Inwersja priorytetów

1 Inwersja priorytetów

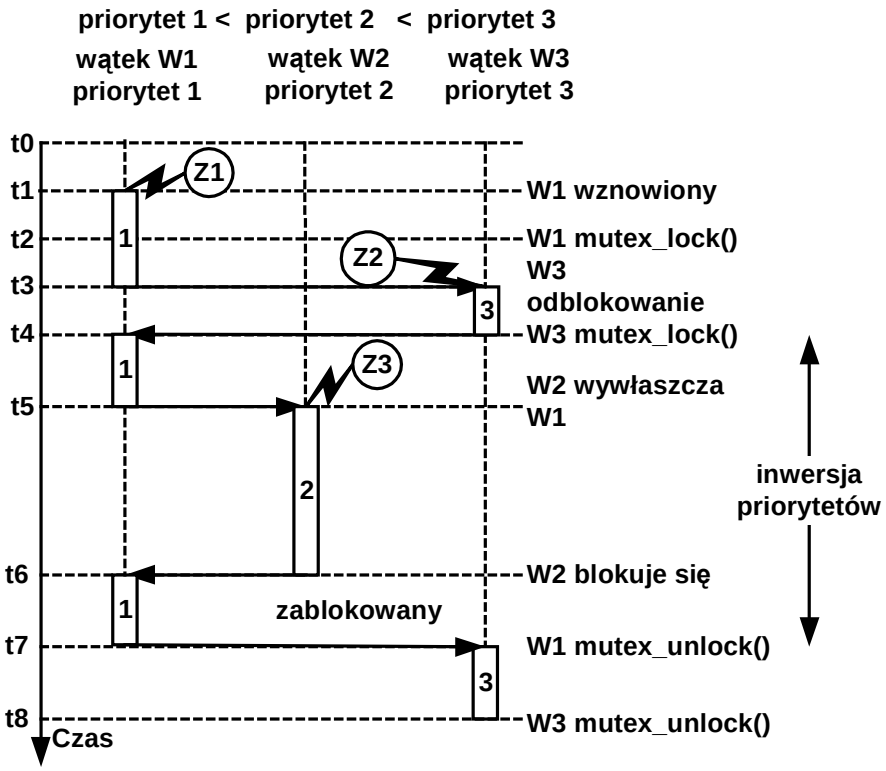
Gdy dwa lub więcej wątki o różnych priorytetach używają wspólnego zasobu chronionego przez pewien mechanizm zapewnienia wzajemnego wykluczania (np. muteks) może dojść zjawiska nazywanego inwersją priorytetów.

Przykład:
Dwa wątki, W3 o priorytecie wyższym i W1 o priorytecie niższym używają zabezpieczonego muteksem wspólnego zasobu.

Scenariusz:
1. Jako pierwszy startuje wątek W1 (niższym priorytecie) a następnie wątek W1 zajmuje muteks.

2. Startuje wątek W3. Muteks jest już zajęty, wątek W3 mimo że posiada wyższy niż W1 priorytet nie będzie wykonywany.

3. Pojawia się wątek W2 o priorytecie pośrednim wywłaszczy on wątek W1 który nie będzie mógł zwolnić muteksu. Wątek W1 pozostanie wciąż zablokowany a wykonywał się będzie wątek W2 o priorytecie niższym niż W3.



Rys. 1-1 Ilustracja zjawiska inwersji priorytetów

t1	Zdarzenie Z1 odblokowuje wątek W1
t2	Wątek W1 zajmuje muteks
t3	Zdarzenie Z2 odblokowuje wątek W3, ma on priorytet wyższy od W1 i go wywłaszcza.
t4	Wątek W3 próbuje zająć muteks. Muteks jest zajęty a więc W3 blokuje się.
t5	Zdarzenie Z3 odblokowuje wątek W2 który wywłaszcza W1 gdyż ma wyższy priorytet.
t6	W2 samoistnie się blokuje. Wątek W1 jest wznawiany.
t7	Wątek W1 zwalnia muteks. Dopiero teraz wątek W3 staje się gotowy i wywłaszcza W1.
t8	Wątek W3 zwalnia muteksy

W okresie od t4 do t7 wykonują się wątki W1 i W2 pomimo że wątek W3 ma wyższy priorytet i pozostaje gotowy. Zachodzi więc inwersja priorytetów.

