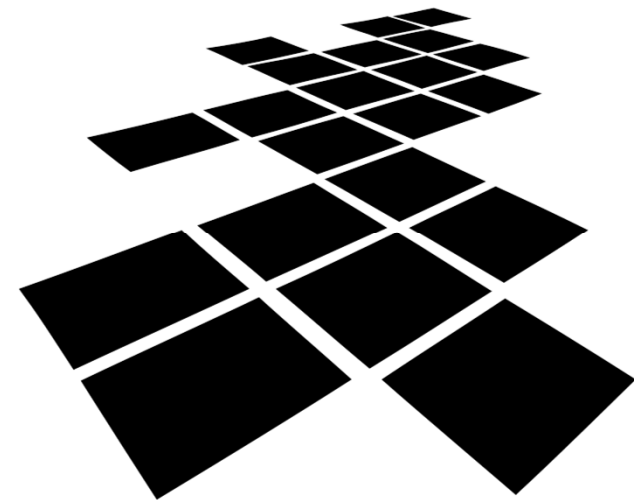


Systemy operacyjne

Przykłady implementacji planowania przydziału procesora

**Wykład prowadzą:
Jerzy Brzeziński
Dariusz Wawrzyniak**

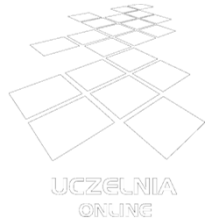


**UCZELNIA
ONLINE**

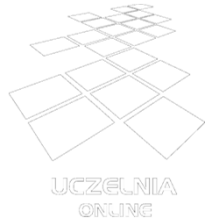


Plan wykładu

- Tradycyjne szeregowanie w systemie UNIX
- Szeregowanie w systemie Linux z jądrem 2.6
- Szeregowanie w systemie Windows 2000/XP



Szeregowanie w systemie UNIX



Informacje wstępne

- Stosowany jest algorytm rotacyjny z wywłaszczaniem, oparty na priorytetach dynamicznych.
- Wartość priorytetu jest z zakresu od 0 do 127, mniejsza wartość liczbową oznacza wyższy priorytet.
- Priorytet (dynamiczny) składa się z części statycznej i części modyfikowanej przez planistę.
- Część statyczna składa się z *bazy* (definiowanej przez system) oraz wartości *nice* (ustalanej przez użytkownika lub nadzorcę).
- Priorytet procesu ustalany jest zawsze, gdy proces ten przechodzi z trybu jądra do trybu użytkownika.
- Okresowo (mniej więcej co 1 sekundę) przeliczane są priorytety wszystkich procesów gotowych.

