Blokady

Stan blokady: każdy proces w zbiorze procesów czeka na zdarzenie, które może być spowodowane tylko przez inny procesu z tego samego zbioru (*zdarzeniem* może być przydział lub zwolnienie zasobu)

Przykłady:
- W systemie są dwie stacje taśmowe; każdy z procesów P1 i P2 ma jedną taśmę i czeka na drugą
- Semafory A i B mają wartość 1:
  P0: *wait*(A); *wait*(B)
  P1: *wait*(B); *wait*(A)

Model systemu

- Różne typy zasobów: Z_1, Z_2, ..., Z_m; W_i jednostek zasobu Z_i ; cykle CPU, pamięć, urządzenia wejścia-wyjścia
- Każdy proces korzysta z zasobu według schematu:
  zamówienie; *użycie*; *zwolnienie*

Charakterystyka blokady

Blokada może powstać wtw, gdy w systemie są spełnione równocześnie cztery warunki:
1. *Wzajemne wykluczanie*: tylko jeden proces może używać zasobu w danym czasie
2. *Przetrzymywanie i oczekiwanie*: proces przetrzymujący co najmniej jeden zasób czeka na przydział dodatkowych zasobów będących w posiadaniu innych procesów
3. *Brak wywłaszczania*: zasoby nie podlegają wywłaszczaniu

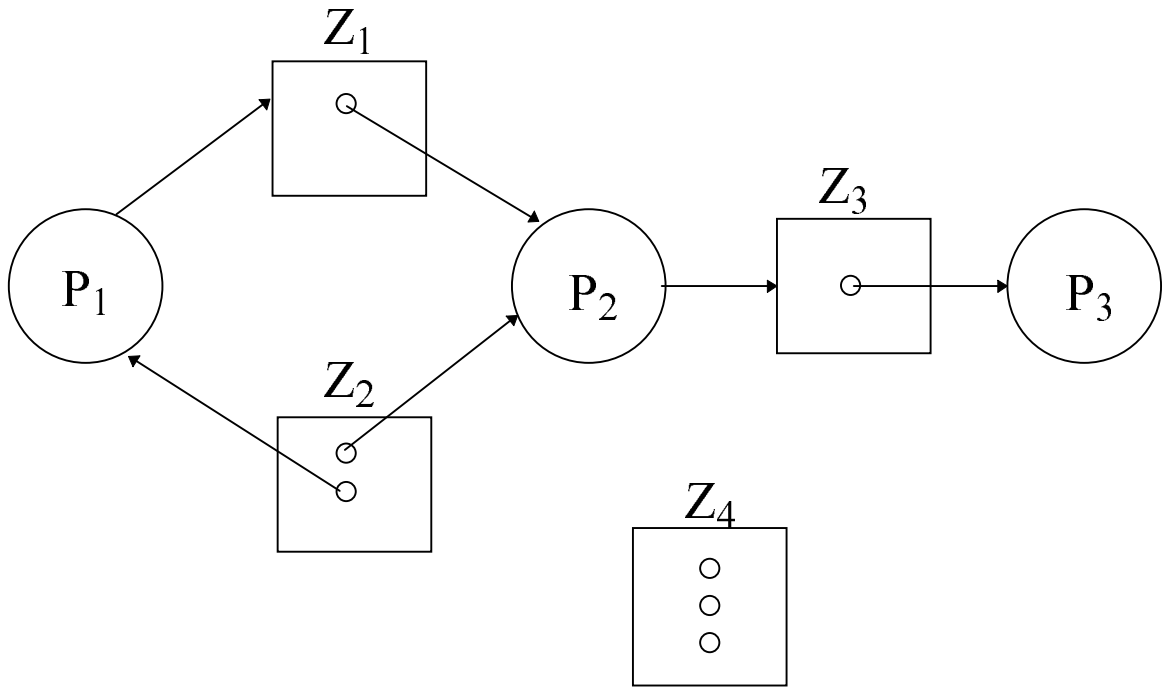
4. *Cykliczne oczekiwanie*: istnieje zbiór czekających procesów $\{P_0, P_1, ..., P_n\}$, takich że P_0 czeka na zasób przetrzymywany przez P_1, P_1 czeka na zasób przetrzymywany przez P_2, ..., P_n czeka na zasób przetrzymywany przez P_0.
(4 implikuje 2, więc podane warunki nie są niezależne)

Graf przydziału zasobów:

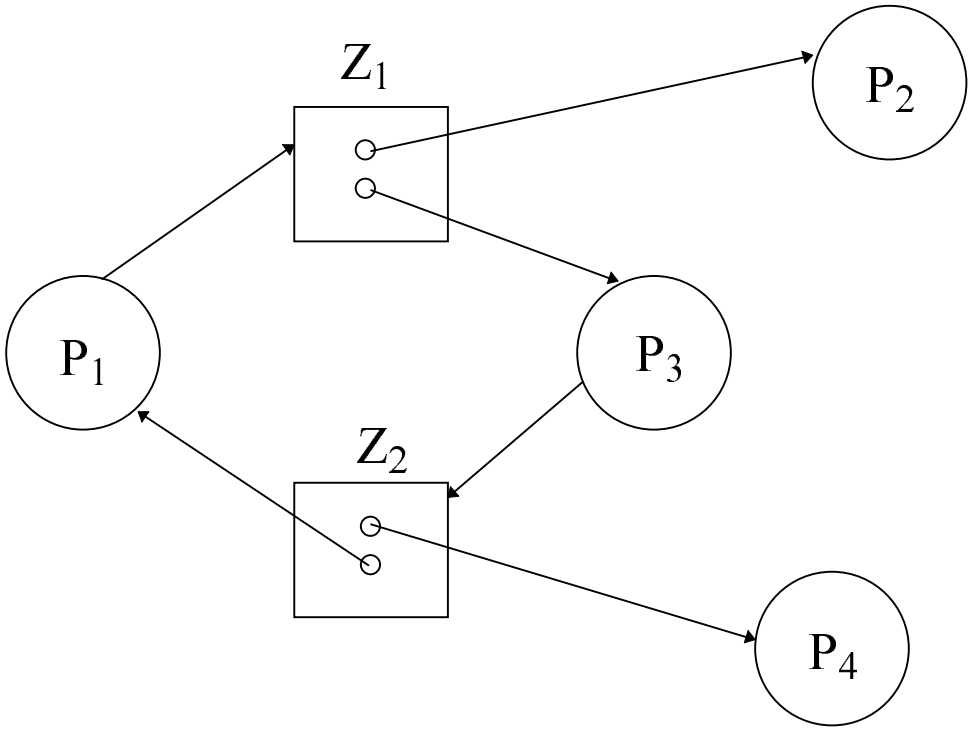
Graf skierowany o zbiorze wierzchołków W i krawędzi K:

- W składa się z dwóch podzbiorów: $P=\{P_1, ..., P_n\}$ - procesy, $Z=\{Z_1, ..., Z_m\}$ - typy zasobów
- *krawędź żądania* - skierowana krawędź $P_i \rightarrow Z_k$
- *krawędź przydziału* - skierowana krawędź $Z_k \rightarrow P_i$

Przykład grafu przydziału zasobów bez cykli:



Przykład grafu przydziału zasobów z cyklem:



1. Jeśli graf nie zawiera cyklu, to nie ma blokady
2. Jeśli graf zawiera cykl, to:
   - jeśli zasoby są w jednym egzemplarzu, to blokada
   - jeśli zasoby są w wielu egzemplarzach, to istnieje możliwość blokady

Metody postępowania z blokadami:

- Zapewnić, że system nigdy nie wejdzie w stan blokady
- Pozwolić na wejście systemu w stan blokady, po czym ją usunąć
- Zignorować problem i udawać, że system nigdy nie wejdzie w stan blokady; stosowane przez większość systemów operacyjnych, w tym Unix

Zapobieganie blokadom

Zapobiec spełnieniu jednego z warunków koniecznych wystąpienia blokady:

- *Wzajemne wykluczanie* - nie jest wymagane dla zasobów dzielonych; musi zachodzić dla zasobów niepodzielnych
- *Przetrzymywanie i oczekiwanie* - trzeba zagwarantować, że gdy proces żąda zasobu, to nie posiada innych zasobów
  - można wymagać, by proces zamawiał i dostawał wszystkie swoje zasoby zanim rozpocznie działanie lub żądał zasobów wtedy, gdy nie posiada żadnych innych
  - niskie wykorzystanie zasobów, możliwość zagłodzenia
- *Brak wywłaszczania*:
  - Jeśli proces będący w posiadaniu zasobów żąda zasobu, którego nie można natychmiast przydzielić, to musi zwolnić wszystkie posiadane zasoby
  - Wywłaszczone zasoby dodaje się do listy zasobów, na które czeka proces
  - Proces zostanie wznowiony, gdy będzie mógł odzyskać posiadane wcześniej zasoby oraz otrzymać nowo żądany zasób
- *Cykliczne oczekiwanie* - narzuca się uporządkowanie całkowite wszystkich typów zasobów i wymaga, by proces zamawiał zasoby we wzrastającym porządku ich numerów