Inwersja priorytetów (*ang. Priority Inversion*)

Inwersja priorytetów – zjawisko polegające na wykonywaniu się wątku o niższym priorytecie mimo że wątek o wyższym priorytecie pozostaje gotowy. Inwersja priorytetów może się pojawić gdy wątki o różnych priorytetach używają wspólnego muteksu lub podobnego mechanizmu synchronizacyjnego.

Powstaje pytanie jak postępować w takim przypadku?

W systemach czasu rzeczywistego stosowane są dwie strategie postępowania z problemem inwersji priorytetów. Jest to:

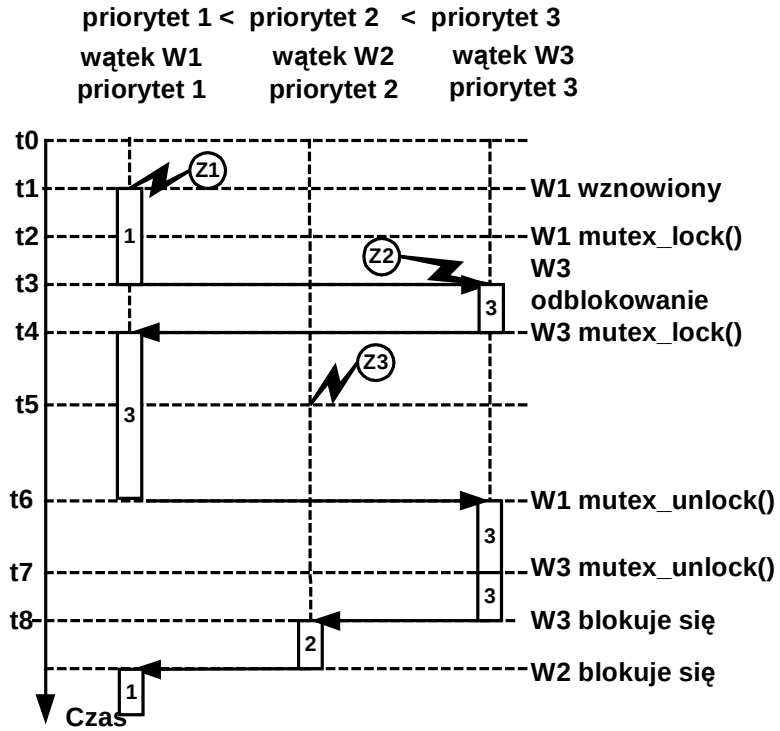
1. Dziedziczenie priorytetu (ang. *Priority Inheritance*)
2. Zastosowanie stosowanie protokołu wykorzystującego tzw. pułap priorytetów (ang. *Priorty Ceiling*)

2 Dziedziczenie priorytetu

Dziedziczeniem priorytetu polega na tym, że gdy wątek W3 o wyższym priorytecie próbuje zająć muteks zajęty już przez wątek W1 o priorytecie niższym, to system podwyższa chwilowo priorytet wątku W1 zajmującego muteks do wysokości priorytetu wątku W3. Dzięki podwyższonemu priorytetowi wątek W1 szybciej wykona swe zadanie i zwolni muteks. Po zwolnieniu muteksu wątkowi W1 zostaje przywrócony pierwotny priorytet.

Dziedziczenie priorytetu (*ang. priority inheritance*)

Dziedziczeniem priorytetu – tymczasowe zwiększenie priorytetu wątku posiadającego zasób do najwyższego priorytetu z priorytetów wątków ubiegających się o zajęcie tego zasobu. Po zwolnieniu zasobu wątkowi przywracany jest początkowy priorytet.