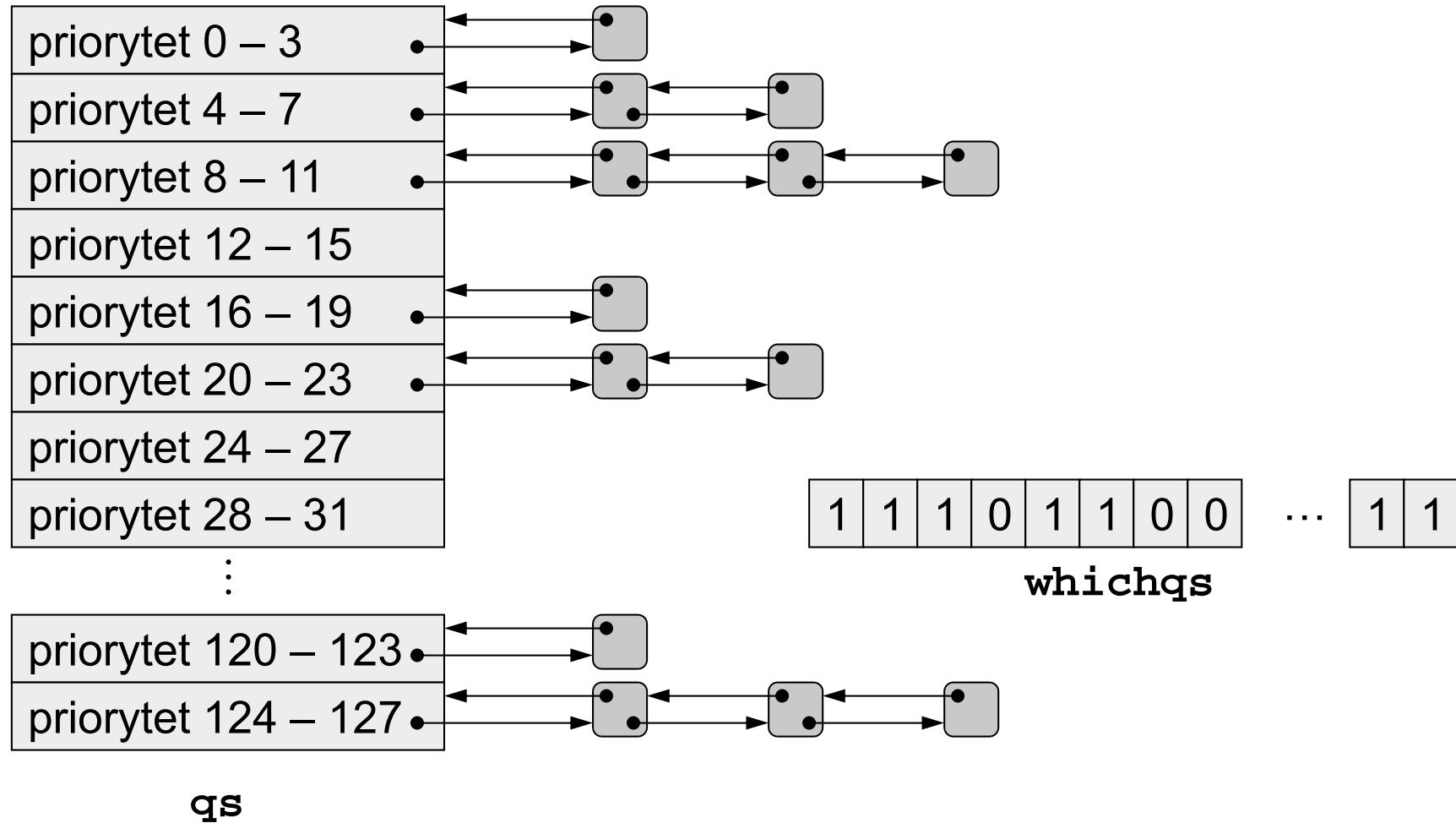


Struktury danych na potrzeby szeregowania

- Na potrzeby szeregowania procesy zorganizowane są w kolejkę priorytetową — qs .
- Każda pozycja tablicy kolejek odpowiada czterem wartościom priorytetu.
- Wektor bitowy $whichqs$ wskazuje pozycje, na których są niepuste kolejki.
- Wyróżnia się 3 zakresy priorytetu:
 - poziom jądra dla procesów nieprzerywalnych,
 - poziom jądra dla procesów przerywalnych,
 - poziom użytkownika.



Kolejka priorytetowa





Parametry funkcji priorytetu

- cpu_i — miara dotychczasowego wykorzystania procesora przez i -ty proces,
- $baza_i$ — priorytet bazowy procesu i -tego,
- $nice_i$ — składowa priorytetu procesu i -tego definiowana przez użytkownika,
- pri_i — priorytet procesu i -tego (mniejsza wartość oznacza wyższy priorytet),
- $usrpri_i$ — priorytet procesu i -tego w trybie użytkownika.



