienie wszystkich przydzielonych mu zasobów,
- wywłaszczenie, czyli odebranie tych zasobów, których potrzebują inne procesy.



Likwidowanie zakleszczeń w przypadku zasobów nieodzyskiwalnych polega na usuwaniu procesów. Nie można to mówić o wywłaszczaniu, ponieważ zasoby są zużywane zaraz po przydzieleniu. Z tego samego powodu, w ramach zapobiegania nie ma sensu przeciwdziałać warunkowi koniecznemu, mówiącemu o braku wywłaszczeń.

W związku z tym, że zasoby nie są zwalniane po przydzieleniu nie ma pojęcia stanu bezpiecznego, który jest podstawą unikania zakleszczenia. A więc w przypadku zasobów nieodzyskiwalnych również tego podejścia nie stosuje się.

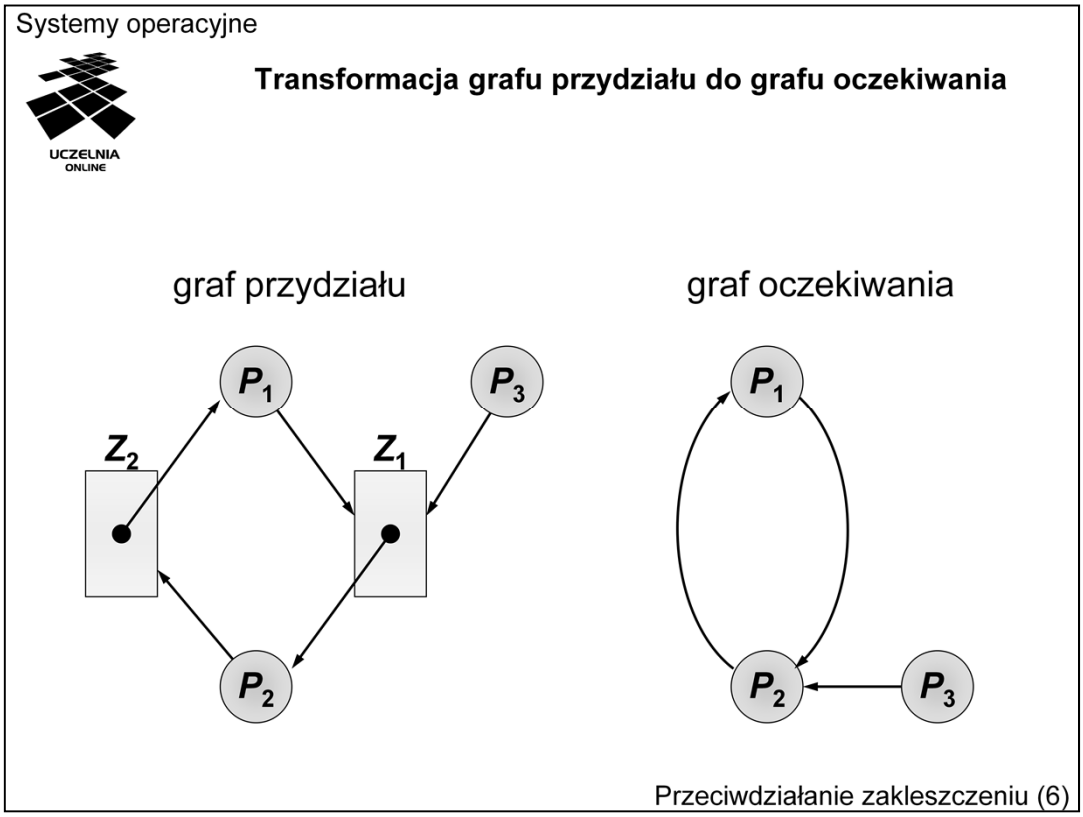


W poprzednim module została omówiona reprezentacja stanu systemu w postaci grafu przydziału zasobów. Na potrzeby detekcji zakleszczenia lub zagrożenia reprezentację tę można jednak uprościć do grafu *oczekiwania*, w którym występują tylko procesy, a skierowane krawędzie wskazują, które procesy od których oczekują zwolnienia jednostek zasobów.

Reprezentacja macierzowa opiera się na wektorach i macierzach reprezentujących stan zasobów oraz żądania procesów. W odpowiednich strukturach macierzowych pamiętane są liczby jednostek zasobów poszczególnych typów, związanych z poszczególnymi procesami. Typom zasobów odpowiadają kolumny, a procesom wiersze.

Reprezentacja macierzowa ułatwia określenie w zwartej formie pewnych relacji na liczbach jednostek zasobów. Może to poprawić czytelność kodu programu, aczkolwiek w praktyce macierze, opisujące stan systemu, mogą zawierać dużo pustych miejsc (tzw. macierze rzadkie). Reprezentacja takiej macierzy w postaci tablicy oznacza spore marnotrawstwo pamięci.

Uogólniając, macierze lepiej nadają się do wyrażania zależności ilościowych, co jest szczególnie przydatne przy analizie zasobów odzyskiwalnych. Zależności pomiędzy konkretnymi procesami natomiast łatwiej odczytać z grafu, co z kolei częściej przydaje się przy analizie zasobów nieodzyskiwalnych.



Z grafu przydziału zasobów można uzyskać graf oczekiwania przez usunięcie wierzchołków zasobowych i złączenie odpowiednich krawędzi. Jeśli zatem z usuwanego wierzchołka wychodzi krawędź przydziału lub produkcji, a dochodzi krawędź zamówienia, w grafie oczekiwania pojawi się krawędź skierowana od procesu zamawiającego do procesu przetrzymującego (lub produkującego) jednostkę zasobu.

Uproszenie grafu przydziału do grafu oczekiwania ułatwia bezpośrednie zastosowanie znanych algorytmów wykrywania cykli lub supłów (węzłów, zatok) w grafie.

Systemy operacyjne

Grafowa reprezentacja stanu — wykrywanie zakleszczenia

- Wykrycie zakleszczenia polega na stwierdzeniu — w zależności od charakterystyki zasobów oraz zamówień procesów — **cyklu** lub **supła** w grafie oczekiwania.
- Zależności pomiędzy własnościami grafu oczekiwania a stanem zakleszczenia są takie, jak zostało to określone dla grafu przydziału zasobów.
- Podejście bazujące na grafowej reprezentacji stanu systemu jest ograniczone do szczególnych przypadków, wyszczególnionych w odniesieniu do grafu przydziału zasobów w poprzednim module.

Przeciwdziałanie zakleszczeniu (7)

Systemy operacyjne

UCZELNIA ONLINE

Macierzowa reprezentacja stanu systemu

- C — m-elementowy wektor liczebności zasobów systemu
  $C[j]$ — całkowita liczba jednostek zasobu Z_j, zarządzanych przez system
- R — macierz $n \times m$ zamówień procesów
  $R[i,j]$ — liczba jednostek zasobu Z_j zamówiona i oczekiwana przez proces P_i
- A — macierz $n \times m$ przydzielonych jednostek zasobów
  $A[i,j]$ — liczba jednostek zasobu Z_j przydzielona procesowi P_i
- F — m-elementowy wektor wolnych jednostek
  $F[j]$ — liczba jednostek zasobu Z_j pozostających w dyspozycji systemu (nie przydzielona procesom)

Przeciwdziałanie zakleszczeniu (8)

Zakres informacji, opisujących stan systemu, musi obejmować zasoby, jakimi w ogóle dysponuje system, zasoby przydzielone poszczególnym procesom, żądania zasobowe procesów, ewentualnie na potrzeby analizy bezpieczeństwa — deklaracje odnośnie maksymalnych żądań. Na tej podstawie można uzyskać inne przydatne informacje, np. liczbę wolnych jednostek zasobów poszczególnych typów.

Macierz C wynika z konfiguracji systemu i nie jest elementem bieżącego stanu.

Macierze R i A związane są bieżącym stanem systemu.

Macierz F wynika z C oraz A (jest to różnica tych dwóch macierzy).

Systemy operacyjne

UCZELNIA ONLINE

Integralność macierzowej reprezentacji stanu systemu

$$\forall_{1 \le j \le m} C[j] \ge \sum_{i=1}^{n} A[i,j]$$

$$\forall_{1 \le j \le m} F[j] = C[j] - \sum_{i=1}^{n} A[i,j]$$

$$\forall_{1 \le i \le n} \; \forall_{1 \le j \le m} A[i,j] + R[i,j] \le C[j]$$

Przeciwdziałanie zakleszczeniu (9)

Z pierwszej formuły wynika, że procesom nie można przydzielić w sumie więcej jednostek zasobów poszczególnych typów, niż liczba jednostek, zarządzana przez system.

Druga formuła właściwie definiuje wektor wolnych jednostek. Wszystkie jednostki, nie będące w posiadaniu któregoś z procesów, są wolne.