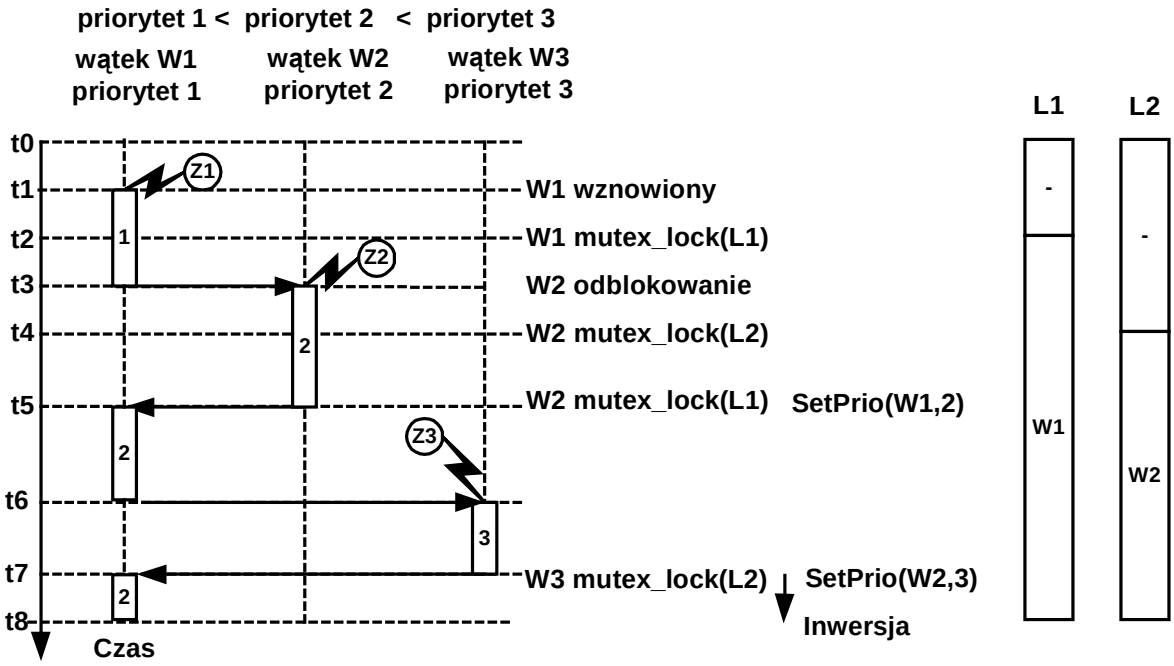
Rys. 2-1 Ilustracja dziedziczenia priorytetów

t1	Zdarzenie Z1 odblokowuje wątek W1
t2	Wątek W1 zajmuje muteks
t3	Zdarzenie Z2 odblokowuje wątek W3. Ma on priorytet wyższy od W1 i go wywłaszcza.
t4	Wątek W3 próbuje zająć muteks. Muteks jest zajęty a więc W3 blokuje się ale wątek W1 dziedziczy priorytet wątku W3 (z 1 wzrósł do 3).
t5	Zdarzenie Z3 odblokowuje wątek W2. Nie powoduje to wywłaszczenia W1 gdyż W2 ma niższy priorytet.
t6	Wątek W1 zwalnia muteks. Dopiero teraz wątek W3 staje się gotowy i wywłaszcza W1.
t7	Wątek W3 zwalnia muteks
t8	Wątek W3 blokuje się i dopiero teraz W2 może być wykonywany.

Sekwencja zdarzeń zachodząca systemie ilustrująca dziedziczenie priorytetów.

Protokół dziedziczenia priorytetów działa prawidłowo w przypadku użycia jednego typu zasobu. Gdy używana jest większa liczba zasobów może dojść do różnych niekorzystnych zjawisk jak:

- blokowanie przechodnie
- zakleszczenie.

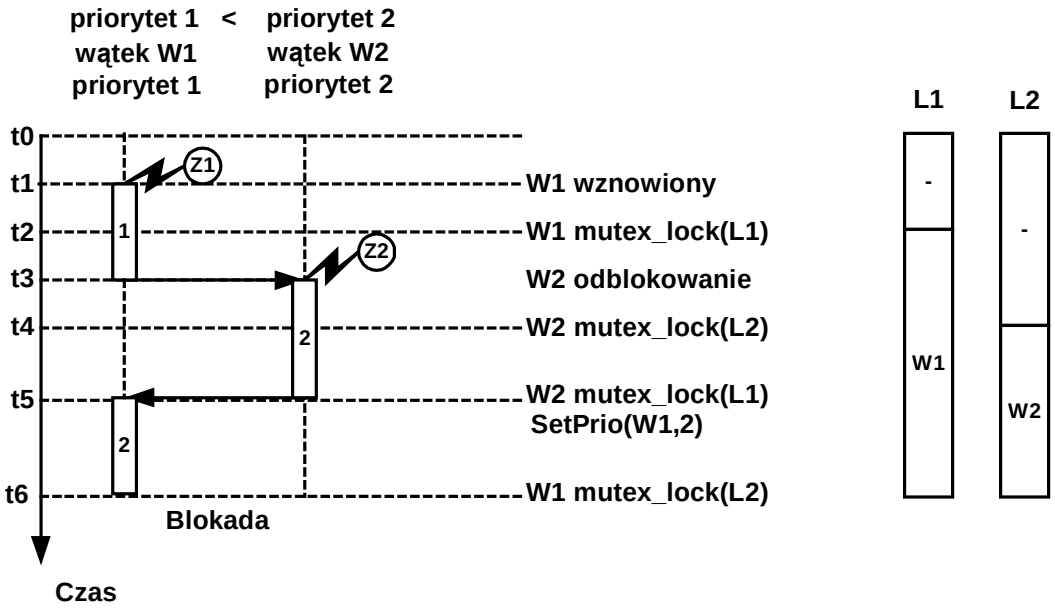


Rys. 2-2 Blokowanie przechodnie (ang. *transitive bloking*)