Przeliczanie priorytetu

$$1) \quad cpu_i := cpu_i + 1$$

$$2) \quad cpu_i := \frac{cpu_i}{2}$$

$$3) \quad usrpri_i := baza_i + \frac{cpu_i}{2} + nice_i$$

$$4) \quad pri_i := usrpri_i$$



Przykład

$usrpri_1$	cpu_1
60	0 1 ... 60
75	30
67	15
63	7 8 ... 67

proces P_1

$usrpri_2$	cpu_2
60	0
60	0 1 ... 60
75	30
67	15

proces P_2

$usrpri_3$	cpu_3
60	0
60	0
60	0 1 ... 60
75	30

proces P_3



Szeregowanie w systemie Linux (jądro 2.6)

Algorytm o stałej złożoności $O(1)$,
zastąpiony wersji 2.6.23 przez CFS



Informacje wstępne

- Stosowany jest algorytm rotacyjny z wywłaszczaniem, oparty na priorytetach dynamicznych.
- Wyróżnia 140 poziomów priorytetu związanych z 3 klasami (politykami) szeregowania:
 - SCHED_OTHER — polityka szeregowania zwykłych zadań,
 - SCHED_FIFO — polityka szeregowania zadań czasu rzeczywistego zgodnie z zasadą FIFO,
 - SCHED_RR — polityka szeregowania zadań czasu rzeczywistego w sposób rotacyjny.
- Większa wartość oznacza niższy priorytet.



Priorytety procesów zwykłych

- Priorytet (dynamiczny) składa się z części statycznej i części modyfikowanej przez planistę.
- Część statyczna definiowana jest przez wartość nice z zakresu od -20 do 19.
- Priorytet dynamiczny decyduje zarówno o pierwszeństwie w dostępie do procesora jak i wielkości kwantu czasu (od 10 ms do 200 ms).
- Preferowane są (nagradzane wyższym priorytetem dynamicznym) zadania ograniczone wejściem-wyjściem.

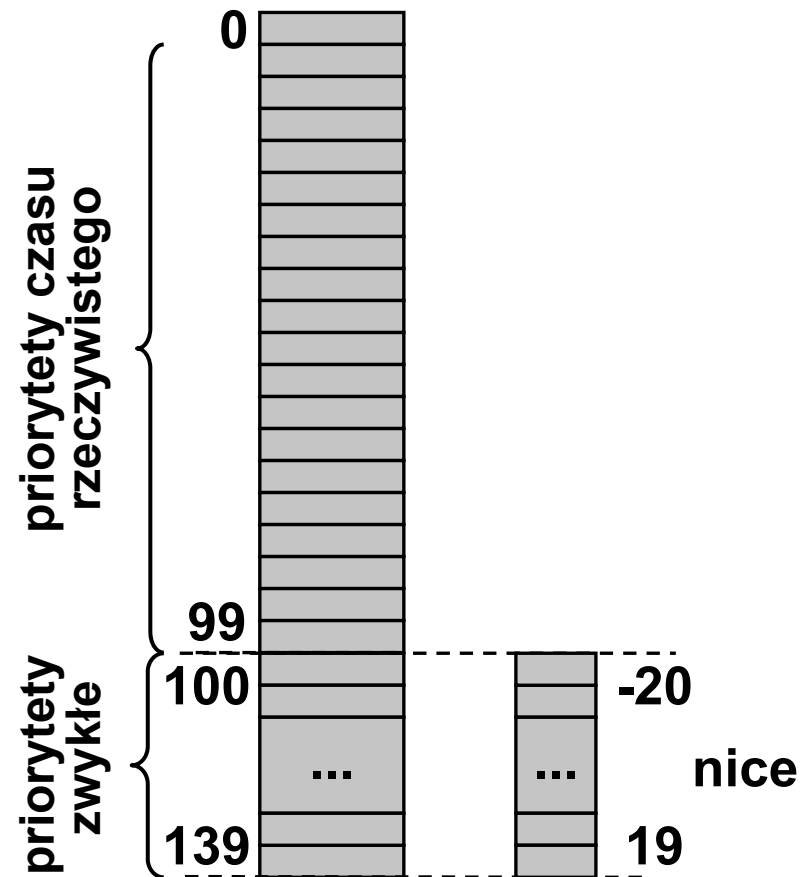


Priorytety procesów czasu rzeczywistego

- Priorytety przydzielane są statycznie (nie zmieniają się) z zakresu od 0 do 99.
- Priorytety procesów czasu rzeczywistego są zawsze większe od priorytetów zadań zwykłych.
- Do grupy zadań czasu rzeczywistego mogą być dołączane tylko procesy nadzorcy (użytkownika uprzywilejowanego root).