Trzecia formuła mówi, że proces nie może żądać więcej jednostek, niż jest w dyspozycji systemu. Zatem to, co już ma przydzielone i to, co jeszcze zamawia, łącznie nie może przekroczyć liczby jednostek, którymi dysponuje system.

Systemy operacyjne

UCZELNIA ONLINE

Macierzowa reprezentacja stanu — zakleszczenie

$$\exists_{\substack{P' \subseteq P \\ P' \neq \emptyset}} \; \forall_{P_i \in P'} \; \exists_{1 \le j \le m} \; R[i,j] > F[j] + \underbrace{\sum_{P_k \notin P'} A[i,j]}_{\text{zasoby zwolnione przez nie zakleszczone procesy}}$$

Przeciwdziałanie zakleszczeniu (10)

Przedstawiona formuła oznacza, że jest jakiś podzbiór procesów — *P'*, oczekujących na zamówione zasoby, ale nawet gdyby wszystkie procesy spoza tego zbioru zakończyły działanie, to i tak dla każdego procesu ze zbioru *P'* jakiś zasób Z_j nie będzie dostępny w wystarczającej liczbie jednostek.

Systemy operacyjne

UCZELNIA ONLINE

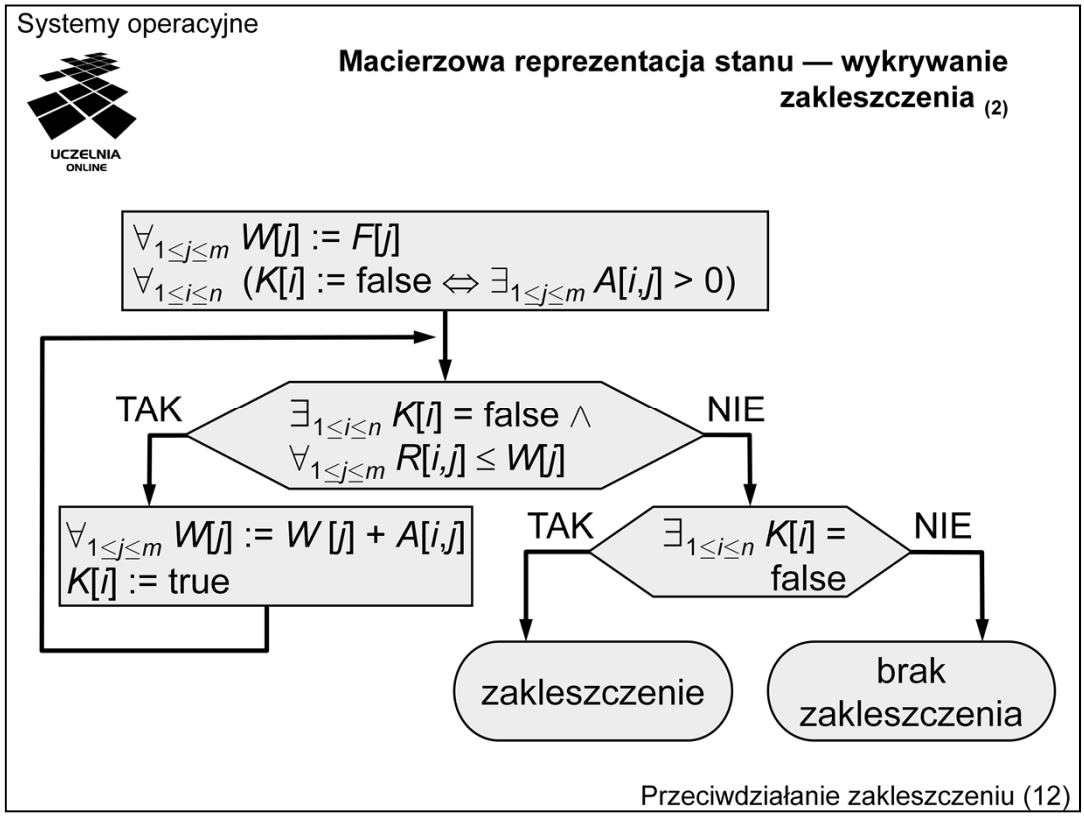
Macierzowa reprezentacja stanu — wykrywanie zakleszczenia (1)

- W — m-elementowy wektor liczby wolnych jednostek zasobów, uwzględniający jednostki zwrócone do systemu przez procesy, które mogą się zakończyć
  $W[j]$ — liczba jednostek zasobu Z_j do rozdysponowania
- K — n-elementowy wektor wartości logicznych, informujący o odzyskaniu zasobów procesu
  $K[i]$ — wartość logiczna informująca, że proces P_i zwrócił do systemu przydzielone mu jednostki zasobów

Przeciwdziałanie zakleszczeniu (11)

Zadaniem algorytmu jest przede wszystkim ustalenie, czy wystąpiło zakleszczenie. Jeśli zakleszczenie wystąpiło, można również uzyskać informację o procesach, które zostały zakleszczone. Stwierdzenie natomiast braku zakleszczenia dotyczy wyłącznie bieżącego stanu (analizowanego przez algorytm) i oznacza, że stan zakończenia przetwarzania wszystkich procesów jest osiągalny. Nie oznacza to jednak, że przy pewnej sekwencji zamówień w przyszłości i ich natychmiastowej (nie kontrolowanej) realizacji do zakleszczenia nie dojdzie.

Oprócz macierzy, opisujących stan systemu, czyli *C*, *R*, *A* i *F*, potrzebne są jeszcze zmienne pomocnicze. W miarę analizy, żądania pewnych procesów można uznać za możliwe do zrealizowania. Procesy takie uznane zostaną zatem za zakończone, co zostanie odnotowane w wektorze *K*. Jednostki zasobów, przetrzymywane przez te procesy, wrócą do systemu i zostaną uznana za wolne. Łączna liczba wolnych jednostek, uwzględniająca jednostki odzyskane po zakończonych procesach, przechowywana będzie w wektorze *W*.



Wektor *W* przechowuje informację o liczbie jednostek poszczególnych rodzajów zasobów dostępną dla procesów. Początkowo zatem jego wartość inicjalizowana jest zawartością wektora *F*. W analizie istotne są tylko te procesy, które mogą zwolnić jakieś jednostki. Wartość na pozycji, odpowiadającej tym procesom, w tablicy *K* jest false. Z kolei procesy, które w tablicy *K* mają wartość true, uznane są za zakończone lub nieistotne.

W bloku decyzyjnym poszukiwany jest proces, który może zwolnić jakieś jednostki (nie jest uznany za zakończony), a którego zamówienie może zostać zaspokojone dostępnymi w systemie jednostkami (przechowywanymi w wektorze *W*).

Jeśli taki proces się znajdzie, zostaje on uznany za zakończony, a zwalniane przez niego jednostki zasobów dołączane są do wektora *W*. Następuje powrót do bloku decyzyjnego i poszukiwanie kolejnego procesu.

Jeśli warunek (koniunkcja) w bloku decyzyjnym jest nie spełniony, to albo nie ma kolejnego procesu „do zakończenia", albo dla żadnego z procesów nie zakończonych nie ma wystarczającej liczby jednostek któregoś z zasobów. Sprawdzane jest to w kolejnym bloku decyzyjnym. Jeśli w wektorze *K* są wartości false, to odpowiednie procesy są nie zakończone, zatem zakleszczone: $\forall P_i$: ($K[i]$ = false $\Rightarrow$ P_i jest zakleszczony).

Jeśli wszystkie procesy udało się doprowadzić do zakończenie ($\forall P_i$: $K[i]$ = true) nie ma zakleszczenia.

Systemy operacyjne

UCZELNIA ONLINE

Przykład działania algorytmu wykrywania zakleszczenia (1)

- System dysponuje 3 typami zasobów: Z_1, Z_2, Z_3 o liczebności odpowiednio 6, 5, 4.
- W systemie współpracują 4 procesy: P_1, P_2, P_3 i P_4.
- Stan systemu:

	$A[1]$	$A[2]$	$A[3]$	$R[1]$	$R[2]$	$R[3]$
P_1	1	0	2	3	1	0
P_2	3	1	0	0	1	3
P_3	1	1	1	1	0	0
P_4	0	2	1	0	1	1

Przeciwdziałanie zakleszczeniu (13)

Jak wynika z opisu stanu, przydział nie przekracza liczby jednostek, którymi dysponuje system. Żaden z procesów nie przekracza też tej liczby w swoich żądaniach.

