# Metoda One-Shot

W celu uniknięcia zakleszczenia trzeba się upewnić żeby kolejkowanie się nigdy nadmiernie nie zwiększało. Jedną z możliwości, aby temu zapobiec jest podział wszystkich procesów na dwie grupy:

* Pierwsza grupa: Procesy posiadające wszystkie zasoby, których kiedykolwiek będą potrzebować.
* Druga grupa: Procesy nieposiadające zasobów.

Procesy należące do pierwszej grupy nigdy nie będą blokowane, natomiast procesy należące do drugiej grupy będą blokowane w trakcie oczekiwania na zwolnienie zasobów przez grupę pierwszą. Nazywa się to alokacją *one-shot*, i implikuje to, że każdy proces w momencie dostępu do zasobów pobiera wszystkie zasoby mu potrzebne. Procesy próbujące dostać zasoby aktualnie używane przez inny proces są blokowane aż do momentu ich zwolnienia przez używający ich aktualnie proces. Każda następna próba dostępu do zasobów po ich poprzednim otrzymaniu skutkuje błędem.

W swojej czystej formie, metoda *one-shot* wymaga, aby każdy proces pobrał wymagane przez siebie zasoby tylko raz, i nigdy więcej nie próbował uzyskać dostępu do nich ponownie.

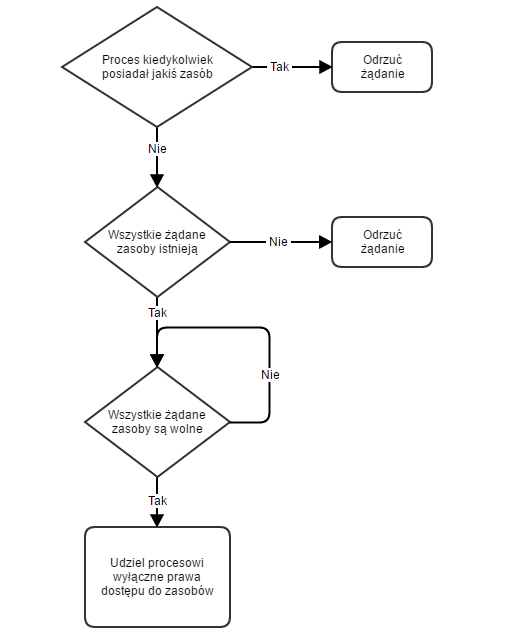
W skrócie metoda *one-shot* podlega jednej bardzo ważnej zasadzie:

***Proces może żądać zasoby tylko, jeżeli nie posiada żadnych.***

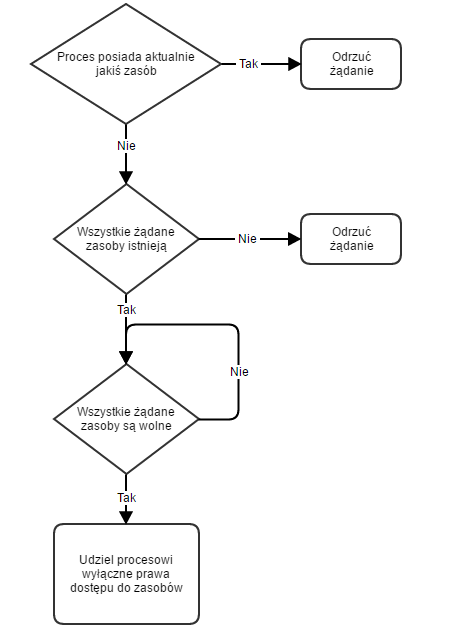
Metoda ta jest bardzo konserwatywna, przez co mało praktyczna. Dla przykładu, jeżeli jakiś proces będzie potrzebował bardzo dużej ilości zasobów, będzie musiał pobrać je wszystkie na raz, tym samym blokując dostęp do nich dla innych procesów, które w międzyczasie mogłyby go używać. Ponadto proces może nie wiedzieć, jakich zasobów może w przyszłości potrzebować.

Na postawie tej metody powstały dwa algorytmy:

* Algorytm One-Shot – algorytm ściśle przestrzegający metody *one-shot*. Posiada złożoność obliczeniową *0(****n****).* Jedną ze znanych implementacji tego algorytmu jest funkcja WaitForMultipleObjects dostępna od Windowsa 2000. *Jego* działanie jest zaprezentowane na Schemat Blokowy 1.
* Algorytm Multishot – algorytm działający wedle metody *one-shot* z jedną małą różnicą. Pozwala on na wielokrotne wysyłanie żądań przez ten sam wątek. Również posiada złożoność obliczeniową 0(**n**). Jego działanie jest zaprezentowane na Schemat Blokowy 2.