Odwzorowanie w tablicy priorytetów





Struktury danych do zarządzania procesami gotowymi

- Tablica priorytetów dla zadań aktywnych — tablica kolejek procesów gotowych, które nie wykorzystały jeszcze kwantu czasu
- Tablica priorytetów dla zadań przeterminowanych — tablica kolejek procesów gotowych, które wykorzystały kwant czasu
- Mapa (maska) bitowa dla każdej z tablic, identyfikująca niepuste kolejki w tablicy priorytetowej



Wywłaszczenie

- Wywłaszczenie procesu w trybie użytkownika może nastąpić przy powrocie z trybu jądra po obsłużeniu przerwania lub zakończeniu wywołania systemowego, gdy znacznik *need_resched* jest ustawiony.
- Znacznik jest ustawiany w następujących przypadkach:
 - po upływie kwantu czasu bieżącego (wykonywanego) zadania,
 - po uzyskaniu gotowości przez zadanie o wyższym niż bieżące priorytecie.



Upływanie kwantu czasu

- Po upływie kwantu czasu wykonywanego procesu zwykłego następuje przeliczenie jego priorytetu oraz wyznaczenie następnego kwantu czasu.
- Jeśli proces charakteryzuje się dużym stopniem interaktywności, a w systemie nie ma zadań przeterminowanych jest on umieszczany na odpowiedniej pozycji w tablicy priorytetów dla zadań aktywnych, w przeciwnym przypadku umieszczany jest w tablicy priorytetów dla zadań przeterminowanych.



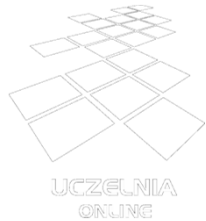
Zmiana epoki

- Jeśli tablica priorytetów dla zadań aktywnych jest pusta (wszystkie procesy gotowe wykorzystały swój kwant czasu), następuje zmiana epoki.
- Zmiana epoki oznacza zamianę tablicy priorytetów: tablica priorytetów dla zadań przeterminowanych staje się tablicą dla zadań aktywnych i odwrotnie.



Zmiana priorytetów dynamicznych

- Początkowy priorytet procesu zwykłego równy jest wartości *nice*.
- Wartość priorytetu może zostać zmieniona następująco:
 - zwiększona o 5 dla zadań ograniczonych procesorem (obniżenia priorytetu),
 - zmniejszona o 5 dla zadań ograniczonych wejściem-wyjściem, rozumianych przez domniemanie jako interaktywne,
 - pozostać bez zmian.
- Proporcjonalnie do priorytetu ustalana jest wielkość kwantu czasu (z zakresu od 10 ms do 200 ms, domyślnie 100 ms).



Ocena interaktywności

- Miarą interaktywności jest względna długość okresów korzystania z procesora oraz przebywania w stanie oczekiwania.
- Implementacją takiej koncepcji w systemie Linux jest atrybut procesu *sleep_avg*, zmniejszany w czasie wykonywania procesu, a zwiększany po wyjściu ze stanu oczekiwania.
- Wartość *sleep_avg* zmienia się od 0 do `MAX_SLEEP_AVG`, a wartością domyślną jest 10 ms.



Planowanie w klasie czasu rzeczywistego

- Na danym poziomie priorytetu najpierw wybierane są procesy klasy SCHED_FIFO.
- Proces klasy SCHED_FIFO wykonuje się tak długo, aż nie odda procesora lub nie pojawi się proces o wyższym priorytecie.
- Proces klasy SCHED_RR wykonuje się podobnie jak proces klasy SCHED_FIFO, ale po upływie kwantu czasu oddaje zasoby innemu procesowi o tym samym priorytecie (jeśli taki proces istnieje).