Systemy operacyjne

UCZELNIA ONLINE

Przykład działania algorytmu wykrywania zakleszczenia (2)

- Wolne zasoby w systemie F = [1, 1, 0].
- Zmiana wektora W oraz K w kolejnych krokach

	W[1]	W[2]	W[3]	K[1]	K[2]	K[3]	K[4]
początkowo	1	1	0	F	F	F	F
po zak. P_3	2	2	1	F	F	T	F
po zak. P_4	2	4	2	F	F	T	T

Przeciwdziałanie zakleszczeniu (14)

Roboczy wektor W początkowo przechowuje wolne jednostki poszczególnych typów zasobów. Liczba wolnych jednostek wystarczająca jest dla zrealizowania żądania procesu P_3. W przypadku pozostałych procesów brakuje jednostek któregoś zasobu. Dla procesu P_1 brakuje jednostek zasobu Z_1, a dla procesów P_2 i P_4, jednostek zasobu Z_3. Po zakończeniu procesu P_3 przydzielone mu jednostki trafiają do systemu i do dyspozycji zarządcy są odpowiednio 2, 2, 1 jednostki. Taka liczba jest wystarczająca dla procesu P_4, ale uzyskany po jego zakończeniu stan wolnych jednostek (odpowiednio 2, 4, 2) nie wystarczy ani dla procesu P_1 (z mało jednostek zasobu Z_1), ani dla procesu P_2 (za mało jednostek zasobu Z_3). Procesy P_1 i P_2 są więc zakleszczone.

Systemy operacyjne

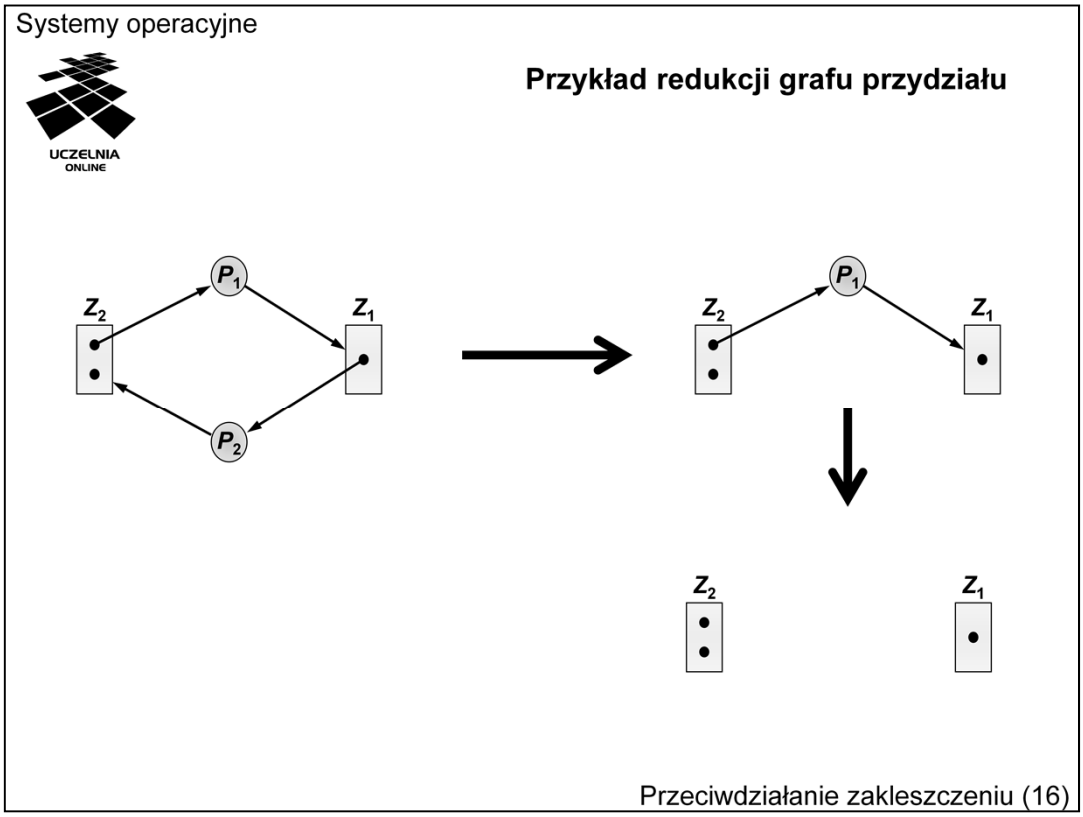
Redukcja grafu przydziału

1. Jeśli nie istnieje taki proces P_i, którego żądania zasobowe mogą zostać zaspokojone przez dostępne jednostki zasobów, przejście do punktu 5.
2. Usunięcie wierzchołka procesu P_i, wraz z wszystkimi jego krawędziami, jeśli żądania można zrealizować.
3. Zwolnienie wszystkich jednostek zasobów odzyskiwalnych, przetrzymywanych przez proces P_i oraz utworzenie odpowiedniej liczby jednostek zasobów nieodzyskiwalnych, których producentem jest P_i.
4. Przejście do punktu 1
5. Jeśli pozostały nie usunięte procesy, to są one zakleszczone.

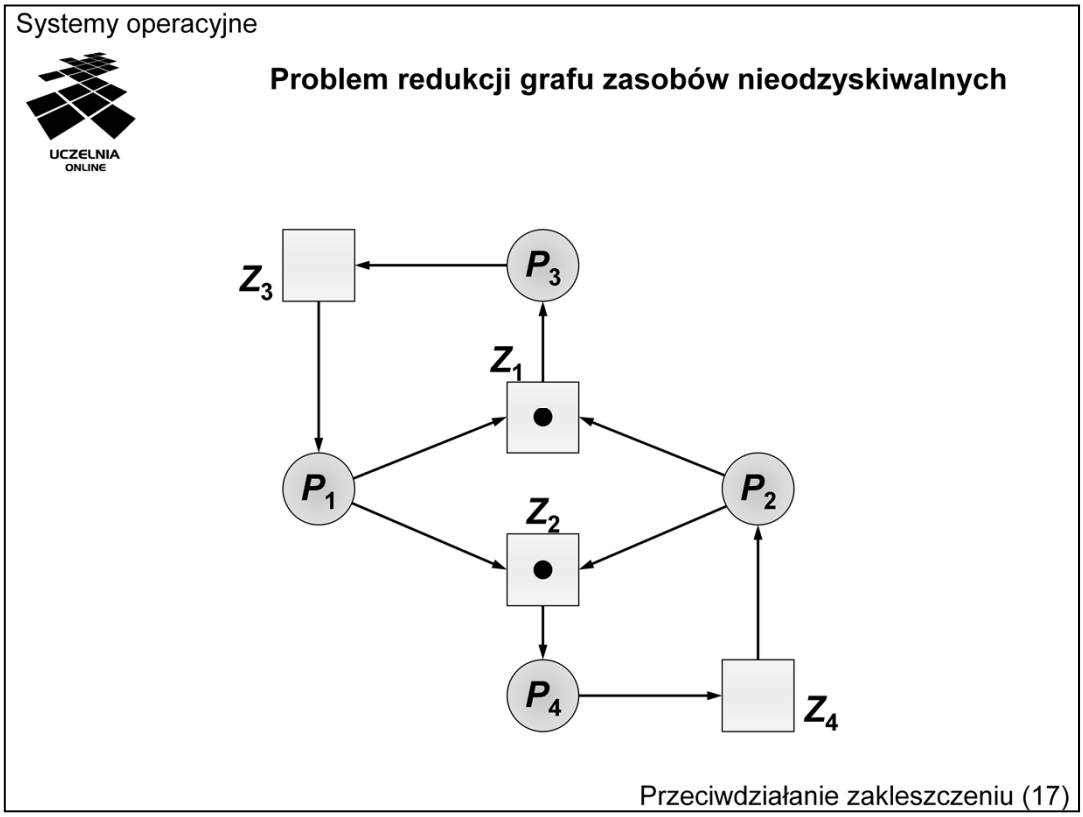
Przeciwdziałanie zakleszczeniu (15)

Redukcja grafu przydziału jest metodą uniwersalną — nadaje się do stosowania w przypadku zasobów odzyskiwalnych i nieodzyskiwalnych, jak również w systemach, gdzie współistnieją zasoby obu tych klas. Redukowalność grafu jest też używana czasami w definicjach stanu zakleszczenia.