W3 jest pośrednio blokowany przez W1 ale priorytet W1 nie jest podnoszony do priorytetu W3. Wady tej pozbawiony jest protokół wykorzystujący pułap priorytetów.

3 Zakleszczenia

Protokół dziedziczenia priorytetów posiada istotny defekt. Gdy wątki potrzebują dwóch różnych zasobów może dojść do ich zakleszczenia. Może się tak zdarzyć gdy wątek W1 zablokuje zasób potrzebny wątkowi W2 a wątek W2 zajmie zasób potrzebny wątkowi W1.



Przypadek prowadzący do zakleszczenia wątków

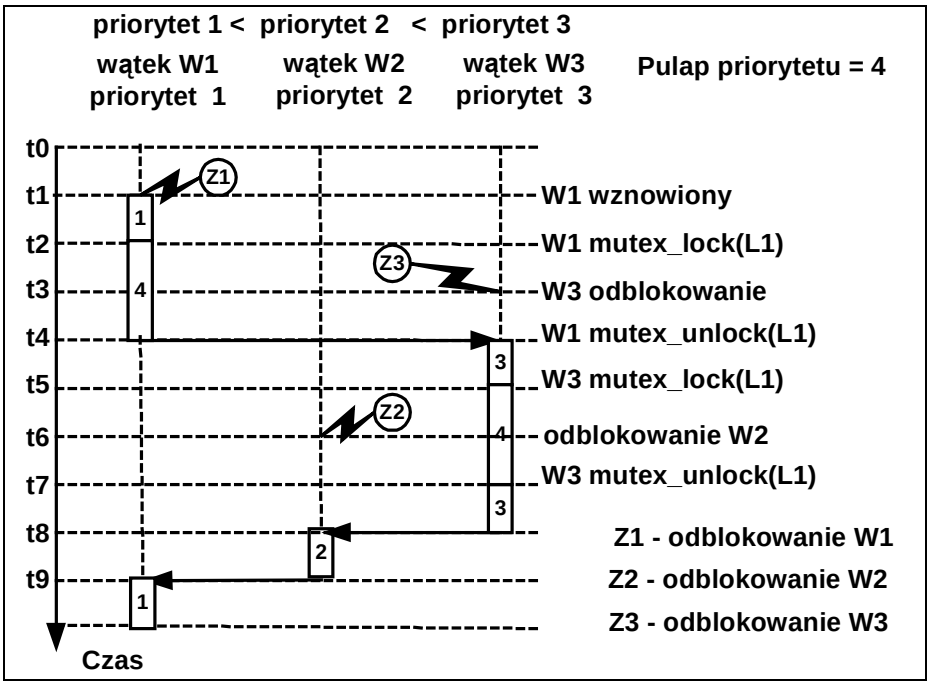
4 Protokół wykorzystujący pułap priorytetów

Drugą strategią zapobiegania inwersji priorytetów jest przyjęcie protokołu wykorzystującego pułap priorytetów.
Każdemu chronionemu zasobowi (w tym przypadku jest to muteks) przypisuje się pewien określony statyczny priorytet. Priorytet ten powinien być wyższy od najwyższego priorytetu z tych wątków które o dany zasób będą konkurowały. Gdy jakiś wątek będzie próbował zająć zasób to zostanie mu tymczasowo przydzielony priorytet związany z tym zasobem. Po zwolnieniu zasobu priorytet wątku wróci do wielkości wyjściowej.

Protokół z pułapem priorytetu (*ang. priority ceiling protocol*)

W protokole z pułapem priorytetu następuje tymczasowe zwiększenie priorytetu wątku usiłującego zająć zasób do pewnego ustalonego priorytetu (wyższego od priorytetu jakiegokolwiek wątku konkurującego o zasób). Wszystkim wątkom konkurujące o zasób zostaje tymczasowo nadany ten jednakowy priorytet.