Szeregowanie w systemie Windows 2000/XP



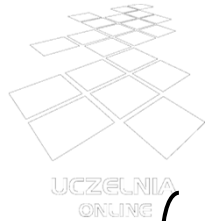
Informacje wstępne

- Szeregowaniu podlegają wątki, stanowiące obiekty w obrębie procesu.
- Stosowany jest algorytm rotacyjny z wywłaszczaniem, oparty na priorytetach dynamicznych.
- Wyróżnia się 32 poziomy priorytetu:
 - 0 — bezczynność (poziom systemowy, niedostępny),
 - 1 do 15 — priorytety dynamiczne (zmiennie)
 - 16 do 31 — priorytety czasu rzeczywistego
- Większa wartość (poziom) oznacza wyższy priorytet.

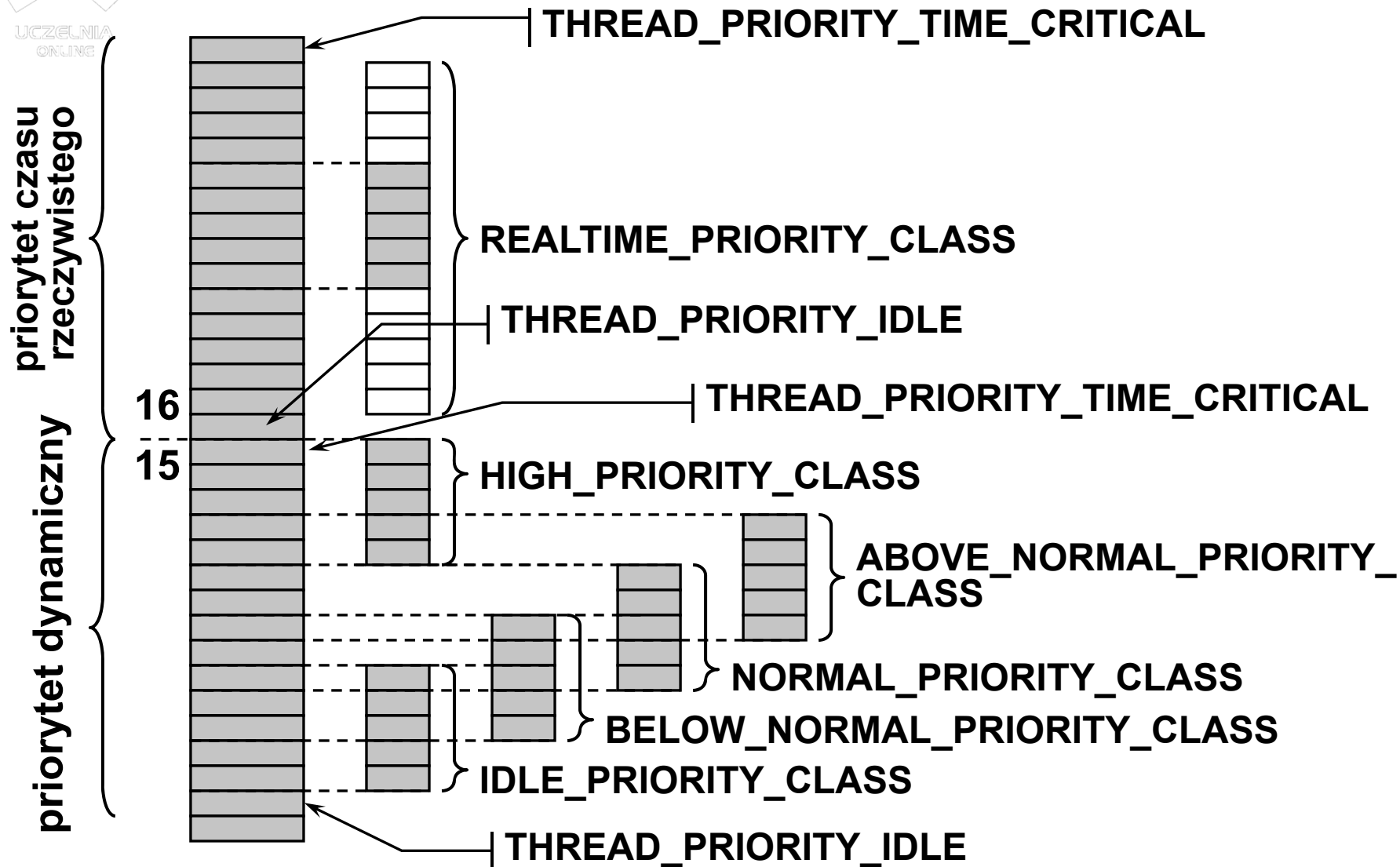


Definiowanie priorytetu

- Zdefiniowanie klasy priorytetu dla procesu: idle (4), below normal (6), normal (8), above normal (10), high (13), realtime (24).
- Zmodyfikowanie priorytetu wątku w ramach klasy priorytetu: idle (-15), lowest (-2), below normal (-1), normal (0), above normal (+1), highest (+2), time critical (+15).
- Priorytet bazowy wątku jest sumą wartości dla klasy oraz modyfikatora (priorytetu wątku).