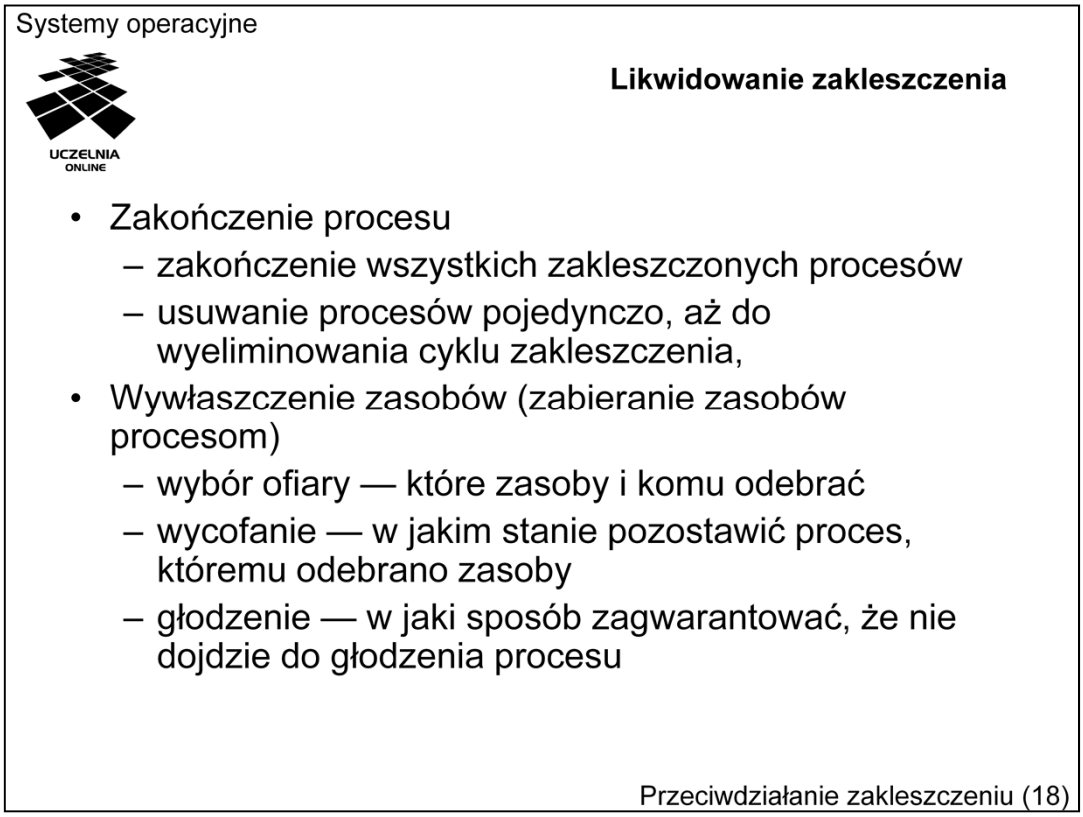
Redukcja polega na usuwaniu wierzchołków procesów wraz z incydentnymi krawędziami w tych przypadkach, w których żądania procesu można zrealizować. Usunięcie wierzchołka procesu oznacza zwolnienie jednostek zasobów odzyskiwalnych i utworzenie w odpowiedniej liczbie jednostek zasobów nieodzyskiwalnych.



Przykład pokazuje redukcję grafu zasobów odzyskiwalnych. Zamówienie procesu P_2 może być zrealizowane, więc odpowiedni wierzchołek ulega redukcji. Po zredukowaniu P_2, jednostka zasobu Z_1 staje się wolna i można zredukować P_1. W wyniku redukcji nie pozostał żaden proces, więc nie ma zakleszczenia.



Jak już wspomniano w poprzednim module, problem sprawiają czasami zasoby nieodzyskiwalne. Przykład powyższy był już analizowany, a wynikiem była niejednoznaczność co do zakleszczonych procesów. Interpretacja wyniku redukcji nastręcza tych samych problemów. Redukując przez P_1 nie da się zredukować procesów P_2 i P_4, a redukując przez P_2 nie da się zredukować P_1 i P_3.



Likwidacja zakleszczenia polega na powiększeniu przydziału zasobowego pewnych procesów, kosztem innych. W ten sposób przynajmniej część procesów uda się zakończyć. W celu odzyskania zasobów, przetrzymujące je procesy można usunąć z systemu. Tracony jest wówczas cały efekt dotychczasowego przetwarzania i to jest zasadniczy koszt tego podejścia. Najlepiej usunąć procesy, które tworzą cykl. Sam wybór procesu może być uzależniony od czasu działania w systemie (im dłużej proces działa, tym potencjalnie więcej jest do stracenia) lub priorytetu procesu. Nie ma natomiast sensu usuwać procesów, które są wprawdzie zakleszczone, ale nie współtworzą cyklu, gdyż żaden z zakleszczonych procesów nie czeka na przetrzymywane przez nie zasoby. W skrajnym przypadku można usunąć wszystkie zakleszczone procesy.

Alternatywą dla usuwania procesów jest odbieranie zasobów. Z odbieraniem zasobów wiąże się problem odtworzenia stanu. Jeśli stan zasobu jest istotny dla przetwarzania procesu, należy stan ten zapamiętać, albo ponowić ciąg instrukcji zmierzających do osiągnięcia tego stanu. Zapamiętanie stanu zasobu jest mniej lub bardziej kłopotliwe w zależności od rodzaju zasobu, a dokładniej od złożoności pamięciowej takiej operacji. Nie stanowi na ogół problemu zapamiętanie stanu procesora — jest to kwestia zapamiętania od kilkudziesięciu do kilkuset bajtów. Można zachować stan pamięci fizycznej pod warunkiem dostępności przestrzeni wymiany. Znaczenie bardziej kłopotliwe może być zapamiętanie stanu urządzeń wejścia-wyjścia, gdyż efekty ich pracy, widziane są na zewnątrz, a więc wymykają się spod kontroli systemu operacyjnego.

Problemem związanym z ponownym wykonaniem pewnego fragmentu przetwarzania jest uniknięcie głodzenia, czy wręcz uwięzienia, gdyż zakleszczenie może wystąpić ponownie.

Systemy operacyjne

UCZELNIA ONLINE

Unikanie zakleszczeń

- Wymagana jest dodatkowa informacja o tym, jakie zasoby będą zamawiane przez proces.
  - W najprostszym przypadku jest to maksymalna liczba jednostek poszczególnych zasobów, niezbędna do zakończenia zadania przez proces.
- Przy każdym zamówieniu zarządca decyduje, czy można je zrealizować, czy należy wstrzymać realizację, biorąc pod uwagę aktualny stan zasobów.
  - W przypadku zadeklarowania maksymalnej liczby jednostek zasobów system musi zapewnić, że nie dojdzie do cyklu w oczekiwaniu na zasoby.

Przeciwdziałanie zakleszczeniu (19)

W podejściach polegających na unikaniu stosowane są takie same algorytmy, jak w przypadku wykrywania, ale nie w odniesieniu do stanu bieżącego, ale do stanów osiągalnych ze stanu bieżącego. Bazując na informacji o **potencjalnych** przyszłych zamówieniach, przewidywany jest najgorszy możliwy stan osiągalny systemu i analizowany pod kątem wystąpienia zakleszczenia. Jeśli nie ma zakleszczenia, stan uważany jest za bezpieczny, w przeciwnym przypadku jest to stan zagrożenia. Opisywane podejście sprowadza się właśnie do **unikania** takich stanów.

Systemy operacyjne

Stan bezpieczny

- Stan systemu jest bezpieczny, jeśli istnieje porządek przydziału zasobów żądającym tego procesom (nawet w stopniu maksymalnym), gwarantujący uniknięcie zakleszczenia.
- Formalnie: system jest w stanie bezpiecznym, jeśli istnieje ciąg bezpieczny, czyli taki ciąg procesów $\langle P_1, P_2, ..., P_n \rangle$, że w danym stanie przydziału zasobów zapotrzebowanie procesu P_i może być zaspokojone przez bieżąco dostępne zasoby oraz zasoby użytkowane przez wszystkie procesy poprzedzające go w ciągu, czyli procesy P_j, gdzie $j < i$.

Przeciwdziałanie zakleszczeniu (20)

Biorąc pod uwagę maksymalne deklaracje zasobowe poszczególnych procesów, można wyznaczyć maksymalne potrzeby zasobowe, do zaspokojenia których musi być przygotowany zarządca zasobów. Zakładając przypadek pesymistyczny, w którym pojawią się zamówienia właśnie z tymi maksymalnymi żądaniami, można stwierdzić (używając algorytmu wykrywania zakleszczenia), czy wystąpi wówczas zakleszczenie. Stwierdzenie zakleszczenia oznacza, że stan bieżący nie jest bezpieczny. W przeciwnym razie można wyznaczyć jedną lub kilka sekwencji w realizacji zamówień, w której proces wcześniejszy zwalnia po swoim zakończeniu zasoby dla procesów następnych.

Bazując na macierzowej reprezentacji stanu, można wykorzystać zaprezentowany wcześniej algorytm detekcji zakleszczenia. Jeśli stan jest bezpieczny, to kolejność, w jakiej procesy wybierane są do realizacji jest właśnie ciągiem bezpiecznym.

Systemy operacyjne

UCZELNIA ONLINE

Przykład stanu i ciągu bezpiecznego

- Procesy P_1, P_2, P_3 ubiegają się o jednostki zasobu Z_1, których łączna liczba jest 12.
- Maksymalne zapotrzebowanie oraz bieżący przydział jest następujący:

proces	max zapot.	bieżący przydział
P_1	10	5
P_2	4	2
P_3	9	2

- Czy istnieje ciąg bezpieczny?
- Czy można zrealizować żądanie przydziału 1 jednostki zasobu Z_1 procesowi P_3?

