Sposoby radzenia sobie z zakleszczeniami

Zakleszczenie (ang. deadlock) to sytuacja w systemie wielozadaniowym, w której dwa lub więcej procesów oczekuje na zasób, który jest blokowany przez inne procesy, co prowadzi do stanu, w którym żaden z procesów nie może kontynuować pracy. Aby skutecznie zarządzać zakleszczeniami, systemy operacyjne stosują różne metody. Można je podzielić na cztery główne grupy:

1. Zapobieganie zakleszczeniom (Deadlock Prevention)

Celem tej strategii jest uniemożliwienie wystąpienia zakleszczenia poprzez zapewnienie, że jedna lub więcej z czterech koniecznych warunków zakleszczenia nie zostanie spełniona. Te warunki to:

- Wzajemne wykluczanie (Mutual Exclusion)
- Trzymanie i oczekiwanie (Hold and Wait)
- Brak wywłaszczenia (No Preemption)
- Cykliczne oczekiwanie (Circular Wait)

Metody zapobiegania:

- **Ograniczenie wzajemnego wykluczania**:
    - Jeśli to możliwe, należy umożliwić współdzielenie zasobów przez wiele procesów (np. dostęp do plików w trybie tylko do odczytu).
- **Unikanie trzymania i oczekiwania**:
    - Wymuszanie, aby procesy rezerwowały wszystkie wymagane zasoby przed rozpoczęciem wykonywania.
    - Wadą jest możliwość marnowania zasobów, jeśli procesy rezerwują więcej, niż aktualnie potrzebują.
- **Wywłaszczenie zasobów**:
    - Jeśli proces czeka na zasób, system może wywłaszczyć inne zasoby od tego procesu i oddać je do puli.
- **Unikanie cyklicznego oczekiwania**:
    - Wprowadzenie porządku numeracji zasobów i wymuszenie, aby procesy żądały zasobów tylko w określonej kolejności.

2. Unikanie zakleszczeń (Deadlock Avoidance)

Metoda ta polega na dynamicznym zarządzaniu zasobami, aby unikać stanów prowadzących do zakleszczenia. Procesy zgłaszają swoje zapotrzebowanie na zasoby przed rozpoczęciem działania, a system analizuje, czy przydzielenie tych zasobów nie doprowadzi do zakleszczenia.

Banker's Algorithm (Algorytm Bankiera):

- Jeden z najpopularniejszych algorytmów unikania zakleszczeń.
- Działa na zasadzie symulacji przydzielania zasobów, sprawdzając, czy przydzielenie nie wprowadzi systemu w stan niebezpieczny.
- Jeśli operacja przydzielenia prowadzi do stanu niebezpiecznego, jest odrzucana, a proces musi czekać na dostępność zasobów.

Wady:

- Algorytm wymaga wcześniejszej wiedzy o maksymalnym zapotrzebowaniu procesów, co nie zawsze jest możliwe.
- Może prowadzić do niskiego wykorzystania zasobów.

3. Wykrywanie zakleszczeń (Deadlock Detection) i ich usuwanie

Jeśli system nie stosuje mechanizmów zapobiegania ani unikania zakleszczeń, konieczne jest wykrywanie, kiedy zakleszczenie wystąpi, i podejmowanie odpowiednich działań w celu jego usunięcia.

Wykrywanie zakleszczeń:

- System okresowo analizuje stan zasobów i procesów w celu wykrycia cyklicznego oczekiwania.
- Do wykrywania cykli w grafie zależności procesów można użyć algorytmów takich jak **DFS (Depth-First Search)**.

Usuwanie zakleszczeń:

1. **Zabijanie procesów (Abort Processes)**:
    - Usunięcie jednego lub więcej procesów biorących udział w zakleszczeniu.
    - Można wybrać proces na podstawie:
        - Priorytetu.
        - Czasu potrzebnego do ukończenia.

        - Ilości zasobów już przydzielonych.
    - Wadą jest utrata danych i pracy procesów.
2. **Wywłaszczanie zasobów (Resource Preemption)**:
    - System odbiera zasoby od procesów, aby rozwiązać cykl zakleszczenia.
    - Wymaga mechanizmów do zapisywania stanu procesów, aby można było je wznowić po odzyskaniu zasobów.

4. Ignorowanie zakleszczeń (Deadlock Ignorance)

Niektóre systemy operacyjne, takie jak Windows czy UNIX, po prostu ignorują problem zakleszczeń, zakładając, że są one rzadkie lub ich wykrywanie i zapobieganie jest zbyt kosztowne.

Podejście "Ostrygowe":

- Uznanie, że zakleszczenia są bardzo rzadkie, i pozostawienie ich rozwiązania administratorowi systemu.
- Przykładem jest system **Linux**, który nie implementuje mechanizmów zapobiegania zakleszczeniom, zakładając, że użytkownik może zidentyfikować i zabić problematyczne procesy.