Priorytet bazowy wątku



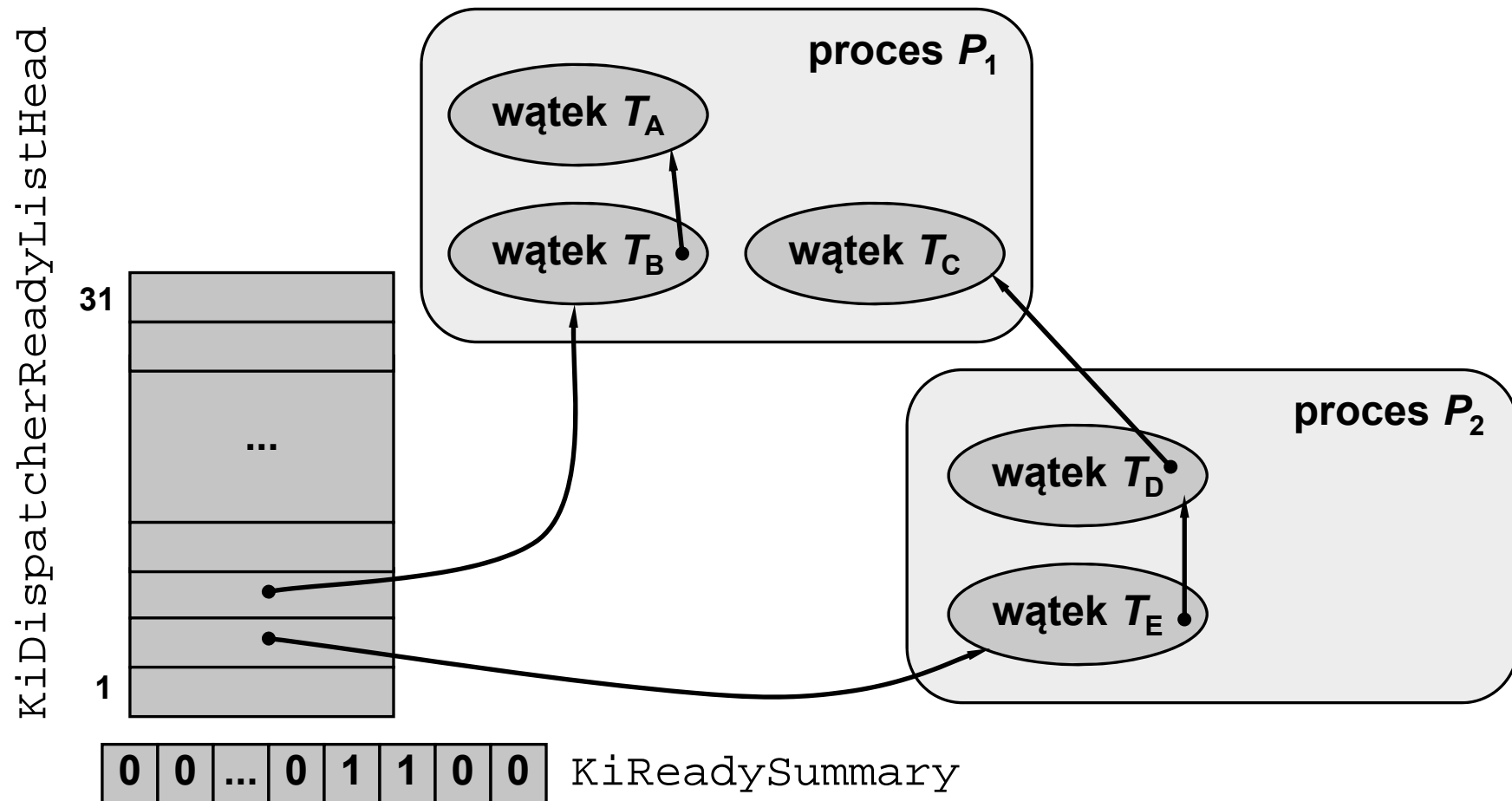


Struktury danych do zarządzania wątkami gotowymi

- *KiDispatcherReadyListHead* — tablica kolejek wątków gotowych.
- *KiReadySummary* — maska bitowa identyfikująca niepuste kolejki w tablicy *KiDispatcherReadyListHead*.



Tablica kolejek wątków gotowych





Przełączanie kontekstu

- Zakończenie działania wątku
- Przejście w stan oczekiwania (samoistnie — w wyniku odwołania do systemu operacyjnego w programie np. wątku w związku z synchronizacją lub operacją wejścia-wyjścia)
- Wywłaszczenie przez wątek o wyższym priorytecie
- Upływanie kwantu czasu



Procedury zarządzania wątkami gotowymi

- *FindReadyThread*
 - uruchamiana po zwolnieniu procesora przez wątek wykonywany,
 - szuka wątku gotowego o najwyższym priorytecie i ekspediuje go na procesor.
- *ReadyThread*
 - uruchamiana dla wątku, który przechodzi w stan gotowości lub zwiększa priorytet,
 - porównuje priorytet gotowego wątku z priorytetem wątku wykonywanego i albo wywłaszcza wątek wykonywany albo kolejkuje wątek gotowy.