Przeciwdziałanie zakleszczeniu (21)

Z maksymalnego zapotrzebowania oraz bieżącego przydziału wynika, że proces P_1 może zażądać jeszcze 5 jednostek (deklaruje 10, a 5 już ma), proces P_2 może zażądać jeszcze 2 jednostek, a proces P_3 jeszcze 7 jednostek. Do rozdysponowania pozostały jeszcze 3 jednostki, gdyż 9 zostało przydzielonych (5+2+2). W wariancie pesymistycznym żądań zasobowych te 3 wolne jednostki wystarczą dla procesu P_2. Po zakończeniu P_2 odzyskane zostaną przydzielone mu jednostki, łącznie będzie więc 5 wolnych jednostek. Taka liczba jednostek będzie wystarczająca dla P_1, a po odzyskaniu przydzielonych mu jednostek ich liczba wzrośnie do 10. Oczywiście uwzględniając ograniczenia na wielkość zamówień, uzyskane jednostki muszą być wystarczające dla P_3. Ciąg bezpieczny to P_2, P_1, P_3.

Po zrealizowaniu żądania procesu P_3 stan systemu jest następujący:

- P_1 ma przydzielone 5 jednostek, może potrzebować jeszcze 5,
- P_2 ma przydzielone 2 jednostki, może potrzebować jeszcze 2,
- P_3 ma przydzielone 3 jednostki, może potrzebować jeszcze 3.

W systemie pozostały 2 wolne jednostki, którymi można zaspokoić potencjalne potrzeby procesu P_2. Po zakończeniu P_2 liczba jednostek do rozdysponowania zwiększy się do 4, ale może się to okazać niewystarczające zarówno dla P_1, jak i dla P_3. Gdyby któryś z nich zażądał więcej niż 4 jednostek (co jest zgodne z ich deklaracjami), doszłoby do zakleszczenia. Zrealizowanie żądania procesu P_2 wprowadza system w stan niebezpieczny. Stan ten nie oznacza jednak jeszcze zakleszczenia — jest to dopiero zagrożenie. Przy sprzyjającym zbiegu okoliczności (sprzyjającej sekwencji żądań) zakleszczenia można uniknąć.

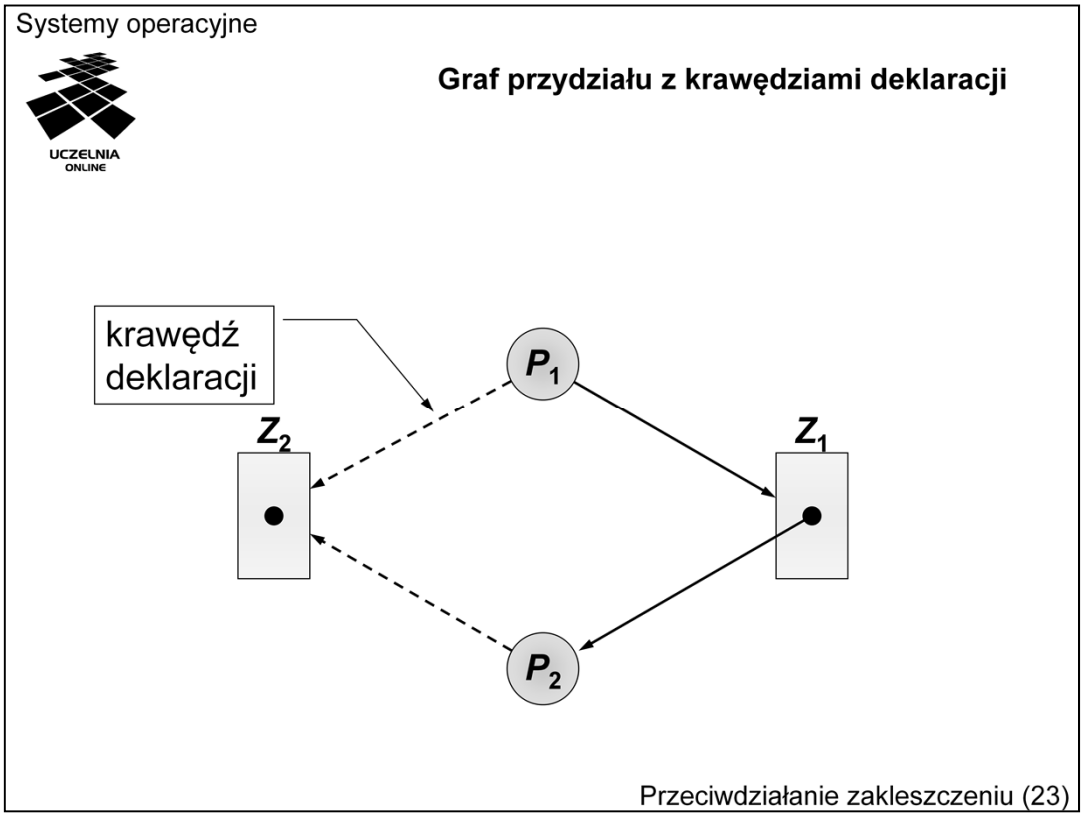
Systemy operacyjne

Grafowa reprezentacja stanu — unikanie zakleszczenia

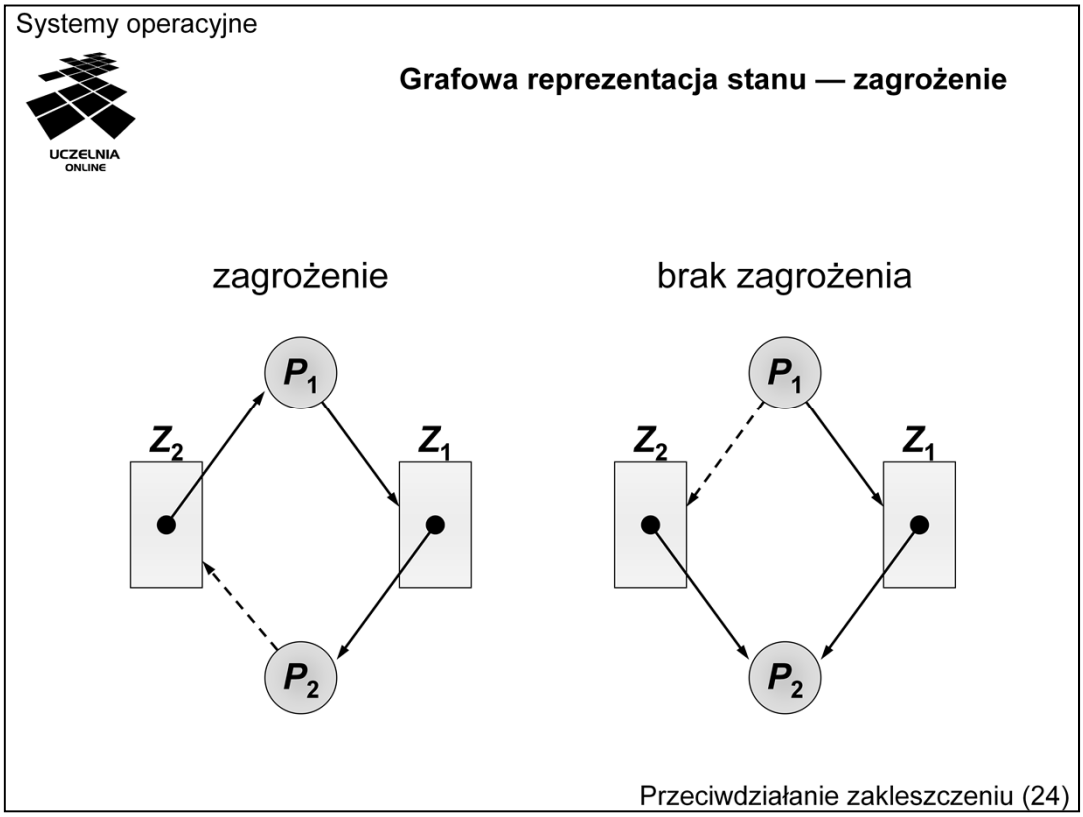
- W celu uwzględnienia potencjalnych żądań, w grafie przydziału zasobów wprowadza się dodatkową krawędź deklaracji, wskazującą, że proces może zamówić egzemplarz zasobu do realizacji zadania.
- Gdy proces zamawia zasób, krawędź deklaracji zamieniana jest na krawędź zamówienia, a gdy go zwalnia, ale się nie kończy, krawędź przydziału zamieniana jest na krawędź deklaracji.
- Zamówienie może być zrealizowane, gdy zamiana krawędzi zamówienia na krawędź przydziału nie spowoduje cyklu lub supła (zależnie od charakterystyki systemu) w grafie oczekiwania, uwzględniającym krawędzie deklaracji.

Przeciwdziałanie zakleszczeniu (22)

Stosowalność algorytmu podlega tym samym ograniczeniom, o których była mowa w przypadku algorytmu wykrywania zakleszczenia, opartym na reprezentacji grafowej. Poza tym, podejścia polegające na unikaniu stosowane są dla zasobów odzyskiwalnych.