Dzięki temu że watek zajmujący zasób zyskuje chwilowo priorytet wyższy niż jakiekolwiek inny wątek konkurujący o zasób – ma on szansę zakończyć operację na zasobie bez wywłaszczenia.



Ilustracja działania protokołu z pułapem priorytetu

t1	Zdarzenie Z1 odblokowuje wątek W1
t2	W1 zajmuje muteks, zyskuje priorytet 4
t3	Zdarzenie Z2 odblokowuje W3. Ma on priorytet niższy od W1 i nie wywłaszcza go.
t4	W1 zwalnia muteks L1 i odzyskuje pierwotny priorytet 1. W1 zostaje wywłaszczony przez W3
t5	W3 zajmuje muteks, zyskuje priorytet 4.
t6	Zdarzenie Z2 odblokowuje W2 ale ma on niższy priorytet niż W3 a więc nic się nie dzieje.
t7	W3 zwalnia muteks L1 i odzyskuje pierwotny priorytet 3.
t8	W3 blokuje się a W2 zostaje wznowiony.
T9	W2 blokuje się a W1 zostaje wznowiony

Zalety protokołu:

- Protokół zapobiega powstawaniu zakleszczeń.
- Zapewnia dobry czas oczekiwania na zasób (dla najgorszego przypadku) poprzez wątek o najwyższym priorytecie. Czas ten równy jest długości najdłuższej sekcji krytycznej wątków o niższym priorytecie.

Wady protokołu:

- Należy z góry wyznaczyć zbiór wszystkich wątków które będą konkurowały o zasób i jako pułap priorytetu przyjąć najwyższy priorytet z zadań z tego zbioru + 1 . Może to być czasochłonne lub nawet niemożliwe.
- Posiada zły średni czas odpowiedzi z związku z narzutami na implementację.

5 Operowanie na pułapie priorytetu w Pthreads.

W bibliotece Pthreads można wybrać protokół zajmowania muteksu. Dokonuje się tego ustawiając atrybuty muteksu przy pomocy funkcji **pthread_mutexattr_ setprotocol().**

int pthread_mutexattr_setprotocol(pthread_mutexattr *attr, int protocol)	
attr	Zadeklarowana wcześniej i zainicjowana zmienna atrybutów
protocol	**PTHREAD_PRIO_INHERID** lub **PTHREAD_PRIO_PROTECT**

Pobranie bieżącego pułapu priorytetu muteksu

int pthread_mutex_getprioceiling(pthread_mutex_t * mutex,int * prioceiling)

Funkcja zwraca bieżący pułap priorytetu **prioceiling** mutexu **mutex**.

Ustalenie pułapu priorytetu dla muteksu

int pthread_mutex_setprioceiling(pthread_mutex_t *mutex, int prioceiling, int *old_ceiling)

Funkcja powoduje że wątek próbuje zająć mutex**.** Gdy muteks wolny zajmuje go. Gdy muteks jest zajęty wątek się blokuje i czeka na na jego zwolnienie. Wtedy ustala nowy pułap priorytetu na **prioceiling** i zwraca poprzedni pułap **old_ceiling** i zwalnia muteks**.**

6 Przypadek sondy Pathfinder

Zjawisko inwersji priorytetów spowodowało awarię sondy marsjańskiej Pathfinder w 1997 roku. Po kilku dniach trwania misji sonda wykonała restart systemu co spowodowało utratę zgromadzonych wcześniej danych.

Komputerem sterującym był system oparty na magistrali VME. Sonda wyposażona była w system VxWorks firmy Wind River Systems. System ten stosuje wywłaszczającą strategię szeregowania wątków którym nadaje się priorytety.

W awarii brały udział trzy wątki:

1. Wątek zarządzający „szyną informacyjna” – wspólnym obszarem pamięci w którym inne wątki umieszczały swe dane. Dane te były dostępne dla innych wątków. Ten watek miał wysoki priorytet.
2. Wątek przekazywania danych meteorologicznych – priorytet niski
3. Watek komunikacyjny – priorytet średni

Wspólny obszar pamięci był zabezpieczony muteksem.

Scenariusz:

1. Watek meteorologiczny zajął muteks
2. Przerwanie odblokowało watek informacyjny który próbował zająć muteks ale że był on zajęty to się zablokował
3. Wątek komunikacyjny o średnim priorytecie wywłaszczył watek meteorologiczny który nie mógł zwolnić muteksu. Było to długotrwałe zadanie.
4. Watchdog timer zaobserwował że „szyna informacyjna” nie działa i przeprowadził restart całego systemu.