Przebieg wywłaszczenia przez ReadyThread

Niech T_g oznacza wątek gotowy, T_w wątek wykonywany, a $\text{pri}(T)$ priorytet wątku T .

if $\text{pri}(T_g) \leq \text{pri}(T_w)$ **then**

if liczba wykorzystanych kwantów przez $T_g \geq 1$ **then**

 umieść T_g na końcu $KiDispatcherReadyListHead[\text{pri}(T_g)]$

else

 umieść T_g na pocz. $KiDispatcherReadyListHead[\text{pri}(T_g)]$

else

 umieść T_w na pocz. $KiDispatcherReadyListHead[\text{pri}(T_w)]$



Upływanie kwantu czasu

- Kwant wyrażany jest *jednostkami kwantu czasu*.
- Liczba jednostek do dyspozycji wątku wynosi:
 - 6 — w wersjach dla komputerów osobistych i stacji,
 - 36 — w wersjach dla serwerów.
- Z każdym taktem zegara odejmowane są 3 jednostki.
- Upływanie kwantu czasu (zredukowanie liczby jednostek do 0) powoduje wyłączenie z procesora pod warunkiem, że jest gotowy inny wątek o takim samym (lub wyższym) priorytecie.
- Jeśli dotychczas działający wątek jest jedynym o tak wysokim priorytecie, przydzielony zostaje mu kolejny kwant.