Z grafu wynika, że zarówno proces P_1 jak i P_2 może zażądać jednostki zasobu Z_2 do realizacji przetwarzania.



Graf po lewej stronie obrazuje stan systemu po zrealizowania zamówienia procesu P_1 na jednostkę zasobu Z_2, a graf po prawej obrazuje stan systemu po zrealizowaniu takiego samego zamówienia na rzecz procesu P_2. Nie tworząc nawet grafu oczekiwania, łatwo można dostrzec cykl w stanie systemu po lewej stronie. W przypadku pojedynczych zasobów odzyskiwalnych jest to warunek konieczny i dostateczny zakleszczenia, a więc zagrożenie (na razie, ponieważ P_2 nie zażądał jeszcze jednostki zasobu Z_2).

Tego ryzyka nie ma w stanie systemu po prawej. Nawet jeśli proces P_1 zażąda jednostki zasobu Z_2, nie będzie zakleszczenia.

Systemy operacyjne

UCZELNIA ONLINE

Macierzowa reprezentacja stanu — unikanie zakleszczenia (1)

- Przed rozpoczęciem realizacji zadania proces musi zadeklarować maksymalną liczbę jednostek poszczególnych typów zasobów, których może potrzebować.
- Na podstawie deklaracji i bieżącego stanu systemu zarządca musi rozstrzygnąć, czy przydział zasobów pozostawi system w stanie bezpiecznym. Jeśli tak, to zasoby mogą zostać przydzielone, a jeśli nie, to proces musi poczekać.

Przeciwdziałanie zakleszczeniu (25)

Prezentowany algorytm — tzw. *algorytm bankiera* — jest sformalizowaną i nieco uogólnioną formą rozumowania przedstawionego we wcześniejszym przykładzie, dotyczącym ciągu bezpiecznego.

Systemy operacyjne

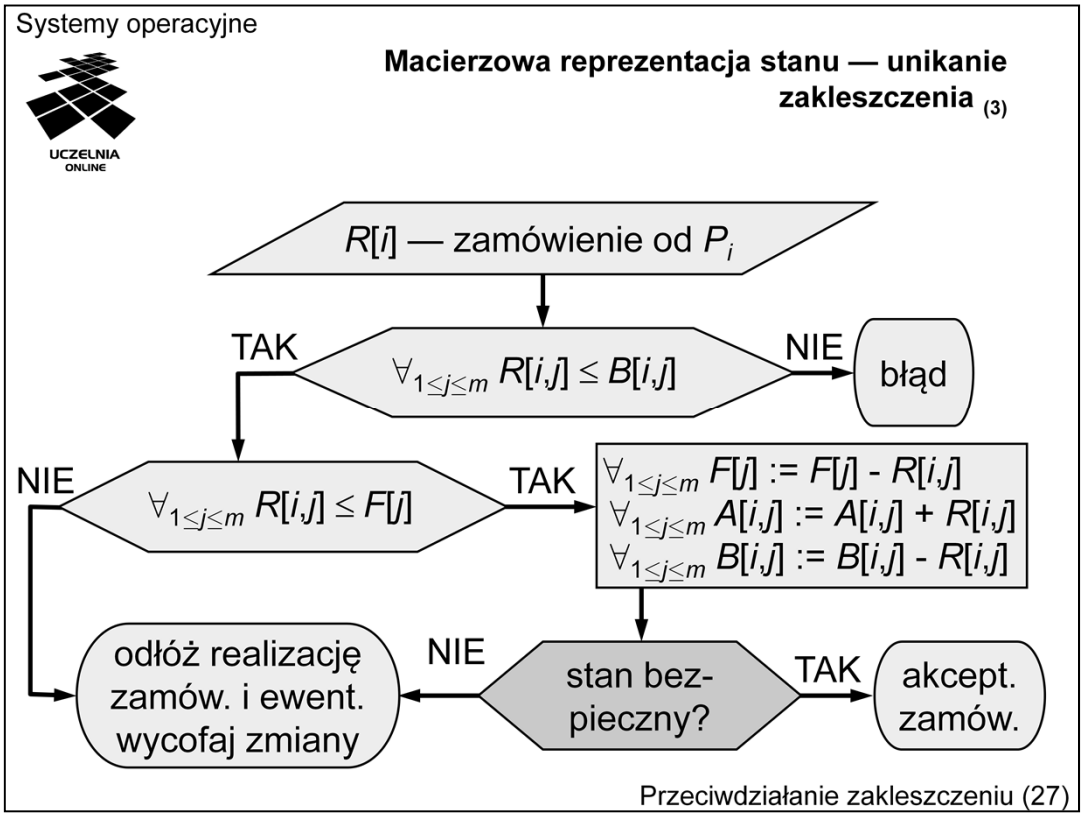
Macierzowa reprezentacja stanu — unikanie zakleszczenia (2)

- *D* — macierz o wymiarach $n \times m$ określająca maksymalne zapotrzebowanie poszczególnych procesów na poszczególne zasoby
- *B* — macierz o wymiarach $n \times m$ określająca zapotrzebowanie poszczególnych procesów na poszczególne zasoby, które jest jeszcze do zrealizowania (nie wykorzystana jeszcze część deklaracji)

Przeciwdziałanie zakleszczeniu (26)

Oprócz macierzy, opisujących stan systemu, czyli *C*, *R*, *A* i *F*, oraz pomocniczych *W* i *K*, potrzebne są jeszcze macierze, związane z maksymalnymi deklaracjami zasobowymi. Deklaracje odnośnie maksymalnego zapotrzebowania procesów na jednostki zasobów poszczególnych typów, przechowywane są w macierzy *D*.

Macierz $B = D - A$. Maksymalne potrzeby odjąć bieżący przydział daje liczbę jednostek, której zażądania zarządca może się jeszcze spodziewać.

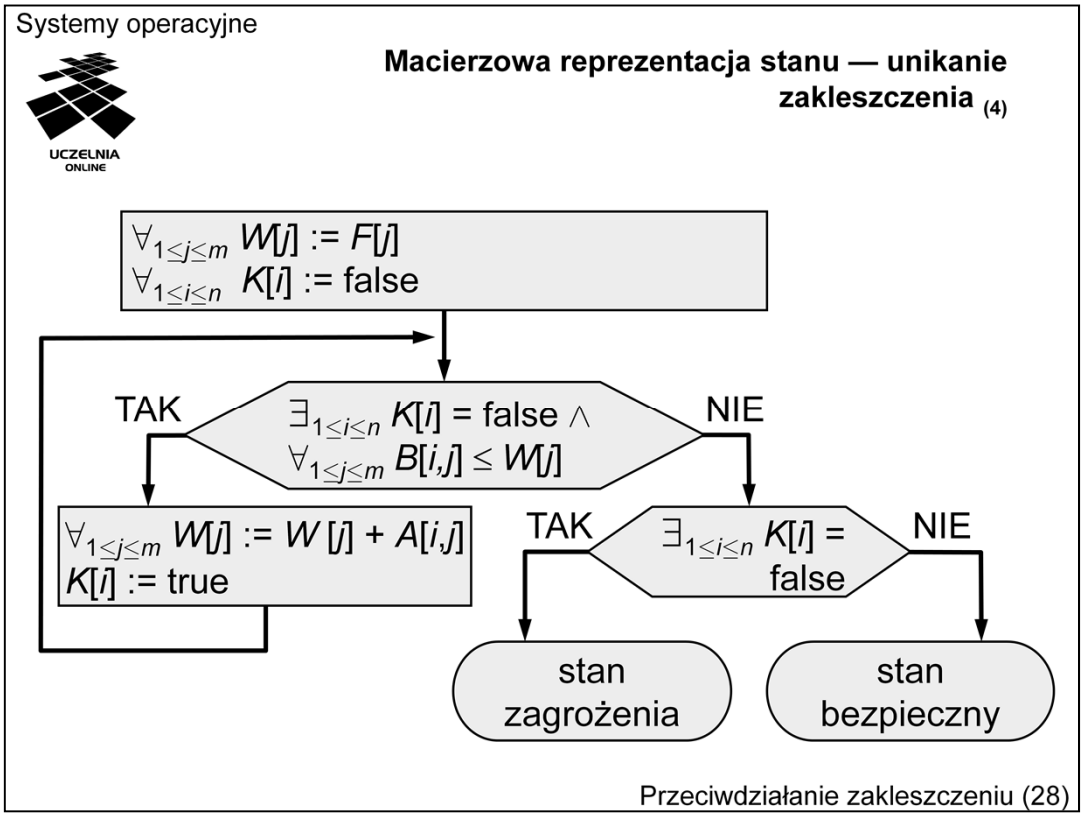


Algorytm uruchamiany jest w reakcji na złożenie zamówienia przez proces.

Najpierw następuje weryfikacja poprawności zamówienia w stosunku do deklaracji. Jeśli żądana liczba zasobów jest większa, niż nie zrealizowana część deklaracji, jest błąd przekroczenia deklarowanych ograniczeń.