Schemat Blokowy 1 Algorytm One-Shot



Schemat Blokowy 2 Algorytm Multishot

# Metoda Hierarchiczna

Aby uniknąć zakleszczenia i nadmiernego wzrostu kolejkowania procesów w celu dostępu do danych używa się metody Hierarchicznej. Polega ona na przydzielaniu procesom numerów rosnąco, oraz asocjacji procesów z najwyższym priorytetem posiadanych zasobów. Metoda hierarchiczna podlega następującej zasadzie:

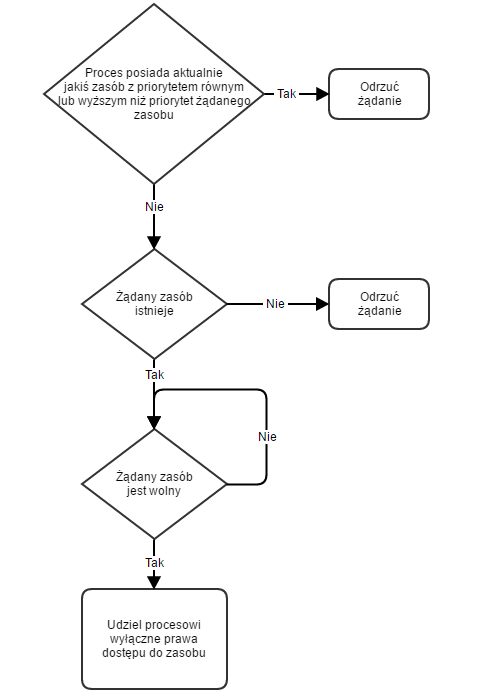
***Proces może żądać zasobów tylko posiadających wyższe priorytety niż***

***aktualnie posiadane zasoby***

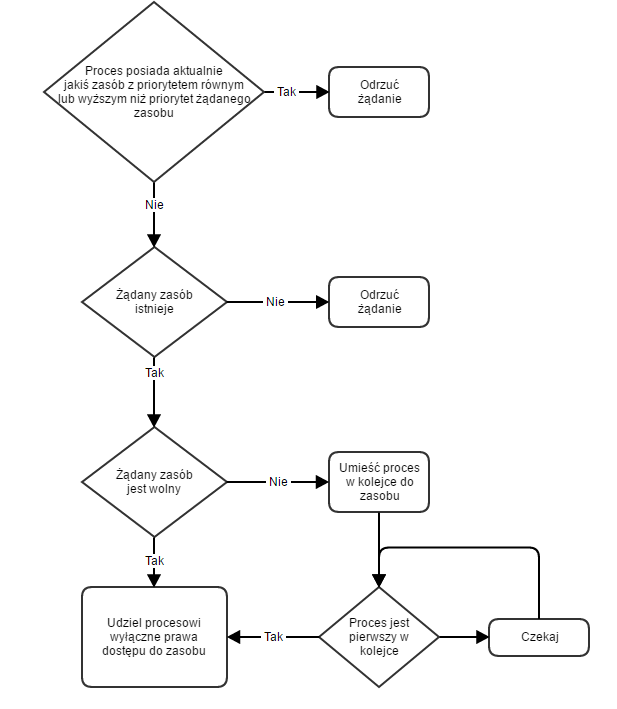
Metoda hierarchicznej alokacji zasobów rozwiązuje problem zakleszczenia. Niestety jest to algorytm dalej konserwatywny, ponieważ proces musi żądać zasobów z niskimi priorytetami dużo wcześniej niż będzie wysyłał żądania o zasoby wysoko priorytetowe. Po dostaniu zasobów wysokopriorytetowych algorytm nie będzie miał możliwości pozyskania zasobów z niskimi priorytetami do momentu aż nie odda wszystkich zasobów posiadających wyższe priorytety. Implikuje to powstanie sztucznego cyklu żądania zasobów z niższymi priorytetami, aby proces nie musiał oddawać zasobów wysokopriorytetowych w celu ich pozyskania. Procesy pozyskujące w ten sposób blokują zasoby nisko priorytetowe, pomimo że ich jeszcze nie potrzebują, przez co zabierają dostęp do nich innym procesom, które mogą ich potrzebować wcześniej. Przy używaniu tej metody rzadkie zasoby powinny dostawać wysokie priorytety, aby żądania one były wykonywane tylko wtedy, gdy są potrzebne.

Na podstawie tej metody powstały dwa algorytmy:

* Algorytm Hierarchiczny – Algorytm ten przestrzega metody Hierarchicznej. Posiada on złożoność obliczeniową 0(**n**). Jego działanie jest zaprezentowane na
* Algorytm Hierarchiczny z kolejką - Algorytm ten przestrzega metody Hierarchicznej, z taką różnicą, że procesy próbujące uzyskać dostęp do danego zasobu zostają zakolejkowane. Posiada on złożoność obliczeniową 0(**n**). Jego działanie jest zaprezentowane na



Schemat Blokowy 3 Algorytm Hierarchiczny



Schemat Blokowy 4 Algorytm Hierarchiczny z kolejką

# Bibliografia

Finkel, R. A. (1998). *Operating Systems Vade-mecum* (Tom 2nd Revised edition). Upper Saddle River, New Jersey: Prentice Hall International Paperback Editions.

Gonzalez, T., Diaz-Herrera, J. i Tucker, A. (2014). *Computing Handbook, Third Edition: Computer Science and Software Engineering.* Londyn: Chapman and Hall/CRC.

Hamilton, H. J. (2016, Czerwiec 11). *Deadlock Prevention Algorithms*. Pobrano z lokalizacji http://www2.cs.uregina.ca/~hamilton/courses/330/notes/synchro/deadlock-ex.html