Unikanie blokad

Wymaga, by system posiadał pewną informację o przyszłym zapotrzebowaniu na zasoby

- W najprostszym modelu wymaga się, by proces deklarował maksymalną liczbę zasobów każdego typu, których będzie potrzebował

- Algorytm unikania blokady dynamicznie bada stan przydziału zasobów, by zapewnić, że nigdy nie dojdzie do cyklicznego oczekiwania
- *Stan przydziału zasobów* jest określony przez liczbę dostępnych i przydzielonych zasobów oraz przez maksymalne zapotrzebowanie procesów

Stan bezpieczny - kiedy proces żąda dostępnego zasobu, system musi ustalić, czy natychmiastowy przydział zasobu zachowa bezpieczny stan systemu

- System jest w stanie bezpiecznym, gdy istnieje *bezpieczna sekwencja procesów*
- Sekwencja $<P_1, P_2, ..., P_n>$ jest bezpieczna, jeśli dla każdego P_i jego potencjalne zapotrzebowanie na zasoby można zaspokoić przez bieżąco dostępne zasoby oraz zasoby będące w posiadaniu procesów P_k, gdzie $k < i$
- Jeśli system jest w *stanie bezpiecznym*, to *nie ma blokady*
- Jeśli system *nie jest w stanie bezpiecznym*, to istnieje *możliwość powstania blokady*
- Można *unikać blokady* zapewniając, że system *nigdy nie wejdzie do stanu niebezpiecznego*

Algorytm korzystający z grafu przydziału zasobów:

- Dla zasobów występujących pojedynczo
- *Krawędź deklaracji* $P_i \rightarrow Z_k$ wskazuje, że proces P_i może zażądać zasobu Z_k; reprezentowana linią przerywaną
- Krawędź deklaracji przechodzi na krawędź żądania, gdy proces zażąda zasobu
- W chwili zwolnienia zasobu krawędź przydziału przechodzi na krawędź deklaracji

- Trzeba a priori zadeklarować zapotrzebowanie na zasoby
- Koszt algorytmu szukania cyklu w grafie zasobów: n^2

Algorytm bankiera

- Dla zasobów wielokrotnych
- Każdy proces musi a priori złożyć maskymalne zapotrzebowanie na zasoby
- Proces żądający zasobu być może będzie musiał czekać, mimo że zasób jest dostępny
- Gdy proces dostanie wszystkie potrzebne zasoby, to zwróci je w skończonym czasie
- Szczegóły działania algorytmu → **na ćwiczeniach!**
- Koszt stwierdzenia czy stan jest bezpieczny: $m \times n^2$

Wykrywanie blokady

- Dopuszcza się wejście systemu w stan blokady
- Algorytm wykrywania blokady
- Algorytm wychodzenia z blokady

Zasoby pojedyncze

- Utrzymuje się *graf oczekiwań*: węzły odpowiadają procesom, $P_i \rightarrow P_k$ jeśli P_i czeka na zasób będący w posiadaniu P_k
- Okresowo wykonuje się algorytm szukania cyklu w grafie
- Koszt algorytmu: n^2, gdzie n - liczba wierzchołków

Zasoby wielokrotne

- Podobny do algorytmu bankiera
- Szczegóły działania algorytmu → **na ćwiczeniach!**
- Koszt algorytmu wykrywania blokady: $m \times n^2$

Jak stosować algorytm wykrywania blokady:

- Kiedy i jak często wykonywać: zależy od tego jak często może wystąpić blokada i jak wiele procesów obejmie
- Jeśli wykonuje się algorytm w dowolnych chwilach, to w grafie może powstać wiele cykli. Wskazanie sprawcy blokady wśród wielu zablokowanych procesów może być niewykonalne

Wychodzenie z blokady

- Zakończenie procesu
  - zakończ wszystkie zablokowane procesy
  - kończ procesy pojedynczo, aż do wyeliminowania blokady
  - jak wybrać proces do zakończenia:
    - ⇒ priorytet procesu
    - ⇒ jak długo proces wykonywał obliczenia i ile potrzebuje czasu do zakończenia
    - ⇒ zasoby używane przez proces
    - ⇒ zasoby potrzebne procesowi do zakończenia
    - ⇒ ile procesów trzeba zakończyć
    - ⇒ czy proces jest interakcyjny, czy wsadowy
- Wywłaszczanie zasobów
  - Wybór *ofiary*
  - *Wycofanie* procesu do pewnego bezpiecznego stanu i późniejsze jego wznowienie z tego stanu
  - *Głodzenie* procesu - pewien proces może być stale wybierany jako ofiara; uwzględnić liczbę wycofań przy ocenie kosztów

Łączenie metod postępowania z blokadą