Następnie sprawdzana jest dostępność żądanej liczby jednostek zasobów. Jeśli nie ma tylu jednostek, ilu żąda proces, żądanie jest oczywiście okładane do czasu zwolnienia odpowiedniej liczby jednostek przez inne procesy.

Jeśli wymagana liczba jednostek jest dostępna, następuje tymczasowa realizacja zamówienia. Jest to swego rodzaju symulacja, chociaż w prezentacji modyfikowane są wszystkie macierze, opisujące bieżący stan. Następnie sprawdza się, czy uzyskany w ten sposób stan jest bezpieczny. Jeśli jest to stan bezpieczny, zamówienie zostaje zaakceptowane (macierze są już zaktualizowane). W przypadku wykrycia zagrożenia, realizacja zamówienia jest odkładana, co wymaga wycofania zmian w opisie stanu systemu.



Sprawdzenie bezpieczeństwa polega na uruchomieniu takiego samego algorytmu, jak w przypadku detekcji zakleszczenia, przy czym zamówieniami są maksymalne potrzeby zasobowe, wynikające z deklaracji i bieżącego przydziału — czyli macierz *B*. Analizowany jest zatem przypadek skrajny, w którym wszystkie procesy oczekują realizacji swoich deklaracji w stopniu maksymalnym.

Systemy operacyjne

UCZELNIA ONLINE

Przykład działania algorytmu (1)

- System dysponuje zasobami Z_1, Z_2, po 8 jednostek.
- W systemie współpracuje 5 procesów: P_1, P_2, P_3 , P_4, P_5.
- Stan systemu:

	A[1]	A[2]	D[1]	D[2]	B[1]	B[2]
P_1	2	2	6	4	4	2
P_2	2	0	7	2	5	2
P_3	0	1	2	3	2	2
P_4	1	0	3	4	2	4
P_5	0	2	2	6	2	4

Przeciwdziałanie zakleszczeniu (29)

Wartości macierzy B wynikają z różnicy pomiędzy D i A.

System dysponuje wolnymi zasobami w liczbie: 3 jednostek zasobu Z_1 oraz 3 jednostek zasobu Z_2 (przydzielone jest 5 z 8). Taka liczba jest wystarczająca tylko do zaspokojenia maksymalnych potrzeb procesu P_3. Jeśli jednak proces P_3 się zakończy, zwolni przydzielone zasoby i liczba wolnych jednostek wzrośnie do 3, 4 (odpowiednio zasobów Z_1 i Z_2). Jak wynika z ostatniej kolumny, wystarczy to procesowi P_4 lub P_5. Po procesach P_4 i P_5 obsłużyć można proces P_1 i na końcu proces P_2. Istnieją zatem 2 ciągi bezpieczne.

Przykład działania algorytmu sprawdzania bezpieczeństwa zaprezentowano na następnym slajdzie.

Systemy operacyjne

UCZELNIA ONLINE

Przykład działania algorytmu (2)

- Wolne zasoby w systemie F = [3, 3].
- Zmiana wektora W oraz K w kolejnych krokach

	W[1]	W[2]	K[1]	K[2]	K[3]	K[4]	K[5]
początkowo	3	3	F	F	F	F	F
po zak. P_3	3	4	F	F	T	F	F
po zak. P_4	4	4	F	F	T	T	F
po zak. P_5	4	6	F	F	T	T	T
po zak. P_1	6	8	T	F	T	T	T
po zak. P_2	8	8	T	T	T	T	T

Przeciwdziałanie zakleszczeniu (30)

Roboczy wektor W początkowo przechowuje wolne jednostki poszczególnych typów zasobów, czyli 3, 3.

po zakończeniu procesu P_3: 3+0=3, 3+1=4,

po zakończeniu procesu P_4: 3+1=4, 4+0=4,

po zakończeniu procesu P_5: 4+0=4, 4+2=6,

po zakończeniu procesu P_1: 4+2=6, 6+2=8,

po zakończeniu procesu P_2: 8+2=8, 6+0=8.

System jest w stanie bezpiecznym, a ciąg bezpieczny to: P_3, P_4, P_5, P_1, P_2 (lub P_3, P_5, P_4, P_1, P_2).

Systemy operacyjne

UCZELNIA ONLINE

Przykład działania algorytmu (3)

- Stan systemu po zrealizowaniu żądania [1,0] procesu P_2:

	$A[1]$	$A[2]$	$D[1]$	$D[2]$	$B[1]$	$B[2]$
P_1	2	2	6	4	4	2
P_2	3	0	7	2	4	2
P_3	0	1	2	3	2	2
P_4	1	0	3	4	2	4
P_5	0	2	2	6	2	4

Przeciwdziałanie zakleszczeniu (31)

Tabela prezentuje stan systemu po przydzieleniu jednej jednostki zasobu Z_1 procesowi P_2. Zmieniła się liczba wolnych jednostek oraz maksymalne potrzeby procesu P_2. Zamówienie takie można zrealizować, pod warunkiem, że zaprezentowany stan jest bezpieczny.

Przykład działania algorytmu sprawdzania bezpieczeństwa zaprezentowano na następnym slajdzie.

Systemy operacyjne

UCZELNIA ONLINE

Przykład działania algorytmu (4)

- Wolne zasoby w systemie F = [2, 3].
- Zmiana wektora W oraz K w kolejnych krokach

	W[1]	W[2]	K[1]	K[2]	K[3]	K[4]	K[5]
początkowo	2	3	F	F	F	F	F
po zak. P_3	2	4	F	F	T	F	F
po zak. P_4	3	4	F	F	T	T	F
po zak. P_5	3	6	F	F	T	T	T

Przeciwdziałanie zakleszczeniu (32)

Roboczy wektor W początkowo przechowuje wolne jednostki poszczególnych typów zasobów, czyli 2, 3.

po zakończeniu procesu P_3: 2+0=2, 3+1=4,

po zakończeniu procesu P_4: 2+1=3, 4+0=4,

po zakończeniu procesu P_5: 3+0=3, 4+2=6

Dla procesów P_1 i P_2 brakuje jednostek zasobu Z_1. Stan nie jest bezpieczny. Przy maksymalnych deklarowanych żądaniach zasobowych jednostek wystarczy tylko dla procesów P_3, P_4 i P_5.

Zgodnie z zasadą unikania zakleszczenia, nie można zrealizować zamówienia procesu P_2.

Systemy operacyjne

UCZELNIA ONLINE

Zapobieganie zakleszczeniom — wzajemne wykluczanie

- Konieczność zagwarantowania wzajemnego wykluczania wynika z charakterystyki zasobu — wzajemne wykluczanie musi być zachowane w przypadku dostępu do zasobów niepodzielnych.
- Zasoby współdzielone nie wymagają dostępu w trybie wyłącznym, więc używanie zasobu przez jeden proces nie blokuje dostępu do niego innym procesom.

Przeciwdziałanie zakleszczeniu (33)

Zapobieganie, podobnie jak unikanie, jest podejściem polegającym na niedopuszczaniu do zakleszczenia. W przeciwieństwie do unikania, decyzja o akceptacji lub odłożeniu realizacji zamówienia podejmowana jest wyłącznie w oparciu o bieżący stan systemu. Nie są tu uwzględniane żadne dane na temat przyszłych zachowań procesów. Podejście takie musi być zatem bardziej zachowawcze — w większym stopniu i często nadmiarowo ograniczające aktywność procesów.

Zapobieganie polega na kontrolowaniu zaistnienia warunków koniecznych. Wystarczy nie dopuścić do spełnienia jednego z nich, a nie dojdzie do zakleszczenia.

Ze względu na sposób korzystania z niektórych zasobów nie można podjąć żadnych kroków w celu przeciwdziałania warunkowi wzajemnego wykluczania.