Regulacja kwantu czasu ⁽¹⁾

- Wątkowi o priorytecie mniejszym niż 16 zabierana jest jedna jednostka kwantu, gdy wchodzi on w stan oczekiwania.
- Jeśli jednak wątek działa na poziomie 14 lub wyższym, przed zredukowaniem jego kwant jest odnawiany.
- Podobne postępowanie jest przeprowadzane w pewnych przypadkach wątków pierwszoplanowych, nawet jeśli ich priorytet jest mniejszy niż 14.
- W przypadku wątków o priorytetach powyżej 15 liczba jednostek kwantu jest odnawiana po wyjściu ze stanu oczekiwania.



Regulacja kwantu czasu (2)

- Wątki procesu pierwszoplanowego mogą uzyskać 3-krotnie dłuższy kwant czasu, zależnie od ustawienia rejestru
HKLM\SYSTEM\CurrentControlSet\Control\PriorityControl\Win32PrioritySeparation
- Wydłużenie kwantu ma na celu zwiększenie preferencji dla wątków związanych z oknem pierwszoplanowym przy jednoczesnym uniknięciu zagłodzenia wątków drugoplanowych.
- W ramach przeciwdziałania głodzeniu wydłużany jest również 2krotnie czas dla wątków oczekujących długo (ponad 300 taktów) na procesor. Celem tego wydłużenia jest przeciwdziałanie inwersji priorytetów.



Zmiana dynamicznych priorytetów wątków

- Podwyższenie priorytetu wątku (maksymalnie do wartości 15) może nastąpić w następujących przypadkach:
 - po zakończeniu operacji wejścia-wyjścia,
 - po oczekiwaniu na zdarzenie lub semafor,
 - po zakończeniu oczekiwania przez wątek pierwszoplanowy,
 - po przebudzeniu wątku GUI,
 - po zbyt długim oczekiwaniu w stanie gotowości.
- Podwyższony priorytet jest obniżany sukcesywnie o 1 po upływie kwantu czasu, aż wróci do wartości bazowej.



Wzrost priorytetu po zakończeniu operacji wejścia-wyjścia

- Operacja związana z dyskiem, CD, portem równoległym lub kartą video — wzrost o 1
- Operacja związana z siecią, portem szeregowym, potokiem, skrytką pocztową — wzrost o 2
- Operacja związana z klawiaturą lub myszą — wzrost o 6
- Operacja związana z kartą dźwiękową — wzrost o 8
- Wartość podwyższenia dodawana jest do **bazowego** priorytetu wątku.



Wzrost priorytetu po oczekiwaniu na zdarzenie lub semafor

- Wartość zwiększenia, 1, dodawana jest do **bazowego** priorytetu wątku.
- Kwant czasu procesora zmniejszany jest o 1 jednostkę.



Wzrost priorytetu po zakończeniu oczekiwania przez wątek pierwszoplanowy

- Priorytet zwiększany jest o wartość zmiennej jądra *PsPrioritySeparation*, dostępnej też jako jedno z pól w rejestrze Windows pod nazwą *HKLM\SYSTEM\CurrentControlSet\Control\PriorityControl\Win32PrioritySeparation*
- Wartość zwiększenia dodawana jest do **bieżącego** priorytetu wątku.
- Zależnie do wartości *Win32PrioritySeparation* może zostać zwiększony kwant czasu dla wszystkich wątków procesu pierwszoplanowego.



Wzrost priorytetu po przebudzeniu wątku GUI

- Wartość zwiększenia — 2 — dodawana jest do **bieżącego** priorytetu wątku.



Przeciwdziałanie głodzeniu

- Co sekundę uruchamiany jest wątek *balance set manager*, którego jednym z zadań jest sprawdzanie czasu bieżącego oczekiwania wątków gotowych.
- Jeśli wątek oczekuje dłużej niż 300 taktów zegara (około 3 – 4 sekundy), jego priorytet uzyskuje wartość 15, a kwant czasu zwiększa się dwukrotnie, a w wersjach „serwerowych” czterokrotnie.
- Po upływie przyznanego kwantu, priorytet wątku wraca do poziomu bazowego.