Systemy operacyjne

UCZELNIA ONLINE

Zapobieganie zakleszczeniom — przetrzymywanie i oczekiwanie

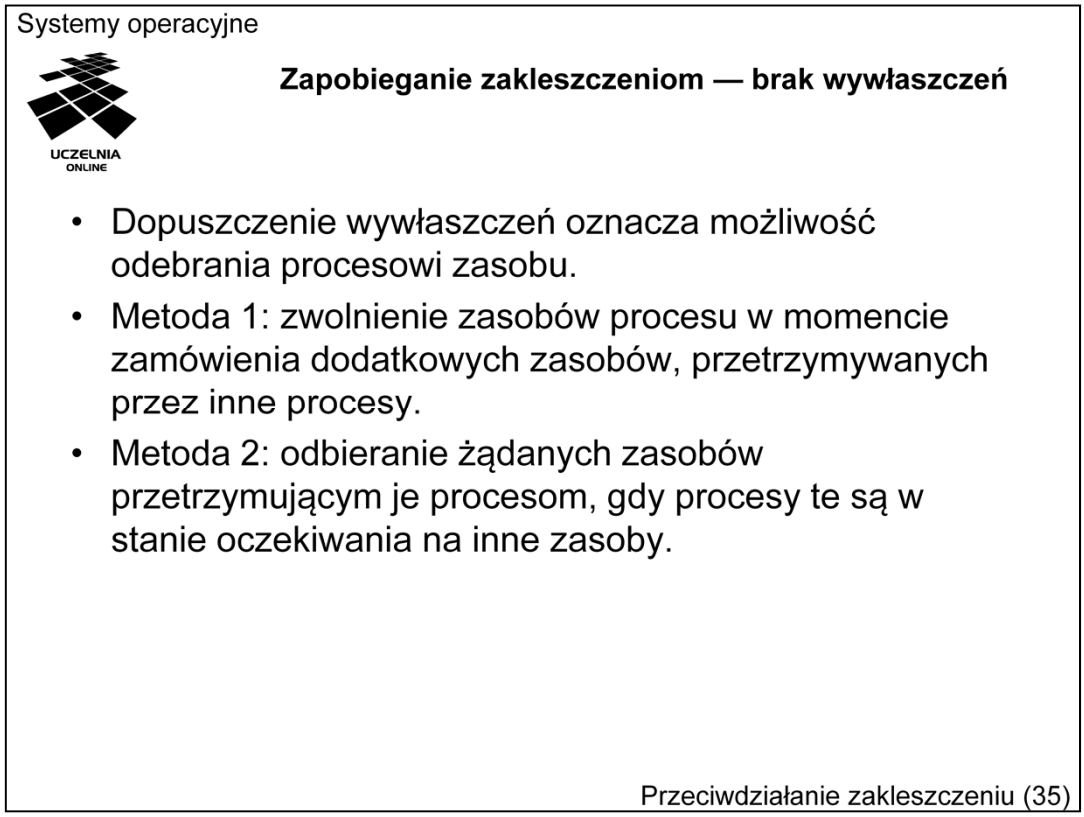
- Przeciwdziałanie warunkowi przetrzymywania i oczekiwania polega na uniemożliwieniu procesowi zamawiania zasobów w czasie, gdy proces sam przetrzymuje jakieś zasoby i blokuje do nich dostęp.
- Metoda 1: proces musi zamówić i uzyskać wszystkie zasoby, zanim rozpocznie realizację zadania, do którego zasoby te są potrzebne.
- Metoda 2: przed zamówieniem dodatkowych zasobów proces musi zwolnić zasoby przydzielone dotychczas (ewentualnie zamówić je ponownie łącznie z nowymi zasobami).

Przeciwdziałanie zakleszczeniu (34)

Podejście ma wiele wad. Uzyskanie wszystkich zasobów przed rozpoczęciem przetwarzania może oznaczać przetrzymywanie zasobów, które przez długi czas nie będą wykorzystywane (np. rezerwacja przestrzeni dyskowej dla stopniowo powiększanego pliku). W konsekwencji mamy do czynienia ze słabym wykorzystaniem zasobów, blokowanie dostępu innym procesom i tym samym zmniejszenie przepustowości systemu.

Sama realizacja praktyczna tego podejścia może być kłopotliwa, gdyż na początku przetwarzania wymagana jest wiedza o zasobach, niezbędnych do realizacji całego przetwarzania, podczas gdy pewne potrzeby ujawniają się dopiero w trakcie samego przetwarzania (np. zapotrzebowanie na dynamicznie alokowaną pamięć, czy przestrzeń dyskową).

Problemem może też być głodzenie procesu ze względu na fakt, że nigdy nie będą jednocześnie dostępne wszystkie zasoby żądane przez proces.



Wywłaszczenie musi zostać zrealizowane we właściwym momencie. Sensowny jest moment, kiedy proces zamawia zasób, ale zasób ten nie jest natychmiast dostępny. Zwalniane są wówczas zasoby przez niego przetrzymywane (od tego momentu są wolne) i dopisywane do bieżącego zamówienia. Wznowienie procesu nastąpi po zrealizowaniu całego zamówienia.

Inny dogodny moment ma miejsce, kiedy proces zamawia zasób, który jest przetrzymywany przez inny oczekujący proces. Zasób ten (lub zasoby) jest wówczas odbierany procesowi oczekującemu i przydzielany zamawiającemu. Jeśli zasób nie jest dostępny, ale jest przetrzymywany przez proces nie zablokowany w oczekiwaniu (domniemamy, że proces działa i używa tego zasobu), proces zamawiający sam przechodzi w stan oczekiwania. W stanie oczekiwania proces może stracić swoje zasoby, o ile pojawią się żądania od inny procesów. Wznowienie procesu następuje po odzyskaniu zasobów utraconych w czasie oczekiwania i uzyskaniu zasobów nowo zamawianych.

Systemy operacyjne

Zapobieganie zakleszczeniom — cykl w oczekiwaniu

- W celu przeciwdziałania cyklowi w oczekiwaniu procesów na zasoby, należy zapewnić, że wszystkie zasoby będą zamawiane w tej samej dla wszystkich procesów kolejności.
- Metoda: nadanie unikalnych numerów wszystkim typom zasobów, czyli odwzorowanie zbiorów $f : Z \to \aleph$, i zamawianie zasobów zgodnie z zasadą:
  $f(Z_j) > f(Z_i) \Rightarrow$ jednostka (lub jednostki) zasobu Z_j są zamawiane po zrealizowaniu zamówienia na jednostki zasoby Z_i
  Inaczej: nie można zamawiać jednostek zasobu Z_i, jeśli są przydzielone jednostki zasobu Z_j, gdy $f(Z_j) \geq f(Z_i)$.

Przeciwdziałanie zakleszczeniu (36)

Wcześniejsze warunki konieczne związane są z projektem systemu lub charakterystyką zasobów. Cykl wynika natomiast z funkcjonowania samych procesów aplikacyjnych, więc znaczenie łatwiej można to kontrolować. Przeciwdziałanie cyklowi jest więc najczęściej stosowaną metodą zapobiegania zakleszczeniom.

Opisana metoda wymaga, żeby jednostki zasobów, oznaczone tym samym numerem, zamawiane były w jednym zleceniu dla zarządcy.

Systemy operacyjne

Łączenie metod postępowania z zakleszczeniami

- W zależności od rodzaju zasobu systemu komputerowego stosowane są różne metody postępowania.
- Zasoby dzieli się na liniowo uporządkowane grupy.
- Żądania procesów realizowane są w kolejności wynikającej z porządku grup, w których znajdują się żądane zasoby.
- W obrębie zasobów w danej grupie stosowana jest właściwa dla tej grupy strategia realizacji żądań.

Przeciwdziałanie zakleszczeniu (37)

Ze względu na różnice w charakterystyce zasobów, pewne metody można stosować w przypadku określonych rodzajów zasobów, ale nie można w przypadku innych rodzajów. Przykłady tego typu przewijały się w omawianych podejściach do rozwiązywania problemu zakleszczenia. Podejście zintegrowane polega na tym, żeby zasoby o podobnej charakterystyce łączyć w grupy, a liniowe uporządkowanie grup z kolei ułatwia przeciwdziałanie cyklowi przy alokacji.

Systemy operacyjne

Przykład grup zasobów

1. Pamięć pomocnicza — obszary pamięci w strefie wymiany na dysku
2. Zasoby zadania — pliki, urządzenia itp.
3. Pamięć główna — obszary pamięci w obrębie fizycznej przestrzeni adresowej
4. Zasoby wewnętrzne — zasoby używane przez system do zarządzania procesami (np. bloki kontrolne, kanały wejścia-wyjścia)

Przeciwdziałanie zakleszczeniu (38)

Systemy operacyjne

UCZELNIA ONLINE

Przykład metod w obrębie grup zasobów

1. Pamięć pomocnicza — wstępny przydział (wymagania odnośnie maksymalnej zajętości przestrzeni adresowej są znane w momencie ładowania procesu)
2. Zasoby zadania — unikanie zakleszczeń (wymagane jest zidentyfikowanie żądań na podstawie opisu procesu) lub uporządkowanie liniowe
3. Pamięć główna — wywłaszczanie (przenoszenie zawartości pamięci w obszar wymiany)
4. Zasoby wewnętrzne — liniowe uporządkowanie zasobów

Przeciwdziałanie zakleszczeniu (39)

1. Wstępny przydział w przypadku pamięci pomocniczej jest możliwy, gdyż wymagany rozmiar obrazu procesu jest na ogół znany w momencie ładowania programu do pamięci.
2. Zapotrzebowanie na pliki i urządzenia może być znane z góry (np. w systemach wsadowych).
3. O możliwości wywłaszczenia pamięci fizycznej wspomniano już przy usuwaniu zakleszczenia.
4. Zasoby wewnętrzne (np. kanały wejścia-wyjścia) można uporządkować np. zgodnie z adresem w pamięci.